UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Vesna S. Pajić

MODELI KONAČNIH STANJA U EKSTRAKCIJI INFORMACIJA

doktorska disertacija

Beograd, 2012.

UNIVERSITY OF BELGRADE
FACULTY OF MATHEMATICS

Vesna S. Pajić

FINITE STATE MODELS IN INFORMATION EXTRACTION

Doctoral Dissertation

Belgrade, 2012.

Mentor:
prof. dr Gordana Pavlović - Lažetić, Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:
1. prof. dr Duško Vitas, Univerzitet u Beogradu, Matematički fakultet
2. prof. dr Ivan Obradović, Univerzitet u Beogradu, Rudarsko-geološki fakultet

Datum odbrane:

Zahvaljujem se svojim saradnicima, na prvom mestu mentoru, profesorki Gordani Pavlović Lažetić, na uloženom trudu, strpljenju i dobroj volji da ova disertacija što pre ugleda svetlost dana, kao i profesoru Dušku Vitasu koji je usmerio moja istraživanja u ovom pravcu.

Posebno bih želela da se zahvalim mami Ljiljani, Milošu, Marku, Darku i Petru što su verovali u mene, pomagali mi i podržavali me u mom radu.

Vesna Pajić

Naslov doktorske disertacije:

Modeli konačnih stanja u ekstrakciji informacija

Rezime

Disertacija je posvećena istraživanju naučne oblasti nazvane **ekstrakcija informacija** (engl. *information extraction*), koja predstavlja podoblast veštačke inteligencije, a u sebi kombinuje i koristi tehnike i dostignuća više različitih oblasti računarstva. Termin "ekstrakcija informacija" će biti korišćen u dva različita konteksta. U jednom od njih misli se na ekstrakciju informacije kao naučnu oblast i tada će se koristiti skraćenica IE, preuzeta iz anglosaksonske literature u značenju "*Information Extraction*". U drugom slučaju, kada se bude mislilo na sam proces i postupak izdvajanja informacija iz teksta, koristiće se oblik "ekstrakcija informacija".

Ova disertacija predstavlja, pored pregleda postojećih metoda iz ove oblasti, i jedan originalni pristup i metod za ekstrakciju informacija baziran na konačnim transduktorima. Tokom istraživanja i rada na disertaciji, a primenom pomenutog metoda, kao rezultat formirana je baza podataka o mikroorganizmima koja sadrži fenotipske i genotipske karakteristike za 2412 vrsta i 873 rodova, namenjena za istraživanja iz oblasti bioinformatike i genetike. Baza i korišćeni metod su detaljno prikazani u nekoliko radova, publikovanih u časopisima ili izlaganih na međunarodnim konferencijama (Pajić, 2011; Pajić i sar. 2011a; Pajić i sar. 2011b)

U glavi 1 dat je uvod u oblast ekstrakcije informacije, unutar koga je opisan istorijat i razvoj metoda ove oblasti. Dalje je opisana klasifikacija tekstualnih resursa nad kojima se vrši ekstrakcija informacija, kao i klasifikacija samih informacija. Na kraju glave 1 oblast ekstrakcije informacije je upoređena sa drugim srodnim disciplinama računarstva.

Glava 2 je posvećena prikazu teorijskih osnova na kojima su zasnovana istraživanja ove disertacije. Razmatrana je teorija formalnih jezika i modela konačnih stanja, kao i njihova uzajamna veza i veza sa ekstrakcijom informacija. Akcenat je stavljen na konačne modele i metode koji su zasnovani na modelima konačnih stanja. Ovi metodi pokazuju veću preciznost od drugih metoda za ekstrakciju informacije, te su

nezamenljivi u situacijama kada je tačnost izdvojenih podataka iz teksta od presudnog značaja. Pojedini pojmovi ekstrakcije informacija - jezik relevantnih informacija, jezik izdvojenih informacija, pravila ekstrakcije, definisani su iz ugla teorije formalnih jezika. Formulisano je i dokazano osnovno svojstvo relacije transdukcije za zadato pravilo ekstrakcije. Definisan je i pojam jezika konteksta informacija i dokazano je njegovo svojstvo regularnosti.

U glavi 3 su prikazana dva različita pristupa projektovanju sistema za ekstrakciju informacija, (1) pristup baziran na mašinskom učenju i (2) pristup baziran na znanju, tzv. *Knowledge Engineering* pristup (u daljem tekstu KE). Sistemi koji koriste prvi pristup podrazumevaju postojanje anotiranih korpusa relevantnih tekstova, dok sistemi zasnovani na KE pristupu zahtevaju postojanje stručnjaka iz određene oblasti, koji su upoznati sa domenom u kome se vrši ekstrakcija informacija i koji definišu jasna pravila koja se koriste za ekstrakciju informacija iz teksta. Većina današnjih sistema za ekstrakciju informacija kombinuje ova dva pristupa, pa su tako pojedine komponente sistema bazirane na mašinskom učenju, a pojedine na pravilima za ekstrakciju. Detaljan prikaz arhitekture sistema za ekstrakciju informacija, kao i pregled postojećih sistema dat je na kraju glave 3, kao i prikaz programskog sistema UNITEX koji je i korišćen u okviru rada na ovoj disertaciji za potrebe ekstrakcije informacija.

Glava 4 je posvećena prikazu originalnog metoda za ekstrakciju informacije koji je zasnovan na konačnim transduktorima i pripada grupi sistema koji su bazirani na KE pristupu. Ovaj metod jasno razlikuje dve faze procesa ekstrakcije. U prvoj fazi se vrši identifikacija i ekstrakcija delova teksta koji odgovaraju jednom entitetu, tj. jednom slogu podataka, dok se u drugoj fazi iz tih manjih delova teksta izdvajaju informacije o vrednostima pojedinih atributa tog entiteta. Prva faza jako zavisi od strukture teksta i ne mora da se izvrši pomoću transduktora. Algoritam prve faze se menja u zavisnosti od teksta iz koga se vrši ekstrakcija. Sa druge strane, druga faza uopšte ne zavisi od strukture teksta, već se u okviru nje primenjuju transduktori koji definišu pravila ekstrakcije. Ove transduktore konstruišu stručnjaci iz specifičnih oblasti. Imajući u vidu da se u okviru velikog broja domena na sličan način opisuju slični koncepti, ovi transduktori mogu biti ponovno korišćeni za neke druge procese ekstrakcije iz iste oblasti. U glavi 4 je detaljno opisan gore pomenuti metod, prikazan je primer njegove

primene na enciklopedijski tekst iz oblasti mikrobiologije kao i rezultujuća baza podataka i data je evaluacija metoda.

Glava 5 posvećena je problemu ekstrakcije informacije i obrade tekstova na srpskom jeziku, kao jednom od morfološki bogatih jezika, koji zbog te svoje osobine zahteva nešto drugačiji pristup u obradi nego što je to slučaj sa jezicima kakav je recimo engleski jezik. Ekstrakcija informacija iz tekstova na srpskom jeziku prikazana je na primeru korpusa tekstova o vremenskoj prognozi i vremenskim prilikama, preuzetih iz većeg broja elektronskih medija.

U glavi 6 dati su neki od zaključaka, kao i smernice za dalja istraživanja iz ove oblasti.

Ključne reči:

ekstrakcija informacija, obrada prirodnih jezika, konačni automati, konačni transdukori, rekurzivne mreže prelaza

Naučna oblast:

Računarstvo i informatika

Uža naučna oblast:

Računarska obrada teksta

UDK klasifikacioni broj:

004.832.2:025.4 (043.3)

Title of doctoral dissertation:

Finite state models in information extraction

Abstract

This dissertation is on research and studying in scientific field called **information extraction,** which can be seen as a sub-area of artificial intelligence and which combines and uses techniques and achievements of several computer science areas. The term „information extraction“ will be used in two different contexts. In the first one, the term will refer to the scientific area and the acronym IE will be used in that case. In the second case, this term will refer to the very process of extracting information.

Beside the IE state-of-the-art survey, an original approach and a method for information extraction based on finite state transducers are presented. A database with microbial phenotype and genotype characteristics, for 2412 species and 873 genera has been created, as a result of the research and the work on the dissertation. The database is intended for research, in bioinformatics and genetics. The method used for the creation of the database and the database itself are described in details and published in several journals and conference proceedings (Pajić, 2011; Pajić et al. 2011a; Pajić et al. 2011b).

In the Section 1, the introduction to IE is given, together with the history of development of methods in this area. The classification of textual resources that are used for information extraction and classification of the information itself are described. At the end of the Section 1, IE is compared with other related disciplines of computer science.

Section 2 contains some excerpts from formal language theory and abstract automata, on which the dissertation is based. The mutual relationship between these two areas and their connection with IE are described. The emphasis is put on the final state models and methods based on them. These methods show higher precision than other methods for extracting information, and are indispensable in situations where the accuracy of data extracted from the text is of crucial importance. Some specific terms of information extraction - the language of the relevant information, the language of

extracted information and extraction rules, are defined from the perspective of formal language theory. The basic feature of the transduction relation for the given rule extraction is formulated and proved. The language of information context is defined and its regularilty is proven.

Two different approaches to designing systems for information extraction, (1) an approach based on machine learning, and (2) knowledge-based approach (hereafter KE) are presented in Section 3. Systems using the first approach require the existence of annotated corpora, while the KE-based systems requires the existence of human experts in specific fields, who are familiar with the domain in which information extraction is to be performed. These experts define rules used to extract information from text. Most of the systems for the information extraction combine these two approaches, so certain components of the system are based on machine learning, and other are based on expert rules for extraction. An overview of an IE system's architecture and of existing systems is given at the end of Section 3. Moreover, the software system UNITEX, which is used for the information extraction in this research, is described in more details.

In the Section 4, an original method for information extraction, based on finite state transducers, is presented, as a KE based method. This method clearly distinguishes between two phases of the extraction. The first phase is carried out to identify and extract smaller parts of the text corresponding to one entity, i.e. one data record, while in the second phase, values of some entity attributes are extracted from those pieces of text. The first phase is highly dependent on the structure of the text and does not have to use transducers. The algorithm of the first phase is different for different texts, from which extraction is performed. On the other hand, the second phase does not depend on the structure of the text. Within the second phase, transducers are applied, defining the rules of the extraction. These transducers are constructed by experts in specific areas. Bearing in mind that similar concepts are often described in a similar way, in the large number of domains, these transducers can be re-used for other processes of extraction from the same area. In the Section 4, the above-mentioned method is described in detail and an example of its application to an encyclopedic text in the field of microbiology is given, as well as the resulting database and the data evaluation.

The information extraction problem from texts in the Serbian language is presented in the Section 5, as an example of the approach needed when the natural language has a rich morphology. The data are extracted from the corpus of weather forecast reports in Serbian, collected from several Serbian electronic media. The whole process is done by UNITEX software, in which the electronic dictionaries for Serbian are applied.

In the Section 6, conclusions and directives for future research in the area are given.

Keywords:

information extraction, natural language processing, finite state automata, finite state transducers, recursive transition networks

Scientific area:

Computer Science and Informatics

Expertise:

Text processing

UDK classification number:

004.832.2:025.4 (043.3)

SADRŽAJ

Glava 1

1 EKSTRAKCIJA INFORMACIJA

1.1. Uvod

U današnje vreme, kada je uobičajeno čuvanje podataka u digitalnom obliku, svedoci smo stvaranja i postojanja velikih kolekcija različitih formata dokumenata i zapisa. Ove kolekcije čuvaju u sebi informacije o najrazličitijim znanjima do kojih je čovek došao. Međutim, pronalaženje tih informacija u kolekcijama podataka je često teško i zahtevno. Sa druge strane, veliki je broj situacija ili problema koji mogu biti uspešno rešeni jedino analizom i upotrebom tih informacija.

Danas je uobičajeno da kompanije koje drže do svoje pozicije na tržištu analiziraju različita mišljenja i osećanja koje kod korisnika izaziva neki njihov proizvod. Sa druge strane, odgovor na pitanje da li je i koliko njihov proizovd omiljen nalazi se u komentarima korisnika na nekom forumu ili veb blogu. Dakle, kompanija se suočava sa velikom količinom tekstualnih poruka, pisanih često i na različitim jezicima, pa čak i korišćenjem različitih grafičkih elemenata, koje treba da analizira. U sličnoj situaciji je i kompanija koja želi da izvuče i analizira sve podatke o konkurentnim preduzećima, a koji se pojavljuju u novinskim tekstovima. Tu je i veliki broj drugih problema, kao što su postavljanje upita nekom računaru pomoću govora, prikupljanje podataka o nekoj katastrofi iz različitih izvora i izdvajanje minimalnog skupa informacija koji tu katastrofu opisuju, analiziranje naučnih studija sa željom da se uspostavi korelacija između nekog medikamenta i uspešnosti terapije, objedinjavanje medicinskih elektronskih kartona pacijenata iz različitih medicinskih institucija u jedinstvenu bazu podataka i drugi.

Svi ovi primeri imaju po nekoliko zajedničkih elemenata:

- postoji zahtev za određenom informacijom,

- odgovor na zahtev se uglavnom nalazi unutar nestrukturiranih izvora podataka, kao što su tekst ili slike,

- nije moguće da čovek obradi te izvore podataka, jer ih ima previše,

- računari nisu u mogućnosti da direktno postave upit nad podacima, jer podaci nisu strukturirani.

Pokušaj rešavanja ovih i sličnih problema doveo je do nastanka čitave jedne oblasti, koja se i danas razvija velikom brzinom i predstavlja podoblast veštačke inteligencije, a u sebi objedinjuje mnoge tehnike drugih naučnih oblasti, kao što su računarska lingvistika, procesiranje prirodnih jezika, verovatnoća i statistika i dr.

Ekstrakcija informacija (engl. *Information extraction*, u daljem tekstu IE) je oblast računarske lingvistike koja obuhvata skup tehnika za pronalaženje informacija u tekstualnim dokumentima, koji su obično nestrukturirani ili polustrukturirani, i predstavljanje tih informacija u strukturiranom obliku. IE se koristi za analiziranje teksta i pronalaženje određenih delova teksta koji sadrže informaciju od interesa, kao i za njenu ekstrakciju iz teksta. Za ekstrakciju informacija iz nestrukturiranih tekstova neophodno je korišćenje tehnika obrade prirodnih jezika (engl. *Natural Language Processing*, u daljem tekstu NLP), koje se često kombinuju sa jezičkim resursima (elektronskim rečnicima i gramatikama).

Nestrukturirani podaci, uglavnom u formi tekstualnih datoteka, su najčešći način čuvanja podataka. To mogu biti različite enciklopedije, novinski članci, naučni radovi i sl. Informacije koje se u njima nalaze su od velikog značaja, ali istovremeno nisu pogodne za analizu i pretraživanje. Ljudsko znanje se sve više nagomilava u ovakvim dokumentima i postaje sve manje dostupno ljudima, jer je pojedinu informaciju teško locirati u tako velikim količinama teksta. Zbog toga je neophodno postojanje alata koji bi omogućili automatsku ekstrakciju informacije i njeno smeštanje u neki od formata pogodan za pretragu i manipulaciju. Ovo

uglavnom podrazumeva kreiranje novih ili dopunu postojećih baza podataka, što je od velikog značaja za razne oblasti ljudskog delovanja.

Za postizanje navedenog cilja do sada je kreiran veliki broj IE sistema, uglavnom korišćenjem dva različita pristupa: (1) pristupa baziranog na mašinskom učenju i (2) pristupa baziranog na znanju, tzv. *Knowledge Engineering* pristupa (u daljem tekstu KE). Sistemi koji koriste prvi pristup podrazumevaju postojanje anotiranih korpusa relevantnih tekstova. Ovi korpusi predstavljaju ulazne podatke za navedene sisteme, koji na osnovu njih treniraju svoje algoritme, tj. vrše podešavanje određenih parametara. Nakon toga, ovi sistemi se primenjuju na ekstrakciju informacija iz novih tekstova, nesadržanih u navedenim korpusima, ali koji obično pripadaju istom domenu. Sa druge strane, sistemi zasnovani na KE pristupu zahtevaju postojanje stručnjaka iz određene oblasti, koji su upoznati sa domenom u kome se vrši ekstrakcija informacija. Oni definišu jasna pravila koja se koriste za ekstrakciju informacija iz teksta i efikasnost takvih sistema u velikoj meri zavisi od stručnosti i sposobnosti stručnjaka koji ta pravila kreiraju.

Najpoznatiji i najčešće korišćeni sistemi zasnovani na mašinskom učenju koriste metode bazirane na skrivenim Markovljevim modelima (engl. *Hidden Markov Models*) i uslovnim slučajnim poljima (engl. *Conditional Random Fields*). Obe metode koriste modele konačnih stanja, ali su zasnovane na teoriji verovatnoće i podrazumevaju veliki skup tekstova potrebnih za treniranje, tj. podešavanje parametara (Burns i sar., 2007; Feng i sar., 2007; Freitag i Mccallum, 2000; Mccallum i sar., 2000; Ray, 2001; Peng i Mccallum, 2006; Zhong i sar., 2007; Zhu i sar., 2005).

Sistemi zasnovani na KE uglavnom koriste modele konačnih stanja kao što su konačni automati, konačni transduktori i rekurzivne mreže prelaza. Ovi modeli se koriste u različitim fazama sistema (Appelt i sar., 1993; Ayuso i sar., 1992; Hobbs i sar., 1992; Jurafsky i Martin, 2008; Krupka i sar., 1992).

Odabir metode, pa i celog pristupa problemu ekstrakcije informacije zavisi od nekoliko ključnih činjenica:

- dostupnost podataka za treniranje – ukoliko anotirani korpusi potrebni za treniranje algoritama postoje ili ih je lako i jeftino napraviti, trebalo bi odabrati pristup zasnovan na mašinskom učenje. Međutim, za kompleksne domene, gde je proces anotacije tekstova spor, skup ili težak, prednost se daje KE pristupu.

- dostupnost jezičkih resursa – ukoliko jezički resursi, kao što su leksikoni, rečnici i gramatike, postoje, ima smisla koristiti KE pristup.

- zahtevani nivo efikasnosti i preciznosti sistema – do sada najefikasniji sistemi su sistemi zasnovani na KE pristupu. Učinak i efikasnost treniranih sistema u velikoj meri zavisi od količine podataka dostupnih za proces treniranja.

Većina metoda iz oblasti IE, zasnovanih na mašinskom učenju, je inicijalno razvijena za tekstove na engleskom jeziku. Ukoliko se isti metodi primene na neke druge jezike, posebno na morfološki bogate jezike ili jezike za koje ne postoji veliki broj anotiranih korpusa, njihova efikasnost se bitno smanjuje. Takođe, sistemi zasnovani na KE pristupu generalno postižu veću preciznost, tj. podaci dobijeni ovim metodima su tačniji od podataka dobijenih mašinskim učenjem, a time i upotrebljiviji kao resursi za neka dalja istraživanja, na primer u oblasti istraživanja podataka.

Uobičajeno je da sistemi za IE imaju nekoliko tipičnih faza:

- tokenizacija – deljenje teksta na rečenice i tokene;

- morfološka analiza i obrada – posebno je značajna kod jezika sa bogatom morfologijom, kakav je i srpski jezik;

- leksička analiza i obrada – određivanje vrste reči, a posebno određivanje posebnih klasa reči od interesa kao npr. ličnih imena;

- rešavanje koreferenci – neophodno je da sistem bude u stanju da prepozna različite varijante jednog imena ('Vesna Pajić' i 'gđa Pajić'), da prepozna na koga se odnose zamenice i slično;

- kombinovanje i spajanje informacija – obično se jedna informacija nalazi na nekoliko mesta u tekstu, ili tako što se ponavlja, ili tako što se njeni delovi pominju na više mesta, pa ih je nakon izdvajanja iz teksta potrebno spojiti u jednu informaciju (na primer, u jednom novinskom članku informacija o učesnicima,

lokaciji i vremenu dešavanja nekog incidenta može da se proteže kroz nekoliko različitih rečenica ili čak i paragrafa).

U zavisnosti od domena i strukture samog teksta moguće je da neka od ovih faza izostane ili bude modifikovana. Sistemi koji će biti opisani u okviru ove doktorske disertacije u većini modula kojima se ove faze realizuju koriste modele konačnih stanja.

1.2. Istorijat i razvoj oblasti ekstrakcije informacije

Iako postoji veliki broj različitih primena metoda ekstrakcije informacije, ova oblast se tradicionalno povezuje sa ekstrakcijom informacije o događajima opisanim u tekstovima na prirodnim jezicima, baziranom na obrascima. Inicijalno, ekstrakcija informacija se razvila unutar MUC (Message Understanding Conferences) konferencija, održavanih kasnih osamdesetih godina (Grishman i Sundheim, 1996). Zadatak je bio da se u unapred definisanim skupovima obrazaca (formularima), od kojih je svaki sadržao specifična polja sa informacijama o nekim događajima, kao što su teroristički napadi u Latinskoj Americi, popune traženi podaci automatski, na osnovu informacija sadržanih u tekstualnim korpusima. Obrasci u obliku gramatika ili pravila (regularnih izraza) su traženi u tekstovima kako bi se identifikovale pojedine informacije.

U okviru MUC-1, u junu 1987. godine i MUC-2, u maju 1989. godine, korpus koji je pretraživan sačinjavale su telegrafske poruke o različitim terorističkim operacijama. Zadatak je bio da sistem za ekstrakciju informacije kreira slogove podataka o tome ko je uradio šta kome, kada i gde. U okviru konferencija MUC-3 i MUC-4 održanih u junu 1991. godine i junu 1992. godine, respektivno, korpus se sastojao od novinskih članaka i transkripata radio emisija, prevednih sa španskog jezika. Težište je bilo na člancima i zapisima o terorizmu u Latinskoj Americi. Članci su se po obimu kretali od jedne trećine strane do dve strane. Zadatak popunjavanja obrazaca zahtevao je identifikovanje, između ostalog, zločinaca i žrtava za svaki teroristički akt opisan u tekstu, tip fizičkog objekta koji je napadnut ili uništen, datum, mesto i efekte. Veliki broj članaka je

opisivao višestruke incidente, dok je ostatak teksta bio potpuno irelevantan. Sledi nekoliko relevantnih isečaka teksta iz jednog izveštaja:

```
San Salvador, 19 Apr 89 (ACAN-EFE) -- [TEXT] Salvadoran President-
elect Alfredo Cristiani condemned the terrorist killing of
Attorney General Roberto Garcia Alvarado and accused the Farabundo
Marti National Liberation Front (FMLN) of the crime.
...
Garcia Alvarado, 56, was killed when a bomb placed by urban
guerrillas on his vehicle exploded as it came to a halt at an
intersection in downtown San Salvador.
...
Vice President-elect Francisco Merino said that when the attorney
general's car stopped at a light on a street in downtown San
Salvador, an individual placed a bomb on the roof of the armored
vehicle.
...
3
According to the police and Garcia Alvarado's driver, who escaped
unscathed, the attorney general was traveling with two bodyguards.
One of them was injured.
```

Ovaj tekst pripada skupu od 1000 poruka korišćenih za finalnu evaluaciju sistema za ekstrakciju informacija prikazanih u okviru MUC-3 konferencije.

Neki od odgovarajućih slogova podataka su bili:

```
Incident: Date 19 Apr 89
Incident: Location El Salvador: San Salvador (CITY)
Incident: Type Bombing
Perpetrator: Individual ID urban guerrillas
Perpetrator: Organization ID FMLN
Perpetrator: Confidence Suspected or Accused by Authorities: FMLN
Physical Target: Description vehicle
Physical Target: Effect Some Damage: vehicle
Human Target: Name Roberto Garcia Alvarado
Human Target: Description attorney general: Roberto Garcia
Alvarado
driver
bodyguards
Human Target: Effect Death: Roberto Garcia Alvarado
```

```
No Injury: driver
Injury: bodyguards
```

Zadatak konferencije MUC-5 održane u julu 1993. godine bio je da se ekstrahuju informacije o zajedničkim poduhvatima iz poslovnih članaka, uključujući učesnike zajedničkih poduhvata, rezultujuće kompanije, vlasništvo i planirane aktivnosti. Ovo je jedan primer tipičnog teksta:

```
Bridgestone Sports Co. said Friday it has set up a joint venture
in Taiwan with a local concern and a Japanese trading house to
produce golf clubs to be shipped to Japan.
The joint venture, Bridgestone Sports Taiwan Co., capitalized at
20 million new Taiwan dollars, will start production in January
1990 with production of 20,000 iron and "metal wood" clubs a
month.
```

Informacije koje je trebalo da budu izvučene iz teksta prikazane su u nastavku:

```
Relationship: TIE-UP
Entities: Bridgestone Sports Co.
a local concern
a Japanese trading house
Joint Venture Company: Bridgestone Sports Taiwan Co.
Activity: ACTIVITY-1
Amount: NT$20000000
ACTIVITY-1:
Activity: PRODUCTION
Company: Bridgestone Sports Taiwan Co.
Product iron and `metal wood' clubs
Start Date: DURING: January 1990
```

MUC je bio prvi značajan pokušaj da se započne istraživanje u oblasti automatske ekstrakcije podataka i u velikoj meri je odredio pravce daljih istraživanja u narednim dekadama. Čak i danas, ekstrakcija informacija se često povezuje sa tehnikama pretrage baziranim na obrascima ili pravilima. Još uvek je veoma aktuelna prvobitna, a danas tradicionalna definicija koju su dali Riloff i Lorenzen još 1999. godine, a koja glasi: "*Sistemi za ekstrakciju informacija izvlače informacije specifične za neki domen iz tekstova pisanih prirodnim jezicima. Domen i tip informacije koja se izvlači mora da bude poznat unapred.*

Sistemi za ekstrakciju informacija često se fokusiraju na identifikaciji objekata, kao što su ljudi, mesta, kompanije i fizički objekti.....Obrasci specifični za pojedini domen se koriste kako bi se relevantne informacije identifikovale." (Riloff i Lorenzen, 1999.) Glavni koncepti na koje ukazuje navedena definicija jesu da sistem za ekstrakciju informacija mora da identifikuje informaciju u tekstu (tj. nestrukturiranom izvoru informacije), kao i da informacija ima neko unapred definisano značenje (čovek, mesto..). Pa ipak, za današnje pojmove ova tradicionalna definicija je postala suviše ograničavajuća.

Ekstrakcija informacija ne mora nužno da podrazumeva da se radi o informacijama u tekstualnom obliku. Čak i ukoliko se ograniči samo na tekstualne resurse, ekstrakcija ne mora da bude specifična za određeni domen, čak naprotiv. Idealan sistem za ekstrakciju informacija bi trebalo da bude nezavisan od domena, ili barem lako prilagodljiv različitim domenima. Takođe, tip informacije bi trebalo da bude što je moguće univerzalniji, baziran na ontologijama i relacijama između objekata.

Kao još jedna posledica MUC konferencije pojavljuje se i to da se svaka tehnika koja koristi poklapanje obrazaca da bi organizovala podatke u strukturirani oblik smatra ekstrakcijom informacija. Na primer, krajem devedesetih godina prošlog veka pojavio se veliki broj pokušaja da se informacije sa veb strana konvertuju u neki strukturirani format. Neki od tih pokušaja analiziraju tekst sa veb strane na prirodnom jeziku, ali većina koristi tehnike poklapanja obrazaca (*pattern matching*) koje analiziraju samo osobine i oznake *mark-up* jezika da bi prikupili podatke sa veb strane. Mi ćemo pretpostaviti da ekstrakcija podataka zahteva barem neki nivo semantičke analize sadržaja.

Dalje, ekstrakcija informacija je često zadužena i za pronalaženje odnosa i relacija između izvučenih podataka, i to onih koje se nalaze u tekstu. Cowie i Lehnert su pokušali da obuhvate navedene osobine procesa ekstrakcije informacija (Cowie i Lehnert, 1996.). Za njih, ekstrakcija informacija je "*proces koji obuhvata izdvajanje fragmenata informacije iz tekstova na prirodnom jeziku i njihovo povezivanje u jedinstveni okvir i strukturu*". Za današnje shvatanje pojma ekstrakcije informacija ovakvo viđenje je prihvatljivije. Pa ipak, nedostaje još

nešto. Iako se u okviru ove disertacije uglavnom obraća pažnja na ekstrakciju informacija iz teksta, tekst nije jedini nestrukturirani izvor podataka (takvi su, npr. slike, video i sl.).

Zbog svega navedenog, u okviru ove disertacije proces ekstrakcije informacije će biti posmatran u svetlu definicije koja je jasna i koncizna, nije ograničavajuća i objedinjuje sve do sada navedeno, a data je u (Moens, 2006.) i glasi: "*Ekstrakcija informacija je proces identifikacije, postupne ili istovremene klasifikacije u semantičke klase, specifične informacije pronađene u nestrukturiranom izvoru podataka, kao što je tekst na prirodnom jeziku, u cilju omogućavanja da ta informacija bude pogodna za obradu*."

1.3. Tipovi tekstualnih resursa

U okviru ove disertacije akcenat će biti stavljen upravo na metode koji se koriste za ekstrakciju informacija iz tekstova, i to tekstova na prirodnim jezicima. S obzirom na to da se i sam proces ekstrakcije informacije razlikuje u odnosu na to da li postoji i koliko je stroga struktura teksta koji se obrađuje, veoma je važno prethodno analizirati strukturu tekstualnog resursa. U tom smislu, tekstualni resursi mogu biti klasifikovani u jednu od tri kategorije: strukturirani, polu-strukturirani i nestrukturirani. Primeri iz poglavlja 1.2 predstavljaju nestrukturirane tekstove, dok je recimo danas izuzetno aktuelno ekstrahovanje podataka sa veb strana, koje opet pripadaju polustrukturiranim tekstovima.

Strukturirani tekstovi. U strukturirane tekstove se svrstavaju tekstualni podaci kod kojih postoji jasna i stroga struktura u kojoj su prikazani, a koja opet određuje značenje tih podataka. Ovakvi podaci se već nalaze u nekoj vrsti relacije. Najčešće su dobijeni iz neke baze podataka, zapisani i označeni tako da se pojedinačnim podacima jasno dodeljuje značenje. Primer ovakvih tekstova je recimo izveštaj iz baze podataka u XML formatu. Strukturirani podaci su relativno jednostavni za obradu u smislu ekstrakcije informacije.

Polu-strukturirani tekstovi. Kod polu-strukturiranih tekstova informacije se nalaze u takvom obliku da nije potrebna nikakva sintaksna analiza na nivou reči ili rečenica. Iako informacija o značenju može da izostane, ona se često

veoma jednostavno otkriva na osnovu same strukture teksta. Primer su oglasi u novinama, ili HTML strane. Prema (Baumgartner i sar. 2001) i (Liu i sar. 2000) i XML i HTML strane sadrže polustrukturirane podatke, s tim što su HTML strane više "razumljive" za čoveka, tj. namenjene za prezentaciju podataka čoveku. Nedostaje im mogućnost da se razlikuje struktura podataka od informacija namenjenih za vizuelni prikaz, što nije slučaj sa XML dokumentima. Većina istraživača i autora danas pod polustrukturiranim tekstovima uglavnom smatra neki od ova dva formata teksta. Međutim, i tekstovi kao što su na primer enciklopedije, kod kojih je retorička struktura (paragrafi, poglavlja, naslovi i dr.) takva da je iz nje moguće zaključiti ponešto i o značenju podataka koji su prikazani u tekstu, takođe se mogu smatrati polustrukturiranim tekstom, što će u okviru ove disertacije biti i demonstrirano.

Nestrukturirani tekstovi. Nestrukturirani tekstovi su tekstovi u slobodnoj formi, kojima obično nedostaje struktura koja bi ukazala na značenje informacija. Postoji jedino retorička struktura, ali je kod ovog tipa tekstova na osnovu nje nemoguće locirati informaciju u tekstu i otkriti njeno značenje. Za izdvajanje relevantnih informacija neophodno je razumevanje teksta, pa se za ekstrakciju informacija iz ovih tekstova koriste tehnike automatskog razumevanje prirodnih jezika (Allen, 1995; Mirhaji i sar., 2006; Soderland, 1999).

Pojam nestrukturiranih podataka, koji se pominje i u definiciji ekstrakcije informacija, a i u prethodnom razmatranju, treba dodatno pojasniti. Pod tim pojmom, podrazumeva se pisani tekst ili govor, video zapisi, audio zapisi, slike. Nestrukturiranost tih podataka ne znači da su podaci nekoherentni, već da su zapisani na takav način da je njihovo automatsko tumačenje teško. Možda bi bilo bolje koristiti termin “ računarski nevidljivi podaci" ili "transparentni podaci”. Upravo ekstrakcija podataka prepoznaje takve podatke i daje im značenje. Kao posledica toga, podaci postaju strukturirani ili polustrukturirani i na taj način je moguće da budu procesirani od strane računara.

U okviru ove disertacije, akcenat će biti na tekstovima pisanim na prirodnim jezicima, koji mogu biti različitog žanra ili tipa (novinski članci, naučni radovi, policijski izveštaji, medicinski kartoni, enciklopedije i sl.). Ukoliko ne

bude navedeno drugačije, smatraćemo da su tekstovi koherentni (logički povezani) i bez grešaka, iako u većini realnih situacija to nije slučaj. Pisani podaci, posebno u elektronskom obliku, su veoma često nekoherentni, sa puno grešaka koje su ili namerne (kao kod spam poruka) ili slučajne. Greške se pojavljuju i u izlazu sistema za prepoznavanje govora. Postoje algortimi koji mogu da budu primenjeni i na takve slučajeve, ali oni nisu u domenu istraživanja ove disertacije.

1.4. Ekstrakcija semantičkih informacija

Tokom procesa ekstrakcije informacije, tražena informacija se identifikuje u tekstu na osnovu lingvističke organizacije teksta. Bilo koji tekst na bilo kom jeziku sastoji se od kompleksnog slaganja rekurzivnih obrazaca koji formiraju koherentnu, smislenu celinu. Ovo je posledica principa kompozitnosti (Szabo, 2004), principa koji se nalazi u osnovi mnogih modernih pristupa jeziku i koji tvrdi da značenje bilo kog kompleksnog lingvističkog izraza jeste funkcija značenja njegovih sastavnih delova. Rečenica u srpskom ili engleskom jeziku se sastoji od manjih delova, subjekta, predikata, jednog ili više objekata. Njihova pojedinačna značenja, kao i redosled i oblik (na primer, glagolsko vreme ili padež) omogućavaju nam da utvrdimo značenje cele rečenice.

Još uvek nije sasvim jasno kako se lingvistički slojevi međusobno odnose, ali razne lingvističke teorije i metodi obrade prirodnih jezika pretpostavljaju da postoji takozvani realizacioni lanac, tj. niz projekcija od ideje do sekvenci karaktera kojima je ta ideja izražena u tekstu (Kiparsky 2002). Ova teoretska tvrdnja ima svoje uporište u gramatici koju je pisao indijski gramatičar Panini u 5. veku pre nove ere. Prema toj tvrdnji, značenje u jeziku se realizuje unutar lingvističke strukture kroz nekoliko različitih lingvističkih nivoa, od kojih je svaki rezultat projekcije osobina višeg, apstraktnijeg nivoa. Na primer, po Paniniju, značenje jednostavne rečenice počinje kao ideja u mozgu pisca. Ona zatim prolazi kroz stanje u kome se događaj i svi njegovi učesnici prevode u skup semantičkih koncepata, od kojih se svaki dalje prevodi u skup gramatičkih i leksičkih koncepata, a oni dalje u sekvence karaktera.

IE (kao i NLP) pretpostavlja da je taj proces projekcija dvosmeran, tj. da je moguće prvobitnu ideju od koje je neka rečenica potekla rekonstruisati na osnovu tih površinskih projekcija serijom inverznih procesa. Drugim rečima, IE pretpostavlja da iako semantičke informacije u tekstu i njihove lingvističke organizacije nisu u prvi mah uočljive računarski, one ipak mogu biti uočene uzimajući u obzir površinske lingvističke pravilnosti. IE sistem će koristiti skup obrazaca za ekstrakciju, koji su ili ručno konstruisani ili automatski naučeni, da bi izvukli informacije iz teksta, a zatim ih prebacili u format koji ima jasniju strukturu. Ovo otkrivanje semantičkih informacija, tj. pridruživanje određenog značenja tekstualnim podacima, u stvari predstavlja osnovu za strukturiranje podataka, jer kada se govori o strukturiranim tekstualnim podacima upravo se misli na delove teksta koji imaju jasno značenje, pri čemu je to značenje na neki način i eksplicitno naznačeno u samom tekstu.

1.5. Ekstrakcija specifičnih informacija

Ekstrakcija informacija se tradicionalno primenjuje u situacijama kada je unapred poznata vrsta semantičkih informacija koja treba da se izdvoji. Na primer, potrebno je utvrditi koji događaji se pominju u tekstu i kada će se oni odigrati. S obzirom da postoji ograničen broj izraza kojima mogu biti opisani događaji i vremenske odrednice u jednom prirodnom jeziku, moguće je dizajnirati metod koji identifikuje određeni događaj i odgovarajuće vreme u tekstu. U zavisnoti od potrebne informacije, moguće je konstruisati različite modele kako bi se prepoznala razlika između različitih klasa na različitom nivou semantičke preciznosti. U nekim primenama, na primer, biće dovoljno da se utvrdi da jedan deo rečenice predstavlja vremensku odrednicu, dok će u drugim slučajevima biti potrebno da se utvrde različite klase vremenskih odrednica, na primer prošlost, sadašnjost ili budućnost.

Specifičnost implicira da ne treba da bude unapred definisana samo semantička priroda ciljane informacije, već i jedinica i polje (prostor, obim) elemenata koji se ekstrahuju. Tipična jedinica ekstrakcije za IE sistem su složene reči i osnovne imenične fraze, ali u nekim drugim primenama moguće je da budu

izdvojene i neke druge lingvističke jedinice, kao što su glagolske fraze, vremenske odrednice, pa čak i rečenice ili veće retoričke strukture. Takođe, informacija može biti izvučena iz jedne ili više klauzula ili rečenica, pre nego što se prikaže kao izlaz nekog IE sistema. Tako je i u primeru kada je potrebno izvući podatke o nekom događaju. Može da se desi da se sam događaj i mesto njegovog održavanja pominju u prvoj rečenici, ali ime učesnika i vreme održvanja u nekoj drugoj rečenici novinskog članka.

Tokom MUC konferencije, IE zadaci su bili veoma uprošćeni i svedeni. Najpopularniji je bio zadatak prepoznavanja imenovanih entita (imena ljudi, organizacija, lokacija, i sl). Iako je zadatak jednostavan na prvi pogled, velika pažnja mora biti posvećena razrešenju koreferenci u tekstu, tj. utvrđivanju da li se dva izraza u tekstu odnose na isti entitet, što ovaj zadatak čini veoma složenim. Pa ipak, ovako jasno postavljanje zadataka u oblasti IE u velikoj meri je uticalo na istraživanja u ovoj oblasti. Zato, iako tradicionalni zadaci ekstrakcije informacija pokrivaju samo uzak skup zahteva koji se danas postavljaju pred istraživače, ipak oni definišu neka od glavnih ciljeva ove oblasti. Danas su veoma aktuelni problemi prepoznavanja relacija između entiteta, zatim neki problemi karakteristični za pojedine domene, kao što su prepoznavanje datuma do kada je neki proizvod dostupan sa veb strane, ekstrakcija naučnih podataka iz različitih publikacija i ekstrakcija simptoma i tretmana iz medicinskih kartona pacijenata.

1.6. Klasifikacija i strukturisanje

Tipično za ekstrakciju informacija jeste da se informacija nakon njenog izdvajanja iz teksta na neki način semantički klasifikuje, kako bi bila jasna njena dalja upotreba u informacionom sistemu. Na taj način, informacija iz nestrukturiranog izvora postaje strukturirana (tj. kompjuterski transparentna i semantički dobro definisana). U ekstremnim slučajevima, informacija koja je doslovno ektsrahovana iz teksta se isključuje iz daljeg procesiranja, ali to nije uobičajeno. Uglavnom se informacija i ekstrahuje iz teksta da bi se kasnije upotrebila u nekom drugom procesu obrade (matematička i statistička analiza, popunjavanje baze podataka i dr.)

Svaki postupak klasifikacije zahteva postojanje određene semantičke klasifikacione šeme, tj. skupa semantičkih klasa koje su organizovane na neki relevantni način (na primer nekom hijerarhijom) i koje se koriste da se izvučeni delovi informacija kategorišu u određene grupe. Ove klasifikacione šeme mogu biti veoma različite, od malih skupova apstraktnih klasa, do veoma detaljno razrađenih i specifičnih klasifikacija.

Na osnovu prirode i namene samog sistema, postoje takozvani IE sistemi zatvorenog i otvorenog domena. Tradicionalno, IE sistemi su bili zatvorenog domena, što znači da su bili dizajnirani za veoma specijalizovane, određene domene znanja, pa su samim tim koristili i veoma specifična pravila za klasifikaciju. Na primer, MUC sistemi su pokrivali veoma uske oblasti, kao što su vojni susreti, terorizam u Latinskoj Americi ili internacionalne zajedničke poduhvate (Grishman i Sundheim, 1996). IE sistemi nezavisni od domena, sa druge strane, su sposobni da obrađuju tekstove veoma heterogene po tipu ili domenu. Oni obično koriste veoma uopštene klasifikacione šeme, koje mogu biti dodatno rafinisane ukoliko je neophodna detaljnija klasifikacija.

Ranije je pomenuto da IE sistemi konvertuju nestrukturirane informacije iz tekstova na prirodnim jezicima u strukturirane. Samim tim mora da postoji unapred definisana struktura, tj. način reprezentacije, u koji se ekstrahovana informacija raspoređuje. Prvi IE sistemi su koristili tzv. šablone za opisivanje pojedinih događaja (a kasnije i složenijih scenarija). Šabloni se sastoje od skupa parova atribut-vrednost (nazvanih *slots*), od kojih svaki predstavlja jedan relevantan aspekt nekog događaja. Zadatak IE sistema tradicionalno je bio da iz teksta izvuče informacije kojima bi popunio odgovarajuće slotove šablona. Da bi ustanovio koji deo informacije odgovara kom slotu, IE sistem koristi skup pravila za ekstrakciju. Ova pravila navode koju formalnu ili lingvističku osobinu neki deo informacije mora da poseduje da bi odgovarao određenoj semantičkoj klasi. Posebno u ranijim sistemima ova pravila su ručno pisana (FASTUS sistem razvijen od strane Appelt i sar. 1993.). Danas, tehnike mašinskog učenja igraju dominantnu ulogu u ekstrakciji informacija. U većini slučajeva koristi se nadgledano učenje, u kome algoritam za učenje koristi korpus za treniranje sa ručno obeleženim primerima kako bi indukovao pravila ekstrakcije (CRYSTAL

sistem razvijen od strane Soderland i sar. 1995.). U nekim slučajevima moguće je primeniti i nenadgledano učenje, i tada nije potreban korpus za treniranje. Na primer, nenadgledano učenje je implementirano u sistemima za rezrešenje koreferenci imeničkih fraza (Cardie i Wagstaff 1999.). Danas postoji veliko interesovanje za slabo nadgledane sisteme koji koriste ograničeni broj primera koji se ručno obeleževa (Shen i sar. 2004.). Primena ovih tehnika omogućila je IE sistemima da postanu nezavisni od domena. Dodatno, tehnike mašinskog učenja dozvoljavaju klasifikaciju probabilističkom dodelom klasa umesto isključivo determinističkog pristupa.

1.7. Discipline srodne ekstrakciji informacija

U pokušajima da se efikasno obradi velika količina informacija koja se čuva u različitim formatima, sem ekstrakcije informacije razvijene su i druge oblasti kojima čovek uz pomoć računara pokušava da pretraži ili istraži podatke. Svaka od ovih oblasti je specifična i po vrsti problema koje pokušava da reši i po svojim metodima. Pa ipak, postoji i uzajamno preklapanje tih oblasti, kao i situacije u kojima se u jednoj oblasti koristi neka druga za ispunjenje nekih specifičnih zadataka.

Information Retrieval. *Information retrieval* (IR) je oblast veštačke inteligencije koja omogućava korisniku da iz velike kolekcije dokumenata, kakve su veb ili kompanijska računarska mreža, dobije samo deo dokumenata baziran na pretrazi po ključnoj reči. IR tehnike mogu efikasno da pretražuju ogromne količine dokumenata zato što unapred kreiraju indekse na osnovu sadržaja dokumenata i time smanjuju kompleksnost svake pojedinačne pretrage. Takvi sistemi su tolerantni prema greškama, nezavisni od domena i iznad svega veoma brzi (Lewis i Jones, 1996). Uspeh IR sistema, kao i veb pretraživača, je uglavnom nastao zahvaljujući njihovoj fleksibilnosti u odnosu na upite koje korisnici postavljaju. Međutim, proširivanje upita korisnika sinonimima i izrazima koji su u vezi sa upitom, iako veoma popularno, sem povećanja broja odabranih dokumenata ima manjkavost što smanjuje preciznost. Za razliku od IR, IE ne pruža korisniku uvid u ceo dokument, već samo u konkretnu informaciju koja je

ekstrahovana iz teksta. Pa ipak, IE i IR su tesno povezani, s obzirom da se upravo IE tehnike koriste u IR sistemima, posebno za automatsko indeksiranje dokumenata, na osnovu koga se dalje vrši pretraga. Više o ovoj oblasti videti u (Manning i sar. 2008).

Sumarizacija. Sumarizacija (ili automatska sumarizacija) je proces u kome se jedan tekst smanjuje po obimu, na način da se odabiraju fraze ili rečenice koje se smatraju reprezentativnim za taj tekst, a da se pri tom ne izgube važne informacije iz teksta. Na sličan način se vrši i sumarizacija više različitih tekstova u kojima ima preklapanja istih informacija. Sumarizacijom se vrši izdvajanje tih informacija iz teksta i njihovo objedinjavanja. IE je veoma korisna u procesu sumarizacije, posebno u njenoj prvoj fazi, u okviru koje se informacije prikazuju u obliku relevantnih fraza sličnih novinskim naslovima. Automatska sumarizacija je detaljno obrađena u više publikacija (Mani i Maybury, 1999; Yatsko i sar. 2010).

Razumevanje teksta *(Text understanding).* Razumevanje teksta je oblast koju čine metodi koji pokušavaju da na neki način rastumače ceo tekst, a ne samo neke njegove delove, kao što je slučaj sa IE sistemima. IE sistemi smatraju za relevantne samo delove teksta koji sadrže unapred definisane vrste informacija, bez uzimanja u obzir značenja čitavog teksta i piščevog cilja ili namere. Sa druge strane, sistemi za razumevanje teksta uzimaju u obzir čitav tekst, njegovo značenje, pa čak i namere pisca. Krajnja reprezentacija izlaznih podataka sistema za razumevanje teksta je mnogo kompleksnija nego što je to slučaj sa izlazom IE sistema, s obzirom da mora da odgovara kompleksnosti jezika na kome je tekst napisan. Više o razumevanju teksta videti u (Allen, 1995) i (Mirhaji i sar. 2006).

Otkrivanje znanja (*Knowledge discovery).* Korišćenje termina "ekstrakcija" implicira da je ciljana informacija prikazana u tekstu eksplicitno, tj. da je dostupna u vidu leksičkih elemenata (reči, grupe reči), gramatičkih konstrukcija (fraze, rečenice, izrazi) i pragmatičnog redosleda i retoričke strukture (paragrafi, odeljci) izvornog teksta. U tom smislu, IE se razlikuje od tehnika koje na određen način zaključuju i produkuju informacije iz teksta, na primer pomoću kreiranja logičkih pravila ili pokušavaju da prikupe znanje iz određenog domena pomoću deduktivnog ili induktivnog zaključivanja. Ove tehnike se obično

razvijaju u okviru oblasti koja se naziva otkrivanje znanja (Fayyad, 1996; Maier i sar. 2010).

Istraživanje podataka (*Data mining).* Razvojem naučnih disciplina, kao i informatičkih tehnologija, došlo je i do kreiranja velikih kolekcija podataka nastalih kao posledica različitih merenja i istraživanja u nauci. Obrada ovih podataka statističkim metodima kako bi se utvrdile relacije između pojedinih veličina često nije bila efikasna, jer postojeći statistički metodi nisu bili u stanju da obrade milione podataka u kratkom vremenskom periodu. Zbog toga je došlo do potrebe za kreiranjem algoritama i alata koji bi bili efikasni u takvim slučajevima. Oblast koja je na taj način nastala naziva se istraživanje podataka (Berry i Liroff, 2004; Han i Kamber, 2006). Ova oblast je izuzetno značajna, posebno u naukama kao što su bioinformatika, medicina i druge.

Istraživanje teksta *(Text mining)*. Istraživanje podataka koji su zapisani kao tekstualni podaci naziva se istraživanje teksta (*R).* Istraživanje teksta i ekstrakcija informacija iz teksta su tesno povezani, i često nije jasno da li se radi o jednom ili o drugom procesu. Pa ipak, istraživanje teksta je širi proces od procesa ekstrakcije informacije, pri čemu je ekstrakcija informacija često sastavni deo sistema za istraživanje teksta. Međutim, i IR, sumarizacija i istraživanje podataka su takođe sastavni delovi sistema za istraživanje teksta. U metodima istraživanja teksta IE je obično neophodan korak u okviru preprocesiranja teksta. Više o tehnikama i rezultatima istraživanja teksta videti u (Roberts i sar. 2009), (Gaizauskas i sar. 2004), (Feldman i Sanger, 2006) i (Bilisoly, 2008).

Glava 2

2 TEORIJSKE OSNOVE

Kao što je prikazano u prethodnoj glavi, oblast ekstrakcije informacije je različito tumačena kroz vreme, pa često nije sasvim jasno da li se neki proces može smatrati ekstrakcijom informacije ili ne. Takođe, njena sličnost sa drugim oblastima čini da se neki zadaci ekstrakcije informacije pripisuju oblastima kao što su istraživanje teksta ili otkrivanje znanja. Zbog toga će u ovom poglavlju biti date osnovne definicije i tvrđenja koja će jasnije da odrede opseg delovanja metoda ekstrakcije informacije.

Za problem ekstrakcije informacije iz teksta od velikog značaja su teorijska znanja o jeziku, i to sa dva stanovišta. Prvo, bilo koji sistem za ekstrakciju informacija, bez obzira na metode kojima se ekstrakcija vrši, obavlja neki vid prethodne obrade teksta iz koga se ekstrahuju informacije. Kako su tekstovi pisani nekim jezikom (često prirodnim, ali je karakterističan primer „tekstova“ koje obrađuje bioinformtika - iz genomike gde jezik predstavljaju niske nad azbukom {A, T, G, C} ili proteomike gde jezik predstavljaju niske nad 20-članom azbukom amino kiselina), ova obrada teksta se uglavnom oslanja na teoriju formalnih jezika. Sa druge strane, skup svih fraza kojima se opisuju informacije koje su prepoznate u tekstu takođe može biti posmatran kao jedan jezik (u daljem tekstu nazvan jezik relevantnih informacija). Zbog toga je deo ovog poglavlja posvećen teoriji formalnih jezika i modela konačnih stanja.

Potpoglavlja 2.1 i 2.9 predstavljaju originalni doprinos ove disertacije, dok je ostatak definicija i tvrđenja unutar ovog poglavlja preuzet uglavnom iz knjiga (Jurafsky i Martin, 2008) i (Vitas, 2006). Takođe, ovo poglavlje se bavi samo

teorijom, dok je upotreba konačnih modela u sistemima za ekstrakciju informacije opisana u poglavlju 3.

2.1. Osnovni pojmovi ekstrakcije informacije

Osnovna definicija problema ekstrakcije informacija, u odnosu na koju se u okviru ove disertacije posmatra i izučava ovaj proces, data je u (Moens, 2006.) i glasi: "*Ekstrakcija informacija je proces identifikacije, postupne ili istovremene klasifikacije u semantičke klase, specifične informacije pronađene u nestrukturiranom izvoru podataka, kao što je tekst na prirodnom jeziku, u cilju omogućavanja da ta informacija bude pogodna za obradu.*"

Polazeći od ove definicije uvešćemo još neke pojmove koji su karakteristični za proces ekstrakcije informacije, sa ciljem da se ovaj proces posmatra iz ugla teorije formalnih jezika.

Specifične informacije koje se pominju u definiciji najčešće su entiteti, odnosi i atributi.

2.1.1. Entitet

Entitet je sekvenca tokena u tekstu koja identifikuje neki objekat višeg semantičkog nivoa i ima određeno značenje koje je od interesa za korisnika. Svaki entitet je određen tipom i vrednošću. Dakle, entitetima se smatraju delovi teksta koji imenuju neki objekat, pa ti delovi teksta najčešće predstavljaju neke imeničke fraze. Tipovi entiteta koji su uobičajeni u različitim procesima ekstrakcije su imena ličnosti, imena kompanija, imena geografskih lokaliteta, adrese, fraze koje opisuju neko vreme (datum ili period), i ovakvi entiteti su najviše obrađivani u literaturi. Posebno su popularizovani u okviru konferencija MUC (Chincor, 1998; Grishman i Sundheim, 1996), ACE (Doddington i sar., 2004) i CoNLL (Sang i Meulder, 2003). Međutim, entiteti mogu biti i imena organizama u literaturi iz oblasti biologije, nazivi enzima ili proteina, hemijska jedinjenja, nazivi medikamenata, bolesti i sl. U okviru ACE takmičenja u izdvajanju odnosa između entiteta definisano je preko 100 različitih tipova entiteta.

Primer. U ovom primeru je prikazan klasičan primer ekstrakcije entiteta.

Nestrukturiran tekst

```
Slobodan Tišma dobitnik je ovogodišnje NIN-ove nagrade za roman
"Bernardijeva soba"
```

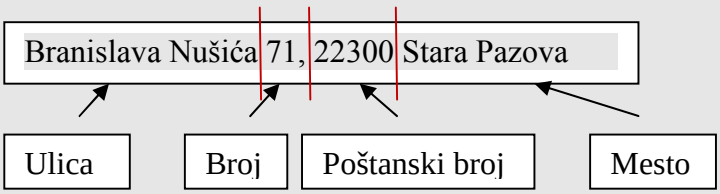
Izdvojeni (označeni) entiteti

```
<osoba>Slobodan Tišma</osoba> dobitnik je ovogodišnje
<casopis>NIN</casopis>-ove nagrade za roman <nagrada>"Bernardijeva
soba"</nagrada>
```

♦

Problem ekstrakcije entiteta ponekad može biti tretiran i kao problem segmentacije nekog teksta. Na primer, adrese koje se nalaze u okviru neke baze podataka je potrebno izdeliti na sastavne, strukturne delove, pri čemu svaki od tih delova predstavlja jednu informaciju (ulica, broj, grad..)

Primer.



♦

Pre svakog procesa ekstrakcije informacije, prvo se pristupa jasnom definisanju tipu entiteta koji se traže u tekstu. Tip entiteta često određuje i sam proces ekstrakcije informacije, dizajn sistema za ekstrakciju informacije kao i pomoćne resurse koji će biti korišćeni u procesu.

O bilo kom tipu entiteta da se radi, entiteti su u svakom slučaju osnovna informacija od koje polazi svaki proces ekstrakcije informacije. Međutim, često nije dovoljno samo izdvojiti entitet iz teksta, već je potrebno uspostaviti određene veze između izdvojenih entiteta ili otkriti neke osobine entiteta zapisane u tekstu.

2.1.2. Odnosi između entiteta

Odnosi su relacije između jednog ili više entiteta, eksplicitno zapisane u tekstu. Entiteti koji se posmatraju mogu međusobno biti u različitim odnosima. Na primer, neka osoba može biti direktor neke kompanije, ili radnik kompanije. Neka kompanija može da se nalazi na nekoj adresi, ili da bude preseljena sa te adrese. Jedan medikament može da bude uspešan u terapiji neke bolest, ili da dovede do neželjenih dejstava. Koji odnos će biti od interesa za jedan proces ekstrakcije određuje korisnik, i on mora takođe biti unapred definisan.

Primer.

Posmatrajmo sledeći tekst:

„Lorens Blok je dobitnik nagrade Grand Master koju dodeljuje udruženje 'Mystery Writers of America'."

Izdvojeni entiteti:

E_1 - Lorens Blok

E_2 - Grand Master

E_3 - Mystery Writers of America

Izdvojeni odnosi:

E_1 <je_dobitnik> E_2

E_3 <dodeljuje> E_2

♦

Ekstrakcija odnosa se razlikuje od ekstrakcije entiteta po svojoj suštini. Entiteti u tekstu su predstavljeni nizom tokena i mogu biti jasno označeni u izvornom tekstu. Za razliku od njih, odnosi ne mogu biti predstavljeni jednostavnim označavanjem sekvenci tokena, već predstavljaju veze (asocijacije) između dva dela teksta (dva entiteta).

Odnosi mogu biti posmatrani na više načina, a sve u zavisnosti od prirode informacije koja se izdvaja. U okviru ove disertacije odnose ćemo posmatrati u zavisnosti od broja entiteta koji u njima učestvuju. **Jednostruki** odnosi su odnosi između entiteta u kojima se vrši povezivanje jednog entiteta E_1 sa jednim entitetom E_2 u jednom trenutku. Entitet E_1 može biti u istom odnosu i sa nekim

drugim entitetom E_3, ali se u tom slučaju svaki od tih odnosa posmatra kao zaseban.

Primer.

Posmatrajmo sledeći tekst:

„Lorens Blok je dobitnik nagrade Grand Master koju dodeljuje udruženje 'Mystery Writers of America'. Napisao je preko 50 romana i višestruki je dobitnik nagrada Edgar, Shamus, i Maltese Falcon. Živi u Njujorku.“

Izdvojeni entiteti:

E_1 - Lorens Blok
E_2 - Grand Master
E_3 - Mystery Writers of America
E_4 – Edgar
E_5 – Shamus
E_6 - Maltese Falcon
E_7 – Njujorku

Neki od izdvojenih odnosa (relacija <je_dobitnik>):

Lorens Blok <je_dobitnik> Grand Master
Lorens Blok <je_dobitnik> Edgar
lorens Blok <je_dobitnik> Shamus
Lorens Blok <je_dobitnik> Maltese Falcon

♦

U prethodnom primeru jedan entitet E_1 je u istoj relaciji sa četiri različita entiteta E_2, E_4, E_5 i E_6. Svaka od tih veza se posmatra kao jedan odnos (jedna informacija), pri čemu se uzajamni odnos entiteta E_2, E_4, E_5 i E_6 ne uzima u obzir.

Za razliku od jednostrukih odnosa, **višestruki** odnosi grupišu više entiteta koji su u međusobnoj vezi. Grupe entiteta koji su u nekom odnosu često nazivamo **slogovima**. Poseban tip ekstrakcije slogova jeste ekstrakcija događaja. Na primer, izdvajanje incidenata iz novinskih tekstova uključuje identifikaciju, povezivanje i izdvajanje povezanih entiteta koji obično predstavljaju vreme, mesto, izvršioce,

žrtve i slično. Jedan takav proces ekstrakcije, u kome su izdvajani slogovi iz teksta, prikazan je u poglavlju 4 ove disertacije.

Ovde je neophodno naglasiti da je domen oblasti ekstrakcije informacije izdvajanje samo onih odnosa koji su eksplicitno dati (zapisani) u tekstu. U nekim složenijim sistemima istraživanja teksta često se koriste dodatni resursi, kao što su različite baze znanja, ontologije i slični, koji u sebi sadrže informacije o odnosima između pojedinih objekata. Takvi odnosi nisu predmet kojim se bavi oblast ekstrakcije informacija, iako mogu da je potpomognu. Dakle, od interesa je samo pronalaženje i izdvajanje onih odnosa koji su zapisani u tekstu koji se obrađuje.

2.1.3. Atributi

Atributi su sekvence tokena koje opisuju neku osobinu entiteta. Atributi koji opisuju neki entitet mogu biti različite strukture. Obično su to imeničke fraze ili pridevi, a često je moguće da atributi jednog tipa entiteta budu u stvari i sami entiteti. Primer koji sledi prikazuje jedan isečak teksta koji u različitim procesima ekstrakcije informacije može biti različito obrađen.

Primer.

„ Jupiter je peta planeta od Sunca i najveća planeta u Sunčevom sistemu. Dobio je ime po vrhovnom bogu starih Rimljana, kojeg su Grci zvali Zevs. Jupiter je udaljen 778,333,000 km od Sunca, ima prečnik 142.984 km i masu 1.900 e27 kg. Četvrto je najsjajnije nebesko telo u Sunčevom sistemu posmatrano sa Zemlje, nakon Sunca, Meseca i Venera. Jupiter ima 2.5 puta veću masu od ukupne mase ostalih sedam planeta u Sunčevom sistemu."

U datom tekstu pojavljuje se veći broj entiteta. U zavisnosti od potrebe za informacijom, različiti procesi ekstrakcije bi mogli da se fokusiraju na sve ili samo na neke od ovih entiteta. Ukoliko je potrebno izvući podatke o nebeskim telima, tada pojavljivanje entiteta „Rimljani“ ili „Grci“ ne bi bilo relevantno. Međutim, ukoliko se radi istraživanje o rečima latinskog ili grčkog porekla, ove informacije bi bile od značaja. Takođe, u procesima ekstrakcije u kojima bi bilo relevantno pojavljivanje božanstva „Zevs“ vrlo verovatno bi i pojavljivanje

božanstva „Jupiter“ bilo značajno, a tada bi bilo neophodno razrešiti da li se entitet „Jupiter“ koji se pojavljuje u tekstu odnosi na božanstvo ili nebesko telo.

Dalje, u zavisnosti od zahteva za informacijom, neki od entiteta koji se pojavljuju bi bili tumačeni kao atributi nekih drugih entiteta. Na primer, udaljenost od Sunca za entitet „Jupiter“ data je sekvencom „778.333.000 km“. Ova sekvence bi u jednim slučajevima bila smatrana za atribut, a u nekim drugim za entitet sam po sebi.

♦

Kao što je prikazano u prethodnom primeru, tumačenje pojmova *entitet*, *odnos između entiteta* i *atribut* jako zavisi od konkretnog procesa ekstrakcije informacije, tj, od potrebe korisnika i njegovog zahteva za određenom informacijom. Zbog toga svaki proces ekstrakcije informacije započinje jasnim i preciznim definisanjem tipova entiteta koji se traže, kao i specifikacijom da li i koju vrstu odnosa između entiteta ili atributa je potrebno ekstrahovati.

2.1.4. Nestrukturiran izvor podataka

Tekstovi u kojima se informacije nalaze mogu biti u različitim oblicima. U (Sarawagi, 2008) tipovi nestrukturiranih izvora podataka klasifikovani su u odnosu na dva aspekta: osnovnu jedinicu nad kojojm se izvodi ekstrakcija (nivo granularnosti) i heterogenost stila i formata u okviru izvora.

Granularnost ekstrakcije je određena veličinom delova teksta u kojima se informacija nalazi. Najniži nivo granularnosti imaju izvori podataka u kojima se informacija izdvaja iz malih, nestrukturiranih slogova teksta kao što su adrese, citati ili oglasi (Agichtein i Ganti, 2004; Borkar i sar., 2001; Michelson i Knoblock, 2008; Peng i McCallum, 2004; Soderland, 1999). U slučaju nestrukturiranih slogova, informacije mogu biti tretirane kao skup strukturiranih polja međusobno povezanih, sa eventualno izmenjenim rasporedom. U tom slučaju svaki token je deo nekog polja, pa je dovoljno izvršiti segmentaciju teksta na entitete.

Nešto složeniji nivo granularnosti imaju izvori podataka kod kojih se informacija nalazi u rečenicama (Borthwick i sar., 1998; Chincor, 1998; Doddington i sar., 2004; Grishman i Sundheim, 1996; Sang i Meulder, 2003). Kod njih postoji veliki broj reči koji nisu deo informacije koja je od interesa i ne ulaze ni u jedan entitet.

Najviši nivo granularnosti imaju izvori podataka kod kojih se informacija proteže kroz nekoliko rečenica, paragrafa, a ponekad i nekoliko dokumenata. Najpopularniji je primer ekstrakcije događaja iz novinskih članaka (Chincor, 1998; Grishman i Sundheim, 1996), ekstrakcija naslova, lokacije i vremena nekog predavanja iz objave (Seymore i sar. 1999) i ekstrakcija naslova rada i citata iz naučnih publikacija (Peng i McCallum, 2004). Ekstrakcija informacija iz dužih segmenata teksta i njena uspešnost najviše zavisi od uspešnosti izdvajanja relevantnih delova teksta od irelevantnih.

Heterogenost izvora podataka određena je razlikom u stilu jezika kojim je pisan tekst i formatu teksta. U tom smislu, sa stanovišta ekstrakcije informacija razlikujemo nekoliko tipova dokumenata:

- mašinski generisani dokumenti - dokumenti koji su generisani od strane računara, uglavnom veoma šablonizovani. Primer su dinamički generisane HTML strane, na osnovu podataka iz neke baze. Ekstrakcija informacija iz ovakvih izvora obavlja se tzv. omotačima (engl. *wrapper*) koji su opisani u poglavlju 3.3.

- delimično strukturirani dokumenti jednog domena – dokumenti koji imaju određenih sličnosti i u strukturi i u stilu (jeziku) kojim su pisani. Kod ovakvih dokumenata obično postoji nekakav neformalni stil kojim su pisani, kao što je slučaj sa novinskim člancima, tehničkom dokumentacijom i sl., pa je moguće razviti relativno uspešne modele ekstrakcije.

- heterogene kolekcije dokumenata – dokumenti među kojima postoje izrazito velike razlike i u formatu i u stilu, koje nisu predvidive. U poslednje vreme postoje pokušaji izdvajanja informacija i iz ovakvih tekstva (Banko i sar., 2007; Cafarella i sar., 2005; Etzioni i sar., 2004; Shinyama i Sekine, 2006)

2.1.5. Strukturiranje informacija

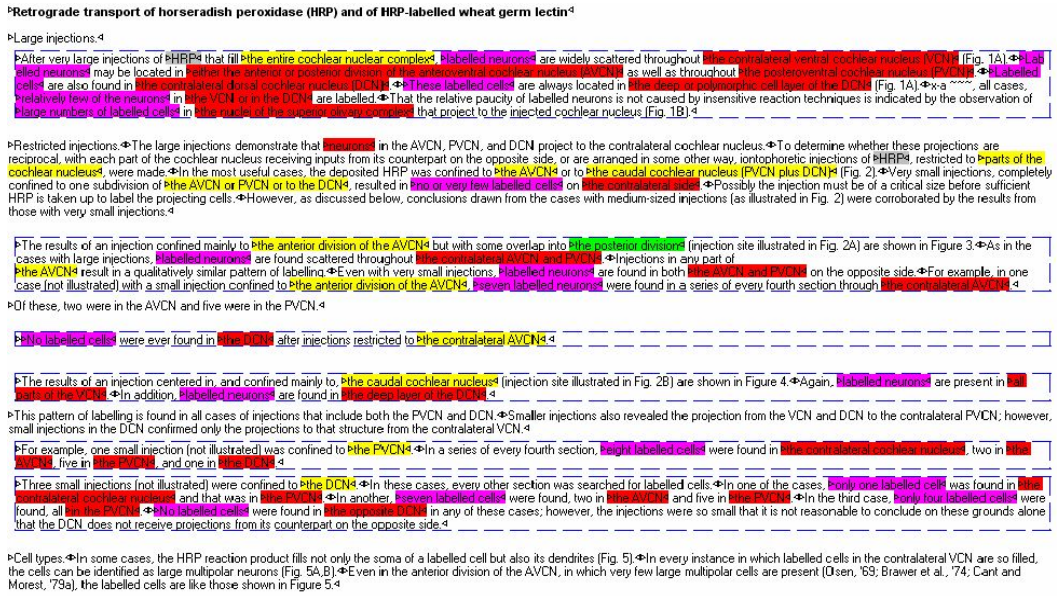
S obzirom da je krajnji cilj procesa ekstrakcije informacija omogućavanje dalje obrade podataka, nakon ekstrahovanja potrebno je izvršiti strukturiranje dobijenih informacija. **Strukturiranje informacija** je proces klasifikovanja informacija u semantičke klase, tj. dodeljivanje značenja dobijenim informacijama.

Strukturiranje informacije se najčešće vrši ili obeležavanjem teksta, tj. umetanjem oznaka (najčešće XML oznaka) koje opisuju značenje delova teksta koji je prepoznat kao informacija, ili kreiranjem i popunjavanjem relacionih baza podataka (jedan takav proces je detaljno prikazan u poglavlju 4), ili na neki drugi način. Sam format izlaznih podataka nije od suštinske važnosti za proces ekstrakcije informacije, jer je moguće naknadno konvertovati podatke iz jednog formata u drugi. Ono što jeste važno i što određuje proces ekstrakcije jesu semantičke klase podataka, tj. njihova specifikacija kao i pravila odlučivanja koja informacija pripada kojoj semantičkoj klasi.

Interesantan je i pristup problemu ekstrakcije informacije opisan u (Feng i sar. 2007). u kome se problem identifikacije informacije iz teksta, u smislu definicije usvojene u ovoj disertaciji, naziva **horizontalni problem** (HP), a problem dodeljivanja značenja, tj. klasifikovanja u semantičke klase, naziva **vertikalni problem** (VP). Naime, ukoliko se posmatraju rečenice jednog teksta od početka ka kraju, problem ekstrakcije je sveden na to da se prvo u rečenicama lociraju reči ili fraze koje opisuju neku informaciju, a zatim da se te informacije grupišu u grupe koje se odnose na jedan entitet (ili slog u bazi podataka).

Primer ovakvog teksta je prikazan na slici 1. Tekst je preuzet iz studije (Feng i sar., 2007) u kojoj su izdvajani podaci o bojenju neurona i delova mozga nakon ubrizgavanja različitih hemikalija. Podaci su izdvajani iz naučne literature u okviru koje su bili opisivani različiti eksperimenti. Cilj je bio da se u tekstu odrede različiti atributi jednog eksperimenta (na slici su obeleženi različitim bojama), a zatim i da se grupišu i dodele određenom eksperimentu na koji se odnose (na slici su delovi teksta koji se odnose na pojedine eksperimente uokvireni isprekidanom linijom). Na primer, treći uokvireni paragraf predstavlja

jedan eksperiment i sadrži tri atributa: „*no labeled cells*”, “*the DCN*”, i “*the contralateral AVCN*”.



Slika 1. Tekst sa obeleženim i grupisanim informacijama

Razdvajanjem horizontalnog i vertikalnog problema i njihovim pojedinačnim rešavanjem u (Feng i sar., 2007), autori su uspeli da postignu mnogo bolje rezultate, nego kada se procesu ekstrakcije informacije pristupi istovremenim razrešenjem ova dva problema.

2.2. Jezik i gramatike

2.2.1. Osnovni pojmovi

Azbuka ili **alfabet** Σ je konačan, neprazan skup simbola. **Reč** ili **niska** nad azbukom Σ je svaki konačan niz simbola x iz Σ, tj.

$$x = (a_1, a_2, \ldots, a_n), n \geq 0, a_i \in \Sigma$$

Broj n naziva se dužina reči x i obeležava se sa $|x|$. **Prazna niska** ili **prazna reč** je reč za koju je $n=0$ i obeležava se slovom ε. Važi da je $|\varepsilon|=0$.

Neka su $x=(a_1,a_2,\ldots a_n)$ i $y=(b_1,b_2,\ldots b_m)$ dve reči nad azbukom Σ. Proizvod **dopisivanja** ili **konkatenacije** reči x i y je reč

$$xy = (a_1, a_2, ..., a_n, b_1, b_2, ..., b_m)$$

Kako je svaka neprazna reč proizvod dopisivanja reči dužine 1, to se reči dužine 1 i elementi azbuke Σ mogu poistovetiti. Zbog toga ćemo ubuduće umesto oznake niske $x=(a_1,a_2,...a_n)$ koristiti oznaku $x=a_1a_2...a_n$.

Klinijevo zatvorenje azbuke je skup svih niski nad azbukom Σ i obeležava se sa Σ^*. Na primer, ukoliko je $\Sigma=(a,b,c)$, tada je $\Sigma^*=(\varepsilon,a,b,c,aa,ab,ac,ba,..,aaa,aab,..)$.

Neka su v i u reči iz Σ^*. Ako postoje reči $v_1 \in \Sigma^*$ i $v_2 \in \Sigma^*$ takve da je:

- $v=v_1uv_2$, u je **faktor** reči v
- $v=uv_2$, u je **levi faktor** ili **prefiks** reči v
- $v=v_1u$, u je **desni faktor** ili **sufiks** reči v
- $v=v_1uv_2$, $v_1 \neq \varepsilon$, $v_2 \neq \varepsilon$, u je **pravi factor** ili **infiks** reči v

Podreč reči $v \in \Sigma^*$ je svaki podniz slova koja čine reč v.

Na skupu Σ^* definiše se više relacija poretka:

- **prefiksni** poredak je parcijalni poredak definisan sa:

$$x \prec y \quad \textit{ako i samo ako je x levi faktor niske y}$$

- **leksikografski** poredak je potpuni poredak koji je uklopljen u prefiksni poredak; definiše se ako je nad Σ definisana relacija potpunog poretka $\leq$ i tada je

$$x \leq y \Leftrightarrow \begin{cases} x \prec y \\ \exists u, v \in \Sigma : x = x_1 u x_2, y = x_1 v y_2, u \leq v \end{cases}$$

- **hijerarhijski** poredak je potpuni poredak u kome su reči uređene prvo po dužini, a zatim reči iste dužine u leksikografskom poretku.

2.2.2. Jezik i operacije nad jezicima

Jezik (**formalni jezik**) L nad azbukom Σ je bilo koji podskup skupa Σ^*, tj. $L \subseteq \Sigma^*$. Ako niska $x \in L$, onda x predstavlja reč jezika L.

Primer.

1. $L_1 = \{a^n \mid n > 0\}$. Jezik L_1 se sastoji od reči u kojima se simbol a pojavljuje proizvoljan broj puta. Ovaj jezik sadrži niske *a, aa, aaa,...* Ukoliko simbol interpretiramo kao cifru dekadnog brojnog sistema, jezikom L_1 bile bi opisane celobrojne neoznačene konstante.

2. $L_2 = \{a^n b^n \mid n > 0\}$. Jezik L_2 sadrži sve niske u kojima se pojavljuje prvo simbol a nekoliko puta, a zatim simbol b isti broj puta. Ako simbole a i b interpertiramo redom kao otvorenu i zatvorenu zagradu, jezik L_2 opisuje jezik umetnutih parova zagrada.

3. Prilikom opisivanja prirodnih jezika formalnim definicijama, simbol azbuke Σ u stvari predstavlja reč prirodnog jezika, a reči nad azbukom odgovaraju rečenicama tog prirodnog jezika.

♦

Jezici mogu biti konačni ili beskonačni. Za precizan opis konačnog jezika dovoljno je navesti sve niske koje u tom jeziku predstavljaju rečenice. Primer konačnih jezika bili bi jezici ključnih reči programskih jezika. Za opis beskonačnih jezika koriste se algoritmi koji utvrđuju da li neka niska pripada datom jeziku ili ne. Najopštiju klasu takvih algoritama opisuju Tjuringove mašine.

S obzirom da su jezici u stvari skupovi niski, na njih je moguće primeniti skupovne operacije: uniju, presek i razliku. Dodatno, nad jezicima je moguće definisati i druge operacije.

Proizvod jezika L_1 i L_2 nad azbukom Σ, u oznaci L_1L_2 je jezik:

$$L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}.$$

n-ti stepen jezika L je jezik:

$$L_0=\{\varepsilon\}$$

$$L_1=L$$

$$L_n=LL_{n-1}$$

Iteracija ili **Klinijevo zatvorenje** jezika L, u oznaci L^*, je jezik:

$$L^* = \bigcup_{n \geq 0} L^n$$

Pozitivno zatvorenje jezika L, u oznaci $L+$, je jezik:

$$L^+ = \bigcup_{n > 0} L^n = L^* - \{\varepsilon\}$$

Od interesa će biti i preslikavanja reči nad nekom azbukom Σ u reči neke druge azbuke Δ, i to ona preslikavanja koja su saglasna sa operacijom dopisivanja.

Neka su Σ i Δ azbuke. Preslikavanje h iz Σ^* u Δ^* je **homomorfizam** iz Σ^* u Δ^* ako za svako $x,y \in \Sigma^*$ važi:

$$h(xy)=h(x)h(y)$$

2.2.3. Gramatika

U okviru ovog poglavlja data je definicija jezika kao podskupa skupa svih reči nad nekom azbukom. Neke jezike možemo da zadamo (opišemo) jednostavno nabrajanjem njihovih elemenata. Druge jezike možemo zadati pomoću osobina reči koje im pripadaju. Takav slučaj je bio sa jezikom L_1 i L_2 iz prethodnog primera. Međutim, postoje jezici koje nije moguće zapisati ili zadati na neki od ovih načina, i za koje je potreban neki kompleksniji mehanizam zadavanja.

Zbog toga se definišu tzv. gramatike ili generativne gramatike, čiju suštinu predstavljaju pravila izvođenja koja jasno govore o tome na koji način, polazeći od početnih simbola dobijamo nove reči jezika. Unutar gramatika, kao pomoćni simboli ili promenljive, koriste se neterminalni simboli. Formalna definicija gramatike data je sledećom definicijom.

Gramatika G je uređena četvorka (Σ, N, P, S) gde je:

- Σ **završna** azbuka, čiji se elementi nazivaju **završni** ili **terminalni** simboli

- N **nezavršna** azbuka, čiji se elementi nazivaju **nezavršni** (**pomoćni** ili **neterminalni** simboli), takva da je $\sum \cap N=\varnothing$,

- $S \in N$ koji se naziva **početni** (**rečenični** ili **startni**) simbol

- $P \subseteq (\sum \cup N)^*N(\sum \cup N)^* \times (\sum \cup N)^*$ - konačni skup **pravila izvođenja** (**produkcija**)

U pravilu $(\alpha,\beta) \in P$, niska α se naziva **leva**, a niska β **desna strana pravila**. Niska α mora sadržati barem jedan element skupa N.

Ako je pravilo $(\alpha,\beta) \in P$ onda se ono zapisuje $\alpha \rightarrow_G \beta$ i čita α postaje (zamenjuje se sa) β u gramatici G.

Ako je u pravilu $\alpha \rightarrow \beta$, $\beta = \varepsilon$ onda se takvo pravilo naziva ε**–pravilo**.

Ako postoji više pravila sa istom levom stranom $\alpha \rightarrow \beta_1$, ... $\alpha \rightarrow \beta_n$, onda se takva pravila pišu u skraćenom obliku $\alpha \rightarrow \beta_1 |.....|\beta_n$.

Praveći paralelu sa prirodnim jezicima, u slučaju prirodnog jezika L neterminalni simboli su oznake gramatičkih kategorija, terminalni simboli su reči jezika L, a produkcije su gramatička pravila pomoću kojih iz početnog simbola S izvodimo rečenice datog jezika.

Primer.

Označimo sa *<recenica>* kategoriju svih rečenica srpskog jezika, sa *<imenicki izraz>* kategoriju imenickih izraza, a sa *<neprelazni glagol>* kategoriju neprelaznih glagola. Posmatrajmo rečenice:

Kiša pada.

Trava raste.

Ona spava.

Sve navedene rečenice imaju istu strukturu i mogu se dobiti sledećim transformacijama:

1. <recenica>→<imenicki izraz><neprelazni glagol>

2. <imenicki izraz>→ kiša|trava|ona

3. <*neprelazni glagol*> $\rightarrow$ *pada*|*raste*|*spava*

Na primer, ukoliko pravila izvođenja primenimo na rečenicu «Kiša pada», dobićemo niz koraka (zamena):

<*recenica*> $\Rightarrow$ <*imenicki izraz*><*neprelazni glagol*> $\Rightarrow$ *kiša*<*neprelazni glagol*> $\Rightarrow$ *kiša pada*

♦

Niz koraka u kojima se primenjuju pravila izvođenja, prikazan u prethodnom primeru, naziva se izvođenje niske «Kiša pada» iz početnog simbola rečenica. Izvođenjem se uspostavlja relacija između početnog simbola i niske leksičkih jedinica. Ovim postupkom se utvrđuje da li neka niska pripada datom jeziku ili ne. Niska leksičkih jedinica pripada nekom jeziku ukoliko postoji izvođenje iz početnog simbola jezika. Ovde treba primetiti da je primenom pravila izvođenja iz prethodnog primera moguće dobiti i rečenice kao što su: «Kiša raste», «Trava spava» i sl. Iako može da se desi da značenje takvih rečenica bude nelogično, one se ne mogu smatrati gramatički neispravnim. Formalno opisivanje jezika se ograničava na sintaksu nekog jezika, a ne i njegovu semantiku.

Prethodno opisani postupak izvođenja formalno je definisan na sledeći način. Ako u gramatici $G= (\sum, N, P, S)$ pravilo $\alpha \rightarrow \beta \in P$ onda za bilo koje dve niske $\gamma, \delta \in (\sum \cup N)^*$ važi:

$$\gamma\alpha\delta \Rightarrow_G \gamma\beta\delta$$

Kaže se da se iz niske $\gamma\alpha\delta$ u gramatici G **neposredno izvodi** niska $\gamma\beta\delta$. Slovo G u oznaci operacije neposrednog izvođenja se često izostavlja, kada je jasno o kojoj gramatici je reč.

Rečenična (ili **gramatička**) **forma** u gramatici G opisuje se rekurzivno sledećim pravilima:

1. Početni simbol S gramatike G je rečenična forma;

2. Ako je $\gamma\alpha\delta$ rečenična forma i $\alpha \rightarrow \beta \in P$ onda je i $\gamma\beta\delta$ rečenična forma.

Rečenica u gramatici G je rečenična forma koja pripada $\sum^*$.

Izvođenje u gramatici G je niz rečeničnih formi $\alpha_0, \alpha_1, \ldots \alpha_n$ takvih da je $\alpha_0 = S$ i $\alpha_{i-1} \Rightarrow \alpha_i$, za svako $i \in [1,n] \subset \mathbb{N}$.

Koristićemo sledeće oznake:

- n-ti stepen relacije $\Rightarrow$ u oznaci $\overset{n}{\Rightarrow}$. Ako $\alpha \overset{n}{\Rightarrow} \beta$, tada se kaže da se β izvodi u n koraka iz α.

- tranzitivno zatvorenje relacije $\Rightarrow$ u oznaci $\overset{+}{\Rightarrow}$. Ako $\alpha \overset{+}{\Rightarrow} \beta$, tada se kaže da se β izvodi u pozitivnom broju koraka iz α.

- tranzitivno i refleksivno zatvorenje relacije $\Rightarrow$ u oznaci $\overset{*}{\Rightarrow}$. Ako $\alpha \overset{*}{\Rightarrow} \beta$, tada se kaže da se β izvodi iz α.

Uvođenjem prethodno definisanih oznaka, rečeničnu fomu u gramatici $G = (\Sigma, N, P, S)$ moguće je definisati kao svaku nisku $\alpha \in (\Sigma \cup N)^*$ takva da važi

$$S \overset{*}{\Rightarrow} \alpha.$$

Jezik opisan gramatikom $G = (\Sigma, N, P, S)$ u oznaci $L(G)$, je

$$L(G) = \left\{ x \mid x \in \Sigma^* \wedge S \overset{*}{\Rightarrow} x \right\}.$$

Kaže se da G **generiše** ili **proizvodi** jezik $L(G)$. Za nisku $x \in L(G)$ kaže se da je **generisana** (**proizvedena**) u gramatici G.

Na osnovu prethodne definicije jasno je da svakoj gramatici odgovara jedan jezik. Međutim, nije teško videti da isti jezik može da bude generisan različitim gramatikama. Drugim rečima, postoje gramatike koje su različite, a generišu isti jezik. Za gramatike $G_1 = (\Sigma, N_1, P_1, S_1)$ i $G_2 = (\Sigma, N_2, P_2, S_2)$ kaže se da su **ekvivalentne** ako je $L(G_1) = L(G_2)$.

2.2.4. Hijerarhija jezika i apstraktni automati

Iako je moguće formalne jezike porediti međusobno na različite načine, kada se u teoriji formalnih jezika govori o hijerarhiji među jezicima (tj. gramatikama) obično se misli na hijerarhiju koju je uspostavio poznati lingvista

Noam Chomsky, a koja je po njemu i dobila ime – hijerarhija Čomskog. Ova hijerarhija je uspostavljena na osnovu strukture gramatičkih pravila.

Hijerarhija Čomskog. Gramatika $G= (\sum, N, P, S)$ je:

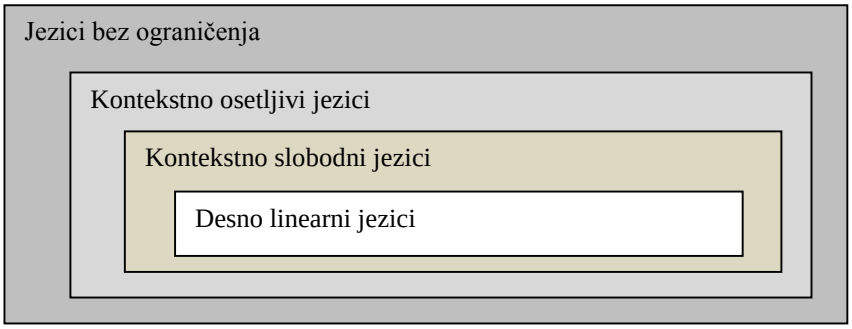
1. *desno-linearna* ili *tipa 3* ako su sva njena pravila oblika $A\rightarrow aB$ ili $A\rightarrow a$, gde su $A, B\in N$, $a\in\sum$. Jezik opisan desno-linearnom gramatikom je *desno-linearni jezik.*

2. *kontekstno slobodna* (skraćeno, *KS*) ili *tipa 2* ako su sva njena pravila oblika $A\rightarrow \alpha$, gde je $A\in N$, $\alpha\in(\sum\cup N)^*$. Jezik opisan kontekstno slobodnom gramatikom je *kontekstno slobodan jezik.*

3. *kontekstno osetljiva* (skraćeno, *KO*) ili *tipa 1* ako su sva njena pravila oblika $\alpha\rightarrow\beta$ *i* $|\alpha|\leq|\beta|$, $\alpha,\beta\in(\sum\cup N)^*$ i α sadrži barem jedan element iz *N*. Jezik opisan kontekstno osetljivom gramatikom je *kontekstno osetljiv jezik.*

4. *bez ograničenja, opšta* ili *tipa 0* ako nema ograničenja na strukturu pravila. Jezik opisan takvom gramatikom je *jezik bez ograničenja.*

Svakoj klasi gramatika odgovara po jedna klasa apstraktnih automata. Apstraktni automati su uređaji koji su u stanju da za zadatu nisku *x* i jezik *L* utvrde da li ta niska pripada jeziku ili ne. Detaljna razmatranja o odnosu između tipova gramatika i jezika koji su njima generisani, kao i odgovarajućim automatima nalaze se u mnogim udžbenicima teorije formalnih jezika i programskih jezika, npr. (Jurafsky i Martin, 2008), (Vitas, 2006), itd. Na slici 2 prikazan je odnos različitih klasa jezika, dok je u tabeli 1 prikazan pregled jezika, odgovarajućih gramatika i automata.



Slika 2. Dijagram koji prikazuje odnos različitih tipova jezika

Tabela 1. Pregled jezika i gramatika i odgovarajućih uređaja za prepoznavanje

Tip	**Klasa jezika**	**Uređaj za prepoznavanje**
0	Jezici bez ograničenja	Tjuringova mašina
1	Kontekstno osetljivi jezici	Linearno ograničeni automati
2	Kontekstno slobodni jezici	Nedeterministički potisni automati
3	Desno linearni jezici	Konačni automati

2.3. Regularni izrazi i jezici

Važnu potklasu jezika čine tzv. regularni jezici. Ova klasa jezika opisuje se posebnom notacijom – regularnim izrazima, široko rasporostranjenim u računarstvu. Međutim, pokazuje se da su regularni jezici u stvari tačno oni jezici koje prihvataju konačni automati, opisani u poglavlju 2.4.

Regularni ili **racionalni izrazi** nad azbukom $\sum$ se opisuju rekurzivno na sledeći način:

1. Prazan skup je regularan izraz koji se predstavlja simbolom $\emptyset$.
2. Regularni izraz ε predstavlja jezik $\{\varepsilon\}$ u oznaci $L(\varepsilon)$.
3. Ako je $a \in \sum$ onda regularni izraz a predstavlja jezik $\{a\}$ i označava se sa $L(a)$.
4. Ako su p i q regularni izrazi jezika $L(p)$ i $L(q)$, onda je:

   - $(p+q)$ regularni izraz koji predstavlja jezik $L(p) \cup L(q)$;

   - (pq) regularni izraz koji predstavlja jezika $L(p)L(q)$;

   - $(p)^*$ regularni izraz koji predstavlja jezik $(L(p))^*$;

   - (p) regularni izraz koji predstavlja jezik $L(p)$.

Obično se u zapisu regularnih izraza koristi oznaka $p|q$ umesto $p+q$, jer se simbolom + označava i pozitivna iteracija. Takođe, zagrade u gornjim izrazima je moguće izostaviti.

Jezik je **regularan** ako se može predstaviti regularnim izrazom. Skup svih regularnih jezika nad azbukom Σ obeležava se sa $Reg(\Sigma^*)$.

Primer.

Regularan izraz $(0|1)^*011$ predstavlja jezik kome pripadaju sve reči nad azbukom $\{0,1\}$ koje se završavaju sa 011.

♦

Regularni izrazi definisani na gore pomenuti način u velikom broju situacija postaju izuzetno nepregledni. Zbog toga se uvode tzv. **regularne definicije**, koje omogućavaju da se pojedinim regularnim izrazima dodeli ime. Na taj način omogućen je i modularni pristup opisivanju regularnog izraza. Regularne definicije se zapisuju u obliku:

$$d_1 \rightarrow r_1$$
$$d_2 \rightarrow r_2$$
$$\ldots\ldots$$
$$d_n \rightarrow r_n$$

gde je svako d_i niska nad azbukom koja je disjunktna sa azbukom Σ, različita od $d_1, d_2, \ldots\ldots d_{i-1}$, a svako r_i regularni izraz nad azbukom $\Sigma \cup \{ d_1, d_2, \ldots\ldots d_{i-1}\}$.

Primer.

Identifikatori u nekim programskim jezicima mogu biti niske alfanumeričkih karaktera, od kojih je prvi karakter obavezno slovo. Ovakvu definiciju identifikatora možemo da izrazimo regularnim izrazom:

$$\begin{aligned} \text{slovo} &\rightarrow A \mid B \mid \ldots \mid Z \mid a \mid b \mid \ldots. \mid z \\ \text{cifra} &\rightarrow 0 \mid 1 \mid \ldots.. \mid 9 \\ \text{identifikator} &\rightarrow \text{slovo}(\text{slovo} \mid \text{cifra})^* \end{aligned}$$

♦

Radi dodatnog pojednostavljenja zapisa regularnih izraza, uvode se sledeće konvencije:

- neka je *r* regularni izraz koji opisuje jezik *L(r)*, tada je $(r)^+$ regularni izraz koji opisuje jezik $(L(r))(L(r))^*$, a *(r)?* regularni izraz koji opisuje jezik $L(r) \cup \{ \varepsilon \}$.
- ako su $c_1, c_2, ..., c_n$ karakteri, tada se regularni izraz $c_1+c_2+ ...+c_n$ može obeležiti sa $[c_1c_2 ...c_n]$
- izraz $[c_1\text{-}c_n]$ označava sekvenciju svih karaktera takvih da je $c_1 \leq c \leq c_n$

Primer.

Imajući u vidu pravila pojednostavljenog zapisivanja, definicija identifikatora iz prethodnog primera bila bi:

slovo → [*A - Z a- z]*
cifra → [0 -9]
identifikator → slovo(slovo | cifra)*

♦

U nekim implementacijama regularnih izraza moguće je koristiti i komplement regularnog izraza r^C kome odgovaraju sve niske koje ne odgovaraju izrazu *r*. Komplement se često označava sa [^]. Tako bi regularnom izrazu *[^abc]* odgovarao bilo koji karakter različit od *a*, *b* i *c*.

Regularni izrazi se koriste u računarstvu najviše za zadavanje upita prilikom pretrage teksta, ali i za opisivanje niski karaktera u bilo kom slučaju gde je to potrebno. Oni omogućavaju relativno lak i brz način zadavanja pravila koja neka niska treba da zadovolji, pa su zbog te svoje osobine široko raspostranjeni u računarstvu. Koriste se u sklopu mnogih editora teksta, programskih alata i programskih jezika. Neki od programskih jezika, kao što su Perl, Ruby, AWK i drugi, su ugradili regularne izraze u samu sintaksu jeziku. Drugi programski jezici, kao što su jezici .NET platforme, Java ili Python omogućavaju implementaciju regularnih jezika preko biblioteka klasa.

U programskom jeziku Java kroz vreme su se pojavljivali različiti paketi klasa za rad sa regularnim izrazima. Počevši od verzije 1.4, Java je uključila paket *java.util.regex* u kome se nalaze klase za rad sa regularnim izrazima. Posebno je interesantan paket *com.stevesoft.pat* (Brandt 2000; Nelson 1997), čije klase za rad sa regularnim izrazima imaju dodatne opcije i funkcionalnost, koje pružaju veće mogućnosti u obradi teksta.

Primer.

Sledi nekoliko primera programskog koda napisanog u programskom jeziku Java, koristeći klasu *Regex* paketa *com.stevesoft.pat.*

```
// kreira se objekat klase Regex
Regex r = new Regex("shells");
// pretražuje se određeni tekst zadatim izrazom r
r.search("She sells sea shells by the sea shore.");
System.out.println(""+r.didMatch());
// štampa "true" -- r.didMatch() je funkcija koja
// vraća boolean vrednost u zavisnosti od toga
// da li je traženi izraz pronađen u tekstu
System.out.println(r.stringMatched());
// štampa "shells" – metod stringMatched() vraća
// pronađenu nisku koja odgovara regularnom izrazu
System.out.println(r.left());
// štampa "She sells sea " – levi kontekst pronađene niske
System.out.println(r.right());
// štampa " by the sea shore." – desni kontekst
**********************************************************
// oznaka (?i) ignoriše mala i velika slova
r = new Regex("(?i)shells");
r.search("SHE SELLS SEA SHELLS BY THE SEA SHORE.");
System.out.println(""+r.didMatch());
// štampa "true"
System.out.println(r.stringMatched());
// štampa "SHELLS"
**********************************************************
Regex r = new Regex("[0123456789][0123456789]");
r.search("How old are you? I'm 35.");
```

```
System.out.println(r.stringMatched());
// štampa "35"
r.search("How old are you? I'm only 8.");
System.out.println(r.stringMatched());
// štampa "null" jer nije naišao na odgovarajuću nisku
r.search("When were you born? In 1963");
System.out.println(r.stringMatched());
// štampa "19"
**********************************************************
//prepoznaje 1, 2, 3, ili 4 cifre
Regex r = new Regex("[0123456789]{1,4}");
r.search("How old are you? I'm only 8.");
System.out.println(r.stringMatched());
// štampa "8"
r.search("How old are you? I'm 35.");
System.out.println(r.stringMatched());
// štampa "35"
r.search("When were you born? In 1963.");
System.out.println(r.stringMatched());
// štampa "1963"
```

♦

Sem dodatnih metoda, klasa Regex omogućava i posebno izdvajanje delova prepoznate niske. Takvi delovi niske se unutar regularnog izraza stavljaju u zagrade. Nakon prepoznavanja niske ovim delovima se dodeljuju indeksi, redom počevši od jedan. Na taj način, ovom klasom može da se simulira funkcionisanje konačnih tansduktora, opisanih kasnije u poglavlju 2.5.

Primer.

```
Regex r = new Regex("[abc]([def])");
r.search("==> be <==");
System.out.println(""+r.didMatch());
// štampa "true"
System.out.println(r.stringMatched());
// štampa celu prepoznatu nisku - "be"
System.out.println(r.stringMatched(1));
// štampa "e" – odgovara sadržaju prve (i jedine) zagrade
**********************************************************
```

```
Regex r = new Regex("([abc])([def])");
r.search("==> be <==");
System.out.println(r.stringMatched(1));
// štampa "b" – odgovara sadržaju prve zagrade
System.out.println(r.stringMatched(2));
// štampa "e" - odgovara sadržaju druge zagrade
************************************************************
// moguće je ugnježdavanje zagrada
Regex r = new Regex("(ab(cd))ef");
r.search("==>abcdef<==");
System.out.println(r.stringMatched());
// štampa "abcdef"
System.out.println(r.stringMatched(1));
// štampa "abcd"
System.out.println(r.stringMatched(2));
// štampa "cd"
```

♦

2.4. Konačni automati

Konačni (**nedeterministički**) **automat** nad konačnom azbukom Σ se sastoji od:

- konačnog skupa Q – **skupa stanja**
- skupa $I \subset Q$ – **skupa početnih, inicijalnih stanja**
- skupa $F \subset Q$ – **skupa završnih, finalnih stanja**
- skupa $\Delta \subset Q \times \Sigma \times Q$ – **skupa relacija prelaza**

Konačni automat se zapisuje kao uređena petorka $A = (\Sigma, Q, I, F, \Delta)$. Element relacije prelaza Δ naziva se **luk**. Ako je luk $l = (p, a, q) \in \Delta$, onda je slovo $a \in \Sigma$ **etiketa** tog luka.

Pod **izračunavanjem** c dužine n u automatu A podrazumeva se niz lukova $l_1, ...l_n$, gde $l_i = (p_i, a_i, q_i) \in \Delta$, tako da je $q_i = p_{i+1}$ za $i \in [1, n-1] \subset \mathrm{N}$. Pod ***etiketom*** **izračunavanja** c, u oznaci $\|c\|$ se podrazumeva niska $a_1...a_n$ sastavljena od etiketa lukova izračunavanja c. Ako je niska $w = \|c\|$ etiketa izračunavanja c, onda se to zapisuje na sledeći način:

$$c : p_1 \xrightarrow{w} q_n \text{ ili } c : p_1 \longrightarrow q_n .$$

Za izračunavanje $c : p \longrightarrow q$ se kaže da je **uspešno** ako važi da je $p \in I$ i $q \in F$. Za reč ω se kaže da je **prepoznata** (ili **prihvaćena**) automatom A ako je ta reč etiketa nekog uspešnog izračunavanja. **Jezik prepoznat (prihvaćen) automatom *A*** je skup svih reči prepoznatih automatom A:

$$L(A) = \left\{ \omega \in \Sigma^* \middle| \exists c : i \to f, i \in I, f \in F, \omega = \|c\| \right\}.$$

Pojam izračunavanja se može posmatrati i na drugi način, kao proširenje relacije prelaza Δ sa slova na reči. Tako **proširena relacija prelaza**, u oznaci Δ^*, opisana je na sledeći način:

$$\Delta^* \subset Q \times \Sigma^* \times Q$$

uz uslove:

- za svako $q \in Q$, $(q, \varepsilon, q) \in \Delta^*$, gde je ε prazna reč;
- ako je $\omega = a_1 a_2 ... a_n$ $(a_i \in \Sigma, n \geq 1)$ i ako postoji $n+1$ stanje $q_0, q_1 ... q_n$ takvo da za svako $i \in N: 1 \leq i \leq n$, važi da je luk $(q_{i-1}, a_i, q_i) \in \Delta$, tada je $(q_0, \omega, q_n) \in \Delta^*$, a ω je **etiketa** puta u automatu koja povezuje stanje q_0 sa stanjem q_n.

  Jezik prepoznat konačnim automatom A je tada

$$L(A) = \left\{ \omega \in \Sigma^* \middle| \exists i \in I, \exists f \in F, (i, \omega, f) \in \Delta^* \right\}.$$

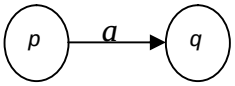
Podskup $X \subseteq \Sigma^*$ je **prepoznatljiv** ako postoji automat A nad azbukom Σ takav da je $X = L(A)$. Familija svih prepoznatljivih podskupova u Σ^* obeležava se sa $Prep(\Sigma^*)$.

Važi sledeća teorema, čiji se dokaz može naći u (Vitas, 2006): Svaki konačan podskup $X \subseteq \Sigma^*$ je prepoznatljiv jezik.

2.4.1. Graf prelaza automata

Konačni automati se najčešće predstavljaju tzv. **grafom prelaza**, iako mogu biti predstavljeni i na druge načine (na primer, matricom prelaza, (Vitas, 2006)).

Reprezentacija automata pomoću grafa se obično naziva **dijagram stanja**. U njoj su stanja automata predstavljena čvorovima grafa, a luk automata *(p, a, q)* je predstavljen lukom iz čvora *p* ka čvoru *q* koji je obeležen etiketom *a* (slika 3).



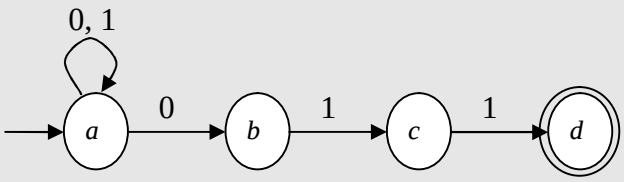
Slika 3. Prelazak iz stanja *p* u stanje *q* nakon što je pročitan ulazni simbol *a*

Izračunavanje je put u grafu, a obeležja lukova koji čine jedan put u grafu su slova etikete izračunavanja.

U grafičkom prikazu automata, početna i završna stanja (elementi skupova *I* i *F*) se obeležavaju na poseban način. Početno stanje je obeleženo strelicom koja pokazuje na njega, dok je završno stanje dvostruko zaokruženo.

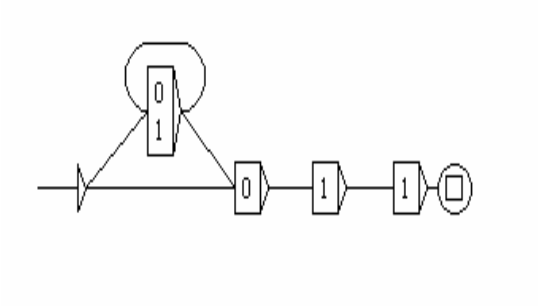
Primer.

Jezik svih reči nad azbukom {0,1} koje se završavaju sa 011, koji je u jednom od prethodnih primera bio opisan regularnim izrazoma $(0|1)^*011$, može biti opisan i sledećim konačnim automatom:



♦

U obradi prirodnih jezika često sama stanja nisu od značaja, već su jedino važne labele po kojima se vrše prelazi. Zbog toga se sreće i nešto drugačija grafička reprezentacija konačnih automata u različitim programskim paketim namenjenim za obradu teksta. Takav primer je softverski paket UNITEX (Paumier, 2011), koji između ostalog, omogućava korisniku da grafički kreira konačne automate. Primer automata koji prepoznaje jezik iz prethodnog primera, kreiran u UNITEX-u prikazan je na slici 4.



Slika 4. UNITEX graf za prepoznavanje jezika svih reči nad azbukom {0,1} koje se završavaju sa 011

2.4.2. Deterministički konačni automati (DKA)

Stanje $q \in Q$ automata $A = (\Sigma, Q, I, F, \Delta)$ je **dostupno** ako postoji izračunavanje $c : i \rightarrow q$, gde je $i \in I$. Stanje $q \in Q$ je **sudostupno** ako postoji izračunavanje $c : q \rightarrow f$, gde je $f \in F$. Ako su sva stanja jednog automata dostupna, kažemo da je automat **dostupan**, a ako su sva stanja automata dostupna i sudostupna, kažemo da je automat **skresan**.

Potkresivanje automata *A* jeste postupak određivanja automata *B* koji je skresan, pri čemu važi da je $L(A)=L(B)$. Automat *B* se sastoji samo od onih stanja automata *A* koja su i dostupna i sudostupna.

Konačni automat $A = (\Sigma, Q, I, F, \Delta)$ je **deterministički** ako skup početnih stanja *I* ima tačno jedan element i ako važi

$$(p, a, q), (p, a, r) \in \Delta \Rightarrow q=r.$$

Deterministički konačni automat se skraćeno obeležava sa DKA.

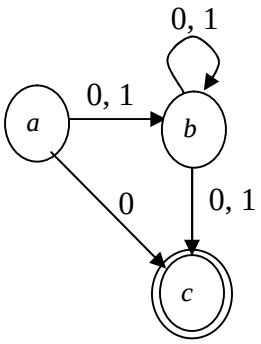
Imajući u vidu definiciju determinističkog konačnog automata, relacija prelaska se u ovom slučaju svodi na parcijalno preslikavanje $\delta: Q \times \Sigma \rightarrow Q$ koje nazivamo **funkcija prelaza** determinističkog konačnog automata. Ova funkcija se dalje proširuje u funkciju δ^* nad rečima iz Σ^*, pa se vrlo često jezik prepoznat determinističkim konačnim automatom A definiše kao:

$$L(A) = \left\{ \omega \in \Sigma^* \middle| \delta^*(i, \omega) \in F \right\}$$

Konačni automat $A = (\Sigma, Q, I, F, \Delta)$ je **potpun** (ili **kompletan**) ako za svako stanje $p \in Q$ i svako slovo $a \in \Sigma$, postoji bar jedno stanje $q \in Q$ takvo da je $(p, a, q) \in \Delta$. Može se pokazati (Vitas, 2006) da za svaki konačan automat A postoji potpun deterministički konačni automat B takav da je $L(A)=L(B)$.

Dato tvrđenje bi moglo da navede na zaključak da je svejedno da li je automat koji se koristi u pojedine svrhe deterministički ili ne. Ovakav zaključak je samo delimično tačan. Naime, prilikom upotrebe automata za generisanje jezika, svejedno je da li će biti korišćen deterministički ili nedeterministički automat. Međutim, ukoliko se automat koristi za prepoznavanje jezika, tada upotreba nedeterminističkog automata može da uzrokuje da ulazna reč ne bude prepoznata, iako pripada jeziku automata. Ovo proizilazi iz činjenice da je kod nedeterminističkog automata iz jednog stanja moguć prelaz u više različitih stanja po istoj etiketi, tj. nije jedinstveno određeno sledeće stanje automata.

Na primer, automat A prikazan na slici 5 generiše jezika $L=\{0, 00, 01, 10, 11, 0000, 0001, \ldots.\}$, tj. jezik svih reči sastavljenih od 0 i 1 različitih dužina, pri čemu je reč „0“ jedina reč dužine $n=1$.



Slika 5. Nedeterministički konačni automat

Kada se ovaj automat koristi za prepoznavanje jezika, reč *„0“* može da bude neprepoznata ukoliko automat iz početnog stanja *a* po etiketi *0* pređe u stanje *b*, umesto u završno stanje *c*. Odluku da li se vrši prelaz u stanje *b* ili *c* nije moguće doneti bez provere da li ulazna reč sem karaktera *0* ima i druge karaktere ili ne. Kao posledica, algoritmi koji koriste nedeterminističke automate za prepoznavanje jezika su znatno kompleksniji.

Sa druge strane, konvertovanje nedeterminističkog automata sa N stanja u ekvivalentan deterministički koji prepoznaje isti jezik može da rezultuje konačnim automatom sa brojem stanja i do 2^N.

2.4.3. Odnos konačnih automata i regularnih jezika

Klinijeva teorema: Neka je Σ konačna azbuka. Tada se familije regularnih i prepoznatljivih jezika nad Σ poklapaju, tj.

$$Reg(\Sigma^*) = Prep(\Sigma^*)$$

Klinijeva teorema je izuzetno značajna, jer daje dve karakterizacije iste familije jezika, jednu preko konačnih automata, a drugu preko regularnih izraza. Tako istovremeno za jedan jezik konačni automat određuje algoritam pomoću koga se utvrđuje da li neka niska pripada jeziku ili ne, dok regularni izrazi opisuju sintaksičku strukturu tog jezika.

Dalje, važe sledeća tvrđenja (Madarasz i Crvenković, 1995):

- Za svaku gramatiku G tipa 3 postoji konačan automat A takav da je $L(A) = L(G)$.

- Neka je A konačan deterministički automat. Tada postoji gramatika G tipa 3 takva da je $L(A) = L(G)$.

Posledica prethodna dva tvrđenja i Klinijeve teoreme jeste da se klasa regularnih jezika poklapa sa klasom jezika tipa 3.

2.5. Konačni transduktori

Za razliku od konačnih automata, koji definišu formalni jezik definisanjem skupa niski karaktera (reči), konačni transduktori definišu relacije između dva skupa niski karaktera. Drugim rečima, konačni automat za datu nisku ω utvrđuje da li pripada jeziku opisanom tim automatom ili ne. Konačni transduktor, sa druge strane, transformiše zadatu nisku u drugu nisku nad istom ili nekom drugom

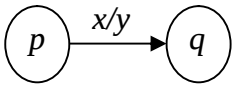
azbukom. Zbog toga se konačni transduktori često nazivaju i konačnim automatima prevodiocima.

Konačni transduktor je uređena šestorka $\tau = (\Sigma_1, \Sigma_2, Q, i, F, \Delta)$, gde su

- Σ_1 i Σ_2 **ulazna** i **izlazna** konačna azbuka
- Q konačan **skup stanja**
- $i \in Q$ **početno stanje**
- $F \subset Q$ skup **završnih stanja**
- $\Delta \subset Q \times \Sigma_1 \times \Sigma_2 \times Q$ relacija **prelaza**, čije elemente nazivamo **lukovima**.

Dakle, konačni transduktori mogu biti posmatrani kao mašine koje prepoznaju (čitaju) jednu nisku karaktera (reč), a generišu neku drugu.

Konačni transduktor je moguće predstaviti grafički grafom prelaza, na sličan način kao i konačne automate (vidi 2.4.1). Tako bi deo grafa koji odgovara prelazu $(p, x, y, q) \in Q \times \Sigma_1 \times \Sigma_2 \times Q$ izgledao kao na slici 6:



Slika 6. Prelazak iz stanja p u stanje q nakon što je pročitan simbol x, pri čemu se generiše (proizvodi, emituje) simbol y

Svakom konačnom transduktoru se može pridružiti konačni automat tako što se parovi simbola na lukovima posmatraju kao simboli konačnog automata. Neka je $\tau = (\Sigma_1, \Sigma_2, Q, i, F, \Delta)$ konačni transduktor. Tada je njegov **pripadajući** konačni automat $A = (\Sigma, Q, I, F, \Delta_1)$ definisan se:

$$\Sigma = \Sigma_1 \times \Sigma_2$$

$$(p, (a, b), q) \in \Delta_1 \Leftrightarrow (p, a, b, q) \in \Delta.$$

Alternativna definicija konačnog transduktora se sastoji u zameni skupa prelaza Δ dvema funkcijama: **funkcijom prelaza** δ: $Q \times \Sigma_1 \to Q$ i **emisionom funkcijom** ε: $Q \times \Sigma_1 \to \Sigma_2^*$ tako da:

$$\delta(p,a) = \left\{ q \in Q \middle| \exists (p,a,b,q) \in \Delta \right\}$$

$$\varepsilon(p,a) = \left\{ b \in \Sigma_2^* \middle| \exists (p,a,b,q) \in \Delta \right\}$$

Takođe, konačni transduktor se može interpretirati i kao relacija nad niskama. Prema ovoj interpretaciji, skup lukova, kao i funkcija prelaza i emisiona funkcija, se primenjuju na niske umesto na pojedinačne simbole.

Prošireni skup lukova $\tilde{\Delta} \subseteq Q \times \Sigma_1^* \times \Sigma_2^* \times Q$ je najmanji podskup takav da:

- za svako $q \in Q$, $(q, \varepsilon, \varepsilon, q) \in \tilde{\Delta}$
- za svako $\omega_1 \in \Sigma_1^*$, $\omega_2 \in \Sigma_2^*$, važi

  $(q_1, \omega_1, \omega_2, q_2) \in_{\tilde{\Delta}}$ i $(q_2, a, b, q_3) \in \Delta \Rightarrow (q_1, \omega_1 a, \omega_2 b, q_3) \in_{\tilde{\Delta}}$.

Ovakva definicija dopušta da se konačnom transduktoru pridruži relacija $L(\tau)$ na sledeći način:

$$L(\tau) = \left\{(\omega_1, \omega_2) \in \Sigma_1^* \times \Sigma_2^* \middle| \exists (i, \omega_1, \omega_2, q) \in \tilde{\Delta} \wedge q \in F \right\} \subseteq \Sigma_1^* \times \Sigma_2^*$$

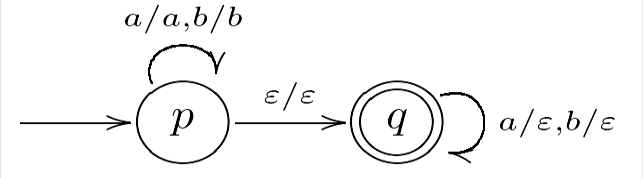
$L(\tau)$ opisuje **prevođenje** ili **transdukciju** koja se vrši transduktorom τ.

Moguća je i nešto drugačija definicija relacije $L(\tau)$, data u (Lawson, 2004). Naime, neka je $e = (q_1, x_1, y_1, q_1')...(q_n, x_n, y_n, q_n')$ jedna sekvenca prelaza transduktora. Stanje q_1 se naziva **početkom sekvence *e***, a stanje q_n' **krajem sekvence *e***. **Labela** sekvence *e* je par niski $(x_1...x_n, y_1..y_n)$. Kaže se da je sekvenca tranzicija *e* **dopustiva**, ukoliko predstavlja putanju transduktora τ. Drugim rečima, sekvenca *e* je dopustiva ako za svaki uzastopni par prelaza iz *e*, (q_i, x_i, y_i, q_i') i $(q_{i+1}, x_{i+1}, y_{i+1}, q_{i+1}')$ važi da je $q_i' = q_{i+1}$. Za dopustivu sekvencu *e* kaže se da je **uspešna**, ukoliko je njen početak početno stanje transduktora, a kraj završno stanje transduktora.

Sada je moguće uvesti relaciju $L(\tau)$ na sledeći način:

$L(\tau) = \{(x,y) \in \Sigma_1^* \times \Sigma_2^* |$ *(x,y) je labela neke uspešne sekvence prelaza* $\}$

Primer. Na slici je dat jedan transduktor:



Slika 7. Transduktor koji vraća prefiks reči

Ulazna i izlazna azbuka transduktora su iste *{a,b}*. Relacija *L(τ)* transduktora *τ* u stvari predstavlja skup svih parova niski *(x,y)* takvih da je *y* prefix od *x*. Ovo je dobar primer da se zaista radi o relaciji, a ne o funkciji, s obzirom da neprazna niska *x* ima više od jednog prefiksa.

♦

Relacije transduktora *τ* u $\Sigma_1^* \times \Sigma_2^*$ mogu biti opisane uopštavanjem Klinijeve teoreme (vidi 2.4.3). Naime, neka su $(x_1,y_1), (x_2,y_2) \in \Sigma_1^* \times \Sigma_2^*$ labele nekog prelaza. Proizvod ovih labela se definiše kao:

$$(x_1,y_1)(x_2,y_2) = (x_1 x_2, y_1 y_2)$$

Operacija proizvoda je analogna operaciji nadovezivanja u Σ_1^*. Takođe, za par $(\varepsilon,\varepsilon)$ važi :

$$(\varepsilon, \varepsilon)(x, y) = (x, y) = (x, y)(\varepsilon, \varepsilon)$$

Dalje, ako je $L, M \subseteq \Sigma_1^* \times \Sigma_2^*$ tada se proizvod *LM* definiše kao skup svih proizvoda elemenata skupa *L* i skupa *M*. Sada je moguće definisati pojam **regularnog** ili **racionalnog skupa**, analogno definiciji datoj u 2.3. Regularni podskup od $\Sigma_1^* \times \Sigma_2^*$ se naziva i **regularna** ili **racionalna relacija** iz Σ_1^* u Σ_2^*.

Važe sledeća tvrđenja (dokaze videti u (Berstel, 1979)), pri čemu je prvo od njih direktna posledica Klinijeve teoreme:

- Relacija $L \subseteq \Sigma_1^* \times \Sigma_2^*$ može biti relacija nekog konačnog transduktora sa ulaznom azbukom Σ_1 i izlaznom azbukom Σ_2 ako i samo ako je *L* regularna relacija iz $\Sigma_1^* \times \Sigma_2^*$.

- Neka su *A*, *B* i *C* konačne azbuke. Neka je *R* regularna relacija iz A^* u B^* i *R'* regularna relacija iz B^* u C^*. Tada je $R' \circ R$ regularna relacija iz A^* u C^*, gde je $(a,c) \in R' \circ R$ ako i samo ako postoji $b \in B^*$ tako da je $(a,b) \in R$ *i* $(b,c) \in R'$.

Neka je *R* regularna relacija iz A^* u B^*. Za datu nisku $x \in A^*$ ili ne postoji niska *y* iz B^* takva da je $(x,y) \in R$, ili postoji tačno jedna takva niska, ili postoji više od jedne niske koje su u relaciji R sa niskom *x*. Ukoliko relacija *R* ima osobinu da za svako $x \in A^*$ postoji barem jedan elemenet $y \in B^*$ takav da su *x* i *y* u relaciji *R*, tj. $(x,y) \in R$, tada *R* može biti posmatrana kao parcijalna funkcija iz A^* u B^*. Takva funkcija se naziva **regularna** ili **racionalna parcijalna funkcija** iz A^* u B^*.

Ukoliko neka regularna relacija nije parcijalna funkcija, to nikako ne predstavalja njen nedostatak. Regularne relacije koje nisu parcijalne funkcije se koriste u procesiranju prirodnih jezika u velikom broju slučajeva; jedan od njih je i za razrešenje višeznačnosti (Lawson, 2004). Sa druge strane, regularne parcijalne funkcije su jednostavnije za korišćenje i imaju neke dodatne osobine. Na primer, postoji algoritam za utvrđivanje da li su dve regularne parcijalne funkcije jednake ili ne, dok takav algoritam ne postoji za dve proizvoljne regularne relacije (Berstel, 1979).

Osnovna osobina transduktora, koja umnogome određuje njihovu upotrebu, jeste da produkuju neki izlaz. Zbog toga se upravo transduktori koriste kao najčešći mehanizam za pisanje pravila ekstrakcije informacije (vidi (Friburger i Maurel, 2004; Sarawagi, 2008)).

2.6. Rekurzivne mreže prelaza

U praksi, primena konačnih automata i transduktora dovodi do velikih problema zahvaljujući tome što automati i transduktori nisu modularni i mogu biti veoma složeni i teški za održavanje. Na primer, ukoliko konačnim mašinama pokušamo da opišemo sintaksu nekog jezika, odgovarajući graf bi bio veoma nepregledan, a pronalaženje svih informacija vezanih za, recimo, imeničke fraze vremenski zahtevno i nepraktično. Dodatno, jezici konačnih automata su regularni jezici, a veliki broj fenomena prirodnih jezika, pa i prirodni jezici u celini, pripadaju širim klasama jezika nego što su to regularni.

Zbog toga se u praksi umesto jednog ogromnog grafa konačnog automata ili transduktora često koristi kolekcija podgrafova (ili podmreža). Tako bi u primeru opisivanja sintakse jezika postojali po jedan graf za svaku kategoriju (rečenica, imenička fraza, glagolska fraza i sl.). Ovakav način grupisanja grafova u kolekcije (mreže) formalno je definisan teorijom rekurzivnih mreže prelaza (*Recursive transition networks – RTN*).

RTN (Woods, 1970.) predstavljaju proširenje konačnih automata na način da su sva stanja imenovana i da je dozvoljeno da lukovi, umesto oznakama koje pripadaju azbuci jezika (tzv. terminalnim oznakama), budu označeni i imenima

stanja. Svaki put kada mašina dođe na takav luk, označen neterminalnom oznakom, ta oznaka se tretira kao poziv potprograma. Trenutna lokacija se stavlja na stek kako bi se nakon obrade neterminalne oznake mašina vratila na nju. Rečenica je prihvaćena kada su završno stanje, kraj rečenice i prazan stek dostignuti u isto vreme.

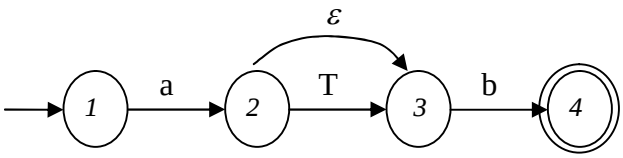
Ovde treba napomenuti da će se u okviru ovog poglavlja na dalje rekurzivne mreže prelaza posmatrati kao mašine za prepoznavanje jezika, tj. kao automati koji ne produkuju nove niske jezika, već samo utvrđuju da li određena niska pripada jeziku koji opisuju. Međutim, imajući u vidu da svaki konačni transduktor ima pripadajući konačni automat (Vitas, 2006.), sve definicije i tvrđenja iz ovog poglavlja mogu lako biti proširena tako da se rekurzivne mreže prelaza posmatraju i kao mašine prevodioci (transduktori).

Formalno, **rekurzivna mreža prelaza** nad konačnom azbukom Σ se sastoji od:

- konačnog skupa Q – **skupa stanja**;
- $i \in Q$ – **početno, inicijalno stanje**;
- skupa $F \subset Q$ – **skupa završnih, finalnih stanja**;
- skupa $\Delta \subset Q \times (\Sigma \cup Q) \times Q$ – **skupa relacija prelaza**;
- skupa $S \subset Q$ – **steka poziva.**

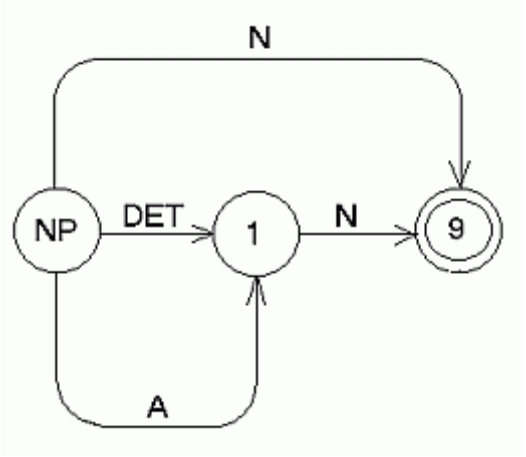
Elementi relacije prelaza Δ (lukovi) kao etikete, sem elemenata azbuke, mogu da sadrže i stanja mreže prelaza. Stanje q smešta se u stek S kada mreža prelaza naiđe na luk koji je označen tim stanjem, a skida se sa steka kada mreža prelaza naiđe na prvo završno stanje nakon poziva stanja q.

U primeru unutar poglavlja 2.2.2 predstavljen je jezik $L_2 = \{a^n b^n \mid n > 0\}$, koji sadrži sve niske u kojima se pojavljuje prvo simbol a nekoliko puta, a zatim simbol b isti broj puta. Ovaj jezik pripada grupi kontekstno slobodnih jezika i nije moguće da bude opisan konačnim automatom. Sa druge strane, jezik L_2 se jednostavno opisuje rekurzivnom mrežom prelaza T prikazanoj na slici 8.



Slika 8. Rekurzivna mreža prelaza *T* koja prepoznaje jezik $L_2 = \{a^n b^n \mid n > 0\}$

Stanja u mrežama prelaza su označena samo da bi mogla da budu identifikovana i njihove oznake nisu u vezi sa azbukom niti gramatikama koje opisuju. Ono što jeste od značaja jesu oznake lukova. Na slici 9 prikazana je mreža koja opisuje imeničke fraze u engleskom jeziku.



Slika 9. Mreža prelaza NP koja prepoznaje imeničke fraze u engleskom jeziku

Pomoću RTN moguće je predstaviti KS gramatike, tj. RTN prepoznaju kontekstno slobodne jezike. Zbog toga se RTN često posmatraju kao grafička reprezentacija kontektsno slobodnih gramatika.

Kao što je pomenuto u 2.2.3, kod kontekstno slobodnih gramatika sva pravila su oblika $A \rightarrow \alpha$, gde je $A \in N$, $\alpha \in (\Sigma \cup N)^*$. Polazeći od pravila izvođenja $(A, \alpha) \in P$ neke gramatike $G=(\Sigma, N, P, S)$, za svaki neterminalni simbol *A* koji se nalazi sa leve strane pravila, kreira se po jedna podmreža koja opisuje to pravilo. Kolekcija takvih podmreža predstavlja RTN koji je ekvivalentan datoj gramatici.

Primer. Posmatrajmo gramatiku koja se sastoji od sledećih pravila:

```
S → NP VP
NP → DET N
```

VP → V NP

Ekvivalentna RTN će se sastojati od najmanje tri podmreže, po jedne za svaki od neterminalnih simbola S, NP i VP. U nekim slučajevima, RTN bi obuhvatila i podmreže koje opisuju leksičke kategorije označene simbolima DET, N i V.

♦

Putanje u podmrežama koje odgovaraju određenim pravilima se sastoje od lukova koji su označeni simbolima sadržanim na desnoj strani pravila. Različite putanje u podmreži odgovaraju različitim pravilima gramatike sa istim simbolom na levoj strani.

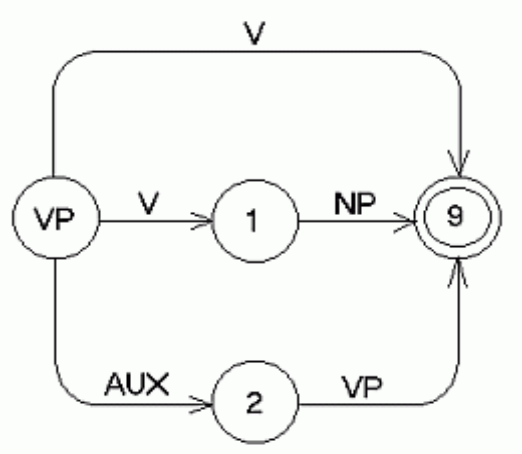
Primer. Neka gramatika sadrži, između ostalih, i sledeća tri pravila:

VP → V

VP → V NP

VP → AUX VP

Simbol na levoj strani sva tri pravila je isti (VP) i njemu će odgovarati podmreža VP. Ova podmreža treba da ima tri moguće putanje, od kojih svaka odgovara po jednom od gornja tri pravila. Tako, podmreža ekvivalentna gornjim pravilama bi izgledala kao na slici:



Gornja putanja podmreže, označena simbolom V, odgovara pravilu VP → V. Srednja putanja, koja se sastoji od dva prelaza označena simbolima V i NP, odgovara pravilu VP → V NP. Donja putanja, sa dva prelaza označena simbolima AUX i VP, odgovara pravilu VP → AUX VP.

♦

RTN prikazana u prethodnom primeru je nedeterministička, s obzirom da postoje dva luka označena istom labelom (V), a koja polaze od inicijalnog stanja VP. Problem neprepoznavanja fraza koji bi mogao da bude prouzrokavan nedeterminizmom rekurzivnih mreža se rešava upotrebom „*backtracking*“ algoritma. Za neki ulazni string testiraju se redom putanje, od vrha na dole. Ukoliko prva putanja ne dovede do završnog stanja, testiranje počinje od početka po sledećoj putanji, i tako redom dok se ne dođe do završnog stanja.

O redosledu putanja treba voditi računa kada se kreiraju grafovi mreža prelaza. Pravilan izbor redosleda može da poboljša performanse i brzinu prepoznavanja, ukoliko se prvo stave putanje koje odgovaraja frazama koje se češće ili sa većom verovatnoćom pojavljuju u tekstu koji se obrađuje.

U zavisnosti od oznake na luku, u okviru RTN mogu postojati dve vrste prelaza:

- CAT prelaz (Woods, 1970.) – prelaz označen simbolom čija podmreža sadrži samo lukove označene elementima azbuke, tj. podmrežom koja u stvari predstavlja konačni automat (u okviru mreže prikazane na slici 9, takav je luk označen sa DET) ili transduktor

- PUSH prelaz – prelaz koji predstavlja izlaz iz jedne mreže i ulaz u drugu (u okviru mreže prikazane na slici u prethodnom primeru, takav je, između ostalih, luk NP). Prilikom prelaza, odgovarajuća labela se smešta na stek (potisnu listu). Ukoliko novopozvana mreža prepozna nisku, tj. dođe do završnog stanja, njena oznaka se skida sa steka.

Potisna lista se često ne prikazuje u okviru specifikacije RTN, već se nalazi u memoriji računara na kome se RTN implementira. Upravo ta potisna lista omogućava računaru prolazak kroz podmreže i povratak na odgovarajuće lokacije u okviru RTN. Na taj način, RTN je u stanju da obradi i prepozna rečenice generisane konteksno slobodnim gramatikama. Otuda su rekurzivne mreže prelaza ekvivalentne kontekstno slobodnim gramatikama, i potisnim automatima (više o potisnim automatima videti u (Vitas, 2006.)).

Iako su u osnovi izgrađeni od konačnih automata ili transduktora, RTN nisu konačni modeli. Rekurzivni pozivi grafova unutar mreže prelaza, kao i mogućnost

postojanja unakrsnih referenci, čine da RTN mogu da imaju beskonačno mnogo stanja. Da bi se u algoritmima koji koriste RTN izbegle pojave beskonačnih petlji, obično se vrši određena vrsta aproksimacije konačnim automatom ili transduktorom. Prilikom ove aproksimacije deo reči jezika koji odgovara određenoj mreži prelaza može da ostane neprepoznat.

Primer.

Softverski paket UNITEX, namenjen za lingvističku obradu teksta, u sebi sadrži program nazvan `Flatten`, koji omogućava konverziju RTN gramatike u konačnu mašinu kad god je to moguće ili konstrukciju aproksimativne konačne mašine, kada to nije (Paumier, 2011). Naime, prilikom kompiliranja dolazi do zamene svakog pojedinačnog poziva nekog podgrafa samim podgrafom. Ova zamena se vrši do određene dubine. Pozivi podgrafova koji se nalaze nakon ove vrednosti bivaju zamenjeni praznom reči, i u tom slučaju rezultat kompiliranja je konačna mašina, ali nije ekvivalentna polaznom grafu. Moguća je i opcija da se pozivi ovih grafova zadrže, kojom prilikom je zadržana ekvivalentnost, međutim rezultat kompiliranja nije konačni transduktor.

♦

2.7. Proširene mreže prelaza

Ovde treba pomenuti još jedan model apstraktnih mašina, proširene mreže prelaza (*Augmented Transition Network* – ATN). Ni ovaj model nije konačan, ali zbog svog značaja, oslanjanja na konačne modele i snage koju ima u prepoznavanju i generisanju jezika, biće u kratkim crtama opisan u okviru ovog poglavlja. Za detaljniji opis čitalac se upućuje na (Woods, 1970).

ATN je rekurzivna mreža prelaza koja je proširena na dva načina. Prvo, lukovima je moguće dodati *akcije*. Akcije omogućavaju eksplicitno stvaranje i imenovanje novih struktura. Imena, nazvana registri, funkcionišu u velikoj meri kao simboličke promenljive u programskim jezicima, oni mogu biti korišćeni u kasnijim akcijama, u okviru lukova koji slede. Za registar se kaže da sadrži strukturu koju imenuje. Akcije određuju promene nad sadržajem registra u smislu

trenutnog ulaznog simbola, prethodnog sadržaja registra i rezultata izračunavanja niskog nivoa (potiskivanje sadržaja steka). Ovo znači da čim se sastavni delovi rečenice otkriju, oni mogu biti čuvani u registru dok se ne spoje sa drugim delovima iz drugih rečenica.

Drugo, lukovima je moguće dodati uslove, koji omogućuju veću kontrolu prelaza. Uslov je bulovska kombinacija predikata u kojima učestvuju ulazni simboli i sadržaj registra. Luk ne može da bude pređen ukoliko izračunavanje njegovog ulaza ima vrednost „netačno“, bez obzira da li ulazni simbol odgovora njegovoj oznaci ili ne. Ovo praktično znači da je omogućeno postavljanje strožijih ograničenja, kao i da informacija o prethodnim stanjima može biti preneta kroz mrežu kako bi se odredili budući prelazi.

Preciznije, specifikacija jezika za opisivanje proširenih mreža prelaza, prvi put predstavljena u (Woods, 1970), prikazana je na slici 10.

⟨transition network⟩ → (⟨arc set⟩⟨arc set⟩*)
⟨arc set⟩ → (⟨state⟩⟨arc⟩*)
⟨arc⟩ → (CAT ⟨category name⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(PUSH ⟨state⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(TST ⟨arbitrary label⟩⟨test⟩⟨action⟩* ⟨term act⟩)|
(POP ⟨form⟩⟨test⟩)
⟨action⟩ → (SETR ⟨register⟩⟨form⟩)|
(SENDR ⟨register⟩⟨form⟩)|
(LIFTR ⟨register⟩⟨form⟩)
⟨term act⟩ → (TO ⟨state⟩)|
(JUMP ⟨state⟩)
⟨form⟩ → (GETR ⟨register⟩)|
*|
(GETF ⟨feature⟩)|
(BUILDQ ⟨fragment⟩⟨register⟩*)|
(LIST ⟨form⟩*)|
(APPEND ⟨form⟩⟨form⟩)|
(QUOTE ⟨arbitrary structure⟩)

Slika 10. Specifikacija jezika za predstavljanje proširenih mreža prelaza (Woods, 1970)

Ova specifikacija je predstavljena u formi kontekstno slobodne gramatike, gde su neterminalni simboli predstavljeni uglastim zagradama. Za detaljniju analizu navedenog formalizma čitalac se upućuje na (Woods, 1970). Ono na šta ovde treba skrenuti pažnju jesu lukovi i njihovo označavanje (<arc>). Prvi element svakog luka jeste reč koja označava tip prelaza. U zavisnosti od tipa

prelaza, druga reč može da označi stanje ili ime neke kategorije. Treća reč u označavanju je `<test>`, što predstavlja određeni uslov koji mora da bude zadovoljen, kako bi se izvršio prelaz po luku.

CAT prelaz i PUSH prelaz su isti kao kod rekurzivnih mreža prelaza (vidi poglavlje 2.6). TST prelaz je prelaz koji u sebi sadrži test koji se proverava. Samo ako je uslov testa zadovoljen, vrši se prelaz po tom luku. Ova vrsta lukova omogućava grananje prilikom prolaska kroz mrežu, slično `if-then` naredbama u algoritmima. U svakom od ova tri prelaza, akcije koje se preduzimaju kreiraju neku strukturu, a završna akcija određuje stanje kome se predaje kontrola daljeg izvršavanja, kao rezultat prelaza. Dve moguće završne akcije su TO i JUMP, koje određuju da li se preuzima sledeća reč sa ulaza ili se ponovo obrađuje ista reč. POP prelaz je prelaz koji određuje pod kojim uslovima stanje treba da bude smatrano kao završno, i koja struktura (forma) se vraća kao rezultat ukoliko jeste završno.

Upisivanje sadržaja u registar obavlja se pomoću tri funkcije, SETR, SENDR i LIFTR, pri čemu postoji razlika u trenutku upisivanja. Funkcija SETR upisuje sadržaj u registar na tekućem nivou. Funkcija SENDR upisuje sadržaj na nižem nivou, tj. koristi se za slanje informacije na niži nivo izračunavanja (na primer, u okviru dela mreže koji se iz nje poziva). Funkcija LIFTR upisuje sadržaj u registar na višem nivou, tj. koristi se za slanje informacije na viši nivo izračunavanja (na primer, podmreži koja je pozvala tekuću).

Ovakvo proširenje mreža prelaza omogućava da ATN mašine funkcionišu kao i Tjuringova mašina, a to znači da su u stanju da prepoznaju i jezike bez ograničenja, tj. jezike tipa 0.

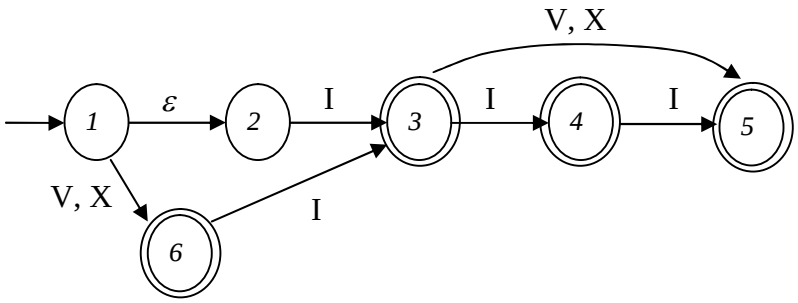
2.8. Značaj i primena konačnih modela u obradi teksta

Konačne mašine se koriste u velikom broju oblasti računarske lingvistike. Njihova upotreba je opravdana i sa stanovišta lingvistike i sa stanovišta računarstva.

Sa stanovišta lingvistike, konačni automati su pogodni kao način za jednostavno opisivanje relevantnih lokalnih fenomena koji se pojavljuju u empirijskim proučavanjima jezika. Iako formalni jezik definisan automatom ne mora da bude ni u kakvoj vezi sa prirodnim jezikom (formalni jezik može biti korišćen za modeliranje stanja, npr. aparata za kafu), ipak se često koristi za modeliranje nekog dela prirodnog jezika (fonologije, morfologije, sintakse i dr.).

Sa stanovišta računarstva, korišćenje konačnih mašina je uglavnom motivisano vremenskom i prostornom efikasnošću. Vremenska efikasnost se uglavnom postiže upotrebom determinističkih automata (Vitas, 2006; Jurafsky i Martin, 2008). Izlaz determinističkih mašina zavisi, uglavnom linearno, samo od veličine ulaza i zbog toga se mogu smatrati optimalnim. Prostorna efikasnost se postiže minimizacijom determinističkih automata (Aho i sar. 1974).

Korišćenje formalnih jezika i konačnih automata obično vodi kompaktnim reprezentacijama leksičkih pravila ili idioma i klišea. Na slici 11 prikazan je konačni automat koji prepoznaje rimske brojeve od 1 do 10.

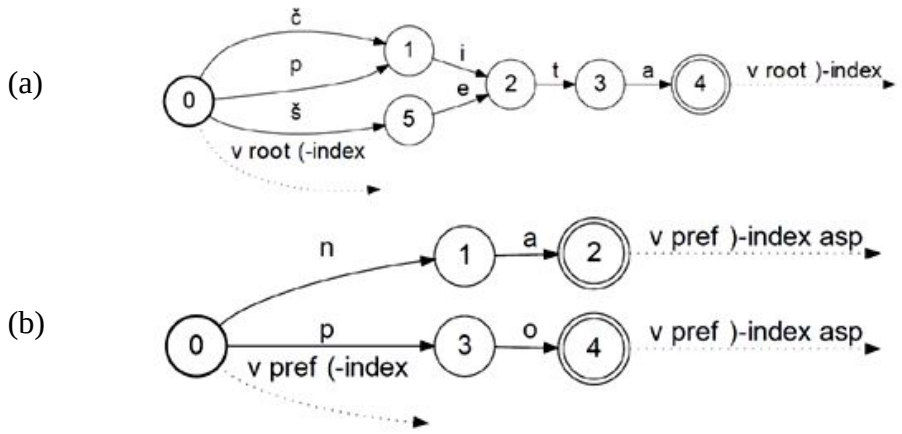


Slika 11. Konačni automat koji prepoznaje rimske brojeve od 1 do 10

Veliki broj drugih primera adekvatne reprezentacije različitih lingvističkih fenomena pomoću konačnih automata je dat u literaturi (Gross i Perrin, 1987; Jurafsky i Martin, 2008; Mohri, 1996).

Grafički prikaz automata omogućava vizuelizaciju i jednostavno modifikovanje automata, koje opet pomaže lingvistima u korigovanju i kompletiranju gramatike. Parsiranje kontekstno slobodnih gramatika može biti rešeno korišćenjem konačnih automata u okviru RTN i ATN (Woods, 1970). Šta više, mehanizmi na koje se oslanja većina metoda za parsiranje su usko povezani sa automatima.

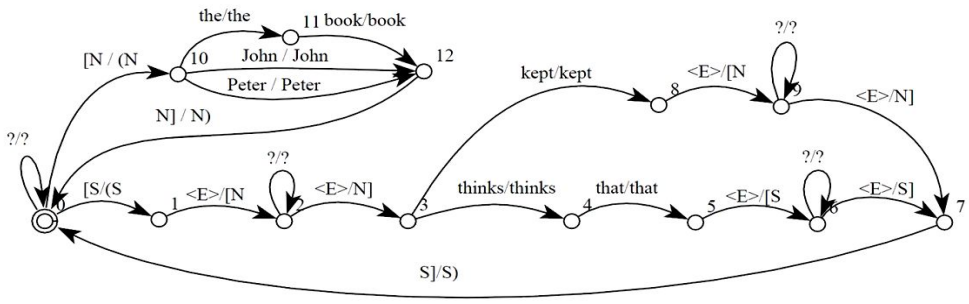
Konačni transduktori se u računarskoj lingvistici koriste za morfološko parsiranje, opisivanje pravopisnih pravila, opisivanje inflekcionih pravila i sl. Slika 12 prikazuje dva transduktora, preuzeta iz (Ćavar i sar. 2009), za izdvajanje glagolske osnove (Slika 12. a) i prefiksa reči (Slika 12.b).



Slika 12. Transduktori za izdvajanje glagolske osnove (a) i prefiksa (b) reči (Ćavar i sar. 2009)

Detaljniji pregled teorijskih i praktičnih primena konačnih transduktora u obradi prirodnih jezika je dat u (Jurafsky i Martin, 2008), (Roche i Schabes, 1997) i (Kornai, 1999).

U (Roche, 1999) je opisan način na koji konačni transduktori pružaju prirodan i unificiran metod parsiranja rečenica u kojima se kombinuju slobodne komponente i idiomi u engleskom jeziku. Jedan takav transduktor je prikazan na slici 13.



Slika 13. Jedan od transduktora za parsiranje rečenica u kojima se pojavljuju idiomi (Roche, 1999)

Transduktor sa slike 13, primenjen na rečenicu „*John thinks that Peter kept the book*”, bi produkovao sledeći izlaz:

```
(S (N John N) thinks that (S (N Peter N) kept (N the book N)S) S)
```

Konačni transduktori su veoma adekvatni kao modeli koji mogu biti automatski proizvedeni tokom treniranja u procesu mašinskog prevođenja. U (Casacuberta i sar. 2005) je opisana tehnika kojom je moguće automatski kreirati konačne transduktore. Slične tehnike automatskog kreiranja transduktora za različite namene, uglavnom zasnovane na verovatnoći i stohastici, prikazane su i u (Bangalore i Ricardi, 2000a), (Bangalore i Ricardi, 2000b), (Knight i Al-Onaizan, 1998), (Makinen, 1999), (Oncina i sar. 1993), (Vidal i sar. 1989) i (Vilar 2000).

U (Friburger i Maurel, 2004) je prikazan način kombinovanja konačnih transduktora u serije, nazvane kaskade, pomoću kojih se transformiše neki tekst i upotreba ovih kaskada na ekstrakciju imenovanih entiteta iz novinskih članaka. Kaskade transduktora su zasnovane na ideji uzastopne primene serije transduktora na tekst, pri čemu je redosled unapred jasno određen. Tako, transduktor T_i parsira tekst L_{i-1} i proizvodi tekst L_i. Zatim transduktor T_{i+1} parsira tekst L_i i tako dalje redom. Sistemi zasnovani na kaskadama transduktora su robusni, precizni i brzi.

Primena rekurzivnih mreža prelaza u obradi prirodnih jezika prikazana je u (Paumier, 2011), (Sastre i Forcada, 2007) i (Sastre, 2009).

2.9. Teorija formalnih jezika i konačnih modela u ekstrakciji informacija

Teorija formalnih jezika opisana u ovom poglavlju u ekstrakciji informacije se koristi u mnogim fazama procesa.

Sa jedne strane, kada se vrši ekstrakcija informacija iz tekstova na prirodnim jezicima neophodna je prethodna obrada teksta. Nivo obrade teksta zavisi od informacija koje treba ekstrahovati, ali je sigurno da će unutar sistema za ekstrakciju informacije postojati barem minimalni nivo obrade jezika. Zbog toga je od velikog značaja izučavanje teorije formalnih jezika, upotreba pravila za opisivanje različitih jezičkih transformacija, gramatičkih pravila i sl.

Sa druge strane, od posebnog je značaja skup niski teksta koje se smatraju relevantnim za određeni proces ekstrakcije informacije. Ovaj skup niski čini takođe jedan jezik, formalno definisan sledećom definicijom.

Definicija 1. *Jezik relevantnih informacija* je skup svih niski koje mogu da se nađu u određenom tekstu, a koje su nosioci informacije od značaja za jedan proces ekstrakcije informacije.

Jasno je iz same definicije da jezik relevantnih informacija zavisi i od zahteva za ekstrakcijom informacija, kao i od tekstualnog resursa. Ovaj jezik ćemo obeležavati sa L_{rel} ili L_{rel} *(PE, D),* gde su *PE* određeni proces ekstrakcije, a *D* tekstualni dokument iz koga se ekstrakcija vrši.

Na početku ovog poglavlja definisali smo proces ekstrakcije informacije kao proces identifikacije i klasifikacije informacija u semantičke klase. Imajući u vidu definiciju 1, jasno je da se problem identifikacije informacije svodi na prepoznavanje niski jezika L_{rel}. Dalja klasifikacija informacija bi odgovarala određivanju značenja niski jezika relevantnih informacija.

Definicija 2. *Pravilo ekstrakcije* je gramatika kojom se vrši prepoznavanje i izdvajanje niski jezika L_{rel} iz teksta koji se obrađuje.

U zavisnosti od samog zahteva za informacijom, jezik L_{rel} može da bude jezik bilo kog tipa u hijerarhiji Čomskog.

Ovde samo treba napomenuti sledeće. Na osnovu definicije 1, s obzirom da je svaki tekstualni resurs konačan, svaki jezik L_{rel} teoretski bi bio konačan, a time i regularan. Međutim, praktično nije uvek moguće pristupiti jeziku L_{rel} kao regularnom, s obzirom da kod velikih kolekcija tekstova nije moguće uočiti i nabrojati sva pojavljivanja relevantnih informacija. Naravno, kod onih tekstualnih resursa i pri takvim zahtevima za ekstrakciju gde je to moguće, prepoznavanje L_{rel} kao regularnog jezika pojednostavljuje proces ekstrakcije. Takođe savremeni sistemi za ekstrakciju pretenduju da budu što je moguće više opšteg tipa, tako da skoro nikada nisu projektovani za samo jedan tekstualni resurs. Takvi sistemi su primenljivi na različite tekstualne resurse, često nepoznate u trenutku dizajniranja

procesa ekstrakcije, tako da u tim slučajevima za jezik L_{rel} se uzima što veći skup niski koje su nosioci informacija.

U savremenim sistemima za ekstrakciju informacija dominiraju dva načina formiranja pravila za ektrakciju: ručno (od strane eksperata) i automatski (algoritmima zasnovanim najčešće na verovatnoći i označenim tekstovima). Bez obzira na koji način nastaju, pravila ekstrakcije su uvek neke gramatike (različitog tipa i različitog nivoa složenosti).

Na osnovu definicije 2 jasno je da je za pravila ekstrakcije potrebno odabrati strukture ili algoritme koji su u stanju da generišu niske, tj. da produkuju neki izlaz na osnovu teksta koji obrađuju. Zbog toga se za pravila ekstrakcije koriste mašine prevodioci (konačni transduktori, rekurzivne mreže prelaza ili proširene mreže prelaza). Prilikom kreiranja pravila ekstrakcije, ponekad čak i u situacijama kada postoji gramatika koja opisuje jezike relevantnih informacija, stalno se pravi kompromis između skupa niski jezika L_{rel} koje će biti prepoznate i izdvojene određenim pravilom i vremena i resursa utrošenih na kreiranje i primenu pravila. Često, ono što je moguće uraditi teoretski, praktično je potpuno neprimenljivo i neekonomično. Proširene mreže prelaza mogu teoretski da opišu bilo koji jezik, međutim sa stanovišta računarstva jedino su konačni modeli vremenski i prostorno efikasni. Zbog toga, pravila ekstrakcije su predstavljena najčešće konačnim mašinama, posebno konačnim transduktorima. RTN gramatike se takođe koriste za opisivanje, ali kako one ne moraju da budu konačne, u samom procesu ekstrakcije se umesto RTN koristi odgovarajuća aproksimacija u vidu konačnog transduktora. Tom prilikom, naravno, dolazi do smanjenja skupa niski jezika relevantnih informacija koje će takvim pravilima biti prepoznate i izdvojene. Stoga uvodimo još jedan pojam.

Definicija 3. *Jezik izdvojenih informacija* u oznaci L_{izdv} je skup svih niski koje pravila za ekstrakciju u jednom procesu ekstrakcije ekstrahuju.

Jasno je da će se u najvećem broju slučajeva jezici L_{rel} i L_{izdv} razlikovati. Razloga za to ima više, ali dva se posebno ističu:

1. Za pravilo ekstrakcije korišćena je gramatika čiji tip ne odgovara tipu jezika L_{rel}. Ovo se dešava u situacijama kada je jezik relevantnih

informacija složen i odgovara na primer jeziku tipa 0 ili 1, a pravila ekstrakcije su tipa 2 ili 3.

2. Pravilo ekstrakcije nije formirano kako treba, tj. prepoznaje i niske koje ne pripadaju jeziku L_{rel}

U prvom slučaju postojaće niske u jeziku L_{rel} koje ne postoje u jeziku L_{izdv}, a u drugom slučaju jezik L_{izdv} će sadržati niske koje ne postoje u jeziku L_{rel}. U većini procesa ekstrakcije dolazi i do prvog i do drugog tipa greške. Proces ekstrakcije je uspešniji što je razlika u skupovima L_{rel} i L_{izdv} manja. Upravo veličina ovih skupova i njihovi odnosi se koriste za evaluaciju sistema za ekstrakciju (vidi poglavlje 3.4).

Imajući u vidu definiciju procesa ekstrakcije informacije, tj. činjenicu da ekstrakcija informacija podrazumeva izdvajanje informacija eksplicitno zapisanih u tekstu, važi sledeća teorema, koja daje jasnu vezu između transduktora i procesa ekstrakcije informacije:

Teorema 1. Neka je $\tau = (\Sigma_1, \Sigma_2, Q, i, F, \Delta)$ transduktor (konačni transduktor ili RTN) i $L(\tau) \subseteq \Sigma_1^* \times \Sigma_2^*$ relacija transdukcije transduktora τ. Ako τ opisuje pravilo ekstrakcije neke informacije, tada važi

$$(x,y) \in L(\tau) \Rightarrow y \text{ je podreč reči } x.$$

Dokaz. Ukoliko pretpostavimo suprotno, tj. da postoji par $(x,y) \in L(\tau)$ takav da y nije podreč reči x, tada postoji barem jedan simbol $a \in \Sigma_2$ koji se sadrži u y, a ne sadrži u x. Međutim, to je u kontradikciji sa definicijom pojma ekstrakcije informacije, jer je izdvojena informacija koja se ne nalazi u ulaznom tekstu. ■

Prethodna teorema omogućava jasnu karakterizaciju transduktora koji se koriste kao pravila ekstrakcije informacija. Osobine transduktora su eksplicitno iskazane sledećim tvrđenjem, koje predstavlja direktnu posledicu teoreme 1. i ovde će biti prikazano bez dokaza.

Posledica 1. Neka je $\tau = (\Sigma_1, \Sigma_2, Q, i, F, \Delta)$ transduktor (konačni transduktor ili RTN) i $L(\tau) \subseteq \Sigma_1^* \times \Sigma_2^*$ relacija transdukcije transduktora τ. Ako τ opisuje pravilo ekstrakcije neke informacije, tada važi:

a) $\Sigma_1 = \Sigma_2 = \Sigma$

b) ako je $L_{prep} = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, (x,y) \in L(\tau)\} \subseteq \Sigma^*$ skup svih niski koje su prepoznate transduktorom τ, tada između jezika L_{izdv} i L_{prep} postoji sledeća veza:

$$x \in L_{prep} \text{ i } y \in L_{izdv} \Rightarrow y \text{ je podreč reči } x.$$

Niske $x \in L_{prep}$ ćemo nazivati i **kontekstom informacije**, a jezik L_{prep} **jezikom konteksta informacija.**

Tvrđenja data u posledici 1 daju karakterizaciju jezika konteksta informacija. Naime, imajući u vidu posledicu Klinijeve teoreme, datu u poglavlju 2.5, važi sledeće tvrđenje:

Posledica 2. Jezik konteksta informacija je regularan jezik.

Glava 3

3 SISTEMI ZA EKSTRAKCIJU INFORMACIJA

3.1. Različiti pristupi projektovanju IE sistema

Postoje dva osnovna pristupa dizajniranju IE sistema. U okviru ovog teksta označićemo ih kao sistema bazirane na znanju i sisteme bazirane na automatskom treniranju.

Sistemi bazirani na znanju (*Knowledge Engineering* - KE) su specifični po postojanju gramatika koje razvijaju ljudi - eksperti, a koje ovi sistemi koriste. Te gramatike predstavljaju formalizam pomoću koga se opisuju pravila kojima se izvlače informacije iz teksta. Obično, ekspert ima pristup tekstovima određenog domena. Na osnovu njihove analize, svog znanja i intuicije on definiše pravila koja sistem zatim koristi za ekstrakciju informacija. U takvim sistemima veština i znanje eksperta igraju ključnu ulogu i bitan faktor koji određuje uspešnost sistema i samog procesa ekstrakcije informacija. Ovaj pristup zahteva i veliki broj sati ljudskog rada. Kreiranje sistema visokih performansi obično je iterativan proces, pri čemu se inicijalno napisana pravila modifikuju i popravljaju i po nekoliko puta, na osnovu analize dobijenog izlaza u svakoj iteraciji.

Sistemi bazirani na automatskom treniranju su po samom pristupu bitno različiti. I ovde je bitna uloga eksperata, ali u mnogo manjoj meri. Ovi sistemi koriste kao polaznu osnovu unapred obeležene korpuse teksta, koji su uglavnom obeleženi od strane eksperata. Obeležavanje zavisi od namene sistema. Ukoliko se projektuje sistem za ekstrakciju imenovanih entiteta, bitno je da se izvrši obleležavanje imenovanih entiteta u nekom inicijalnom korpusu. Kada se jednom

izvrši anotacija korpusa, pokreću se algoritmi za treniranje koji na osovu anotiranih korpusa produkuju informacije potrebne za ekstrakciju iz proizvoljnog teksta. Korisnik ima udela u ovom procesu, tako što vrši procenu da li je sistem ispravno ekstrahovao informaciju. Ukoliko nije, sistem sam modifikuje svoja pravila i ponavlja proces.

Kada je oblast ekstrakcije informacije nastajala, dominirala su rešenja bazirana na znanju (Ayuso i sar. 1992, Hobbs i sar. 1992, Krupka i sar. 1992). Ovi sistemi su predstavljeni u okviru MUC konferencija i bili su veoma efikasni za ekstrakciju informacija o terorističkim napadima iz novinskih članaka. Vremenom su počeli da se pojavljuju poluautomastki i automatski sistemi, od kojih je prvi bio sistem PROTEUS (Yangarber i Grishman, 1998.). Ovaj sistem u sebi sadrži skup unapred definisanih pravila ekstrakcije događaja, ali isto tako omogućava korisniku da unese primere novih događaja, kao i način na koji treba da budu tretirani (strukturirani). Na osnovu korisnikovog unosa sistem kreira nova pravila, kojima proširuje postojeću biblioteku pravila.

U današnje vreme dominiraju sistemi bazirani na automatskom treniranju. Iako su česte rasprave o tome koji sistemi su bolji, ipak je opšti zaključak da takvu ocenu nije moguće doneti. I jedni i drugi imaju svoje prednosti i mane, koje treba uzeti u obzir kada se donosi odluka koji sistem odabrati za konkretan proces ekstrakcije informacije.

U situacijama kada ne postoje korpusi za treniranje, ili kada je potrebna visoka preciznost, sistemi bazirani na znanju postižu bolje rezultate. Ipak, oni zahtevaju više ljudskog rada nego sistemi bazirani na automatskom treniranju, kao i postojanje određenih lingvističkih resursa, kao što su elektronski rečnici i gramatike, pa je u nekim situacijama bolje odabrati sistem baziran na automatskom treniranju.

Tako na primer, u situaciji kada je potrebno napraviti sistem za ekstrakciju imenovanih entiteta, lakše je naći osobu koja je u stanju da u tekstu označi imenovane entitete, nego osobu koja je ekspert u oblasti lingvistike, koja bi kreirala pravila na osnovu kojih se izvalače imenovani entiteti iz teksta. Ali, isto tako treba imati na umu i druge probleme koji tu mogu da se jave. Prilikom

anotiranja neophodno je sačuvati konzistentnost između anotacija izvedenih od strane različitih osoba. Na primer, da li će se "Crveni krst" smatrati za imenovani entitet neke kompanije? Ne postoji tačan odgovor na ovo pitanje, već je on stvar dogovora, ali mora biti unapred poznat. Sa podacima za treniranje koji nisu konzistentni nije moguće postići veliku preciznost sistema. Sve ovo navodi na činjenicu da pribavljanje anotiranih podataka koji bi se iskoristili za treniranje nije tako jednostavno, i da je daleko skuplje nego što se u početku može učiniti. Za mnoge domene je samo obeležavanje podataka podjednako komplikovano, dugotrajno i skupo kao i kreiranje pravila za ekstrakciju informacija.

Još jedan problem koji treba uzeti u obzir prilikom odabira pristupa projektovanju sistema za ekstrakciju informacije, jeste mogućnost izmene inicijalnih zahteva za informacijom. Veoma često krajnji korisnik traži jednu vrstu podataka, ali nakon nekog vremena shvata da su mu potrebni nešto izmenjeni podaci. U zavisnosti od vrste promene, ona može bitno da utiče na sistem za IE. Recimo da je razvijen sistem za prepoznavanje imenovanih entiteta u okviru teksta koji sadrži velika i mala slova, ali da korisnik sada treba da obradi tekst pisan samo velikim slovima. Sistem baziran na automatskom anotiranju može ovu izmenu veoma lako da obradi, jednostavnim prebacivanjem korpusa za treniranje u velika slova. Sa druge strane, sistem baziran na znanju koji je informaciju o velikom i malom slovu koristio u okviru pravila za prepoznavanje imenovanih entiteta ne može više da bude upotrebljen tj. potrebno je napisati potpuno nova pravila.

U drugom slučaju, kada je razvijen sistem za prepoznavanje imena država i gradova, on veoma lako može da se transformiše u sistem koji prepoznaje i toponime ukoliko je baziran na znanju. Potrebno je samo dodavati ili proširiti skup pravila. Sa druge strane, sistem baziran na automatskom treniranju zahteva ponovno anotiranje čitavog korpusa.

Treba još istaći i to da ne mora svaki moduo sistema za IE da bude zasnovan na istom principu, već je moguće kombinovanje ova dva pristupa. Ukratko, pristup baziran na znanju je opravdan kada:

- su dostupni lingvistički resursi (elektronski rečnici, gramatike, leksikoni)

- postoji ekspert za pisanje pravila
- podaci za treniranje su nekompletni ili skupi
- važno je postići čak i malo povećanje performanse
- zahtevi za ekstrakcijom će se verovatno menjati

U suprotnim situacijama, bolje je koristiti sisteme bazirane na automatskom treniranju.

3.2. Arhitektura i komponente IE sistema

Način na koji će biti projektovan jedan IE sistem u velikoj meri zavisi od sledećih faktora:

- jezika kojim je napisan tekst

- vrsta teksta (transkript govornog teksta se analizira drugačijim tehnikama od pisanog teksta, da li je pisan i malim i velikim slovima ili samo velikim, neformalani tekst preuzet sa nekog foruma može da sadrži gramatičke i greške u kucanju, i sl.)

- osobine teksta (dugačak tekst zahteva korišćenje IR tehnika za identifikaciju manjih delova teksta koji će biti procesirani, da li tekst sadrži slike ili tabele i sl.)

- zadatak ekstrakcije informacije (neke informacije je jednostavnije izvući od nekih drugih)

Bez obzira na to što se, u zavisnosti od prethodno navedenih fakotra, sistemi za ekstrakciju informacija međusobno veoma razlikuju, ipak postoje određene faze u okviru svakog od njih koje su im zajedničke. To su:

- normalizacija

- tokenizacija

- leksička i morfološka analiza

- sintaksna analiza (parsiranje)

- razrešenje koreferenci

- identifikacija informacije

- objedinjavanje delimičnih rezultata

Neki sistemi, koji su dizajnirani za prikupljanje informacija na gramatičkom nivou (na primer, za ekstrakciju imenica ili glagola), sastoje se samo od tokenizatora, leksičko-morfološkog procesora i identifikatora informacije. Ipak, velika većina sistema je namenjena za ekstrakciju informacija višeg, semantičkog, nivoa, tako da su skoro sve faze zastupljene u njima. One predstavljaju osnovu sistema, ali je veoma verovatno da će pojedini sistemi imati i dodatne module, zavisno od njihove namene.

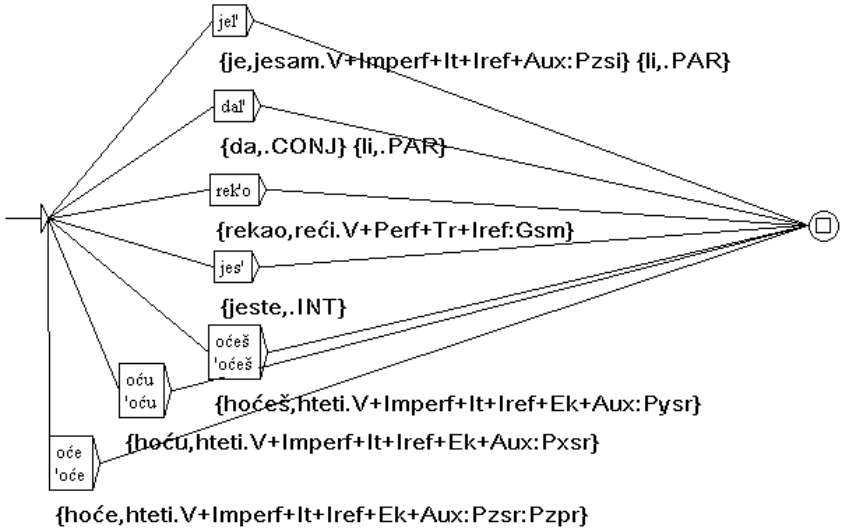
Za neke od navedenih faza IE sistema, kao što su tokenizacija ili sintaksna analiza, postoje razvijeni softverski alati. To su uglavnom alati namenjeni za lingvističku obradu teksta na prirodnim jezicima, kao što je UNITEX. Sistem za ekstrakciju informacija može da za neku od NLP faza koristi već postojeće alate, ili da implementira svoje algoritme. U svakom slučaju, obrada teksta pre identifikacije infomacije je od izuzetnog značaja, pa je neophodno znati na koji način se u dotičnom alatu izvode navedene operacije.

3.2.1. Normalizacija

Normalizacija teksta je proces u kome se ujednačavaju različiti pojavni oblici nekog lingvističkog fenomena. Ovo može da se odnosi na normalizaciju Unicode karaktera, zatim na prebacivanje svih slova u mala ili velika, uklanjanje dijakritika ili znakova interpunkcije i slično. Proces normalizacije nije uvek potreban, već se vrši u situacijama i za one fenomene čiji različiti oblici nisu od značaja za proces ekstrakcije, već ga samo otežavaju. Tako je u primeru kreiranja baze podataka namenjene za biološka istraživanja, opisanom kasnije u poglavlju 4, normalizacija izostavljena.

Najčešća je normalizacija separatora teksta. Standardni separatori su razmak, tabulator i novi red. U tekstu je moguće da se pojavi nekoliko separatora jedan za drugim. U situacijama kada ovakav niz separatora nije značajan za lingvističku analizu niti za proces ekstrakcije, on biva zamenjen, i to novim redom ukoliko je barem jedan od separatora u nizu oznaka za novi red, ili razmakom

ukoliko se u nizu separatora ne pojavljuje oznaka za novi red. Moguće je izvršiti i normalizaciju teksta u drugom smislu, po nekom proizvoljnom pravilu. Tako je na slici 14 prikazan primer transduktora koji odgovara gramatici za normalizaciju teksta uklanjanjem elizija[1].



Slika 14. Transduktor za normalizaciju elizija u tekstu na srpskom jeziku

3.2.2. Tokenizacija

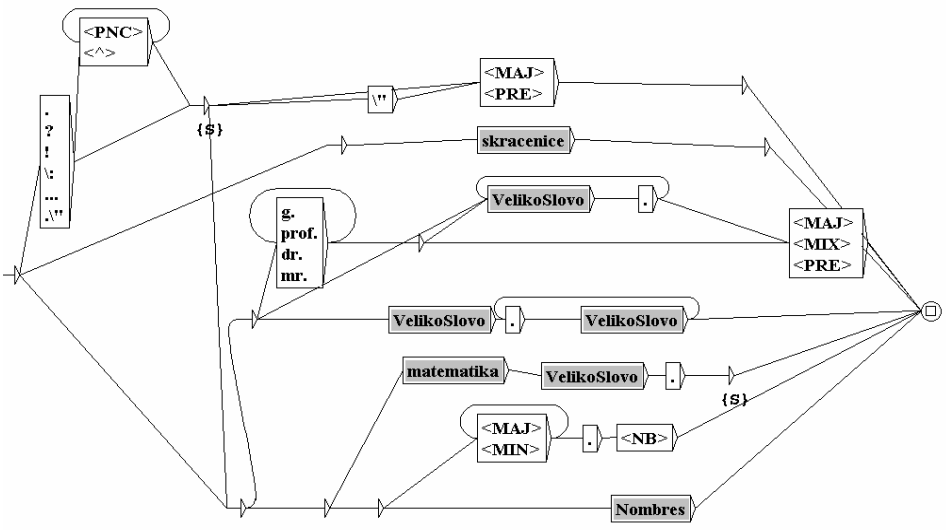
Tokenizacija je postupak razdvajanja teksta na tokene, nedeljive leksičke jedinice. Najčešće se pod tokenima smatraju reči nekog jezika, dok se kod niski nealfabetskih karaktera svaki karakter smatra za token. Tokenizacija kao postupak je relativno jednostavna za neke grupe jezika, kao što je većina evropskih jezika, dok je za neke druge jezike izuzetno složena. Kod jezika kao što su japanski ili kineski, granice jedne reči nisu evidentne na osnovu ortografije. Tokenizacija ovih jezika se u nekim NLP softverskim alatima često obavlja slovo po slovo, ili se koriste posebni podmoduli. Zajedno sa tokenizacijom ili neposredno pre nje, obavlja se često i razdvajanje teksta na rečenice. Nakon završene tokenizacije, lista dobijenih tokena se koristi u procesu leksičke i morfološke analize.

Tokenizacija se u velikom broju sistema obavlja tako što se za tokene smatraju niske slova razdvojene razmakom i cifre. Ovakav pristup je prilično

[1] elizija – izostavljanje glasova ili slogova iz reči

pojednostavljen. U slučajevima kada je proces ekstrakcije informacije složen i zahteva i određeni nivo semantičke analize ili korišćenje leksikona i rečnika, važno je da se određivanje tokena obavi što je moguće preciznije. Glavni problem u postupku tokenizacije jeste odlučivanje šta će se smatrati za token, a zatim i kreiranje pravila koja bi prepoznala takve tokene. Ukoliko se tokenizacija vrši na osnovu separatora teksta, pojaviće se problem sa nazivima kao što su „Novi Sad“ ili „Banja Luka“. Ovi nazivi nekad mogu biti smatrani za jedan token, a nekad za dva, što sve zavisi od konkretnog procesa ekstrakcije. Često se tokenizacija vrši i na osnovu znakova interpunkcije, međutim tada treba biti oprezan sa izrazima kao što su „O'Reilly" i "Reilly's".

Pravila po kojima se vrši tokenizacija obično se zadaju regularnim izrazima ili konačnim mašinama, kojima se detaljno opisuje način na koji se tekst razdvaja na tokene. Na slici 15 prikazana je jedna gramatika za razdvajanje na rečenice, koja opisuje u kom kontekstu znakovi interpunkcije treba da budu shvaćeni kao separatori rečenica.



Slika 15. RTN koji opisuje način razdvajanja na rečenice teksta na srpskom jeziku

Sekvenca `{S}` se produkuje kao izlaz transduktora na mestima gde je prepoznat kraj rečenice. Sivom bojom su označeni pozivi podgrafova. Podgraf *matematika* opisuje neke matematičke pojmove kao što su *skup*, *promenljiva* i slično. Podgraf *Nombres* opisuje različite formate brojeva u kojima mogu da se nađu interpunkcijski znaci kao što su „*1,234,567.00*“ ili „*3.14*“. Etikete <MAJ>,

<MIN>, <PRE> i <PNC> u UNITEX-u označavaju redom reč pisanu velikim slovima, reč pisanu malim slovima, reč sa velikim početnim slovom i interpunkcijski znak.

3.2.3. Leksička i morfološka analiza

Morfologija je podoblast lingvistike koja se bavi proučavanjem strukture oblika reči. Većina sistema za ekstrakciju informacija iz tekstova pisanih na jezicima sa jednostavnom morfologijom nema komponentu za morfološku analizu. U engleskom jeziku, moguće je jednostavno eksplicitno navesti sve oblike neke reči u leksikonu. Za morfološki bogatije jezike, kao što je francuski, morfološka analiza ima smisla, dok je za morfološki jako bogate jezike, kao što su nemački ili srpski, morfološka analiza neizostavna u procesu IE.

Tokom leksičke analize, tokeni dobijeni u procesu tokenizacije se traže u rečnicima kako bi se utvrdila njihova vrsta reči i druge leksičke osobine potrebne za dalje procese u okviru IE sistema. Označavanje vrste reči od strane nekih od najboljih razvijenih programskih alata se obavlja sa maksimalnom preciznošću od 95%. Iako ovo može da zvuči impresivno, ispostavlja se da tih 5 % situacija u kojim ovi sistemi greše čine upravo one situacije u kojima bi pravilno označavanje reči bilo od suštinskog značaja za proces ekstrakcije informacije.

Postoji veliki broj različitih elektronskih, mašinski čitljivih formata rečnika i leksikona, uglavnom formiranih od strane timova lingvista za pojedine jezike (za engleski jezik (Chrobot i sar., 1999; Klarsfeld i Hammany-Mc Carthy, 1991; Monceaux, 1995; Savary, 2000), za francuski jezik (Courtois, 1996; Courtois i Silberztein, 1990; Labelle, 1995), za srpski jezik (Krstev i Vitas, 2005; Vitas i sar., 2003)). Na primer, softverski alat UNITEX, koji je u okviru ovog istraživanja korišćen za pripremu teksta, koristi elektronske rečnike u DELA (*Dictionnaires Electroniques du LADL - LADL electronic dictionaries*) formatu (Silberztein, 1993). Više o DELA rečnicima i UNITEX-u videti u poglavlju 3.6.

Jedan od najvažnijih zadataka modula za leksičku i morfološku analizu jeste da dobro obrađuje vlastita imena. U primeru koji sledi prikazane su četiri situacije

u kojima je veoma bitno da sistem bude u stanju da odredi imenovani entitet, kao i njegov tip.

Primer.

"Ivančić i sinovi su danas objavili ....." – ime kompanije

"Gadafiju i sinovi okreću leđa ..." – ime osobe

"Zoran Starčević i sinovi, Nikola i Željko..." – ime osobe

"Prosjaci i sinovi je televizijska serija snimljena 1971. ..." - ime serije

♦

Prepoznavanje imena se obavlja ili KE pristupom ili automatski, na osnovu pravila izvedenih iz anotiranih korpusa. U zavisnosti od namene sistema, leksikoni i rečnici koje sistem koristi mogu pokušati da obuhvate što je više moguće reči i izraza. Ukoliko je sistem namenjen za ekstrakciju informacija jednog uskog domena, leksikoni i rečnici mogu da obuvate samo deo jezika koji se koristi u tom domenu. Odluka o tome da li koristiti leksikone opšte namene ili specifične za domen zavisi od mnogo stvari, ali svakako treba izbegavati dugačke liste reči. U njima je obično veliki procenat greške, a često dovode i do višeznačnosti. Neki sistemi, između ostalih i UNITEX, omogućavaju primenu više rečnika na tekst, pri čemu se jasno definiše redosled njihove primene.

3.2.4. Sintaksna analiza - parsiranje

Sintaksna analiza ima za cilj da identifikuje sintaksnu strukturu analiziranog dokumenta. Koji nivo sintaksne analize, ili bilo koje druge NLP faze, će biti korišćen u okviru IE sistema, zavisi od namene samog sistema i potrebe da procesira velike količine teksta u razumnom preiordu vremena. S obzirom da je većina IE sistema usmerena na ekstrahovanje relativno jednostavnih veza između entiteta i strukturalno prostih fragmenata teksta, pravila za ekstrahovanje mogu biti opisana različitim gramatikama baziranim na konačnim modelima koji mogu biti obrađeni brzo i efikasno, pa je često sintaksna analiza kompletnog teksta nepotrebna.

Većina IE sistema vrši samo jednostavnu, delimičnu analizu teksta, pa čak postoje i sistemi koji potpuno izostavljaju sintaksnu analizu. Razlog tome je što IE sistemi obično ekstrahuju samo male delove teksta, pa je uspostavljanje irelevantnih gramatičkih veza u tekstu potpuno nepotrebno.

3.2.5. Koreference

U tekstu koji se analizira često postoji mnogo različitih načina da se referiše na određeni entitet. Zbog toga uspeh IE sistema u velikoj meri zavisi od toga koliko je sposoban da razlikuje i utvrdi da li se neka informacija odnosi na jedan ili na drugi entitet. Razrešenje koreferenci je od velikog značaja, a sa druge strane često veoma kompleksan zadatak. Višestruke reference na isti entitet u tekstu se samo u retkim slučajevima vrše pomoću istih termina i fraza. Najčešće se koriste zamenice, opisi, skraćenice, akronimi, različite ortografske varijante. Izuzetno je važno da IE sistem bude u stanju da poveže što veći broj ovakvih različitih referenci istog entiteta sa njim samim.

Primer.

1. "Slavni glumac Rade Šerbedžija otkriva da će njegova prijateljica Anđelina Džoli tokom sledeće godine doći u Srbiju, ali se osvrće i na saradnju sa Lijamom Nisonom s kojim trenutno snima u Turskoj. Uspevši da preciznom organizacijom uskladi svoje porodične odnose kako se njegovo odsustvo ne bi znatno osetilo, slavni glumac otputovao je prošlog četvrtka na višenedeljni boravak u Tursku gde je počelo snimanje trilera Taken 2, u kojem igra jednu od glavnih uloga zajedno sa Lijamom Nisonom. Poslednjih meseci, kada je domaća javnost u pitanju, osim pomisli na vrsne role u holivudskim sagama, pominjanje imena ovog umetnika iziskuje i pitanje da li će i kada njegova prijateljica Anđelina Džoli pristići i u Beograd."

2. "Boris Tadić", "Tadić", "Predsednik"

3. "Sjedinjene Američke Države", "SAD", "Amerika"

4. "Gram positive", "Gram +", "Gram pos"

5. "alpha helix", "alfa heliks", "alfa-heliks", "α heliks"

♦

U okviru IE sistema, i moduo za razrešenje koreferenci može biti dizajniran na bazi znanja ili na bazi učenja. KE pristup je primenjen u okviru FASTUS sistema (Appelt i sar. 1993.). Za razrešenje koreferenci na bazi učenja mogu biti korišćena stabla odlučivanja (McCarthy i Lehnert, 1995.), koja u suštini predstavljaju `if-then` izraze, pri čemu se uslovi u `if` izrazu određuju na različite načine.

Primer. Sistem RESOLVE kreiran od strane (McCarthy i Lehnert, 1995) koji koristi stabla odlučivanja za razrešenje koreferenci bio je dizajniran za potrebe MUC-5 konferencije. Informacije su bile sadržane u rečenicama kao što je:

FAMILYMART CO. OF **SEIBU SAISON GROUP** WILL OPEN A CONVENIENCE STORE IN TAIPEI FRIDAY IN A **JOINT VENTURE** WITH **TAIWAN'S LARGEST CAR DEALER**, THE COMPANY SAID WEDNESDAY.

U datom primeru podebljani su delovi tekta koji sadrže relevantne informacije. Prva od fraza („FAMILYMART CO.“) bi trebalo da bude strukturirana od strane sistema na sledeći način:

```
:string   "FAMILYMART CO."
:slots    ENTITY  (name "FAMILYMART CO.")
                  (type COMPANY)
                  (relationship JV-PARENT CHILD)
```

Slično, fraza „TAIWAN'S LARGEST CAR DEALER“ bi bila obrađena na sledeći način:

```
:string   "TAIWAN'S LARGEST CAR DEALER"
:slots    ENTITY  (type COMPANY)
                              (relationship JV-PARENT)
                              (nationality "Taiwan (COUNTRY)")
```

Da bi koristili stabla odlučivanja, autori su podatke predstavili u obliku vektora, tj. parova atribut-vrednost. Prilikom razrešenja koreferenci za dve izdvojene fraze posmatrani su prvo pojedinačni atributi, a zatim i atributi tog para fraza. Za prethodna dva entiteta, vrednosti atributa bi bile:

Pojedinačne fraze:	*Parovi fraza:*
NAME-1 = YES	ALIAS = NO
JV-CHILD-1 = NO	BOTH-JV-CHILD = NO
NAME-2= YES	COMMON-NP = NO
JV-CHILD-2= NO	SAME-SENTENCE = NO

Pojedinačni atributi:

NAME-i – Da li referenca *i* sadrži ime? Moguće vrednosti *YES* i *NO*

JV-CHILD-i – Da li se referenca *i* odnosi na rezultat nekog zajedničkog ulaganja? Moguće vrednosti *YES, NO* i *UNKNOWN*.

Atributi parova:

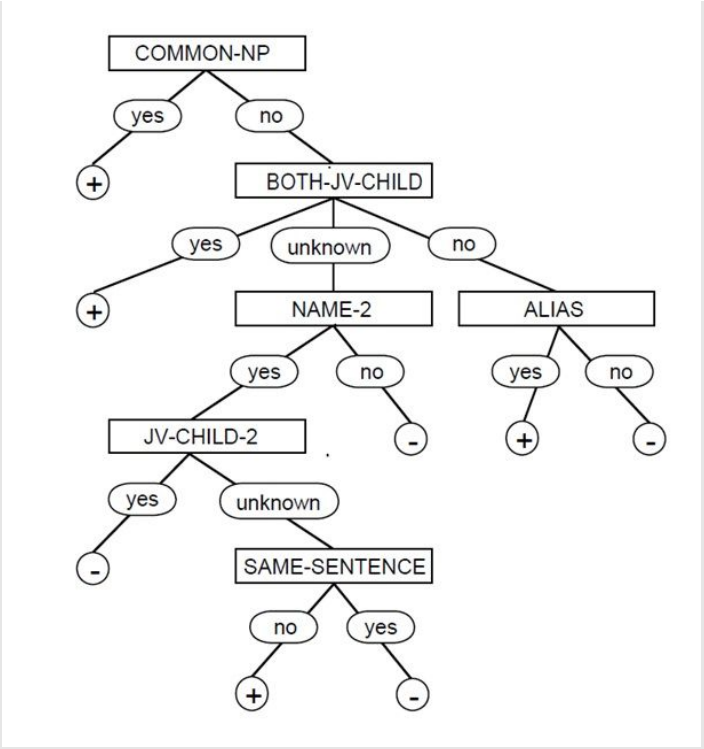
ALIAS - Da li jedna referenca sadrži alijas druge? Moguće vrednosti *YES* i *NO*.

BOTH-JV-CHILD - Da li obe reference ukazuju na rezultat nekog zajedničkog ulaganja? Moguće vrednosti *YES, NO* i *UNKNOWN*.

COMMON-NP - Da li obe reference pripadaju istoj imeničkoj frazi? Moguće vrednosti *YES* i *NO*.

SAME-SENTENCE - Da li obe reference pripadaju istoj rečenici? Moguće vrednosti *YES* i *NO*.

Na osnovu ovih atributa formirano je drvo odlučivanja prikazano na slici 16. Konkretno, kada se ovo drvo odlučivanja primeni na vrednosti iz tabele, za dve reference iz primera rezultat je negativan.



Slika 16. Drvo odlučivanja za razrešenje koreferenci (McCarthy i Lehnert, 1995)

♦

3.2.6. Identifikacija informacije

Nakon lingvističke obrade teksta, pristupa se postupku identifikovanja informacija iz teksta, njihovom izdvajanju i strukturiranju (dodeljivanju semantičkog značenja). Postoji nekoliko preduslova da bi se ova faza uspešno izvršila.

Prvo je potrebno precizno i jasno odrediti koje informacije u tekstu se smatraju relevantnim, tj. koje informacije sistem treba da ekstrahuje. U zavisnosti od namene IE sistema, obično se vrši ekstrakcija entiteta, atributa, relacija između entiteta i događaja. Tipovi entiteta koji su najčešće ekstrahovani od strane IE sistema su imenovani entiteti (organizacije, osobe, lokacije, knjige, filmovi, institucije, hoteli i dr.), zatim drugi imenovani koncepti (proteini, hemijska jedinjenja, lekovi, bolesti i dr.), vremenske odrednice (datumi, dani u nedelji i sl.) ili mere (monetarni izrazi, razdaljine, veličine, težine i sl.). Za svaku tekstualnu

referencu prepoznatu kao relevantnu potrebno je odrediti njen tip, kao i fragment teksta koji je nosilac informacije.

U procesima ekstrakcije informacija, često je potrebno za određeni entitet izdvojiti i vrednosti pojedinih atributa koji su od interesa. Na primer, ukoliko se entitet odnosi na neku kompaniju, tada je često potrebno izdvojiti informacije o nazivu kompanije, tipu, vlasniku, državi iz koje je kompanija i sl. Izdvajanje atributa je u tesnoj vezi sa razrešenjem koreferenci, kako bi se tačno utvrdilo na koji entitet se neki atribut odnosi.

Nakon izdvajanja entiteta i atributa, a u zavisnosti od namene IE sistema, često se pristupa i utvrđivanju relacija između entiteta. Iako uspostavljanje nekih kompleksnijih odnosa između entiteta može da prevaziđe okvire ekstrakcije informacija, ukoliko su ti odnosi eksplicitno prikazani u tekstu i IE sistemi mogu da ih ekstrahuju. Tako je, na primer moguće iz teksta izdvojiti informacije o tome ko je zaposlen u nekoj kompaniji, ili ko je vlasnik nekog preduzeća. Ukoliko se radi o medicinskoj dokumentaciji ili naučnim radovima, česta je primena IE sistema za uspostavljanje relacije između dijagnoze i prepisane terapije, ili genomskih sekvenci i proteina.

Još kompleksniji zadatak ekstrakcije informacija jeste određivanje informacije o događajima, kao što su teroristički napadi, transferi sportista iz jednog kluba u drugi, plasiranje novih proizvoda, interakcije između enzima, saobraćajne nesreće i drugo. Ovakvi i slični događaji mogu biti, sa stanovišta ekstrakcije informacije, posmatrani kao kompleksne relacije kod kojih je ključno vreme kada su se odigrali.

Nakon jasnog i preciznog utvrđivanja vrste entiteta koji je od interesa i atributa i relacija koje je potrebno izdvojiti, potrebno je jasno odrediti i format izlaznih podataka, tj. ekstrahovanih informacija. Struktura rezultujućih podataka mora da bude:

- reprezentativna, tj. da jasno predstavlja i definiše značenje pojedinih informacija

- dostižna u odnosu na ulazne podatke, tj. da sva polja u izlaznoj strukturi mogu da se generišu na osnovu ulaznih podataka, i

- pogodna za dalju obradu i analizu.

Ova faza se uglavnom obavlja definisanjem i primenom pravila ekstrakcije, tj. opisivanjem jezika relevantnih informacija i njegovim mapiranjem u semantičke kategorije. Pravila ekstrakcije mogu biti kreirana od strane eksperata ili automatski, na osnovu anotiranih tekstova. Bez obzira na način generisanja pravila, da bi sistem bio što uspešniji, neophodno je prvo izvršiti analizu domena u kome će se neki IE sistem primenjivati. Za dizajniranje pravila ekstrakcije relevantnih za specifičan domen koriste se dva pristupa, tzv. *atomski* i *molekularni* (Abolhassani, 2003; Kaiser i Miksch, 2005).

Molekularni pristup je i najčešći. On uključuje poklapanje (pronalaženje) svih ili većine atributa jednog entiteta ili događaja u pojedinačnom fragmentu teksta, tj. pokušava da pronađe i izdvoji kompletan slog podataka u jednom koraku. Razvojni ciklus ovog pristupa počinje malim brojem visoko pouzdanih pravila koji obuhvataju većinu slučajeva u određenom domenu, ignorišući pritom veliki broj slučajeva koji se ređe pojavljuju. Ova pravila se obično generišu tako da obuhvate one tekstualne fraze koje su od strane ljudi-eksperata, a na osnovu njihovog uvida u tekst, određene kao najčešći obrasci ili oblici u kojima se informacija nalazi u tekstu. Daljim razvojem, pravila se modifikuju ili kreiraju nova kako bi obuhvatila i ove ređe slučajeve. Tako sistem počinje sa velikom preciznošću i niskim odzivom, a zatim postepeno smanjuje preciznost kako bi povećao odziv. Ukoliko se i pravila generišu od strane eksperata, dakle KE pristupom, moguće je u određenom broju slučajeva zadržati visoku preciznost.

Atomski pristup kreira moduo za analizu domena koji prepoznaje argumente događaja i kombinuje ih u strukturu obrasca strogo i isključivo na osnovu inteligentnog pogađanja (umesto na osnovu sintaksnih veza). Ovaj pristup je baziran na pretpostavci da svaka imenička fraza određenog oblika i svaki glagol odgovarajućeg tipa, bez obzira na njihove uzajamne sintaksne veze, indikuju neki događaj ili odnos koji je od ineteresa za proces ekstrakcije informacija. Na taj način, sistemi započinju proces esktrakcije sa velikim odzivom, da bi u kasnijim iteracijama formirali filtere za uklanjanje pogrešno prepoznatih fraza (eng. *false positive*). Poboljšanje filtera i heuristika za kombinovanje atomskih elemenata

dovodi do povećanja preciznosta, ali smanjenja odziva. Atomski pristup je opravdan u situacijama i za domene u kojima je lako ustanoviti tip entiteta, kao i tip atributa kome odgovara prepoznata informacija. Atomski pristup je češći kod sistema zasnovanih na automatskom kreiranju pravila.

Nakon analize domena i definisanja strukture rezultujućih podataka, generišu se pravila ekstrakcije. Ova pravila mogu biti pisana ručno, od strane ljudi, ili automatski generisana, od strane sistema za ekstrakciju različitim metodama učenja. Zbog svoje osobine da produkuju neki izlaz, za pravila ekstrakcije se najšešće koriste konačni transduktori i rekurzivne mreže prelaza bazirane na transduktorima.

Osnovna forma jednog pravila ekstrakcije je:

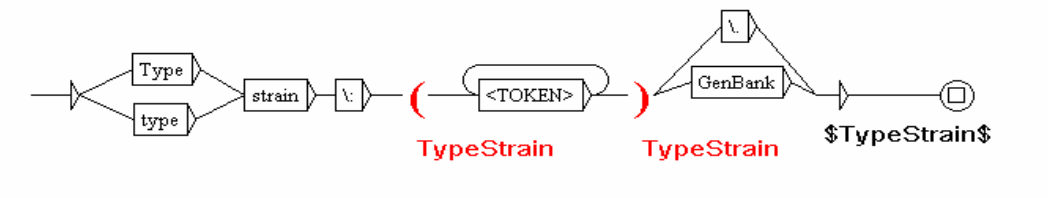
`Konteksni obrazac → Akcija`

Konteksni obrazac se sastoji od jednog ili više obrazaca koji opisuju različite osobine pojedinih entiteta i konteksta u kome se oni pojavljuju u tekstu. Opisivanje konteksta se obavlja nekim od konačnih modela. Deo pravila nazvan **akcija** predstavalja klasifikovanje izdvojene informacije u neku od unapred definisanih semantičke klasa.

Postoji veliki broj različitih formata za predstavljanje pravila ekstrakcije. To su Common Pattern Specification Language (CSPL) (Appelt i sar., 1993) i formati izvedeni iz njega kao što su JAPE (Cunningam i sar., 2002), obrasci i liste kao u sistemu Rapier (Califf i Mooney, 1999), regularni izrazi kao u sistemu WHISK (Soderland, 1999), SQL izrazi (Jayram i sar., 2006; Reis i sar., 2008) i Datalog izrazi (Shen i sar., 2007). Sve te reprezentacije imaju dosta zajedničkih osobina i mogu biti jednostavno uopštene. U stvari, suštinski sve navedene reprezentacije mogu biti predstavljene konačnim modelima, na prvom mestu transduktorima i rekurzivnim mrežama prelaza. U okviru ove disertacije korišten je programski sistem UNITEX za kreiranje pravila ekstrakcije.

Na slici 17 prikazan je graf za ekstrakciju *TypeStrain* osobine (videti poglavlje 4) iz enciklopedijskog teksta o mikroorganizmima. Ceo graf opisuje kontekst u kome informacija može da se nađe u tekstu, dok je zagradama označen deo teksta koji će biti izdvojen. U konkretnom slučaju to je bilo šta (bilo koji

token, u UNITEX-u označen sa <TOKEN>) što se nalazi u određenom kontekstu. Izdvojeni deo teksta se smešta u promenljivu *TypeStrain*, čija vrednost se zatim dodeljuje izlazu transduktora (na grafu označeno sa *$TypeStrain$*).



Slika 17. Graf za ekstrakciju *TypeStrain* osobine mikroorganizma iz teksta

Graf sa slike 17 bi prepoznao sledeće fragmente teksta, pri čemu su podebljani delovi teksta koje bi transduktor generisao kao izlaz:

Type strain: ***Breinl strain, ATCC VR-142****.*

Type strain: ***Z9-Hu (Reference Center for Rickettsioses, Marseille, France)****.*

Type strain: ***NIAID Phillips 32****.*

Type strain: ***RML 369-C (Rocky Mountain Laboratories Collection)****.*

Type strain: ***2678, ATCC VR-610****.*

Konteksni obrazac može, ali ne mora da sadrži kontekst pojavljivanja informacije. Ovo sve zavisi od prirode same informacije. Na primer, izdvajanje e-mail adresa iz radne biografije kandidata zahteva pronalaženje reči koja u sebi ima karakter @, bez obzira na kontekst u kome se pojavljuje. Sa druge strane, ukoliko se vrši ekstrakcija e-mail adresa sa veba, problem je složeniji. U okviru veb strana pojavljuju se i drugi tokeni koji sadrže znak @, a ne predstavljaju e-mail adrese. Takođe, danas je široko raspostranjeno navođenje e-mail adresa u drugim formatima, kao što su „*pajic(at)agrif.bg.ac.rs*“, „*pajic at agrif dot bg dot ac dot rs*“ ili čak u obliku „*ime.prezime@agrif.bg.ac.rs*“. Imajući sve to u vidu, izdvajanje e-mail adresa sa veba zahteva opisivanje šireg konteksta u kome se informacija pojavljuje.

U najvećem broju sistema vrši se kaskadna primena pravila ekstrakcije. Pravila se primenjuju na tekst u nekoliko različitih, uzastopnih faza, pri čemu izlaz jedne faze predstavlja ulaz za sledeću. Na primer, sistem za izdvajanje adresa iz teksta bi mogao da ima dve faze. U prvoj fazi bi se primenjivala pravila za označavanje imenovanih entiteta u tekstu, kao što su imena ljudi, ulica,

gradova. U drugoj fazi bi zatim ti entiteti bili grupisani u blokove adresa, koje bi tom fazom bile izdvajane.

3.2.7. Objedinjavanje delimičnih rezultata

Veoma često se neka informacija nalazi u okviru nekoliko rečenica ili na nekoliko mesta u tekstu. U tom slučaju biće ekstrahovani delovi informacije sa više mesta, koji onda treba da budu objedinjeni pre ubacivanja u obrazac ili bazu podataka. Zbog toga neki IE sistemi koriste poseban moduo za objedinjavanje rezultata.

Objedinjavanje delimičnih rezultata je prikazano u (Appelt, 1999; Hirschman 1992; Hobbs i sar. 1997; Surdeanu i Harabagiu, 2002). Interesantan je i primer sistema WHIES (Xu i Krieger, 2003), koji se sastoji od dva odvojena modula, jedan za popunjavanje obrazaca sa podacima (eng. *template filling*) i drugi za objedinjavanje delimično popunjenih obrazaca (eng. *template merging*). Nakon što se izvrši izdvajanje informacija i njihovo ubacivanje u obrasce, takvi delimično popunjeni obrasci se objedinjavaju. Pomenuti sistem je vršio ekstrakciju informacija pomoću dve različite metode, pa je u okviru njega prvo vršeno objedinjavanje na nivou rečenice (informacija izdvojena iz iste fraze, ali različitim metodama), a zatim i na nivou diskursa (ista informacija koja je izdvojena iz različitih fraza i rečenica).

Objedinjavanje rezultata uglavnom predstavlja vid unifikacije informacija. Način na koji će ono biti izvršeno zavisi od načina na koji izdvojena informacija treba da bude strukturirana, ali skoro uvek se vrši određeno upoređivanje informacija izdvojenih različitim metodama ili iz različitih delova teksta, a zatim i određeni vid normalizacije (svođenja na isti oblik). Prilikom poređenja informacija dolazi i do potrebe razrešenja koreferenci, što i postupak objedinjavanja čini veoma složenim.

Primer. Posmatrajmo primer teksta o nekom terorističkom napadu:

"*The bank was the target of the attack....The lobby was completely destroyed*."

Ukoliko se vrši izdvajanje informacija o događaju, tada bi sistem za isti događaj prepoznao dve grupe informacija. Iz prve rečenice, kao meta napada, bila bi izdvojena banka („*bank*“). Iz druge rečenice, kao prouzrokovana šteta, bilo bi izdvojeno potpuno uništeno predvorje („*completely destroyed lobby*“). Tek razrešenjem koreferenci, kojim bi se ustanovilo da se radi o predvorju banke na koju je izvršen napad, bilo bi moguće i objedinjavanje izdvojenih informacija u jedan slog podataka.

♦

3.3. Omotač (wrapper)

Omotač je poseban program koji prikuplja informacije iz različitih izvora, objedinjuje ih i unificira. Pa ipak, omotači se najčešće primenjuju samo za poslednja dva zadatka. Cilj omotača je da locira relevantnu informaciju u polustrukturiranim tekstovima i da ih prebaci u samo opisnu reprezentaciju (u smislu HTML označavanja) za dalje procesiranje (Kushmerick i sar. 1997.). Čini se da IE sistemi i omotači rade iste stvari, ali oblasti primene su potpuno različite. Osim toga, omotači ne koriste NLP tehnike u ovom procesu.

Najveća oblast primene omotača jeste veb, sa svojim neograničenim brojem veb sajtova koji većinom predstavljaju polustrukturirane podatke, te se omotači koriste upravo za izdvajanje tih podataka sa veb strana. Izdvajanje informacija je uglavnom zasnovano na analizi HTML oznaka i vizuelne reprezentacije veb strane, a ne na osnovu leksike ili sintakse jezika. U tom smislu, uspešnost omotača zavisi od same strukture veb strane. Razlike između strukture veb strana i frekvencija kojom se one menjaju čine posao kreiranja omotača netrivijalnim zadatkom. Ova činjenica dovodi do dva problema: generisanje omotača i održavanje omotača. Naime, generisanje omotača se uglavnom vrši na osnovu uvida u strukturu ciljanih veb strana. Međutim, promena strukture strane je moguća i tada omotači više nisu uspešni. Ovaj problem, kao i mogućnosti njegovog prevazilaženja su prikazani u (Chidlovskii, 2001.), na primeru omotača za veb sajt *Database and Logic Programming*[2] *(DBLP).*

[2] http://www.informatik.uni-trier.de/~ley/db/index.html

Omotači koji izdvajaju informacije sa veb strana na osnovu pravila ekstrakcije posebno popularni poslednjih godina, a neke od tehnika za generisanje pravila u oblasti ekstrakcije informacija se koriste kao metodi za indukciju omotača. Ove tehnike su se pokazale veoma uspešnim i za IE zadatke u pojedinim domenima, u situacijama kada su kolekcije veb strana generisane na osnovu jednog skripta, što je pokazano u (Muslea i sar., 1999.), (Kushmerick, 2000.), (Liu i sar., 2000) i (Baumgartner i sar., 2001.).

Uspešnost omotača zavisi od jedinstvenosti i kompletnosti formata strane i kvaliteta nivoa HTML-a.

Jedinstvenost i kompletnost formata strane podrazumeva određenu vrstu homogenosti među stranama, na primer kada zahtev za pretragom uvek vraća isti skup elemenata unutar rezultujuće veb strane. Na osnovu ove osobine, Kaiser i Miksch 2005. godine klasifikuju veb strane na:

- strane sa rigoroznom strukturom – jedinstven format i kompletna informacija, na primer AltaVista[3] uvek vraća HTML element *name* i *body*, čak i kod jako nestrukturiranih podataka

- strane sa polurigoroznom strukturom – jedinstven format i nekompletna informacija, na primer kada se u okviru strane pojavljuju stavke iz baze podataka, pri čemu se za svaku stavku vraća i lista atributa, ali ta lista nije ista za svaku stavku

- strane sa polurelaksiranom strukturom – nema jedinstvenog formata, ali je informacija kompletna

- relaksirana - nema jedisntvenog formata, niti kompletne informacije

Nivo kvaliteta HTML-a podrazumeva kvalitet samog HTML koda, tj. da li su sve oznake zatvorene, da li svaki atribut odgovara tačno jednoj oznaci ili je moguće da se u okviru jedne oznake nađe nekoliko atributa i sl.

Za generisanje omotača koriste se slične metode kao i za IE sisteme: ručno generisanje korišćenjem pravila pisanih od strane eksperata, polu-automatsko

[3] http://www.altavista.com

generisanje koristeći pristup baziran na mašinskom učenju i automatsko generisanje sa nenadgledanim učenjem.

3.4. Evaluacija sistema za ekstrakciju informacije

Tokom nekoliko MUC konferencija postignut je dogovor o tome kako bi trebalo izvršiti evaluaciju sistema za ekstrakciju informacije (Lehnert i sar. 1994.). Od tada, uspešnost IE sistema se meri pomoću dve vrednosti, preciznosti (engl. *precision*) i odziva (engl. *recall*). I jedna i druga vrednost se meri na osnovu veličine skupa informacija koje su ekstrahovane (S_{ext}) i skupa informacija koje se nalaze u tekstu (S_{rel}), i to na sledeći način:

$$precision = |S_{ext} \cap S_{rel}| / |S_{ext}|$$

$$recall = |S_{ext} \cap S_{rel}| / |S_{rel}|$$

Ovde treba napomenuti da skupovi S_{ext} i S_{rel} u stvari predstavljaju jezike L_{prep} i L_{rel}, definisane u poglavlju 2.7. Na taj način, preciznost meri količinu koliko je korektnih, relevantnih podataka među ekstrahovanim podacima. Sa druge strane, odziv meri količinu ekstrahovanih relevantnih informacija u odnosu na sve relevantne informacije u tekstu. Tako na primer, ukoliko se u tekstu nalazilo 100 informacija koje bi trebalo ekstrahovati, a sistem je ekstrahovao 80 informacija, od čega samo 60 relevantnih, tada bi odziv bio 60%, a preciznost 75%. Preciznost i odziv u većini slučajeva ne mogu da budu tačno izračunati, jer uglavnom nije poznata tačna količina relevantnih informacija u tekstu. Zbog toga se obično vrši procena ovih vrednosti na osnovu vrednosti dobijenih iz nekog skupa tekstova namenjenih za evaluaciju.

Iako su i preciznost i odziv veoma važni sa stanovišta uspešnosti IE sistema, često nije moguće optimizovati oba parametra istovremeno. Uglavnom modifikacije na IE sistemu koje dovode do povećanja odziva, smanjuju preciznost i obrnuto. U zavisnosti od namene sistema, ponekad će biti data veća važnost jednom od ova dva kriterijuma. Tako na primer u slučajevima kada će se ekstrahovani podaci koristiti za neka dalja istraživanja veoma je važno da te

informacije budu tačne, čak i po cenu da veći deo informacija ostane neprepoznat u tekstu od strane IE sistema.

S obzirom da će ipak često biti potrebno proceniti uspešnost IE sistema u odnosu na obe vrednosti, ustanovljena je još jedna metrika koja omogućava poređenje različitih IE sistema. Ova mera se naziva F mera. Ona koristi težinske koeficijente dodeljene odzivu i preciznosti pomoću kojih je moguće dati veći ili manji značaj jednom od parametara u odnosu na drugi.

$$F = \frac{(\beta + 1) \cdot P \cdot R}{\beta \cdot P + R}$$

pri čemu je P preciznost, R odziv, a β težinski koeficijent.

F mera predstavlja geometrijsku sredinu između odziva i preciznosti. Pomoću parametra β moguće je dodeliti veći značaj odzivu ili preciznosti. Ukoliko je $\beta = 1$, data je podjednaka važnost i preciznosti i odziva. Ako je $\beta > 1$, značajnija je preciznost; ako je $\beta < 1$, značajniji je odziv.

3.5. Pregled postojećih sistema za ekstrakciju informacija

U okviru ovog poglavlja biće dat pregled IE sistema koji koriste pravila ekstrakcije zasnovana na modelima konačnih stanja, najčešće konačnim transduktorima ili rekurzivnim mrežama prelaza. Ova pravila mogu da budu generisana od strane ljudi - eksperata, metodima mašinskog učenja ili kombinacijom ova dva pristupa.

U ranoj fazi razvoja IE oblasti, većina sistema je koristila generisanje pravila ekstrakcije od strane ljudi - eksperata. Međutim, s obzirom da je proces ručnog generisanje pravila ekstrakcije često glomazan i vremenski zahtevan, istraživanja u oblasti ekstrakcije informacije su kasnije usmerena u pravcu automatizacije ovog procesa. Automatizaciju je moguće izvršiti na dva načina: 1.) nadgledanim učenjem, gde su neophodni veliki skupovi podataka za treniranje koji indukuju pravila koristeći tehnike mašinskog učenja i 2.) nenadgledanim

učenjem, gde se pravila formiraju na osnovu malog skupa inicijalnih pravila i anotiranih korpusa.

3.5.1. Generisanje pravila ekstrakcije od strane eksperata

Jedan od prvih sistema za ekstrakciju informacija kod kojih su pravila ekstrakcije bila pisana od strane eksperata je *FASTUS* (Hobbs i sar. 1992.; Appelt i sar. 1993; Hobbs i sar. 1997). Reč *FASTUS* je akronim za *Finite State Automaton Text Understanding System*. Pa ipak, ovaj sistem se na bavi razumevanjem teksta, već ekstrakcijom podataka. Baziran je na kaskadnim, nedeterminističkim konačnim automatima.

U okviru FASTUS sistema ekstrakcija informacija se obavlja u nekoliko faza. Prvo se vrši prepoznavanje vlastitih imena i složenica, zatim prepoznavanje osnovnih fraza (rečenice se dele u imeničke i glagolske grupe) i na kraju prepoznavanje kompleksnih imeničkih i glagolskih fraza. Na osnovu izdvojenih tekstualnih sekvenci kreiraju se strukture koje opisuju pojedine entitete i događaje. Na kraju se vrši i objedinjavanje informacija izdvojenih iz različitih delova teksta, ukoliko se odnose na isti entitet ili događaj. Na ovaj način, sistem analizira i obrađuje sve veće segmente teksta. Autori ističu upravo pristup podele na nivoe prepoznavanja kao bitnu osobinu sistema FASTUS, koja je dovela do velike preciznosti.

FASTUS je bio primenjen u okviru MUC-4 konferencije, kada su domen primene IE sistema bili novinski članci sa informacijama o terorističkim incidentima. Objedinjavanje informacija je vršeno za incidente u okviru iste rečenice, kao i incidentima iz sledećih rečenica, sve dok ne dođe do neslaganja tipa incidenta, datuma ili lokacije. Sam sistem je bio relativno mali, iako je koristio veoma velike rečnike. Ručno kreirana pravila ekstrakcije su bila veoma efikasna.

GE NLTOOLSET (Krupka i sar. 1992.) je IE sistem koji koristi alate zasnovane na znanju, nezavisne od domena. Ekstrakcija informacija se sastoji iz tri koraka, pre-procesiranje, lingvistička analiza i post-procesiranje. U prvom koraku tekst se deli na manje delove pri čemu se ti delovi filtriraju tako da se

izbace nerelevantni delovi teksta. Istovremeno se vrši identifikacija fraza koje odgovaraju određenim obrascima i obeležavaju se delovi teksta koji bi mogli da nose određenu informaciju. U drugom koraku vrši se parsiranje i semantička interpretacija. U trećem koraku vrši se odabir obrazaca i mapiraju se semantičke kategorije koje odgovaraju tim obrascima. Baza znanja sistema sastoji se od leksikona sa opisanim konceptima i gramatičkim pravilima. Leksikon sadrži oko 10000 stavki od kojih je samo mali broj ograničen na domen. Gramatičko jezgro leksikona se sastoji od oko 170 pravila.

PLUM (*Probabilistic Language Understanding Model*) (Ayuso, 1992.), tj. njegova verzija korišćena u okviru MUC 4 primenjivala je ručno generisana pravila. Arhitektura sistema sadrži moduo za pre-procesiranje, morfološku analizu, moduo za parsiranje, semantički interpreter, moduo za procesiranje diskursa i moduo za generisanje šablona. Unutar modula za preprocesiranje utvrđuje se granica poruke, identifikuje se zaglavlje poruke, kao i granice paragrafa i rečenice. Moduo za morfološku analizu dodeljuje informacije o vrsti reči i to koristeći tehnike zasnovane na treniranju i probablilističkim metodima. Moduo za parsiranje generiše jedan ili više fragmenata koji se ne preklapaju a koji obuhvataju neku rečenicu. Ovi fragmenti se nakon toga obrađuju pomoću semantičkog interpretera. Ovaj moduo koristi semantičke komponente, kao što su različite leksičke semantike i semantička pravila. Leksičke semantike su konstruisane od strane procedure za automatsko indukovanje slučajeva, tj. mogućih fraza reči, kojom se navodi semantički tip reči i predikati koji idu uz nju. Semantička pravila su uopšteni sintaksni obrasci. Njihov glavni element su semantičke forme, koje mogu biti ili entiteti određenog domena, događaji ili stanje stvari. Entiteti se odnose na ljude, mesta, stavri i vremenske intervale nekog domena i prikazani su imeničkim frazama. Događaji (ko je uradio šta kome) i stanje stvari može biti opisano određenim klauzulama. Moduo za procesiranje diskursa konstruiše objekte događaja koji odgovaraju događajima u poruci. Zato on mora da uključi indirektne veze koje nisu eksplicitno pronađene od strane interpretera i razreši refernce u tekstu. Generator šablona zatim koristi strukturu kreiranu od strane modula za procesiranje diskursa, kako bi kreirao konačne šablone.

PROTEUS (Yangarber i Grishman, 1998.) je sistem za ekstrakciju informacija koji se sastoji od sedam modula: (1) leksička analiza (2) prepoznavanje imena, (3) delimična sintaksna analiza, (4) analiza obrasca scenarija, (5) razrešenje referenci, (6) analiza diskursa i (7) generisanje izlaznih podataka. Moduo za leksičku analizu deli dokument na rečenice i tokene. Svakom tokenu je dodeljena stavka iz rečnika koja mu odgovara. Opciono, moduo za određivanje vrste reči može biti pokrenut kako bi se eliminisale manje verovatne stavke koje odgovaraju tokenu. Moduli za prepoznavanje imena, delimičnu sintaksnu analizu i obrasce scenarija koriste determinističko, od dna ka vrhu, delimično parsiranje ili uklapanje obrazaca. Kao obrasci koriste se regularni izrazi. Obrasci za prepoznavanje imena identifikuju vlastita imena. Moduo za delimičnu sintaksnu analizu pronalazi imeničke i glagolske fraze. Moduo za obrasce scenarija pronalazi sintaksne konstrukcije višeg nivoa. Moduo za razrešenje referenci povezuje zamenice sa entitetima koji im prethode i razrešuje problem koreferenci. Moduo za analizu diskursa kreira kompleksnu strukturu događaja koristeći pravila zaključivanja višeg nivoa. Na taj način je omogućeno da nekoliko klauzula sadrži informaciju o istoj kompleksnoj činjenici. Moduo za generisanje šablona transformiše prikupljene informacije u konačan šablon (strukturu). Prikupljanje obrazaca se odvija u nekoliko koraka. Prvo korisnik unosi rečenicu koja sadrži događaj i odabira događaj iz liste događaja. Zatim, sistem primenjuje trenutne obrasce na dati primer kako bi izveo inicijalnu analizu. Na taj način, identifikovane su grupe imenica i glagola i njihovi semantički tipovi i primenjena je minimalna generalizacija. Sistem predstavlja rezultate korisniku, koji onda može da modifikuje polazni obrazac, tj. da odabere odgovarajući nivo generalizacije, učini neki element opcionim, ukloni ga i sl. Zatim korisnik specificira na koji način će ekstrahovani elementi biti korišćeni za popunjavanje šablona, na osnovu čega sistem kreira novi obrazac koji odgovara izmenama i datom primeru.

3.5.2. Nadgledani sistemi

Nadgledano učenje koristi podatke namenjene za treniranje kako bi se indukovala pravila za ekstrakciju. Tako, skoro da nije potrebno znanje iz

određenog domena, već samo veliki skupovi podataka koji treba da budu anotirani u skladu sa strukturom informacije koja se prikuplja. Sa druge strane, priprema podatka za treniranje predstavlja usko grlo nadgledanih IE sistema. Većina sistema zahteva veliku količinu anotiranih dokumenata za određeni zadatak ekstrakcije, što dodatno dovodi do smanjene prilagodljivosti jednog IE sistema drugim zadacima ekstrakcije.

AutoSlog (Riloff, 1993.) je prvi sistem koji uči pravila ekstrakcije na osnovu primera namenjenih njegovom treniranju. On ekstrahuje rečnike određenog domena koji sadrže čvorove koncepata za ekstrakciju informacija iz teksta. Čvor koncepta je pravilo koje uključuje reč ili reč i okidače i semantička ograničenja. Ukoliko sistem nađe određeni okidač u tekstu i uslov naveden u čvoru koncepta je zadovoljen, taj čvor koncepta je aktiviran i na osnovu njegove definicije deo teksta je ekstrahovan iz konteksta. Dalje, sistem identifikuje rečenice anotirane alatima za popunjavanje slotova i semantičkim tagovima. Zatim, pretražuje se lista heuristika i pokušava da se pronađe ona koja odgovara izdvojenom delu teksta. Svaka heuristika izdvaja podatke za jedan atribut. *AutoSlog* takođe koristi semantički tager i semantička ograničenja unutar pravila ekstrakcije, ne objedinjuje slične koncepte i obrađuje samo slobodan tekst.

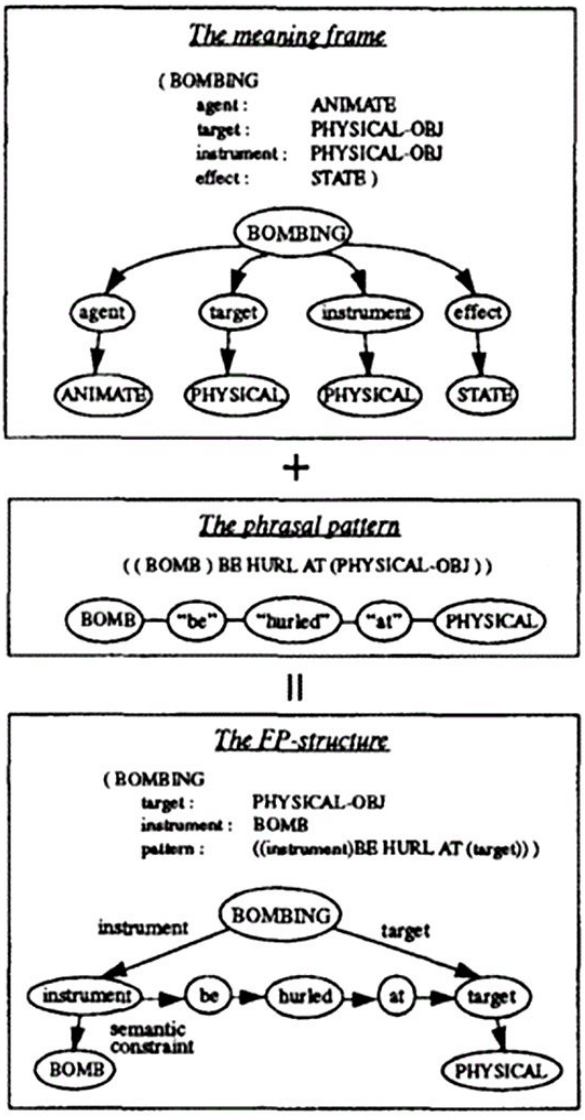
Na slici 18 prikazan je skup obrazaca (leva kolona) i skup fraza izdvojenih sistemom *AutoSlog* (desna kolona).

1. <subj> passive_verb	<subj> was acquired
2. <subj> active_verb	<subj> sold
3. <subj> active_verb dobj	<subj> saw profit
4. <subj> verb infinitive	<subj> wanted to sell
5. <subj> auxbe noun	<subj> is CEO
6. <subj> auxhave noun	<subj> has debt
7. active_verb <dobj>	sold <dobj>
8. infinitive <dobj>	to sell <dobj>
9. verb infinitive <dobj>	agreed to buy <dobj>
10. noun auxbe <dobj>	CEO is <dobj>
11. noun auxhave <dobj>	buyer has <dobj>
12. infinitive prep <np>	to sell for <np>
13. active_verb prep <np>	will sell for <np>
14. passive_verb prep <np>	was sold for <np>
15. noun prep <np>	owned by <np>

Slika 18. Fraze izdvojene sistemom *AutoSlog* (desna kolona) na osnovu heuristika (leva kolona) (Riloff i Phillips, 2004)

Obrasci su podeljeni u tri grupe, u zavisnosti da li se odnose na subjekat, direktni objekat ili predložnu frazu. U zavisnosti od toga koji deo teksta je izdvojen, primenjuju se heuristike iz odgovarajuće grupe.

PALKA sistem (Kim i Moldovan, 1995.) proizvodi pravila za ekstrakciju kao parove okvira značenja i frazalnih obrazaca, nazvanih FP struktura (eng. *FP-structure* u značenju *Frame-Phrasal pattern structure*). Slika 19 prikazuje primer FP strukture.



Slika 19. Formiranje FP strukture (Kim i Moldovan, 1995.)

Ukoliko postojeća pravila ne mogu da budu primenjena, *PALKA* kreira novo pravilo i pokušava da ga generalizuje sa postojećim kako bi uključio nove

pozitivne instance. On prilagođava postojeća pravila kako bi izbegla negativne instance, ako sistem generiše pogrešnu interpretaciju prilikom primenjivanja pravila. Nova pravila se generišu na osnovu korisnikovog izbora. Naime, deo teksta za koji nije pronađeno odgovarajuće pravilo šalje se korisniku, koji ručno bira koji deo teksta treba i na koji način da bude strukturiran. Na osnovu korisnikovog označavanja, sistem PALKA kreirea novu FP strukturu kojom bi ta sekvenca teksta mogla da bude obrađena. Glavna strategija prilikom kreiranja novog pravila jeste da se u pravilo uključe svi elementi koje je korisnik označio, kao i glavni glagol rečenice, a da se izostave svi neoznačeni elementi. Pri tome se svi označeni elementi zamenjuju njihovim semantičkim kategorijama. Takođe, ukoliko je korisnik označio jednu imenicu u imeničkoj frazi kao jedan element, cela grupa (fraza) se zamenjuje tim elementom. Svi glagoli se zamenjuju svojom glagolskom osnovom.

CRYSTAL sistem (Soderland i sar., 1995.) obrađuje tekstove koji su već obrađeni sintaksnim parserima. Potrebni su mu dokumenti za treniranje, anotirani od strane eksperata, kao i semantička hijerarhija. *CRYSTAL* počinje proces učenja kreirajući pravila za ekstrakciju za svaku instancu ciljnog događaja u podacima za treniranje. Koristeći pristup induktivnog učenja, sistem pronalazi najsličniji par ili pravila i objedinjuje ih zajedno pronalazeći najstroža ograničenja koja obuhvataju oba pravila.

LIEP sistem (Huffman, 1996.) koristi heuristike slično kao i *AutoSlog*, ali uči pravila za izdvajanje vrednosti više atributa i nije u stanju da obradi ekstrakciju jednog sloga. On dozvoljava korisniku da interaktivno identifikuje događaje u tekstu, pri tom pretpostavljajući da je teško formirati velike anotirane korpuse za treniranje. Za svaku potencijalnu rečenicu za treniranje, entiteti od interesa (ljudi, kompanije, i sl.) se identifikuju. *LIEP* pokušava da odabere pravilo za ekstrakciju koje će imati maksimalan broj ekstrakcija pozitivnih primera i minimalan broj negativnih ekstrakcija. Ukoliko novi primer ne odgovara ni jedonom postojećem obrascu, *LIEP* pokušava da generalizuje poznata pravila kako bi obuhvatio primer. Ukoliko generalizacija nije moguća ili dobijeni obrazac smanjuje kvalitet, konstruiše se novi obrazac.

Sistem *WHISK* (Soderland, 1999.) koristi algoritme mašinskog učenja poznate kao algoritmi pokrivanja ili obuhvatanja., kako bi indukovao regularne izraze indukcijom od vrha na dole. Počinje sa najopštijim pravilima koje nastavlja da specijalizuje. Proces se zaustavlja kada skup pravila generisanih na ovaj način može da obuhvati sve pozitivne primere za treniranje. Nakon toga se obavlja redukcija broja pravila, uklanjanjem onih koja generišu preklapanje rezultata.

RAPIER sistem (Calif i Mooney, 1999; Calif i Mooney, 2003) koristi parove dokumenata i popunjava šablone kako bi indukovao pravila za poklapanje obrazaca koji direktno izdvajaju vrednosti određenih atributa u šablonu. U tu svrhu sistem koristi algoritam za učenje od dna prema vrhu kako bi ograničio pretragu bez nametanja veštačkih granica konstantama, i kako bi povećao preciznost dajući prednost određenijim pravilima. Parovi pravila se slučajno biraju i vrši se pretraga kako bi se našla najbolja generalizacija ta dva pravila, uzimajući najmanje opštu generalizaciju, dodavajući ograničenja dok pravilo pavilno ne obradi sve podatke za treniranje. *RAPIER* može da obradi samo ekstrakciju jednog atributa iz polustrukturiranih tekstova.

GATE (Cunningham i sar. 2002) je grafičko razvojno okruženje koje omogućava korisnicima da razviju komponente i resurse za procesiranje prirodnih jezika. Skup resursa namenjenih za procesiranje je zajedno upakovan u sistem nazvan *ANNIE* (*A Nearly-New IE* sistem). Ovi resursi mogu da budu korišćeni individualno ili u kombinaciji sa novim modulima. *ANNIE* se sastoji od tokenizatora, modula za rastavljanje na rečenice, obeleživača vrste reči, rečnika imena, semantičkog obeleživača, i modula za razrešenje koreferenci. Tokenizator, moduo za rastavljanje na rečenice i obeleživač vrste reči funkcionišu slično kao kod ostalih IE sistema. Semantički obeleživač se sastoji od ručno pisanih pravila, koja opisuju obrasce čije poklapanje se traži i anotaciju koja se kreira kao rezultat. Baziran je na konačnim transduktorima. Rečnik imena sadrži liste imena gradova, organizacija, dana u nedelji itd. Moduo za razrešenje koreferenci pronalazi koreference ili entitete, tako što prepoznaje relacije između entiteta. *GATE* pruža jednostavan i lako proširiv alat za anotaciju teksta kako bi se podaci anotirali za NLP algortime. Anotacija može biti izvršena ručno od strane korisnika ili poluautomatski pomoću nekog od resursa nad korpusom, a zatim korigujući i

dodavajući nove anotacije ručno. Zavisno od informacije koja treba da anotira, neki moduli u okviru *ANNIE* mogu biti korišćeni za zadatak anotacije korpusa. *GATE* omogućava razvoj NLP aplikacija na fleksibilan i proširiv način. Odlikuje ga robustnost, pouzdanost, višestrukost korišćenja i proširivost.

(LP)^2 (Ciravegna 2001) je alat namenjen za ekstrakciju informacija sa veb strana koji indukuje simbolička pravila iz korpusa označenih SGML oznakama. Indukcija se izvodi na osnovu atomske generalizacije (od dna ka vrhu) na osnovu primera u korpusu za treniranje. Treniranje se obavlja u dva koraka. Inicijalno se utvrđuju pravila za označavanje, a u drugom koraku se indukuju nova pravila kako bi se korigovale greške i nepreciznosti. Algoritam se u trenutku objavljivanja pokazao boljim od svih dotadašnjih u smislu efektivnosti, smanjenja vremena potrebnog za treniranje i veličine korpusa.

3.5.3. Nenadgledani sistemi

Nenadgledani sistemi od korisnika preuzimaju jedino zahtev za određenom informacijom. Dalja interakcija korisnika i sistema ne postoji, tj. korisnik ne vrši bilo kakvu intervenciju ili podešavanje pravila ekstrakcije. Nikakvi obrasci se ne daju korisniku unapred. Glavni izazov je korisnikovu potrebu za informacijom pretvoriti u niz obrazaca za ekstrakciju. Ovi sistemi su bazirani na samorazvijajućim (engl. *bootstrapping*) metodima, tj. metodima koji su u stanju da praktično „od nule“ ili od inicijalno malog skupa obrazaca ekstrakcije razviju kolekciju pravila i na taj način sami izgrade sistem.

AutoSlog-TS (Riloff, 1996; Riloff i Philips, 2004) predstavlja proširenje sistema *AutoSlog*. *AutoSlog-TS* zahteva jedino postojanje korpusa koji je unapred klasfikovan u odnosu na relevantnost svakog dokumenta za određeni proces ekstrakcije. Sistem generiše obrasce za ekstrakciju za svaku imeničku frazu u korpusu za treniranje različitim heuristikama. Veliku novinu u odnosu na sistem *AutoSlog* predstavlja mogućnost korišćenja više pravila ukoliko se više njih poklapa sa kontekstom i na taj način omogućava generisanje višestrukih pravila. Statističkim metodima se otkriva koji obrazac je pouzdan za domen. U drugoj fazi vrši se evaluacija ovih obrazaca. Na taj način, sistem procesira korpus po drugi

put i generiše statistiku relevantnosti za svaki obrazac, na osnovu koje se oni kasnije rangiraju.

Riloff i Jones (1999) su predložili algoritam za treniranje koristeći uzajamno pokretanje (*mutual bootstrapping*) za otkrivanje informacija. Ovaj algoritam uči obrasce ekstrakcije informacije i istovremeno generiše imeničke fraze koje pripadaju nekoj semantičkoj klasi. Polazeći od nekoliko inicijalnih imenica koje pripadaju ciljanoj semantičkoj klasi, algoritam uzajamnog pokretanja iterativno uči nove obrasce ekstrakcije, a zatim na osnovu njih pretpostavlja koje nove imenice pripadaju datoj semantičkoj klasi. Ovako naučeni obrasci su više slični obrascima za prepoznavanje imenovanih entiteta nego obrascima za prepoznavanje događaja. Kasnije, Philips i Riloff (2007) su pokazali da metodi uzajamnog pokretanja mogu biti korišćeni za obrasce događaja koristeći imenice koje identifikuju neku od uloga u događaju kao inicijalne. Imenice koje identifikuju neku od uloga u događaju su reči koje identifikuju ulogu koju neka imenica ima u nekom događaju. Na primer, imenica "kidnaper" identifikuje učesnika događaja kidnapovanja. Koristeći ovakve imenice kao inicijalne, Basilisk algoritam (Thelen i Riloff, 2002) je korišćen i za obrasce ekstrakcije događaja i za imenice koje identifikuju uloge. Kod algoritama uzajamnog pokretanja mala početna greška može da dovede do velikog broja grešaka tokom iteracija.

EXDISCO (Yangarber i sar, 2000; Yangarber, 2003.) takođe primenjuje strategiju uzajamnog pokretanja. Ova strategije je zasnovana na pretpostavci da prisustvo relevantnih dokumenata znači da je obrazac dobar, kao i da dobri obrasci mogu da nađu relevantne dokumente. Polazeći od neoznačenog korpusa i malog broja inicijalnih obrazaca, skup dokumenata se deli na skup relevantnih dokumenata koji sadrže barem jednu instancu obrasca, i skup nerelevantnih dokumenata koji ne sadrže ni jedan od inicijalnih obrazaca. Nakon toga, kandidati-obrasci se generišu iz rečenica i izjava u dokumentu i rangiraju u odnosu na relevantne dokumente. Obrazac sa najvišim rangom se dodaje u skup obrazaca i svaki dokument se ponovo rangira na osnovu novodobijenog skupa obrazaca. Skup dokumenata se opet deli na dva podskupa i nastavlja sa iteracijama.

Sistem *Snowball* (Agichtein, 2000) je baziran na DIPRE algoritmu (Brin, 1998), koji ima izuzetno dobre rezultate sa podacima koji sadrže odnose između dva entiteta (na primer, imena autora i naslove knjiga). DIPRE algoritam polazi od inicijalnih parova entiteta koji su u poznatom odnosu. Na osnovu poznatih informacija obeležava se tekst namenjen za treniranje, a zatim se indukuju obrasci koji opisuju obeležena pojavljivanja. Obrasci se zatim primenjuju na neoznačen tekst, da bi se izdvojili novi parovi entiteta (autor, knjiga), koji se zatim dodaju inicijalnom skupu parova. Postupak se ponavlja sve dok se ne zadovolji neki konvergencioni kriterijum.

Dakle, polazeći od inicijalnih instanci relacija i uopštenog regularnog izraza koji entiteti treba da zadovolje, *Snowball* generiše obrasce iz tekstualnih dokumenata. Ključno unapređenje u odnosu na DIPRE je što obrasci *Snowball* sistema uključuju oznake imenovanih entiteta. Obrasci se generišu klasterovanjem sličnih parova, korišćenjem algoritma za klasterovanje u jednom prolazu. Nakon što su obrasci generisani, sistem otkriva nove parove koji odgovaraju obrascima do određenog nivoa poklapanja. Ova informacija pomaže sistemu da odluči koje parove će da uvrsti u finalni šablon.

Ekstrakcija informacija na osnovu upita (*The Query-Driven Information Extraction - QDIE*) (Sudo, 2004) pokušava da minimizira intervenciju od strane korisnika koristeći skup ključnih reči kao ulazne podatke. Sistem parsira dokument pomoću parsera zavisnosti i označivača imenovanih entiteta i vraća skup relevantnih dokumenata određenih korisnikovim upitom. Drvo zavisnosti između rečenica se koristi za obrasce ekstrakcije. Svako poddrvo rečenice koja odgovara modelu obrasca postaje obrazac-kandidat. *QDIE* računa relevantnost za svaki obrazac kandidat koristeći *tf-idf* (eng. *term frequency – inverse document frequency*) vrednost, poznatu u IR literaturi (Salton i Buckley, 1988). Naime, obrazac je više relevantan što se više pojavljuje u skupu relevantnih dokumenata i manje u celom skupu dokumenata.

(Stevenson i Greenwood, 2005) takođe koriste pristup u kome polaze od inicijalnog skupa obrazaca i koriste semantičku meru sličnosti da iterativno rangiraju i odaberu novi obrazac-kandidat na osnovu njihove sličnosti sa

inicijalnim obrascima. Pri tom koriste predikat-argument strukture kao reprezentaciju za IE obrasce, kao i (Surdeanu i sar.,2003) u ranijim radovima. (Sudo i sar., 2003) kreirali su čak i bogatiji model poddrveta kao reprezentaciju za IE obrasce, gde IE obrazac može biti proizvoljno poddrvo drveta zavisnosti. Obrasci poddrveta se otkrivaju na osnovu relevantnih i irelevantnih dokumenata namenjenih za treniranje.

Sistem KNOWITALL (Etzioni i sar. 2005) je sistem za ekstrakciju informacija sa veba koji uči kako da obeležava svoje sopstvene podatke za treniranje koristeći mali skup obrazaca za ekstrakciju, nezavisnih od domena. Ovaj sistem je u stanju i da obradi heterogene korpuse oslanjajući se na moduo za označavanje vrste reči umesto parsera. Nedostatak ovog sistema je što, sa druge strane, zahteva veliki broj upita sistemima za pretraživanje i veliki broj preuzimanja veb strana sa Interneta. Posledica toga jeste da eksperimentalni procesi ekstrakcije ovim sistemom mogu da traju i nedeljama.

TextRunner sistem (Banko i sar. 2007) je prvi prilagodljiv i nezavisan od domena *IE* sistem. Autori su ga predstavili kao novu paradigmu nazvanu „otvorena ekstrakcija informacija“ (eng. *Open IE – OIE*). Osobina otvorenih IE sistema je da samo jednim prolaskom kroz tekstualni korpus generišu veliki broj obrazaca za izdvajanje informacija (posebno relacija između entiteta), bez potrebe za ljudskom intervencijom. Namenjeni su prvenstveno za upotrebu nad vebom kao korpusom, i kao takvi, pokušavaju da obrade neograničen broj relacija i da se izvršavaju dovoljno brzo kako bi bili efikasni.

Sistem *TextRunner* izdvaja imena relacija kao i njihove argumente, tj. entitete koji su u relaciji. Svakom izdvojenom paru se dodeljuje verovatnoća i indeksiraju se kako bi se podržala efikasna ekstrakcija i iskorišćenje od strane korisnika putem upita.

Interesantan je i sistem WOE (*Wikipedia-based Open Extractor*) opisan u (Lange, 2007), prvi koji generiše primere za treniranje karakteristične za pojedine relacije tako što uparuje atribute iz Infoboksa sa Wikipedia veb strane sa odgovarajućim rečenicama. Preciznost i odziv ovog sistema zavisi od dodatnih

modula koji se koriste za obradu teksta, čijim odabirom je moguće dati na značaju brzini sistema nad preciznošću, ili obratno, u zavisnosti od potrebe.

3.5.4. IE sistemi višeg semantičkog nivoa

Većina pomenutih sistema ima prilično lokalizovan pristup procesu ekstrakcije informacija, tj. uzima u obzir samo manji deo konteksta u kome se neki entitet pojavljuje u tekstu. Takvi obrasci za ekstrakciju analiziraju samo lokalni kontekst koji okružuje neki tekst da bi doneli odluku o ekstrakciji. U poslednje vreme sve je veći broj sistema koji pokušavaju da vrše analize većih delova teksta, obično čitavih rečenica, pa često i nekoliko rečenica teksta koje se odnose na istu temu, tj. pripadaju istom diskursu.

Ovakvi sistemi su prvobitno pristupali problemu ekstrakcije sa stanovišta istovremenog pojavljivanja entiteta (Xiao i sar. 2004) i sa stanovišta relacije zavisnosti između entiteta (Zhang i sar., 2006). Otkrivanje zavisnosti omogućava ekstrakciju korektnih eniteta i relacija na nivou izjave (klauzule). Maslenikov i sar. (2006) pokazuju da povećanje dužine putanje neke relacije dovodi do značajnog smanjenja performansi, najčešće zato što postoji mogućnost da entiteti pripadaju različitim klauzulama. S obzirom da su klauzule u rečenicama povezane klauzalnim relacijama (Halliday i Hasan, 1976), pokazuje se da analiza diskursa rečenice može da doprinese poboljšanju rada IE sistema. Tako su Taboada i Mann (2005) pokazali da analiza diskursa pomaže u dekompoziciji dugačkih rečenica na klauzule, a time dalje i u razdvajanju relevantnih od nerelevantnih klauzula. Miltsakaki (2003) tvrdi da odnos klauzula (glavna i zavisna) odražava i odnos entiteta koji se u njima pojavljuju. Takođe, određena znanja o strukturi teksta pomažu da se pravilno interpretira i odredi značenje entiteta u tekstu (Grosz i Sidner, 1986). Ovo je posebno značajno kod tekstova kao što su sudske odluke ili novinski članci, kod kojih segmenti diskursa imaju tendenciju da budu u nepromenljivom redosledu (Moens i sar. 2002).

U dve studije (Gu i Cercone, 2006; Patwardhan i Riloff, 2007) prikazani su IE sistemi koji koriste klasifikatore da prvo identifikuju rečenice relevantne za neki događaj u dokumentu, a zatim primenjuju IE metode samo na te rečenice.

Finkel i sar. (2005) primenjuju penale u okviru modela za učenje kako bi obezbedili konzistentnost oznaka između informacija ekstrahovanih iz različitih delova teksta. Maslenikov i Chua (2007) koriste zavisnost i teoriju retoričke strukture (Taboada i Mann, 2005) za analizu diskursa kako bi povezali entitete u različitim izjavama i utvrdili relacije između entiteta koji su u tekstu udaljeni jedni o drugih.

3.6. Programski sistem UNITEX u ekstrakciji informacija

Za obradu teksta i pisanje i primenu pravila ekstrakcije u okviru ove disertacije korišćen je softverski alat UNITEX.

UNITEX *(Paumier, 2011)* je kolekcija programa razvijenih za analizu teksta na prirodnim jezicima korišćenjem lingvističih resursa i alata. Resursi se sastoje od elektronskih rečnika i gramatika, posebno razvijenih za pojedine jezike. Ovaj sistem je otvorenog koda (engl. *open source*). Dizajniran je tako da bude prenosiv, tj. da je moguće njegovo izvršavanje na različitim operativnim sistemima, kao što su *Windows, Linux, MacOS* i drugi. Programi u okviru UNITEX-a su pisani na programskim jezicima C/C++, dok je grafički korisnički interfejs pisan u programskom jeziku JAVA. UNITEX je projektovan tako da može da podržava različite prirodne jezike.

3.6.1. Lingvistički resursi – elektronski rečnici i gramatike

Sama upotreba UNITEX sistema počinje upravo odabirom jezika koji će se koristiti. Ovaj odabir jezika utiče na dve bitne stvari u okviru dalje obrade teksta, a to su elektronski rečnici koji će se koristiti i gramatike za tokenizaciju i normalizaciju tekstova u fazi predprocesiranja.

Elektronski rečnici sadrže reči i složenice nekog jezika zajedno sa njihovim lemama i skupom gramatičkih (semantičkih i flektivnih) kodova. Upravo korišćenje ovakvih rečnika predstavlja glavnu prednost sistem UNITEX nad drugim sistemima za pretragu i analizu teksta, s obzirom da informacije koje se

nalaze u njima omogućavaju definisanje čitavih klasa reči i izraza korišćenjem veoma jednostavnih obrazaca.

Gramatike koje se koriste unutar UNITEX sistema reprezentuju lingvitičke fenomene na osnovama rekurzivnih mreža prelaza, predstavljenih grafovima koji se veoma jednostavno kreiraju i prilagođavaju od strane korisnika.

Posle odabira jezika, pristupa se obradi samog teksta. UNITEX zahteva da tekst koji treba da se obradi bude u *Unicode Little Endian 16bit* (UTF16LE) formatu[4]. U okviru UNITEX-a postoje Java klase pomoću kojih je moguće izvršiti konverziju teksta, ukoliko format nije adekvatan (klasa *fr.umlv.unitex.io.UnicodeIO).*

Kada je tekst snimljen u odgovarajućem formatu, pristupa se njegovoj pripremi. Prvo se vrši normalizacija separatora teksta. Standardni separatori su razmak, tabulator i novi red. U tekstu je moguće da se pojavi nekoliko separatora jedan za drugim. Ukoliko raspored separatora nije bitan za leksičku analizu, prilikom normalizacije se niz uzastopnih separatora zamenjuje jednim separatorom, i to novim redom ukoliko je barem jedan od separatora u nizu oznaka za novi red, ili razmakom ukoliko se u nizu separatora ne pojavljuje oznaka za novi red. Međutim, ukoliko je raspored separatora važan za proces ekstrakcije informacije (omogućava određivanje značenja neke informacije), normalizaciju separatora je moguće izostaviti. Sa druge strane, moguće je izvršiti i normalizaciju teksta u drugom smislu, po nekom proizvoljnom pravilu (na primer, za uklanjanje elizija, grafom prikazanim na slici 12 u okviru poglavlja 3.2.1).

Zatim se pristupa tokenizaciji teksta. S obzirom da separatori leksičkih jedinica mogu biti različiti u različitim jezicima, UNITEX vrši tokenizaciju na način specifičan pojedinom jeziku, pa se na primer za tajlandski jezik tokenizacija vrši karakter po karakter.

Posle faze tokenizacije kreirana je lista svih tokena koji su se pojavili u tekstu. Takođe, tekst je proširen ubacivanjem separatora rečenica. Na ovako pripremljen tekst dalje se primenjuju elektronski rečnici, tj. određuje se podskup reči iz rečnika koje se pojavljuju u tekstu.

[4] *The Unicode Standard,* http://www.unicode.org

Elektronski rečnici u UNITEX-u su u DELA formatu i konstruisani su od strane timova lingvista za odgovarajući jezik (za engleski jezik (Chrobot i sar., 1999; Klarsfeld i Hammany-Mc Carthy, 1991; Monceaux, 1995; Savary, 2000), za francuski jezik (Courtois, 1996; Courtois i Silberztein, 1990; Labelle, 1995), za srpski jezik (Krstev i Vitas, 2005; Vitas i sar., 2003)). DELA sintaksa opisuje proste i složene reči pridruživanjem koda koji opisuje flektivna, sintaksna i semantička svojstva reči. Svaka reč ili složena forma je opisana u jednoj liniji elektronskog rečnika, tj. predstavlja jednu stavku u rečniku. U zavisnosti od toga da li se čuvaju reči u izmenjenom obliku ili samo njihove leme, razlikujemo dve vrste DELA rečnika, DELAF i DELAS.

DELAF format je rečnik oblika reči, u kome jedna stavka (jedan ulaz) izgleda ovako:

`lekara,lekar.N+Hum+Ek:ms2v:ms4v:mp2v`

Navedena linija elektronskog rečnika je formirana od nekoliko delova, sa sledećim značenjem:

`lekara` – izmenjen oblik reči; ova informacija je obavezna

`lekar` – odgovarajuća lema (može da bude izostavljena)

`N+Hum+Ek` – sekvenca kodova koji određuju vrstu reči i njena gramatička svojstva, npr. **N** označava da se radi o imenici (*Noun*), **Hum** da se pojam odnosi na osobu, **Ek** da se radi o ekavskom izgovoru

`ms2v:ms4v:mp2v` – sekvenca kodova koja određuje odnos oblika reči i leme, tj. opisuju njen rod, broj, deklinaciju, konjugaciju i sl. Na primer, kod **ms2v** bi značio da se radi o imenici muškog roda (**m**-masculine), u drugom padežu jednine (**s**-singular), dok slovo **v** označava animalnost (v-živo biće).

DELAS format je veoma sličan formatu DELAF, s tom razlikom što se u ovu vrstu rečnika upisuju samo leme reči. Na primer,

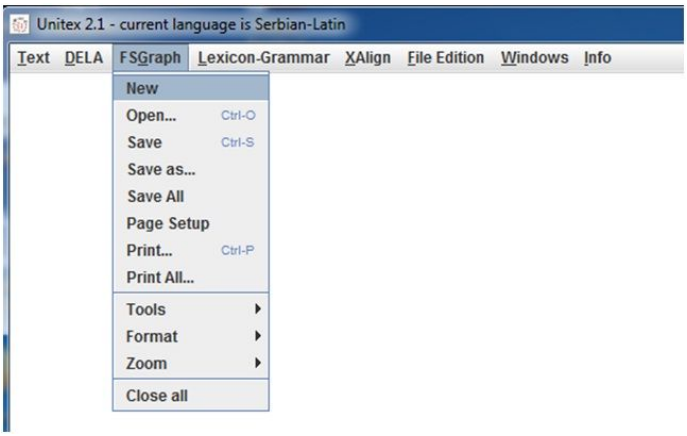
`konj, N4 + Zool`

Prvi gramatički kod (**N4**) određuje koja gramatika će biti korišćena za promenu reči, dok **Zool** označava da se radi o životinji.

Nakon primene rečnika na tekst, UNITEX sistem kreira tri pripadajuće datoteke, jednu sa prostim rečima koje se pojavljuju u tekstu, drugu sa složenim rečima i treću sa neprepoznatim rečima (Vitas, 2007). Ovako pripremljen tekst moguće je dalje pretraživati regularnim izrazima ili grafovima, ili obrađivati na neki drugi načina, na primer primenom pravila ekstrakcije.

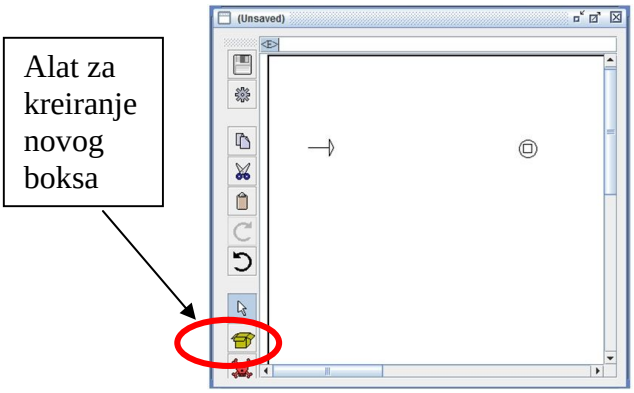
3.6.2. Grafički korisnički interfejs za kreiranje grafova

UNITEX ima veoma dobro razvijen, jednostavan za korišćenje i intuitivan grafički interfejs namenjen za kreiranje grafova. Kreiranje grafa započinje odabirom opcije *New* iz menija *FSGraph* (slika 20).



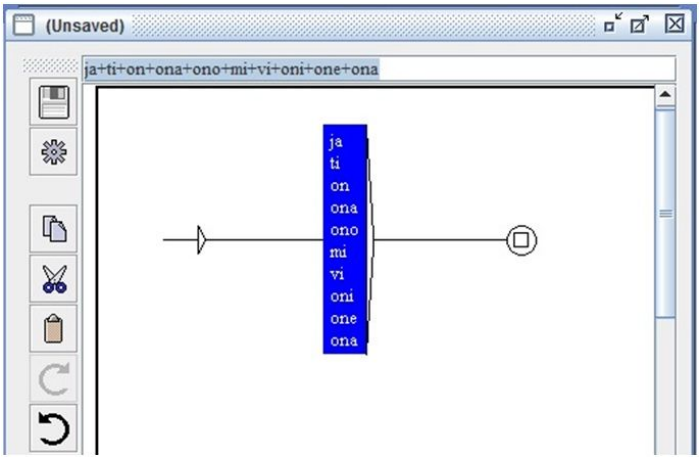
Slika 20. Meni *FSGraph* programskog sistema UNITEX

Nakon odabira ove opcije otvara se prozor u kome je moguće kreirati graf (slika 21).



Slika 21 . Novokreirani graf sa početnim i završnim stanjem. Sa leve strane prozora nalazi se paleta alata za rad sa grafovima

Simbol u obliku strelice (slika 21) predstavlja inicijalno stanje grafa. Okrugli simbol sa kvadratom predstavlja završno stanje grafa. Sa leve strane prozora nalazi se paleta alata za rad sa grafom. Izborom simbola “Box” iz palete alata i pritiskom levog tastera miša bilo gde na radnoj površini unutar prozora grafa kreira se boks, koji odgovara prelazu grafa sa jednog stanja na drugo. Inicijalno, novokreirani boks sadrži simbol <E>, što odgovara praznoj reči (ε).



Slika 22. Kreiranje boksa koji sadrži više reči po kojima je moguć prelaz iz jednog stanja u drugo

Tekstualno polje na vrhu prozora omogućava da se u boks unesu različite oznake i simboli. Tako je, na primer, moguće uneti “ja+ti+on+ona+ono+mi+vi+oni+one+ona” (slika 22). Pritiskom levog tastera miša na stanje grafa, a zatim na boks kreira se veza između stanja i boksa. Graf kreiran na taj način bi prepoznavao nominativ ličnih zamenica u srpskom jeziku. Ovaj graf odgovara regularnom izrazu “ja|ti|on|ona|ono|mi|vi|oni|one|ona”.

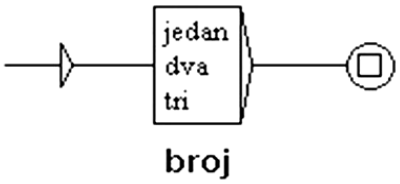
Unosom različitih vrednosti u boks dobijaju se i različiti prelazi, a time i različite vrste grafova. U jednom boksu moguće je uneti sledeće vrste oznaka:

- leksičke maske – označavaju se simbolima < i >. Mogu da se odnose na referencu iz rečnika ili da sadrže specijalne simbole. Tako bi upit `<pevati.V>` odgovarao svim oblicima glagola čija je lema reč *pevati*. Specijalnih simbola ima više, a neki od njih su <TOKEN> (bilo koji token), <MOT> (reči sastavljene samo od slova), <MAJ> (reči pisane velikim slovima), <NB> (neprekidne sekvence cifri) i drugi.

- morfološki filteri – označavaju se simbolima << i >>, unutar kojih se navode regularni izrazi koji opisuju neki token. Tako su mogući filteri `<<izam$>>` (sve reči koje završavaju na `izam`), `<<^a>>` (reči koje počinju sa `a`), `<<a.*s>>` (reči koje sadrže karakter `a`, a zatim bilo koju sekvencu karaktera praćenu karakterom `s`) itd.

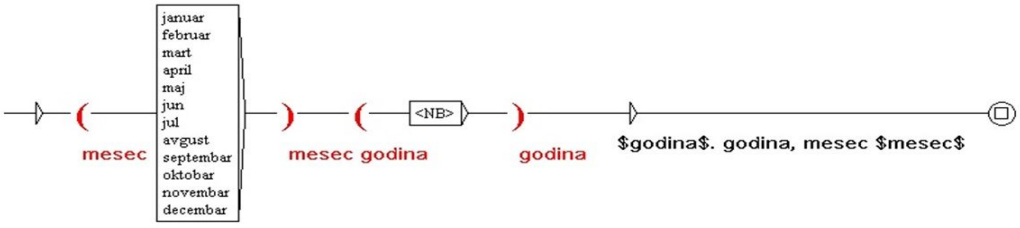
- pozivi podgrafova – označavaju se imenom grafa koji se poziva i karakterom : koji mu prethodi (na primer "`alpha + :beta + gamma + :E:\greek\delta.grf`"). Moguće je navesti punu putanju do grafa, koristiti relativne putanje ili koristiti tzv. repozitorijume grafova (posebne direktorijume koji sadrže kolekcije grafova, definisane na nivou UNITEX-a, videti (Paumier, 2011))

- izlaz transduktora – dodeljivanjem izlaza nekom boksu kreira se graf koji odgovara transduktoru. Da bi se definisao izlaz koristi se karakter /. Svi karaketri desno od njega predstavljaju izlaz transduktora. Na primer, unos sekvence "`jedan+dva+tri/broj`" u tekstualno polje boksa rezulturalo bi grafom kao na slici 23.



Slika 23. Transduktor koji prepoznaje reči *jedan*, *dva* i *tri* i pri tom generiše reč *broj* kao izlaz

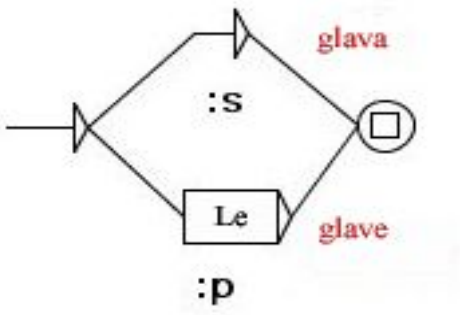
- promenljiva – posebnim boksovima moguće je označiti početak i kraj promenljive koja biva generisana prolazom kroz graf, a koja kasnije može da bude korišćena u izlazu transduktora. Promenljiva dobija vrednost na osnovu prepoznate sekvence, tj. jednog njenog dela koji je definisan početkom (boks koji sadrži oznaku **`$var1(`**) i krajem (boks koji sadrži oznaku **`$var1)`**). Na slici 24 je prikazan graf koji prepoznaje jedan format datuma ("*januar 2012*") i prebacuje ga u drugi ("*2012. godina, mesec januar*").



Slika 24. Primer transduktora sa dve promenljive, *mesec* i *godina*

U zavisnosti u kojoj fazi obrade teksta se koriste, grafovi mogu da budu različito interpretirani ili da u nekima od njih ne bude dozvoljena upotreba svih simbola. Na primer, transduktori se koriste, između ostalog, i za opisivanje morfoloških varijacija koje odgovaraju pojedinim klasama reči, dodeljujući flektivne kodove svakoj varijanti. Takvi transduktori se nazivaju flektivni transduktori. Putanje ovih transduktora opisuju modifikacije koje treba primeniti na kanonski oblik reči, nakon kojih se produkuje određeni izlaz koji odgovara izmenjenom obliku reči (slika 25). Putanje mogu da sadrže operatore i slova. Jedini dozvoljeni specijalan simbol jeste prazna reč `<E>`.

U okviru flektivnih transduktora koriste se operatori koji definišu način na koji se neka reč menja. Mogući operatori su predstavljeni slovima L, R, C i D. Operator L uklanja poslednje slovo reči. Operator R ubacuje (vraća) slovo reči u izmenjeni oblik. Operator C duplira slovo u reči (kao na primer u *permitted* ili *hopped*). Operator D briše slovo iz reči i pomera sve što se nalazi desno od slova za jedno mesto. U primeru na slici 25, putanja `Le` znači da se polaznoj lemi reči uklanja poslednje slovo, a zatim dodaje slovo e kako bi se formirala množina (plural) imenice glava.



Slika 25. Primer flektivnog transduktora

3.6.3. Kompilacija grafova

Nakon kreiranja grafa, korisnik ga čuva u datoteci sa ekstenzijom *.grf*. Ova datoteka je u stvari tekstualna datoteka koja sadrži informacije o vizuelnoj prezentaciji skupa stanja i funkcije prelaza grafa, uključujući font, boju slova, boju pozadine i slično. Ovaj format podataka je odgovarajući dokle god korisnik radi na kreiranju grafa, poziva podgrafove i slično. Jednom kada je taj posao obavljen, i kada graf treba da posluži kao ulazni podatak za dalju pretragu teksta, neophodno je konvertovati ga u pogodniji format. UNITEX za tu svrhu koristi format *.fst2*.

Datoteka u *.fst2* formatu je tekstualna datoteka koja opisuje jedan ili više grafova, ali samo njihove skupove stanja i funkcije prelaza, a ne i vizuelne elemente.

Sledi primer jedne datoteke u *.fst2* formatu, dobijene nakon kompilacije grafa sa slike 24.

Primer.

```
0000000001
-1 datum
: 14 1
: 12 2 11 2 10 2 9 2 8 2 7 2 6 2 5 2 4 2 3 2 2 2 1 2
: 15 3
: 16 4
: 13 5
: 17 6
: 18 7
t
f
%<E>
%januar
%februar
%mart
%april
%maj
%jun
%jul
%avgust
%septembar
%oktobar
%novembar
%decembar
%<NB>
%$mesec(
%$mesec)
```

```
%$godina(
%$godina)
%<E>/$godina$. godina, mesec $mesec$
f
                                    ♦
```

Prva linija predstavlja broj grafova koji su kodirani u datoteci. Početak svakog od njih je označen linijom koja sadrži redni broj grafa i njegovo ime. U prethodnom primeru to je linija `-1 datum.` Da je unutar grafa bilo poziva drugih grafova, njima bi odgovarale linije koje počinju sa `-2`, `-3` i tako redom. Linije koje slede iza njih opisuju stanja grafa. Ukoliko je stanje finalno, linija počinje karakterom `t`, a ukoliko nije, karakterom `:`.

Za svako stanje, lista prelaza je predstavljena sekvencom parova celobrojnih vrednosti. Prva celobrojna vrednost predstavlja oznaku ili podgraf koji odgovara prelazu. Oznake su numerisane počevši od 0. Podgrafovi su predstavljeni negativnim vrednostima. Druga celobrojna vrednost predstavlja broj rezultujućeg stanja nakon prelaza. U svakom grafu stanja su numerisana počevši od broja 0. Po konvenciji, nulto stanje je početno stanje. Svaka linija koja predstavlja definiciju prelaza završava se razmakom. Kraj svakog grafa označen je linijom `f`.

Oznake su definisane nakon poslednjeg grafa. Ukoliko oznaka sadrži i izlaz transduktora, ulazna i izlazna sekvenca su razdvojene znakom `/` (npr. `the/DET` ili `<E>/$godina$. godina, mesec $mesec`).

Dakle, u okviru rada sa UNITEX sistemom, svaki graf je neophodno kompilirati u datoteku formata *.fst2*, kako bi bilo moguće koristiti grafove pomoću UNITEX-ovih programa. Ovaj format zadržava strukturu sa podgrafovima unutar gramatika, te se razlikuje od konačnih transduktora. UNITEX-ov program `Flatten` omogućava konverziju FST2 gramatike u konačni transduktor kad god je to moguće ili konstrukciju aproksimativnog transduktora, kada to nije (Paumier, 2011). Naime, prilikom kompiliranja dolazi do zamene svakog pojedinačnog poziva nekog podgrafa samim podgrafom. Ova zamena se vrši do određene dubine. Pozivi podgrafova koji se nalaze nakon ove vrednosti bivaju zamenjeni praznom reči, i u tom slučaju rezultat kompiliranja je konačni transduktor, ali nije

ekvivalentan polaznom grafu. Moguća je i opcija da se pozivi ovih grafova zadrže, kojom prilikom je zadržana ekvivalentnost, međutim rezultat kompiliranja nije konačni transduktor.

3.6.4. Primena grafova za ekstrakciju informacija

Upotreba UNITEX-a za ekstrakciju informacija moguća je na dva načina. Prvi način je da se pomoću grafičkog korisničkog interfejsa UNITEX-a i radnog okruženja koje ovaj sistem obezbeđuje izvede ceo proces ekstrakcije informacija, dakle i kreiranja pravila ekstrakcije, i predprocesiranje teksta, i samo izdvajanje inoformacija. Ukoliko se ekstrakcija vrši na taj način, tada je jedino moguće umetanje određenih oznaka u tekst koje bi odredile semantičke klase ekstrahovanih informacija. Za detaljnija objašnjenja o tome kako je moguće obaviti ekstrakciju informacija na ovaj način čitalac se upućuje na (Paumier, 2011).

Drugi način jeste da se UNITEX-ov korisnički interfejs koristi samo za kreiranje grafova i eventualno za predprocesiranje, a da se sama ekstrakcija informacija izvrši pomoću posebno kreiranih algoritama, implementiranih u nekom od programskih jezika, pri čemu bi se UNITEX-ovi spoljašnji programi koristili za obradu teksta i primenu grafova na tekst. Na ovaj način je moguće izvršiti proizvoljno strukturiranje podataka (na primer, formiranje i popunjavanje relacione baze podataka na osnovu ekstrahovanih informacija).

Ukoliko se UNITEX koristi na gore pomenuti drugi način, tj. u okviru nekog drugog softverskog sistema za ekstrakciju, tada je ekstrakciju moguće obaviti na mnogo različitih načina (i u pogledu algoritama i u pogledu njihove implementacije). Jedan takav metod je detaljno opisan u poglavlju 4 ove disertacije. Zato će ovde biti opisani samo neki od koraka u ekstrakciji informacija, a koji se tiče upravo UNITEX-a i njegovih spoljašnjih programa

Normalizacija. Kao što je napomenuto ranije, normalizacija teksta može biti obavljena na različite načine, a u nekim situacijama i potpuno izostavljena. Pa ipak, ukoliko se tekst normalizuje u okviru predprocesiranja, ovaj proces se obavlja pomoću UNITEX-ovog spoljašnjog programa *Normalize*.

Poziv programa *Normalize* ima sledeću sintaksu:

```
Normalize txt [-no_CR] [-f=norm]
```

Parametar `txt` predstavlja kompletnu putanju do tekstualnog fajla u kome se nalazi tekst koji se normalizuje. Nakon obrade, program *Normalize* kreira nov, izmenjen tekst i smešta ga u datoteku sa ekstenzijom `.snt`. Opcioni parametar `-no_CR` zamenjuje bilo koju sekvencu separatora jednim praznim mestom, dok se parametar `-f=norm` koristi da odredi datoteku u kojoj su smeštena dodatna pravila normalizacije.

Nakon normalizacije, neophodno je da se kreira i poseban direktorijum potreban za dalji tok procesa obreda teksta. Spoljašnji programi UNITEX-a očekuju da ovaj direktorijum ima tačno određeno ime. Naime, ukoliko datoteka koja se obrađuje nosi naziv `tekst1.txt`, tada nakon procesa normalizacije bivaju kreirana datoteka `tekst1.snt` i direktorijum `tekst1_snt`.

Tokenizacija. Tokenizacija se obavlja UNITEX-ovim spoljašnjim programom *Tokenize*, čiji poziv ima sledeću sintaksu:

```
Tokenize text alphabet [-char_by_char]
```

Parametar `text` predstavlja kompletnu putanju do tekstualnog fajla u kome se nalazi tekst dobijen nakon normalizacije (sa ekstenzijom `.snt`). Parametar `alphabet` predstavlja kompletnu putanju do datoteke koja sadrži definiciju alfabeta za jezik teksta. Ova datoteka definiše koji karakteri pripadaju alfabetu jezika na kome je tekst koji se obrađuje. Opcioni parametar `-char_by_char` dopušta mogućnost da se tokenizacija vrši karakter po karakter, što je neophodno u slučaju nekih azijskih jezika. Za tekstove na srpkom jeziku ova opcija se isključuje, pa se za jednu leksičku jedinicu smatra sekvenca slova, pri čemu su slova definisana pomoću datoteke `alphabet.txt`, zatim karakter koji nije slovo, separator rečenice i leksičke labele (Paumier, 2011). Lista leksičkih jedinica (tokena) biva snimljena u tekstualnu datotetu `tokens.txt`, pri čemu se svakom tokenu dodeljuje jedinstveni kod. Zatim se i ceo tekst kodira pomoću kodova tokena i biva snimljen u binarnu datoteku `text.cod`. Program `Tokenize` proizvodi i sledeće četiri datoteke:

- `tok_by_freq.txt` – tekstualna datoteka koja sadrži tokene sortirane po učestalosti pojavljivanja u tekstu
- `tok_by_alph.txt` - tekstualna datoteka koja sadrži tokene sortirane alfabetski
- `stats.n` – tekstualna datoteka koja sadrži informacije o broju separatora rečenica, broju tokena, broju reči i brojeva
- `enter.pos` – binarna datoteka koja sadrži listu pozicija novog reda u tekstu; ova datoteka se koristi u Uniitex radnom okruženju za usaglašavanje konkordanci sa originalnim tekstom

Primena elektronskih rečnika. Na tekst koji je podeljen na leksičke jedinice moguće je primeniti elektronske rečnike pomoću UNITEX-ovog spoljašnjeg programa *Dico*. Poziv programa *Dico* ima sledeću sintaksu:

```
Dico text alphabet [-md=XXX] dic_1 [dic_2 ...]
```

Ovaj program primenjuje elektronski rečnik (ili više njih) na tekst, pri čemu je neophodno da tekst bude podeljen na leksičke jedinice pomoću programa *Tokenize*. Parametar `text` predstavlja kompletnu putanju do datoteke u kojoj se nalazi tekst. Parametar `alphabet` predstavlja datoteku koja definiše slova za jezik na kom je tekst koji se obrađuje. Opcioni parametar `-md=XXX` se koristi da opiše koji morfološki rečnici će biti korišćeni, pri čemu `XXX` predstavlja listu rečnika u *.bin* formatu. Parametar `dic_i` predstavlja putanju i ime rečnika koji se koristi. Rečnik mora biti ili u *.bin* formatu, koji se dobija nakon primene programa *Compress* (videti (Paumier, 2011)) ili u formatu *.fst2*, tj. u obliku grafa. Program *Dico* generiše nekoliko fajlova i snima ih u direktorijum teksta:

- `.dlf` – rečnik prostih reči iz tekstu;
- `dlc` – rečnik složenih reči iz tekstu;
- `err` – rečnik neprepoznatih reči;
- `tags.ind` – sekvence koje je potrebno umetnuti u automat teksta (videti (Paumier, 2011))
- `stat_dic.n` – datoteka koja sadrži broj prostih, složenih i neprepoznatih reči

Primena grafa na tekst. Primena grafova na tekst (bez obzira da li se vrši samo pretraga teksta ili se primenjuju transduktori) obavlja se pomoću UNITEX-ovog spoljašnjeg programa *Locate*, koji ima sledeću sintaksu

```
Locate text fst2 alphabet s/l/a i/m/r n [dir] [-thai] [-
space] [-md=XXX]
```

Ovaj program primenjuje određenu RTN opisanu grafom na tekst i kreira indeks pronađenih izraza koji odgovaraju zadatom grafu. Njegovi parametri su:

- `text` - kompletna putanja do teksta nad kojim se vrši pretraga (`datoteka.snt`)
- `fst2` - kompletna putanja do RTN opisane grafom
- `alphabet` - putanja do datoteke u kojoj je opisan alfabet jezika
- `s/l/a` - parametar koji određuje da li se traži najduže (`l`), najkraće (`s`) ili sva (`a`) poklapanja
- `i/m/r` - parametar koji određuje način primene transduktora; i (*ignore*) – ne uzima u obzir izlaz transduktora, m (*merge*) – dodaje izlaz transduktora u originalni tekst, r (*replace*) - zamenjuje prepoznatu sekvencu izlazom transduktora
- `n` – parametar koji određuje maksimalan broj pojavljivanja; ukoliko je njegova vrednost `all`, neophodno je pronaći sva pojavljivanja
- `dir` - opcioni parametar koji određuje direktorijum koji može da zameni podrazumevani direktorijum teksta
- `thai` – opcioni parametar, određuje da se vrši pretraga teksta na *Thai* jeziku
- `space` – opcioni parametar koji određuje da pretraga može da počne na bilo kojoj poziciji u tekstu; ovaj parametar se koristi za morfološke pretrage
- `md=XXX` – opcioni parametar koji određuje koji morfološki rečnici treba da se koriste

Nakon pretrage, program `Locate` proizvodi dve datoteke koje bivaju snimljene u direktorijumu teksta: `concord.ind` koja sadrži reference do pronađenih pojavljivanja izraza koji odgovaraju grafu i `concord.n`, koji sadrži broj pojavljivanja kao i procenat prepoznatih tokena unutar teksta. Ukoliko je broj pojavljivanja veći od 0 znači da su pornađene fraze koje odgovaraju grafu. Tada se u datoteci `concord.ind` nalaze sva pronađena pojavljivanja. U zavisnosti od samog procesa ekstrakcije informacije, vrši se dalja obrada datoteke `concord.ind`, kako bi se izdvojeni delovi teksta obradili i strukturirali (videti poglavlje 4).

Glava 4

4 DVOFAZNI METOD ZA EKSTRAKCIJU PODATAKA ZASNOVAN NA KONAČNIM TRANSDUKTORIMA

4.1. Opis metoda

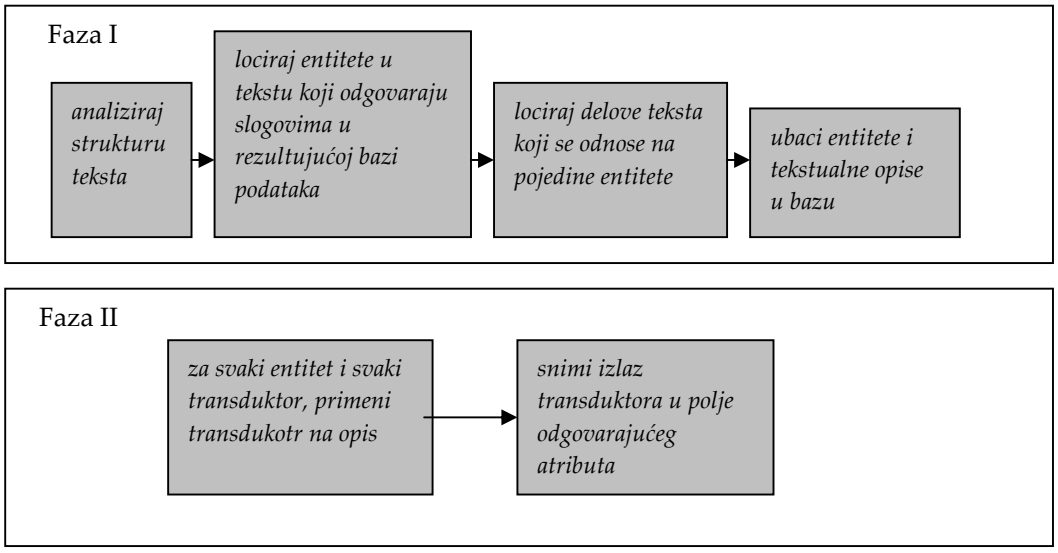
U većini metoda za ekstrakciju informacija lociranje i identifikacija slogova unutar teksta i izdvajanje podataka iz njih se obavlja istovremeno, u okviru jednog istog logičkog procesa. Ovakav pristup često otežava proces ekstrakcije informacija. I horizontalni i vertikalni problem (poglavlje 2.1) ekstrakcije informacije su sami za sebe veoma složeni i zahtevni, a njihovo kombinovanje i istovremeno rešavanje još više povećava mogućnost pojave greške u podacima, ili neprepoznavanje podataka u tekstu.

Kako bi se prevazišli, ili barem pojednostavili navedeni problemi, razvijen je poseban, nov metod nazvan *Dvofazni metod za ekstrakciju informacija baziran na konačnim transduktorima*. Ovaj metod je namenjen za ekstrakciju određenih entiteta i njihovih osobina (atributa) iz teksta. Pogodan je za korišćenje u situacijama kada se obrađuju tekstovi kao što su veb strane, enciklopedije, neki udžbenici, tj. za sve one resurse kod kojih je moguće na neki način, na osnovu njihove retoričke strukture (poglavlja i odeljci u enciklopediji) ili HTML i XML oznaka (za veb strane), odrediti koji delovi teksta se odnose na koji entitet.

Metod razlikuje dve, logički i vremenski potpuno odvojene faze (slika 26). Kao ulaz u algoritam prve faze koristi se ceo tekst koji se obrađuje. Zatim

algoritam prve faze locira entitete i na osnovu same strukture tekstualnog resursa deli celokupan tekst na manje celine, od kojih svaka odgovara po jednom entitetu (objektu) koji se obrađuje, tj. svaka će predstavljati jedan slog u bazi podataka. Kao izlaz algoritma prve faze dvofaznog metoda kreira se baza podataka sa slogovima koji su identifikovani na neki način (svaki odgovara po jednom entitetu, pa se često upravo prepoznati entitet koristi kao identifikator sloga) i koji sadrže manje delove teksta sa informacijama o objektu. Ove informacije su i dalje u nestrukturiranom obliku, ali su pridružene odgovarajućem entitetu na koji se odnose. Na taj način je razrešen vertikalni problem ekstrakcije informacije.

U drugoj fazi cilj je rešiti horizontalni problem ekstrakcije informacija, tj. iz teksta izvući vrednosti pojedinih atributa i dodeliti im značenje. Za izdvajanje atributa u drugoj fazi koriste se konačni transduktori i rekurzivne mreže prelaza, gde se za svaki atribut koji je potrebno izdvojiti, tj. za svaku osobinu entiteta koja je od interesa u nekom procesu ekstrakcije, kreira po jedan transduktor. Na taj način se u stvari pomoću transduktora zadaju pravila ekstrakcije. Značenje izvučene informacije definisano je samim transduktorom koji je tu informaciju prepoznao, pa se izlaz svakog transduktora ubacuje u odgovarajuće polje baze podataka. Druga faza ne zavisi od strukture samog dokumenta, pa isti transduktori mogu biti primenjeni i na druge tekstualne dokumente koji sadrže odgovarajući tip informacija.



Slika 26. Šematski prikaz ekstrakcije informacija primenom dvofaznog metoda

Podela procesa ekstrakcije informacije u dve faze omogućava ne samo bolju efikasnost i veću preciznost, već i korišćenje različitih softverskih alata i sistema za njenu implementaciju. Tako je moguće za implementaciju prve faze metoda korisiti jedan alat (na primer, neki od programskih jezika koji omogućavaju rad sa relacionim bazama), a za implementaciju druge faze neki sasvim drugi alat (na primer, neki od softverskih alata koji su efikasni i jednostavni u radu sa konačnim modelima).

U svakom slučaju, pri projektovanju sistema za ekstrakciju informacija koji bi implementirao dvofazni metod treba uzeti u obzir nekoliko činjenica:

- algoritam prve faze jako zavisi od strukture tekstualnog resursa, pa je potrebno odabrati ili alate koji mogu efikasno da obrade jedan tip resursa (ukoliko se projektuje sistem za određeni tip dokumenata i tekstova) ili alate koji su u mogućnosti da efikasno obrađuju različite formate tekstualnih dokumenata (ukoliko se projektuje sistem opšte namene). Tako je moguće koristiti i neke od već razvijenih programa za preuzimanje teksta sa veb strana (omotači, eng. *wrappers*), ili posebno razviti algoritme koristeći funkcije za rad sa tekstom koje obezbeđuje programski jezik koji se koristi, regularne izraze, grafove konačnih modela ili sl.

- u okviru prve faze vrši se kreiranje slogova podataka relacione baze, pa je neophodno odabrati alat koji je u mogućnosti da komunicira sa bazama podataka ili na neki drugi način obezbediti kreiranje i popunjavanje baze

- pre primene samih transduktora neophodno je izvršiti pripremu teksta, tj. njegovu tokenizaciju, normalizaciju i primenu rečnika. Tek nakon toga, primenjuju se transduktori koji prepoznaju kontekst neke informacije i na osnovu njega određuju vrednost odgovarajućeg atributa

Iako je implementaciju dvofaznog metoda moguće izvršiti na razne načine i različitim postojećim ili novoformiranim softverskim rešenjima, ipak postoje određeni elementi koje svaki sistem za implementaciju dvofaznog metoda mora da obezbedi. To su:

a) podsistem za razdvajanje teksta na slogove podataka (za implementaciju prve faze)

b) struktura za čuvanje podataka

c) podsistem za predprocesiranje teksta

d) podsistem za kreiranje transduktora

e) podsistem za izdvajanje informacija primenom transduktora

f) podsistem za upravljanje procesom ekstrakcije informacija

Podsistem za razdvajanje teksta na slogove podataka preuzima tekstualni resurs koji se obrađuje i iz njega izdvaja manje delove teksta t_j, $j=1..m$ koji se odnose na pojedine objekte (entitete). Pri tome, svakom izdvojenom delu teksta se dodeljuje određeno značenje, tj. jasno se utvrđuje na koji entitet se taj deo teksta odnosi.

Izdvojeni delovi teksta se čuvaju u nekom formatu pogodnom za dalju obradu, tj. **strukturi za čuvanje podataka**. Dvofazni metod predviđa da to bude relaciona baza podataka, ali je moguće koristiti i neki drugi format.

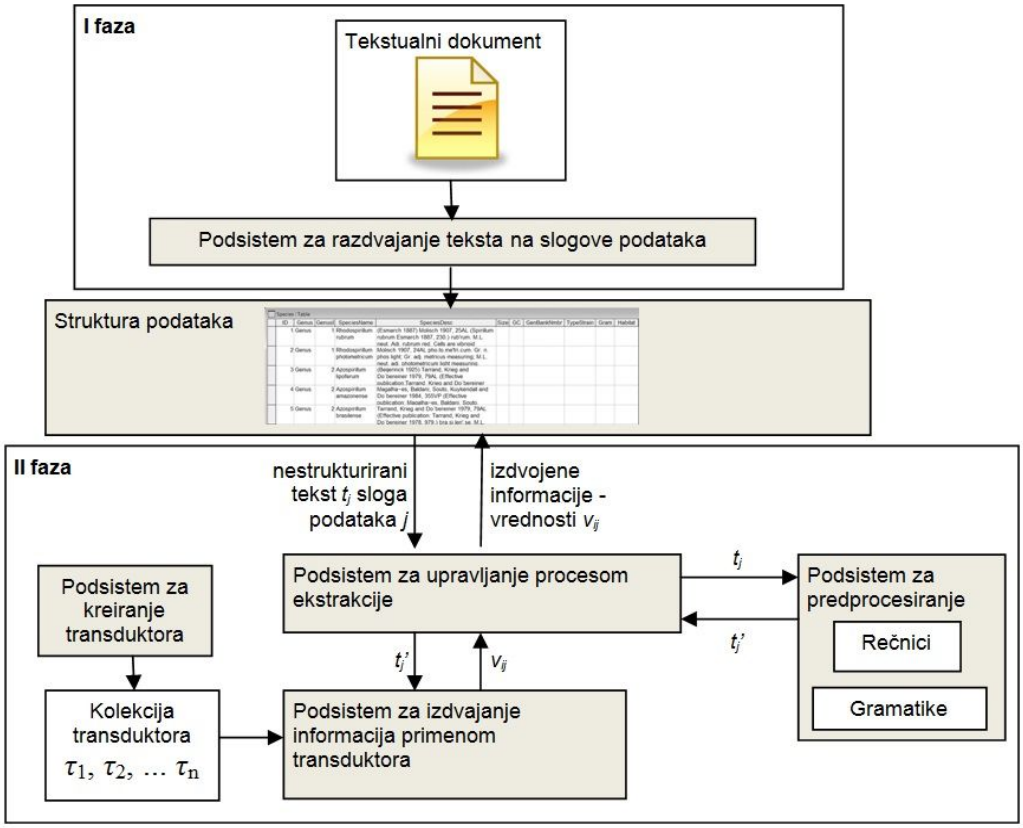
Pomoću **podsistema za kreiranje transduktora** se kreiraju transduktori ($\tau_1, \tau_2, \ldots \tau_n$) namenjeni za izdvajanje konkretnih informacija, tj. vrednosti v_{ij}, $i=1..n$, određenih atributa entiteta iz teksta, za svaki izdvojeni deo teksta t_j, $j=1..m$. Osobine nekih od transduktora za ekstrakciju informacija su opisane u 4.2.3. Ukoliko postoje već razvijene biblioteke transduktora za ekstrakciju, ovaj podsistem je moguće izostaviti. Takođe, zbog svoje specifičnosti i određene veštine potrebne za izradu transduktora, moguće je potpuno odvojiti sistem za kreiranje transduktora od ostatka sistema za ekstrakciju. Na taj način je omogućeno angažovanje stručnjaka specijalno obučenih za rad sa konačnim modelima i odgovarajućim alatima, nezavisno od ostatka procesa ekstrakcije.

Da bi u okviru konačnih modela koji se koriste za ekstrakciju informacija bili korišćeni lingvistički resursi ili leksički i morfološki filteri i maske (videti 3.6.2), neophodno je da tekst na koji se primenjuju bude prethodno lingvistički obrađen. **Podsistem za upravljanje procesom ekstrakcije informacija** preuzima jedan po jedan izdvojeni deo teksta t_j iz strukture u kojoj je čuvan i šalje ga prvo **podsistemu za predprocesiranje** na lingvističku obradu. Podsistem za predprocesiranje može da koristi ili da sadrži i različite lingvističke resurse, kao

što su elektronski rečnici i gramatike. Lingvistički obrađeni delovi teksta t_j' se dalje prosleđuju podsistemu za izdvajanje informacija primenom transduktora.

Podsistem za izdvajanje informacija primenom transduktora primenjuje niz transduktora (τ_1, τ_2, ... τ_n) na deo teksta t_j' koji mu je prosleđen, pri čemu svaki od transduktora produkuje određeni izlaz. Upravo niske koje predstavljaju izlaze transduktora čine izdvojene informacije (v_{ij}) i bivaju smeštene u polja strukture podataka koja odgovaraju slogu *j* koji se obrađuje.

Arhitektura opšteg sistema za ekstrakciju informacija pomoću dvofaznog metoda baziranog na transduktorima prikazan je na slici 27



Slika 27. Arhitektura sistema za ekstrakciju informacija koji implementira dvofazni metod baziran na transduktorima

U nastavku ovog poglavlja detaljno je opisan razvoj jednog sistema za ekstrakciju informacija i primena dvofaznog metoda unutar njega. Kao podsistem za predprocesiranje, za kreiranje transduktora i za izdvajanje informacija korišćen je UNITEX. Podsistem za razdvajanje teksta na slogove podataka i podsistem za

upravljanje procesom ekstrakcije su pisani posebno, koristeći programski jezik Java. Programski kod klase koja predstavlja osnov podsistema za upravljanje procesom ekstrakcije prikazan je u prilogu 1 ove disertacije. Za strukturiranje i čuvanje podataka korišćena je MS Access baza podataka.

4.2. Primena dvofaznog metoda na naučnu enciklopediju kao polustrukturirani resurs – kreiranje i dopuna baze podataka o mikroorganizmima

Jedan od veoma aktuelnih problema u oblasti biologije i genetike jeste pridruživanje fenotipskih karakteristika nekog organizma (npr. tip staništa, optimalna temperatura rasta, patogenost i sl.) proteinima i metabolitima koji su kodirani genomom tog organizma. Predviđanje fenotipa na osnovu genotipskih karakteristika (tj. molekularnog sastava) je veoma važno. Postoji nekoliko studija koje se bave ovim problemom (Chang i sar., 2011; Goh i sar., 2006; Jim et i sar., 2004; Jimeno-Yepes i sar., 2009; Korbel i sar.. 2005; Li i sar., 2010; MacDonald i Beiko, 2010; Seki i Mostafa, 2009; Tamura i D'haeseleer, 2008), a koje uključuju upotrebu tehnika za istraživanje teksta i ekstrakciju informacija iz literature kako bi se otkrile te veze .

U poslednje vreme dolazi do nagomilavanja genotipskih podataka i genomskih sekvenci različitih organizama. Ovi podaci su obično dobro strukturirani i smešteni u baze podataka (npr. GenBank, Bilofsky i Christian, 1988). Da bi se genotipski podaci iskoristili za uspostavljanje odnosa sa karakteristikama odgovarajućih organizama, neophodno je da postoje i odgovarajući podaci o fenotipskim karakteristikama u sličnom obliku. Pa ipak, za razliku od genotipskih podataka, podaci o fenotipskim karakteristikama se najčešće nalaze u različitim tekstualnim dokumentima, kao što su naučni radovi, udžbenici ili enciklopedeije, i to u okviru slobodnih tekstualnih opisa, nepogodnih za automatsku, računarsku obradu. Takvim podacima je veoma teško pristupiti, te su oni nepogodni za analiziranje matematičkim metodima i računarima, pa postoji velika potreba za alatima koji bi bili u stanju da takve podatke prikupe i obrade (Gopalacharyulu, 2008; Mahdavi, 2010). Postoji nekoliko javno dostupnih baza

podataka koje kombinuju informacije o fenotipskim i genotipskim karakteristikama organizama (Chen i sar., 2010; Palakal i Pavithra, 2009; Sayers i sar., 2009, Sprague i sar., 2008; Thorisson i sar., 2009). Pa ipak, veliki deo podataka o mikroorganizmima ostaje u nestrukturnom ili polustrukturnom obliku, u okviru različitih naučnih tekstova.

U okviru ovog poglavlja biće opisano jedno istraživanje u okviru koga je glavni zadatak bio formirati bazu podataka sa podacima prikupljenjim iz enciklopedije o mikroorganizmima. Ovo istraživanje je sprovedeno u okviru Grupe za bionformatiku[5] Matematičkog fakulteta, Univerziteta u Beogradu. Za automatsko prikupljanje podataka odabran je pristup istraživanja teksta i ekstrakcije informacije. Kao izvor podataka korišćena je enciklopedija "*Systematic Bacteriology*" (Garrity, 2005; Garrity i sar., 2005; Krieg i sar., 2010; Vos i sar., 2009) koja sadrži podatke o mikrobima (njihovim fenotipskim i genotipskim karakteristikama) u obliku opisnog teksta, tj. teksta u slobodnoj formi. Pomoću tehnika ekstrakcije podataka kreirana je i popunjena baza podataka sa informacijama o fenotipskim i (u manjoj meri) genotipskim karakteristikama mikroorganizama. Enciklopedija je korišćena samo i isključivo za potrebe naučnog istraživanja. Cilj je bio da se pokaže kako je moguće dobiti strukturirani resurs sa relevantnim podacima, koji je moguće integrisati sa drugim sličnim postojećim resursima i koristiti za dalja istraživanja u oblasti biologije i genetike.

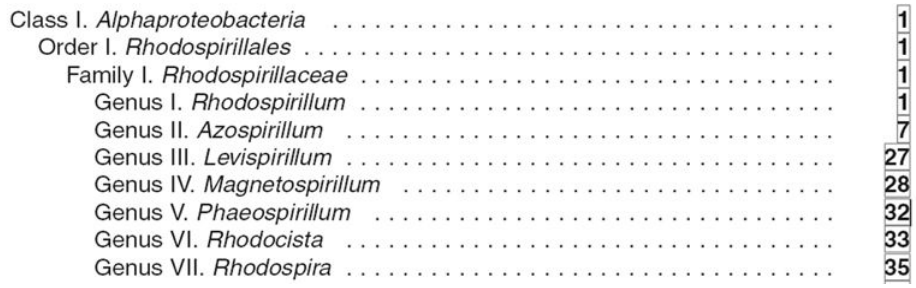
S obzirom da je struktura enciklopedije takva, da je bilo moguće na osnovu retoričke strukture teksta uspostaviti određene veze između entiteta, tj. mikroorganizama u ovom slučaju, i informacija koje se na njih odnose, ovom tekstualnom resursu pristupili smo kao polustrukturiranom a samu strukturu teksta iskoristili u procesu ekstrakcije. Takođe, podaci dobijeni iz enciklopedije trebalo je da budu što je moguće tačniji, kako bi bilo moguće nad njima sprovoditi dalja istraživanja iz oblasti mikrobiologije i genetike. Zbog toga je odabran pristup baziran na znanju i kreiran je i primenjen dvofazni metod baziran na konačnim transduktorima, prethodno opisan u 4.1, za ekstrakciju informacija iz teksta.

[5] http://bioinfo.matf.bg.ac.rs/

4.2.1. Analiza strukture korišćenog resursa i kreiranje modela baze podataka

S obzirom da prva faza dvofaznog metoda zavisi od strukture samog resursa iz koga se izvlače podaci, prvi korak u ekstrakciji informacija jeste analiziranje tekstualnog resursa. Podaci koje smo želeli da izvučemo nalazili su se u četiri toma enciklopedije "*Systematic Bacteriology*", u obliku opisnog, nestrukturiranog teksta na engleskom jeziku. Transformisali smo ovaj tekst iz *.pdf* fromata u *.txt* format, kako bismo obezbedili brži i jednostavniji pristup. Za konverziju teksta korišćen je softverski alat *Abbyy PDF Transformer*[6], i tom prilikom je došlo do gubitka određenih informacija zasnovanih na strukturi dokumenta (neki paragrafi su pogrešno protumačeni, podaci smešteni u tabelama su izgubili strukturu tabele i sl.). Međutim, ovakvo narušavanje strukture nije značajno uticalo na proces ekstrakcije informacije.

Iako su se razlikovali po sadržaju, sva četiri toma enciklopedije su imala veoma sličnu strukturu, koja je iskorišćena u procesu ekstrakcije informacija. Poglavlja enciklopedije su odgovarala taksonomskim kategorijama carstva *Bacteria*. Na primer, prvo poglavlje drugog toma (Garrity, 2005) sadrži informacije o taksonomskoj klasi *Alphaproteobacteria*, sledeće poglavlje je o prvom redu ove klase (*Rhodospiralles*), a zatim slede poglavlja o familijama reda *Rhodospiralles*. Svako poglavlje koje se odnosi na neku familiju praćeno je poglavljima o rodovima te familije. Slika 28 prikazuje izvod iz sadržaja drugog toma enciklopedije.



Slika 28. Izvod iz sadržaja enciklopedije "*Systematic Bacteriology*", Tom 2, deo C

[6] http://pdftransformer.abbyy.com/

Opisi vrsta, koji sadrže i podatke koje želimo da ekstrahujemo, dati su na kraju svakog poglavlja o rodu, i to na samom kraju poglavlja. Prethodi im linija teksta "*List of species of the genus ...*". Broj vrsta se razlikuje od roda do roda, ali svaki od opisa vrsta počinje rednim brojem, praćenim imenom vrste i opisom u obliku opisnog teksta. (Slika 29).

9. **Hyphomicrobium zavarzinii** Hirsch 1989b, 495[VP] (Effective publication: Hirsch 1989, 1903.)
za.varzin'i.i. M.L. gen. n. *zavarzinii* of Zavarzin, named for G.A. Zavarzin, the Russian microbiologist who isolated these bacteria.

Mother cells drop- or pear-shaped, somewhat slender, with hyphae that rarely branch. Mother cells 0.63 × 1.8 µm (range: 0.5–0.9 × 0.7–2.5 µm). Swarmer cells with 1–3 subpolar flagella. In liquid media under most growth conditions, rosettes are formed, since mother cells produce a polar holdfast. Growth in liquids initially as turbidity and later as a pellicle, with precipitation on the bottom. Colonies on solid media are colorless to light brownish or beige, smooth and shiny, with entire edges.

Chemoorganotrophic, aerobic, oligocarbophilic. Good growth with the following carbon sources: methanol, methylamine·HCl, formate, *n*-butyrate, isovalerate, crotonate, β-hydroxybutyrate, ethanol, *n*-propanol, isobutanol, and glycerol. Growth is stimulated significantly by acetate, *n*-valerate, α-oxoglutarate, galacturonate, formaldehyde, D-glucose, D-mannose, D-melibiose, amygdalin, esculin, chitin, Bacto peptone, DL-lysine, DL-aspartate, and dilute human urine. Nitrogen sources utilized are: NH_4^+, NO_2^-, NO_3^-, and (poorly) Bacto peptone. There is slow growth in the absence of added nitrogen sources (oligonitrophily). Poor growth on sheep blood agar with α-hemolysis. The following antibiotics inhibit growth at 30 µg (per disc): kanamycin, neomycin, and tetracycline. Streptomycin at 10 µg is also inhibitory. There is growth in the presence of 3.5% NaCl. Temperature range: 15–37°C. Optimal pH: 6.5–7.5. Visible light inhibits growth slightly.

Grow anaerobically with nitrate and gas formation (with methanol as the carbon source). With methylamine·HCl and thioglycolate, there is little growth. Catalase and cytochrome oxidase are positive; gelatin liquefaction is negative. Poly-β-hydroxybutyrate is a storage product.

Not pathogenic for mice or guinea pigs.

Genome size: 2.73 × 10^9 Da (strain ZV-580; Kölbel-Boelke et al., 1985).

Habitat: peaty and moist soil near Moscow, Russia.

The mol% G + C of the DNA is: 61.8–64.8 (Bd, T_m, HPLC) (Mandel et al., 1972; Gebers et al., 1986; Urakami and Komagata, 1987b; Urakami et al., 1995b).

Type strain: ATCC 27496, IFAM ZV-622.

GenBank accession number (16S rRNA): Y14305.

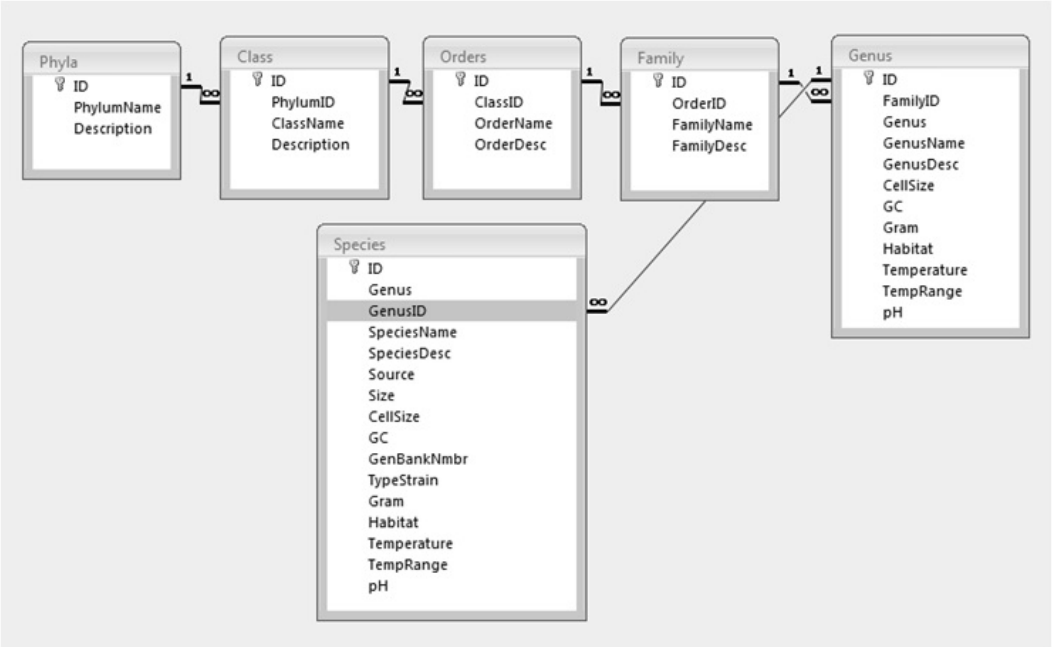
Additional Remarks: Additional strains include IFAM ZV-580, ZV-620, MY-619, MC-625, MC-629, MC-630, and MC-627.

Slika 29. Primer opisa vrste. Podvučeni delovi teksta predstavljaju podatke koje želimo da izvučemo iz teksta.

Pojedinačni atributi o organizmima nalazili su se u slobodnoj tekstualnoj formi, unutar opisa jedne vrste (oni koji su karakteristični za vrstu) ili roda (ukoliko su zajedničkim svim vrstama tog roda). Ti opisi su sadržali različite podatke o organizmima, kao što je njihov oblik, stanište, *Gram* osobina, dominantni soj (*type strain*), veličina genoma, procenat nukleotida guanin i citozin (G+C) u DNK lancu, veličina ćelije, optimalna pH vrednost, GenBank identifikacioni broj, i sl. Upravo su to podaci koje smo želeli da izvučemo iz enciklopedije i smestimo u strukturni oblik, tj. u bazu podataka. Ovi podaci su prikazani podvučeno na Slici 29.

Nakon analize strukture, a pre pristupanja prvoj fazi ekstrakcije, kreirana je baza podataka sa tabelama koje odgovaraju taksonomskim kategorijama: *Phyla, Class, Order, Family, Genus, GenusIncSed* (za "*Genus Incertae Sedis*", taksonomsku grupu čiji je odnos sa drugim grupama nepoznat ili

nedefinisan) i *Species*. Tabele *Phyla*, *Class*, *Order* and *Family* će biti korišćene za čuvanje informacija o taksonomskim kategorijama i njihovim relaicjama. Tabele *Genus*, *GenusIncSed* i *Species*, sem kolona koje čuvaju relacije među grupama, sadrže i kolone koje će čuvati izvučene podatke o organizmima. Dijagram baze je prikazan na slici 30.



Slika 30. Dijagram rezultujuće baze podataka

4.2.2. Prva faza: kreiranje odgovarajuće baze i razbijanje teksta na delove koji odgovaraju pojedinačnim entitetima

Na osnovu analize strukture enciklopedije, razvijen je algoritam prve faze. On koristi činjenicu da svako poglavlje odgovara jednoj sistematskoj kategoriji (*Class*, *Order*, *Family* i *Genus*).

Algoritam čita liniju po liniju teksta sadržaja enciklopedije. Svaka linija ima istu strukturu: prva reč je naziv taksonomske kategorije, praćen rimskim brojem i tačkom, iza koje sledi naziv kategorije (npr. "*Family I. Rhodospirillaceae*"). Algoritam koristi prvu reč u liniji sadržaja kako bi odredio tabelu u koju slog treba da bude ubačen (u navedenom primeru to bi bila tabela "*Family*)". Ime kategorije se koristi kao vrednost polja *FamilyName* sloga koji će

biti kreiran (u ovom slučaju kreira se slog za familiju "*Rhodospirillaceae").* Redosled u kome se različite kategorije pojavljuju u sadržaju iskorišćen je za uspostavljanje veza između kategorija i određivanje pripadnosti, tj. za uspostavljanje veza između tabela. Na primer, struktura tabele *Family* je prikazana u tabeli 2. Polje *OrderID* se koristi za čuvanje veze sa odgovarajućim slogom iz tabele *Order*, tj. sa redom kome neka familija pripada.

Tabela 2. Struktura tabele *Family*

Field Name	Data Type
ID	Integer
OrderID	Integer
FamilyName	Text

Tabela sa opisima vrsta i rodova su popunjavane u sledećem koraku. U okviru I faze popunjavane su samo vrednosti u prva četiri polja (*ID, GenusID, SpeciesName* i *SpeciesDesc* za tabelu *Species*), dok su ostala polja popunjavana u okviru druge faze. Preciznije, I fazom metoda je izvršeno izdvajanje i strukturiranje većih delova teksta, koji su sadržali informacije o organizmima, a na koje će kasnije biti primenjeni transduktori kako bi se izdvojili konkretni atributi koji opisuju organizme.

Dakle, u okviru I faze je trebalo izvući sledeće informacije o organizmima, za tabelu *Species*:

- ime vrste,
- da li pripada rodu iz tabele Genus ili *Genus Incertae Sedis*,
- ime roda kome pripada,
- opis vrste.

Da bi popunio tabelu Species, algoritam prolazi kroz tekst enciklopedije u potrazi za linijom koja počinje sa "*List of species*". Iza ove linije počinju opisi vrsta koje izdvajamo u bazu podataka u okviru prve faze tj. kao rešenje vertikalnog problema. Svaki opis počinje rednim brojem i imenom vrste, na primer

1. Rhodovulum sulfidophilum (Hansen and Veldkamp 1973) .....

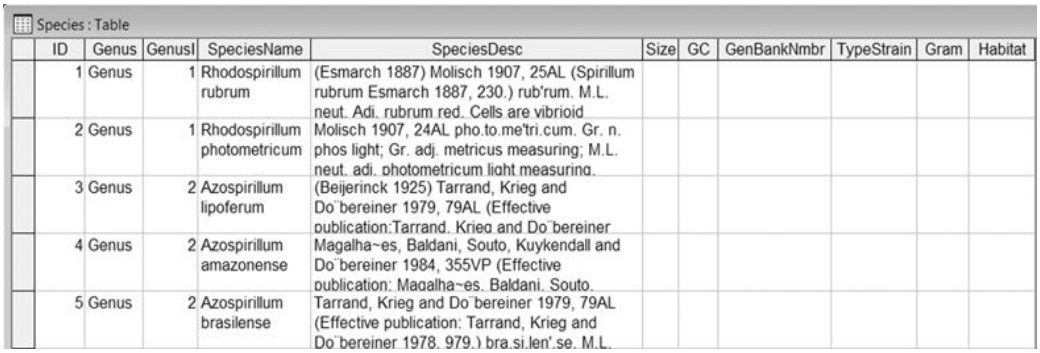
Ova činjenica je iskorišćena kako bi algoritam otkrio slogove za tabelu *Species*, tj. kako bi identifikovao početak i kraj opisa jedne vrste. Na nesreću, tokom konverzije iz `.pdf` u `.txt` format neki paragrafi su pogrešno protumačeni, tako da su u tekstualnom fajlu postojale linije koje počinju brojem i tačkom, ali ne predstavljaju početak opisa vrste. Zbog toga je algoritam dizajniran tako da koristi činjenicu da prva reč u imenu vrste predstavlja ime roda kome ona pripada. Samo linije koje zadovoljavaju taj uslov su prepoznate od strane algoritma i tretirane kao početak opisa vrste. Zbog navedene modifikacije, algoritam je imao odličnu efikasnost. Svaki opis vrste koji je postojao u originalnom tekstu je prepoznat i ubačen u bazu podataka. Ovakva vrsta finog podešavanja algoritma je moguća na osnovu analize struktura tesktualnog resursa koji se koristi. Na istraživaču je da odluči do kog nivoa će modifikovati algoritam, kako bi postigao željeni nivo efikasnosti.

U nastavku je prikazan algoritam prve faze dvofaznog metoda korišćen u slučaju opisanog istraživanja. Ovde treba napomenuti još jednom, da ovaj algoritam u velikoj meri zavisi od strukture tekstualnog resursa, i da će stoga biti različit u slučaju nekog drugog tekstualnog dokumenta. Zbog toga ovaj algoritam treba posmatrati kao primer kako je moguće razreštii vertikalni problem ekstrakcije informacije, a ne kao strogi niz koraka koje je potrebno preduzeti u ovom procesu.

```
set file to the encyclopedia content file
while not end of file
        read the line from the file
        if line starts with the name of some taxonomic category
                read the first word and set as category
                read the second word and set as name
                insert the record in the table corresponding to the category
        end if
end while
set file to the encyclopedia text
while not end of file
        read the line from the file
        if line starts with “List of species”
                set desc to true
        end if
        if line starts with the name of some category
                if line starts with “Genus”
                        set the name of the Genus to be current
```

```
                end if
                if newSpecies
                        insert the speciesDesc and the speciesName
                                to the database
                        set speciesDesc to empty string
                        set speciesName to empty string
                end if
                set desc to false
                set newSpecies to false
        end if
        if desc
                if line starts with number followed by dot
                        if the first word after dot not equal current genus
                                break
                        end if
                        if newSpecies
                                insert the speciesDesc and the speciesName
                                        to the database
                                set speciesDesc to empty string
                                set speciesName to empty string
                        end if
                        set newSpecies to true
                        set speciesName to the first two words after the dot
                        set speciesDesc to the rest of the line
                else
                        appent speciesDesc with the line
                end if
end while
```

Na sličan način je kreirana i tabela *Genus*, pa je nakon završetka prve faze, formirana baza podataka koja je sadržala slogove sa podacima o vrstama i rodovima. Konkretno, za tabelu Species kreirana su i popunjena polja *SpeciesName* (ime vrste), *Genus* (rod kome pripada) i *SpeciesDesc* (tekstualni opis vrste koji sadrži vrednosti pojedinih atributa od interesa). Deo podataka je prikazan na slici 31. Slično, za tabelu *Genus* u popunjena polja *GenusName* (ime roda), *Family* (familija kojoj rod pripada) i *GenusDesc* (tekstualni opis roda koji sadrži vrednosti pojedinih atributa).



Species : Table

ID	Genus	GenusI	SpeciesName	SpeciesDesc	Size	GC	GenBankNmbr	TypeStrain	Gram	Habitat
1	Genus	1	Rhodospirillum rubrum	(Esmarch 1887) Molisch 1907, 25AL (Spirillum rubrum Esmarch 1887, 230.) rub'rum. M.L. neut. adj. rubrum red. Cells are vibrioid						
2	Genus	1	Rhodospirillum photometricum	Molisch 1907, 24AL pho.to.me'tri.cum. Gr. n. phos light; Gr. adj. metricus measuring; M.L. neut. adj. photometricum light measuring.						
3	Genus	2	Azospirillum lipoferum	(Beijerinck 1925) Tarrand, Krieg and Do¨bereiner 1979, 79AL (Effective publication:Tarrand, Krieg and Do¨bereiner						
4	Genus	2	Azospirillum amazonense	Magalha~es, Baldani, Souto, Kuykendall and Do¨bereiner 1984, 355VP (Effective publication: Magalha~es, Baldani, Souto,						
5	Genus	2	Azospirillum brasilense	Tarrand, Krieg and Do¨bereiner 1979, 79AL (Effective publication: Tarrand, Krieg and Do¨bereiner 1978, 979.) bra.si.len'se. M.L.						

Slika 31. Izgled tabele *Species* nakon završene prve faze ekstrakcije

4.2.3. Transduktori za ekstrakciju informacija

Nakon završetka prve faze, a pre početka druge, tekst je izdeljen na manje celine koje su prepoznate kao sadržaoci informacija koje treba da se izvuku iz teksta, pri čemu je za svaku od tih celina identifikovan slog u bazi podataka, tj. mikroorganizam na koji se te informacije odnose. Ti manji delovi teksta se sada nalaze u bazi podataka. U nastavku procesa je potrebno iz njih izvući konkretne podatke.

Pristup ekstrakciji informacija baziran na konačnim transduktorima koristi upravo konačne transduktore za prepoznavanje i izdvajanje pojedinačnih informacija, tj. vrednosti određenih atributa iz teksta. Teoretski osnov upotrebe transduktora kao pravila ekstrakcije prikazan je u poglavljima 2.5 i 2.9. U okviru druge faze navedenog metoda, od strane eksperata kreiraju se konačni transduktori, po jedan za svaki atribut koji je potrebno izvući iz teksta. U okviru pomenutog istraživanja korišćen je softver UNITEX (Paumier, 2011) za kreiranje transduktora, iako je moguće koristiti i druge softvere za tu namenu.

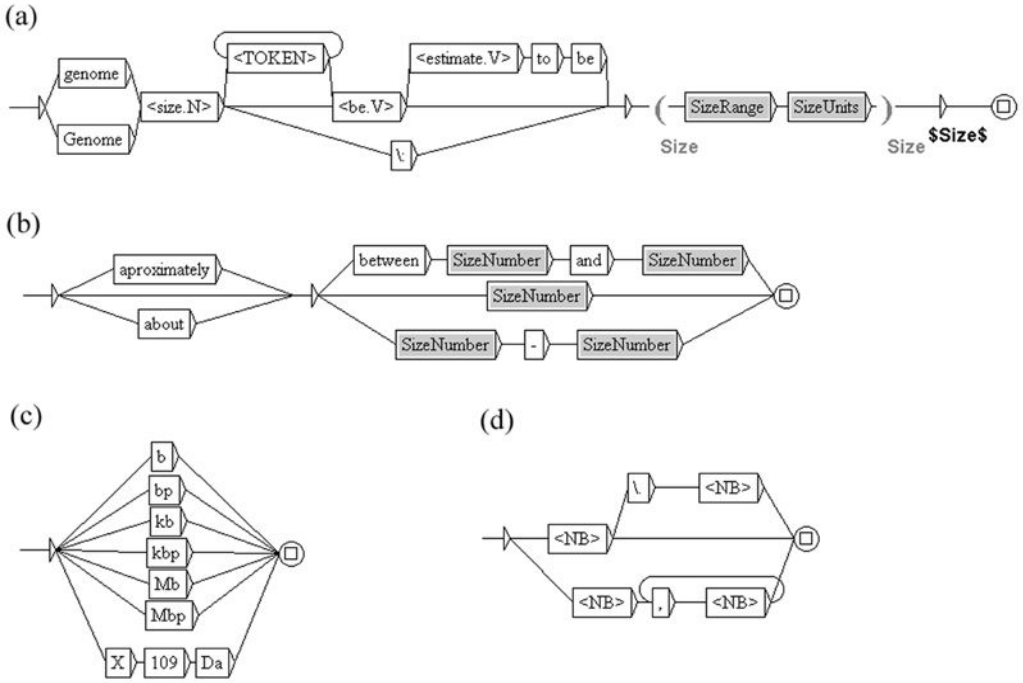
Slika 32 prikazuje nekoliko grafova korišćenih za ekstrakciju informacije o veličini genoma. Glavni transduktor (Slika 32.(a)) opisuje kontekst u kome se očekuje da se informacija o veličini genoma pojavi unutar teksta, i definiše deo teksta koji će biti ekstrahovan kao podatak. Ovaj kontekst je utvrđen od strane eksperata i baziran je na analizi enciklopedijskog teksta. Reči u grafu transduktora su povezane linijama, koje vizuelno predstavljaju relaciju prelaza (poglavlje 2.5). Relacija prelaza je neophodna za definisanje tekstualnih obrazaca. Ukoliko postoji

linija od reči *word1* do reči *word2*, znači će obrazac "*word1 word2*" biti prepoznat od strane tog dela transduktora.

Na primer, algoritam koji koristi transduktor prikazan na slici 32 izdvaja veličinu genoma i to tako što prepoznaje izraze kod kojih je reč "*genome*" prva reč. Sledeća reč u izrazu treba da odgovara imenici "*size*" u bilo kom obliku, što je označeno sa `<size.N>`. Jedino u tom slučaju algoritam nastavlja sa radom, tj. sa prepoznavanjem izraza pomoću transduktora. Informacije o različitim oblicima reči "*size*" se preuzimaju iz elektronskih rečnika. Glavni transduktor koristi i leksičke maske kao što su `<be.V>` and `<estimate.V>`, koje prepoznaju bilo koji oblik glagola *to be* ili *to estimate*, kako bi opisao kontekst u kome informacija o veličini genoma može da se pojavi. Specijalan simbol `<TOKEN>` se odnosi na bilo koji token (bilo koju reč ili karakter koji nije slovo) u tekstu. Samo sekvenca pročitanih reči, koja odgovara putanji definisanoj pomoću transduktora je prepoznata i u tom slučaju se na osnovu tog prepoznatog izraza produkuje i izlaz transduktora.

Izlaz je definisan pomoću zagrada u transduktoru (Slika 32.(a)). Deo teksta koji odgovara delu transduktora između zagrada smešta se u promenljivu, u ovom slučaju nazvanu *Size*. Deo transduktora obeležen karakterom `$` definiše šta će biti izlaz. Transduktor prikazan na slici 32 će kao izlaz vratiti vrednost promenljive *Size*. U nekim drugim slučajevima moguće je da se izlaz definiše kao neki složeniji izraz, a ne samo vrednost promenljive.

Glavni transduktor poziva dva podgrafa, nazvana *SizeRange* i *SizeUnits*. Pozivi podgrafova su obeleženi sivim pravougaonicima, kao na slici 32.(a) i slici 32.(b). Odgovarajući podgrafovi, *SizeRange* i *SizeUnits* su prikazani na slici 32.(b) i slici 32.(c). Graf *SizeRange* opisuje kontekst koji odgovara različitim načinima na koje je moguće izraziti veličinu genoma u enciklopedijskom tekstu, kao šro su "*between 2240 and 3787*", "*1256–1276*" ili "*approximately 4061*". Graf *SizeUnits* opisuje sve jedinice korišćene za izražavanje veličine genoma, a koje se pojavljuju u enciklopedijskom tekstu. Graf *SizeNumber* opisuje različite refernce na brojeve koji se pojavljuju u tekstu. Specijalni simbol `<NB>` prepoznaje bilo koju neprekidnu sekvencu cifara.



Slika. 32. Transduktor za ekstrakciju informacija o veličini genoma kreiran pomoću UNITEX softvera (a) Glavni transduktor sadrži pozive podgrafova *SizeRange* i *SizeUnits* i produkuje izlaz $*Size*$; (b) Podgraf *SizeRange* za opisivanje različitih načina za specifikaciju vrednosti veličine genoma; (c) Podgraf *SizeUnits* koji prepoznaje različite jedinice za veličinu genoma; (d) Podgraf *SizeNumber* prepoznaje različite formate brojeva

Sledeće fraze su prepoznate od strane transduktora prikazanog na slici 32. Ekstrahovani podaci, koji se smeštaju u bazu podataka, označeni su podebljanim slovima.

"*genome sizes of four G. oxydans strains were estimated to be* ***between 2240 and 3787 kb***"

"*genome size of R. prowazekii is* ***1,111,523 bp***"

"*genome size of R. africae is* ***1.248 kb***"

"*genome size of R. australis is* ***1256–1276 kbp***"

"*genome size is* ***2.62 X 109 Da***"

"*Genome size:* ***2.73 X 109 Da***"

"*genome size is* ***1.713 Mbp***"

"*genome size was estimated to be* ***approximately 4061 kb***"

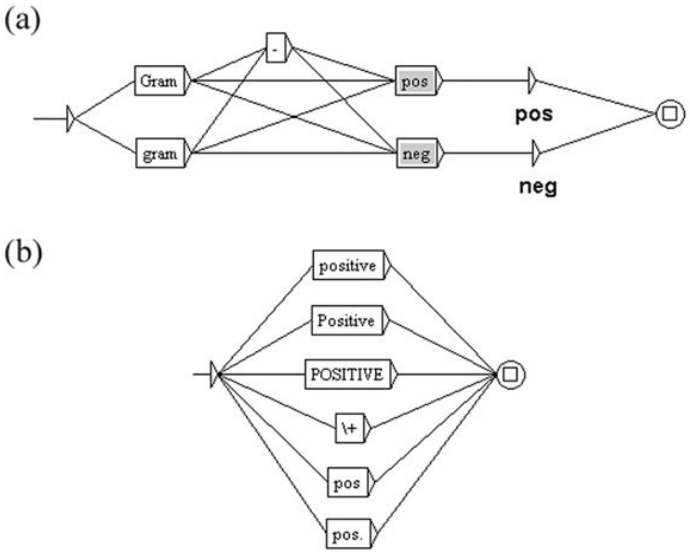
"*genome size of all the classical strains examined was* ***about 3000 kb***"

Dakle, transduktor prepoznaje celu frazu, ali izdvaja samo označeni deo teksta. U smislu definicija datih u 2.9, za transduktor dat na slici 32 važi da upravo gore navedene fraze (i njima slične koje su prepoznate transduktorom) sačinjavaju jezik L_{prep}, dok se jezik L_{izdv} sastoji od delova teksta koji su izdvojeni, tj. L_{izdv} = {"between *2240 and 3787 kb", "1,111,523 bp", "1.248 kb", "1256–1276 kbp", "2.62 X 109 Da", "2.73 X 109 Da", "1.713 Mbp"...}.*

Na slici 33 prikazan je još jedan transduktor, koji je namenjen za ekstrakciju *Gram stain* osobine. Ovaj transduktor prepoznaje, između ostalih, fraze kao što su:

"*Gram positive*", "*gram-positive*", "*Gram +*", "*Gram-pos.*", "*gram negative*", "*Gram -*"

Ove fraze će biti prepoznate od strane jedne od nekoliko mogućih putanja grafa. Zavisno od informacije sadržane u tekstu, a samim tim i putanje grafa koja je prepoznala određenu frazu, transduktor će da proizvede izlaz "*pos*" za *Gram* pozitivne baketrije, ili "*neg*" za *Gram* negativne. Izlaz transduktora predstavlja aktuelnu informaciju koju želimo da ubacimo u bazu podataka.



Slika 33. (a) Graf koji predstavlja transduktor za ekstrakciju informacije o Gram stain svojstvu mikroorganizma; (b) podgraf nazvan "*pos*" koji opisuje tokene koji se odnose na Gram pozitivne bakterije

Konačni transduktori, kao sredstvo za ekstrakciju informacije u okviru dvofaznog metoda su odabrani iz nekoliko razloga. Prvo, po svojim osobinama su

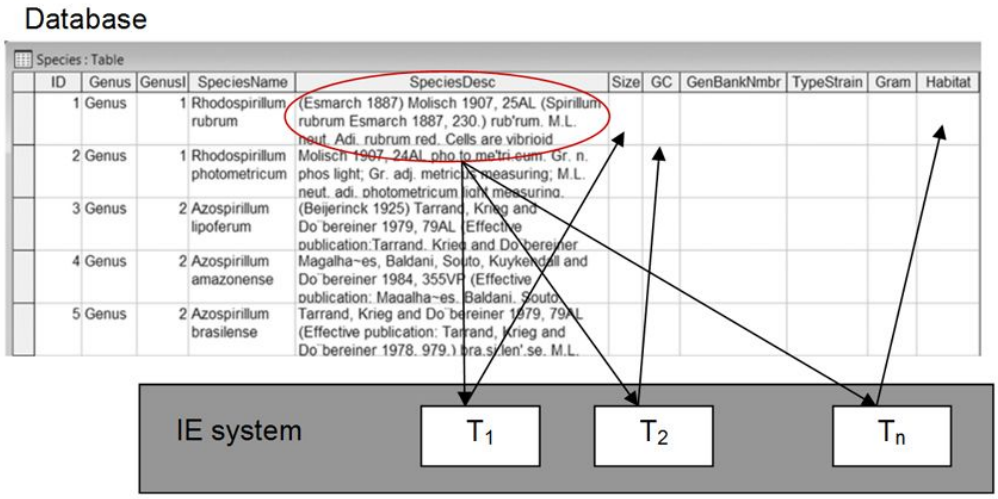
veoma pogodni za modeliranje jezika releventnih informacija i pravila ekstrakcije, s obzirom da uzimaju u obzir kontekst informacije, kao i da su u stanju da produkuju izlaz.

Drugo, pretpostavlja se da u tekstovima kao što su naučni ili pravni tekstovi ne postoji veliki broj različitih fraza ili izraza kojima je moguće opisati pojedini fenomen, već se uglavnom koristi karakteristična terminologija i slične rečenične konstrukcije. Na primer, nema baš mnogo različitih načina da se kaže da je neka bakterija Gram negativna. Zbog toga, dizajniranjem transduktora koji prepoznaje deo teksta koji opisuje Gram osobinu bakterije i koji produkuje izlaz "*pos*" ili "*neg*" u zavisnosti od pročitane informacije, moguće je procesirati ne samo konkretan enciklopedijski tekst, već i bilo koji drugi tekstualni resurs o mirkoorganizmima. Zbog toga je moguće kolekciju transduktora kreiranih tokom jednog procesa ekstrakcije informacija, kao što je slučaj enciklopedije "*Systematic Bacteriology*", ponovo iskoristiti i primeniti na neki drugi tekst iz iste oblasti.

Treće, i možda najvažnije, u okviru opisanog istraživanja dobijena baza podataka je trebalo da bude upotrebljena kao komplementarni resurs za dalja istraživanja iz oblasti bioinformatike i mikrobiologije, pa je bilo od presudnog značaja da dobijeni podaci budu što je moguće tačniji, tj. da preciznost IE procesa bude velika. Rezultati pokazuju da su upravo transduktori omogućili mali procenat greške među podacima, pa dobijena baza predstavlja pouzdan resurs.

4.2.4. Druga faza: ekstrakcija informacija i njihovo smeštanje u rezultujuću bazu podataka

U okviru druge faze sistem uzima iz slogova baze podataka delove teksta u nestrukturiranom obliku, u kojima se nalaze informacije o entitetima (vrednosti pojedinih atributa) i analizira ih pomoću grafova transduktora. Za svaki atribut koji je potrebno izvući iz teksta kreira se poseban transduktor. Izlaz koji taj transduktor produkuje nakon primene na tekstualni opis, smešta se u bazu podataka kao vrednost odgovarajućeg atributa za odgovarajući slog. Ovaj postupak je šematski prikazan na slici 34.



Slika 34. Ilustracija primena transduktora T_1, T_2, ..., T_n za ekstrakciju informacija

Kao što je već pomenuto, za kreiranje transduktora korišćen je programski paket UNITEX. Isti programski paket je korišćen i za primenu pojedinačnih transduktora na tekst, kao i za prethodnu obradu teksta.

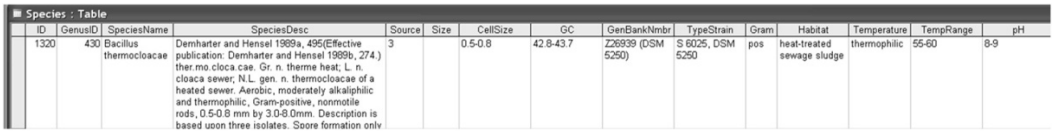
U okviru pripremne obrade tekstualnih opisa izvršena je tokenizacija, normalizacija i primena leksičkih resursa (elektronski rečnik engleskog jezika), kako bi u transduktorima bilo moguće koristiti leksičke i morfološke maske.

Imajući u vidu do sada opisan postupak, pseudo kod druge faze izgleda ovako:

```
for each record in the table “Species”
        read the description from the record
        for each transducer tᵢ
                apply transducer tᵢ onto description
                insert the output of the transducer tᵢ into attribut aᵢ column
        end for
end for
```

Konkretna implementacija druge faze metoda, korišćena u okviru ovog istraživanja, data je u prilogu 1, gde je prikazan kod klase *EkstrakcijaInformacija*, u okviru koje se vrši primena transduktora na opise iz tabele *Genus* i ubacivanje izdvojenih informacija u tabelu.

Kao rezultat, nakon druge faze rezultujuća baza podataka je popunjena podacima, pa je tabela *Species* izgledala kao na slici 35.



Species : Table

ID	GenusID	SpeciesName	SpeciesDesc	Source	Size	CellSize	GC	GenBankNmbr	TypeStrain	Gram	Habitat	Temperature	TempRange	pH
1320	430	Bacillus thermocloacae	Demharter and Hensel 1989a, 495(Effective publication: Demharter and Hensel 1989b, 274.) ther.mo.cloca.cae. Gr. n. therme heat; L. n. cloaca sewer; N.L. gen. n. thermocloacae of a heated sewer. Aerobic, moderately alkaliphilic and thermophilic, Gram-positive, nonmotile rods, 0.5-0.8 mm by 3.0-8.0mm. Description is based upon three isolates. Spore formation only	3		0.5-0.8	42.8-43.7	Z26939 (DSM 5250)	S 6025, DSM 5250	pos	heat-treated sewage sludge	thermophilic	55-60	8-9

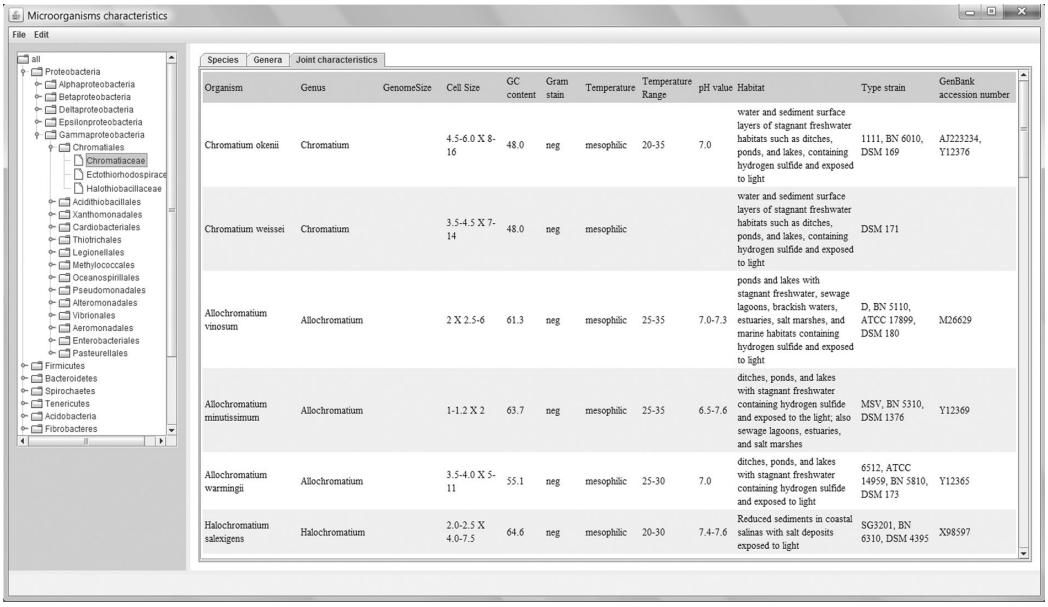
Slika 35. Tabela *Species* nakon završene druge faze procesa ekstrakcije informacije

4.2.5. Struktura rezultujućih podataka

Nakon primene modifikovanog dvofaznog metoda na enciklopedijski tekst, svi ekstrahovani podaci su smešteni u relacionu bazu podataka. Pri tom je korišćena Microsfot Access baza za čuvanje i manipulisanje podacima, mada podaci mogu biti veoma lako i jednostavno izvezeni u bilo koji format baze podataka.

Za svaki mikroorganizam izvučeni su sledeći podaci iz opisa vrsta: veličina genoma, veličina ćelije, sadržaj G+C, GenBank pristupni broj, *Type strain, Gram stain,* stanište, optimalna temperatura, raspon temperature i raspon pH vrednosti. Neki od podataka o krakteristikama vrste su se nalazili u opisima rodova kojima te vrtse pripadaju, pa su isti transduktori primenjeni i na opise u tabelama *Genus* i *GenusIncSed*.

Kako bi se omogućio bolji pregled podataka i iskombinovale vrednosti iz tabela *Species* i *Genus (*ili *GenusIncSed)*, razvijen je jednostavan korisnički interfejs prikazan na slici 36. Ovaj korisnički interfejs omogućava posmatranje grupa podataka, i to tako što je korisniku omogućeno da odabere taksonomsku grupu u panelu na levoj strani aplikacije, nakon čega se na desnoj strani prikazuju podaci organizama koji pripradaju odabranoj grupi. U desnom panelu aplikacije postoje tri kartice, nazvane "*Species*", "*Genera*" i "*Joint characteristics*". Kartica "*Species*" prikazuje podatke iz tabele *Species*, kartica "*Genera*" prikazuje podatke iz tabele *Genus*, a kartica "*Joint characteristics*" kombinuje podatke iz obe prethodno pomenute tabele: ako postoji vrednost nekog atributa u tabeli Species, ona se prikazuje; ako ne postoji, prikazuje se vrednsot istog atributa iz tabele Genus.



Slika 36. Korisnički interfejs za pregledanje podataka. Podaci o organizmima iz taksonomske grupe odabrane u panelu sa leve strane, se prikazuju u panelu sa desne strane. Korisnik može da posmatra podatke o vrstama, rodovima ili združene podatke, selektujući jednu od tri kartice sa desne strane.

4.3. Evaluacija dvofaznog metoda baziranog na konačnim transduktorima

Postupak ekstrakcije informacija o mikroorganizmima iz teksta enciklopedije "*Systematic Bacteriology*" je sem kreiranja baze podataka koja bi se koristila u budućim biološkim istraživanjima, imao za cilj i da izvrši evaluaciju dvofaznog metoda koji je tom prilikom korišćen. Imajući u vidu pomenute ciljeve, izvršena je procena sledećih stavki:

- količina dobijenih podataka – opisani proces ekstrakcije informacija je trebalo da porizvede bazu podataka sa dovoljnom količinom podataka kako bi neka buduća istraživanja mogla da budu izvedena nad njom

- tačnost podataka (preciznost metoda) – dobijeni podaci je trebalo da budu što je moguće tačniji i pouzdaniji

- odziv metoda – zbog uspešnosti same metode bilo je važno postoći dobar odziv, tj. pronaći što više podataka u nestrukturnom tekstu

- ekonomski aspekt celokupnog procesa - da li je proces kreiranja baze podataka dvofaznim metodom ekstrakcije informacija ekonomski opravdan, tj. jeftin, brz i nezahtevan što se tiče resursa

4.3.1. Količina dobijenih podataka

Iako je deo informacija izgubljen tokom procesiranja (tokom konverzije iz *.pdf* u *.txt* format ili zbog izraza koje transduktori nisu prepoznali), većina podataka je uspešno ekstrahovana iz enciklopedijskog teksta i transformisana u strukturni format, koji jednostavno može biti eksportovan u široki raspon formata baza podataka.

Rezultujuća baza podataka je sadržala 2412 slogova o vrstama i 873 slogova o rodovima mikroorganizama, sa opisima izvučenim iz teksta i sačuvanim u okviru baze podataka. Razvijeno je 10 transduktora i primenjeno na opise u tabeli *Species*. S obzirom da neke karakteristike nisu postojale u opisima rodova (GenBank pristupni broj, *type strain*, veličina genoma), 7 preostalih transduktora je primenjeno na opise u tabeli *Genus*.

Tabela 3 prikazuje broj atributa ektsrahovanih od strane svakog transduktora iz tabela *Species* i *Genus*. Kolona *Joint Characteristics* prikazuje koliko vrednosti je raspoloživo za svaku karakteristiku ukoliko se podaci iz ove dve tabele združe.

Tabela 3. Broj dobijenih vrednosti od strane svakog transduktora primenjenog na tabele *Species* i *Genus*; treća kolona prikazuje broj združenih rezultata

Characteristic	*Table Species*	*Table Genus*	*Joint Characteristics*
Size	54	-	54
CellSize	1136	529	1893
GC	2091	755	2364
GenBankNmbr	2014	-	2014
TypeStrain	2343	-	2343
Gram	488	742	2157
Habitat	1633	284	1837
Temperature	465	257	921
TempRange	648	170	1072
pH	730	190	1060

Broj ekstrahovanih vrednosti se razlikuje za različite transduktore, pa ipak verujemo da ovako dobijena baza ima veliki potencijal kao resurs za istraživanja u

oblasti odnosa i veza između genotipskih i fenotipskih karakteristika. Postoje primeri uspešnih istraživanja koja su izvedena na mnogo manjim uzorcima (86 sekvencionisanih genoma u (Kim i sar., 2004), 182 sekvence gena u (Root i sar., 2011), i 891 sekvenci gena bakterija i arhea u (Roske i sar., 2010)). Sve ove studije su pokazale veoma dobre rezultate i korist od postojanja i manjih količina podataka, nego što je to slučaj sa rezultujućom bazom podataka dobijenom u navedenom istraživanju.

4.3.2. Preciznost i odziv metoda

S obzirom da je enciklopedija koja je korišćena kao izvor podataka veoma obimna, i sadrži skoro 5000 strana, nije moguće tačno izračunati preciznost i odziv metoda. Umesto toga, uzet je slučajan uzorak od 100 slogova vrsta, za potrebe evaluacije. Ovi slogovi su pojedinačno analizirani. Upoređeni su njihovi opisi u enciklopediji i vrednosti atributa koje su se nalazile u tim opisima, sa vrednostima atributa koje su izdvojene u bazu podataka, a koji su dobijeni automatski od strane sistema za ekstrakciju informacija. Rezultati ove analize prikazani su u tabeli 4.

Tabela 4. Preciznost i odziv transduktora

Transduktor	Size	Cell Size	GC	Gen Bank Nmbr	Type Strain	Gram	Habitat	Temperature	Temp Range	pH
Preciznost	0.92	0.91	1.00	1.00	0.99	1.00	0.97	1.00	0.84	0.81
Odziv	0.90	0.96	0.96	1.00	0.96	1.00	0.79	0.69	0.66	0.77

Preciznost je bila veoma visoka, što rezultujuću bazu čini vema pouzdanim resursom za dalja biološka istraživanja. Ovako velika preciznost je posledica činjenice da su transduktori dizajnirani od strane eksperata kako bi izvlačili vrednosti pojedinih atributa, i stoga prepoznaju samo one informacije koje su se nalazile u opisanom kontekstu.

Odziv se razlikovao za različite transduktore, zavisno do kompleksnosti konteksta u kome inforamcija može da se nađe. Na primer, transduktor za Gram osobinu organizama je bio veoma efikasan; ekstrahovao je korektno sva pojavljivanja ovog atributa u tekstu. Neki drugi transduktori nisu bili tako

efikasni, kao što je slučaj sa transduktorom za stanište mikroorganizma (*Habitat*). Međutim, daljim podešavanjem, modifikacijom i proširivanjem transduktora, kako bi oni prepoznali i pojavljivanja koja nisu prepoznata u prvom prolazu, proces ekstrakcije informacije bio bi poboljšan i podignut na željeni nivo efikasnosti. Ovo je svakako jedna od važnih prednosti metoda zasnovanih na konačnim transduktorima u odnosu na druge metode ekstrakcije informacije, posebno na one bazirane na probabilističkim modelima (Burns i sar., 2007; Feng i sar., 2007; Freitag i Mccallum, 2000; Mccallum i sar., 2000; Ray, 2001; Peng i Mccallum, 2006; Zhong i sar., 2007; Zhu i sar., 2005).

4.3.3. Ekonomski aspekti procesa kreiranja baze podataka metodima ekstrakcije informacije

Glavni cilj opisanog istraživanja bio je kreiranje iscrpnog izvora podataka sa minimalnim brojem radnih sati, kako bi ceo proces bio ekonomski isplativ. Iako nije moguće tačno izračunati, procenjuje se da je ceo proces kreiranja baze podataka trajao 20 radnih dana ili 160 radnih sati. Još detaljnije, to znači da je bilo potrebno 576000 sekundi za popunjavanje baze podataka sa 2412 slogova u jednoj tabeli (tabela *Species*) i 873 u drugoj (tabela *Genus*). Imajući u vidu da je za svaku vrstu bilo potrebno ili pronaći vrednost nekog atributa u tekstu i upisati ga u bazu ili utvrditi da se vrednost tog atributa ne nalazi u tekstu, kao i da je ovih atributa bilo 10 za vrste i 7 za rodove, lako se može izračunati da je na ovaj način određena vrednost za 30231 atribut.

Ukoliko bi ovi podaci bili uneti ručno umesto korišćenjem automatske ekstrakcije informacija, bilo bi potrebno odrediti iz teksta i upisatu u bazu vrednost jednog atributa za manje od 20 sekundi, što je potpuno nemoguće.

Dakle, za 20 radnih dana dobili smo veoma iscrpnu bazu podataka o mikroorganizmima sa zadovoljavajućom tačnošću podataka. Postojala je mogućnost da se na kreiranje transduktora potroši više vremena, što bi dovelo do veće efikasnosti procesa ekstrakcije. Stvar je procene istraživača da li je i koliko je isplativo vremena provesti na kreiranju i naknadnom modifikovanju

transduktora, imajući u vidu da se na taj način povećava ukupno vreme, pa samim tim i troškovi procesa, ali da se povećava preciznost i odziv procesa.

Glava 5

5 EKSTRAKCIJA INFORMACIJA IZ TEKSTOVA NA SRPSKOM JEZIKU

5.1. Specifičnost srpskog jezika

Kao što je struktura dokumenta iz koga se izdvaja informacija od velikog značaja za proces ekstrakcije informacija, još više na proces izdvajanja utiču osobine jezika na kome je tekst pisan. Srpski jezik, i drugi njemu slični jezici, zahtevaju poseban pristup kada se vrši bilo kakva obrada tekstova pisanih ovim jezikom, i to sa tri vrlo bitna stanovišta: način kodiranja, morfološke osobine i postojanje elektronskih resursa. I prilikom ekstrakcije informacija o tome treba voditi računa.

5.1.1. Način kodiranja karaktera

Srpski jezik je jedan od retkih jezika koji ravnopravno koristi dva alfabeta, ćirilicu i latinicu. U poslednje vreme na vebu se sve češće susreću i tekstovi na nekim drugim jezicima pisani različitim alfabetima (na primer, ruski tekstovi pisani latinicom), tako da je problem ekstrakcije informacija iz tekstova pisanim različitim alfabetima uočljiv i u nekim drugim situacijama, a ne samo u slučaju srpskog jezika.

Upravo korišćenje različitih alfabeta dovodi i do upotrebe različitih kodnih šema za predstavljanje istog teksta ili iste informacije. Međutim, različito kodiranje je moguće i u nekim drugim slučajevima, što ilustruje sledeći primer.

Primer. Primer je preuzet iz (Vitas i Pavlović-Lažetić, 2007).

U tekstovima na srpskom jeziku zapisivanje toponima *New York* moguće je na više načina. Prvo, korišćenje različitih pisama dovodi do dva različita oblika:

Njujork i *Њујорк*. Za ćirilično i latinično pismo u upotrebi su različite kodne šeme. Takođe, u latiničnom pismu digraf "Nj" ima dve interpretacije, kao jedan karakter koji odgovara ćiriličnom њ, ili kao grupa suglasnika N+j. Korišćenjem *Unicode* kodiranja, slovo Nj može biti predstavljeno kao digraf pomoću dva koda ili kao ligatura pomoću jednog koda. Dalje, slovo Nj može biti zapisano kao Nj ili NJ, pa postoji četiri različita načina za predstavljanje slova Nj koristeći *Unicode* kodiranje.

♦

Dodatno, postojeći elektronski resursi za obradu srpskog jezika, na prvom mestu elektronski rečnici (Vitas i sar. 2003), su iskodirani na osnovu latiničnog alfabeta i ASCII koda, pri čemu se slova sa dijakritičkim karakterima koja nedostaju u ASCII kodu zapisuju kao dvoslovne kombinacije (*š* kao *sx, đ* kao *dx, č* kao *cy, ć* kao *cx, ž* kao *zx, nj* kao *nx, lj* kao *lx, dž* kao *dy*).

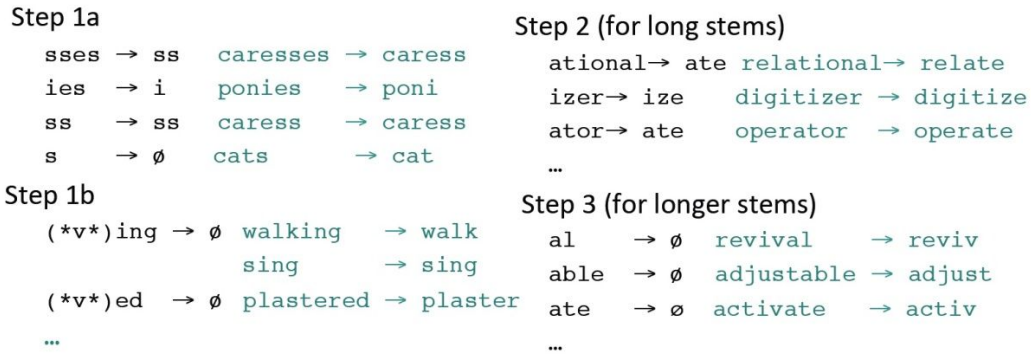
5.1.2. Morfološke osobine

Srpski jezik spada u jezike sa veoma bogatom morfologijom. Zbog toga često postoji velika razlika u efikasnosti algoritama za rešavanje problema iz oblasti obrade prirodnih jezika. Pojedini algoritmi daju odlične rezultate kada se primene na tekstove na engleskom jeziku, ali veoma loše kada se radi o tekstovima na jeziku sa bogatom morfologijom, kakav je i srpski jezik.

U nekim slučajevima primena algoritama već razvijenih za engleski jezik praktično nije moguća kada je u pitanju srpski. Primer takvog algoritma je Porterov algoritam (*Porter Stemming Algorithm)* namenjen za svođenje reči na njen koren (engl. *stemming*) (Porter, 1980). Svođenjem reči na koren bi engleske reči *automate, automates, automatic, automation* sve bile svedene na nisku *automat*.

Porterov algoritam je izvorno nastao za potrebe IR sistema za tekstove na engleskom jeziku, jer se pošlo do pretpostavke da će sve reči sa istim korenom imati i slično značenje (*connect, connected, connection*...). I danas se često koristi u velikom broju sistema koji obrađuju tekstove na engleskom jeziku. Osnovni

koncept Porterovog algoritma jeste da sukcesivno zamenjuje sufikse reči, ili nekom drugom niskom, ili praznom niskom, dok ne dođe do korena reči. Na slici 37. ilustrativno je prikazano nekoliko koraka Porterovog algoritma.



Slika 37. Koraci Porterovog algoritma sa primerima zamene sufiksa

Suštinski, Porterov algoritam eksplicitno navodi pravila zamene i pri tom koristi neke dodatne uslove za situacije kada se pravila ne primenjuju kako bi poboljšao efikasnost (na primer, u slučaju reči *king* ili *thing* ne vrši se odbacivanje nastavka *ing*). Ovi uslovi nisu bazirani na lingvističkim pravilima, već su procenjeni iskustveno i uglavnom proveravaju dužinu preostale osnove reči, nedopuštajući da bude prekratka.

Ovakav pristup za obradu srpskog jezika je neprikladan. Eksplicitno navođenje svih mogućih flektivnih i derivacionih pravila srpskog jezika, uzimajući u obzir sve ili bar većinu situacija, je praktično neizvodljiv. Većina prideva u mnogim slovenskim jezicima, pa i u srpskom jeziku može da poprimi i preko 40 različitih oblika. Pored toga, postoji problem velikog broja višeznačnih flektivnih sufikasa, tj. sufikasa koji, zavisno o tome s kojim se osnovama kombinuju, označavaju različite morfosintaktičke jedinice. Na primer, flektivni sufiks *–e* se koristi za formiranje genitiva jednine većine imenica ženskog roda (*vode, ruke*). Međutim, isti sufiks kod imenica muškog roda može da izražava akuzativ množine (*vojnike, zakone*). Zbog ove višeznačnosti svi slovenski jezici su bogati homografima, tj. oblicima reči koji su flektivno povezani sa dve ili više različite lekseme.

Dodatno, u obradi tekstova na srpskom jeziku postoje i problemi prouzrokovani glasovnim promenama na granicama morfema. Tako na primer,

dok sufiks *-e* u obliku *vojnike* označava akuzativ množine, u reči *vojniče* isti sufiks se koristi u kombinaciji sa glasovno izmenjenom osnovom kako bi označio vokativ jednine.

U kontekstu računarske obrade teksta na srpskom jeziku, pa i ekstrakcije informacija, ovako veliki nivo morfološke složenosti veoma je problematičan.

5.1.3. Nedostatak elektronskih resursa za srpski jezik

Lingvistički resursi za srpski jezik su razvijeni u okviru Grupe za jezičke tehnologije[7] formirane na Matematičkom fakultetu Univerziteta u Beogradu. Do danas je razvijen određen broj različitih resursa (Krstev, 2008), među kojima su najznačajniji jednojezički i višejezički korpusi, sistem morfoloških rečnika srpskog jezika i semantička mreža za srpski jezik (Pavlović-Lažetić, 2006). Iako je uložen veliki napor da ovi resursi budu što obimniji, njihov obim je ipak relativno mali u poređenju sa nekim drugim jezicima.

Za jezike sa velikim govornim područjem, kakav je na primer engleski, franscuski ili španski, postoji daleko veći broj razvijenih jednojezičkih i višejezičkih korpusa, koji su pri tom i mnogo obimniji. Poređenja radi, paralelni srpsko – engleski korpus sadrži oko 2 miliona reči, a srpsko – francuski oko milion (Vitas i sar. 2003), dok sa druge strane paralelni korpus *JRC-Acquis* (Steinberger i sar. 2006) trenutno sadrži povezane tekstove prevedene na 22 jezika iz preko 4 miliona dokumenata, sa preko 60 miliona reči za engleski, španski i francuski (po jeziku). Paralelni korpus *Europarl,* koji sadrži tekstove iz zapisnika Evropskog parlamenta (Koehn, 2005), prvobitno je obuhvatao paralelne tekstove na 11 evropskih jezika, a kasnije je proširen sa još 10 jezika. Sadrži preko 50 miliona reči za svaki jezik.

Uopšte, količina tekstova na srpskom jeziku u elektronskoj formi, bez obzira da li su lingvistički anotirani ili ne, a koja je dostupna istraživačima, mnogo je manja u poređenju sa količinom tekstova na nekom od svetskih jezika kakav je engleski.

[7] http://www.korpus.matf.bg.ac.rs/index.html

Primer. Tačan odnos količine elektronskih tekstova na srpskom i na engleskom jeziku, dostupnih preko Interneta, nije moguće utvrditi. Međutim, dobar pokazatelj tog odnosa može da bude i broj rezultata koje vraća pretraživač Google[8] za iste ključne reči na srpskom i na engleskom jeziku. Tako je na primer, za ključnu reč „*family*“ Google vratio oko 6 milijardi rezultata, dok je za ključnu reč „*porodica*“ vratio 9 miliona, a za ključnu reč „*familija*“ 3 miliona. Slično, pretraga po ključnoj reči „*agency*“ vratila je 1 milijardu rezultata, dok je pretraga „*agencija*“ vratila oko 30 miliona.

♦

Zbog svega navedenog, srpski jezik spada u grupu jezika u literaturi često nazivanih *less resourced languages*.

5.1.4. Pristup ekstrakciji informacija iz tekstova na srpskom jeziku

Specifičnosti srpkog jezika, opisane u ovom poglavlju, u velikoj meri određuju koji će pristup i koji metod biti odabran za ekstrakciju informacija iz tekstova pisanih srpskim jezikom.

Bogat morfološki sistem srpskog jezika zahteva upotrebu korišćenja dodatnih lingvističkih resursa, kao što su elektronski rečnici i gramatike kojima se opisuju gramatička pravila jezika, za procesiranje teksta. Jedino na taj način je moguće razviti sisteme za ekstrakciju informacija koji bi bili efikasni kada se primene na tekstove na srpskom jeziku.

Nedostatak resursa, na prvom mestu anotiranih korpusa koji bi se koristili za metode mašinskog učenja, nameću upotrebu metoda baziranih na konačnim modelima, pre nego metoda zasnovanih na verovatnoći.

U nastavku ovog poglavlja biće opisan jedan proces ekstrakcije informacija iz tekstova na srpskom jeziku, koji ima za cilj da demonstrira na koji način je moguće prevazići navedene probleme prouzrokovane specifičnošću srpskog jezika.

[8] http://www.google.com

5.2. Ekstrakcija informacija o meteorološkim pojavama

Tekstovi o vremenskim prilikama na nekom užem ili širem lokalitetu, bez obzira da li se radi o vremenskoj prognozi ili o osmotrenim podacima, istraživani su tokom niza godina u okviru oblasti kao što su ekstrakcija informacija, istraživanje teksta ili razumevanje teksta (Slocum, 1985; Kononenko i sar. 1999; Kononenko i sar. 2000; Brkić i Matetić, 2007; Labsky i sar. 2007). Ovakva vrsta tekstova je interesantna zbog svojih osobina sa jedne strane, kao i zbog različite mogućnosti upotrebe dobijenih podataka, sa druge strane. Dobijeni podaci su najčešće korišćeni za potrebe nekih drugih informacionih sistema, na primer za automatsko prevođenje sa jednog jezika na drugi, za vizuelizaciju podataka, za objedinjavanje podataka dobijenih iz više izvora i slično.

U okviru ove disertacije biće predstavljen proces ekstrakcije podataka o vemenskim prilikama, koji može biti upotrebljen u različite svrhe (na primer za automatsko kreiranje leksikona ili za automatsku anotaciju teksta). Kao tekstualni resurs korišćena je kolekcija tekstova o vremenskim prilikama prikupljena tokom 2010., 2011. i 2012. godine iz nekoliko izvora na srpskom jeziku. U okviru poglavlja 5.3 korpus tekstova vremenskih prilika je detaljnije objašnjen.

Cilj proces ekstrakcije bio je obeležavanje pojedinačnih informacija koje su se nalazile u jednom tekstualnom opisu. Od interesa su bila tri tipa informacija: lokacija, vreme i meteorološka pojava. Detaljnije o strukturi i semantičkim klasama informacija videti u 5.4. Sve pronađene informacije su označavane u tekstu i na taj način strukturirane. Nije izvršeno njihovo transformisanje u neke druge formate podataka (na primer, relacionu bazu podataka), s obzirom da je taj postupak trivijalan, a pri tome zavisi od potreba daljih istraživanja.

Pravila ekstrakcije su zadavana konačnim transduktorima i mrežama prelaza koje produkuju izlaz. Za kreiranje i primenu transduktora korišćen je programski sistem UNITEX.

5.3. Osobine izvornog tekstualnog resursa

Tekstovi o vremenskim prilikama su prikupljani tokom 2010., 2011. i 2012. godine iz nekoliko izvora (Republički hidrometeorološki zavod Republike Srbije[9], agencija Meteos[10], dnevni list Politika[11], B92[12], SMedia[13] i Internet portal Krstarica[14]). Preuzeto je ukupno 13705 tekstualnih opisa, koji su se sastojali od ukupno 45862 rečenice. Svi opisi su smešteni u bazu podataka, iz koje je kasnije formiran jedan tekstualni dokument sa objedinjenim opisima nad kojim je vršena analiza i obrada.

Većina preuzetih tekstova je sadržala vremensku prognozu za jedan ili više dana, mada je bilo i tekstova u kojima su opisivani osmotreni podaci. S obzirom da su tekstovi preuzimani iz više izvora, njihova struktura je bila dosta heterogena. Pa ipak, postojale su i određene zakonitosti koje su važile za sve tekstove, bez obzira iz kog izvora dolaze.

Sledi primer nekoliko opisa vremenskih prilika:

```
Oblačno i hladnije, povremeno sa kišom koja će krajem dana preći u
susnežicu i sneg. Vetar slab i umeren severni i severozapadni.
Jutarnja temperatura oko 3 °C, najviša dnevna oko 6 °C, tokom noći
u padu.

U Srbiji danas pretežno sunčano, posle podne u brdsko-planinskim
predelima umereno oblačno. Vetar slab, severni. Maksimalna
temperatura od 16 do 23 °C.

Narednih dana pretežno sunčano, temperatura oko 20 °C.
```

Opisi vremenskih prilika su se sastojali od manjih fragmenata (rečenica i delova rečenica) koji su u sebi nosili tri vrste informacije (meteorološku pojavu, lokaciju i vreme), zaokružene u pojedinačnim izjavama. Dakle, svaka jedinica semantičke strukture teksta (pojedinačna izjava) može biti posmatrana kao trojka <lokacija, vreme, pojava>. Tako bi idealnim izdvajanjem iz opisa „Ujutru i pre podne u nižim delovima grada magla ili sumaglica. Tokom dana

[9] http://www.hidmet.gov.rs
[10] http://www.meteos.rs
[11] http://www.politika.rs
[12] http://www.b92.net
[13] http://www.smedia.rs
[14] http://www.krstarica.com

umereno do pretežno oblačno i uglavnom suvo.“ bile izdvojene sledeće izjave:

<"niži delovi grada", "ujutru", "magla ili sumaglica">

<"niži delovi grada", "pre podne", "magla ili sumaglica">

<"-", "tokom dana", "umereno do pretežno oblačno">

<"-", "tokom dana", "suvo">

Izjave su se u tekstualnim opisima međusobno preplitale, bez jasnih granica između dve različite izjave. Postojala su dva različita odnosa u kome su dve izjave mogle da se nađu unutar jednog tekstualnog opisa:

- sukcesivni (redni) odnos – izjave slede jedna drugu „ujutru u Srbiji kiša, tokom dana na severu zemlje razvedravanje“

- rekurentni (preklopljeni) odnos – jedna izjava preuzima neku od informacija od prethodne, najčešće vreme ili lokaciju – „u petak i subotu promenljivo oblačno i svežije mestimično sa kišom, pljuskovima i grmljavinom“

Ovakva semantička struktura zahteva i poseban pristup, semantički orijentisan, kako bi se razrešile koreference između izdvojenih delova, ukoliko je glavni cilj izdvajanje pojedinačnih izjava. Međutim, prvi korak u tom procesu jeste otkrivanje i izdvajanje pojedinačnih obeležja. U ovoj disertaciji biće opisan upravo taj postupak, dok će povezivanje izdvojenih obeležja i njihovih vrednosti u izjave biti predmet budućih istraživanja.

5.3.1. Osobine podjezika vremenskih prilika

Način na koji se vrši opisivanje vremenskih prilika je veoma specifičan i lako prepoznatljiv. Ograničen skup reči prirodnog jezika (u ovom slučaju srpskog, ali slično važi i za druge prirodne jezike) koji se koristi za opisivanje pojava može biti posmatran kao podjezik prirodnog jezika, zajedno sa svojim osobinama:

- ograničena leksika – za opisivanje neke pojave u većini opisa se koriste iste reči, bez upotrebe velikog broja sinonima. Tako je uobičajeno da se kaže da je vreme promenljivo ili nestabilno, a skoro nikad varijabilno, nestalno.

- nepoštovanje gramatičkih i sintaksnih pravila prirodnog jezika – rečenice i izjave u opisima vremenskih prilika po pravilu ne sadrže pomoćne glagole, često nemaju predikat („`Vetar slab, jugoistočni`.“) niti priloge.

- struktura teksta – nije moguće jasno odvajanje izjava samo na osnovu znakova interpunkcije, jedna rečenica često sadrži više izjava, a često se nekoliko rečenica (izjava) spaja u jednu pomoću zareza („`U većem delu promenljivo oblačno, mestimično kratkotrajna kiša, pljuskovi i grmljavina, a u oblasti Sredozemlja i Crnog mora pretežno sunčano i toplo`.“)

Sa jedne strane, postojanje ovakvog podjezika i njegova upotreba olakšavaju proces obrade tekstova, samim tim što su mnoga sintaksna pravila pojednostavljena u odnosu na prirodni jezik. Sa druge strane, upravo nepoštovanje sintaksnih pravila prirodnog jezika onemogućava korišćenje već postojećih elektronskih gramatika koje su za dati prirodni jezik razvijene i dostupne.

5.4. Semantičke klase korišćene za strukturiranje informacija

Informacije koje su se nalazile u tekstualnim opisima vremenskih prilika, a koje su bile od interesa prilikom istraživanja grupisane su u semantičke klase različitog nivoa. Svakom izdvojenom fragmentu iz teksta trebalo je dodeliti neku semantičku klasu, pri čemu su neke od njih sadržale i dodatnu klasifikaciju. Hijererhija klasa prikazana je u tabeli 5.

U okviru procesa ekstrakcije informacija koji će biti opisan u ovoj disertaciji cilj je bio identifikovati segmente teksta koji su nosioci nekog od gore definisanih obeležja. Obeležavanje prepoznatog segmenta teksta i semantičke klase koja mu je dodeljena vršeno je umetanjem posebnih oznaka direktno u tekst.

Za nazive oznaka korišćene su semantičke klase prikazane u koloni *Obeležje* u tabeli 5. Njihovo objedinjavanje u klase višeg nivoa, kao i razrešenje koreferenci među njima biće predmet daljih istraživanja. Korišćene oznake su imale sledeću sintaksu:

```
<obelezje>segment teksta</obelezje>
```

Tabela 5. Hijerarhija klasa korišćenih za strukturiranje informacija izdvojenih iz teksta

Tip informacije	Element	Obeležje	Vrednosti
Meteo	Padavine	TipPadavina	*kiša, sneg, susnežica, grad ...*
		ObimPadavina	*slaba, jaka, ...*
	Oblačnost	PrisustvoOblaka	*sunčano, oblačno*
		ObimOblačnosti	*promenljivo, potpuno, delimično..*
	Vetar	PravacVetra	*jugoistočni, severni ...*
		JačinaVetra	*jak, slab...*
		BrzinaVetra	*16 m/s*
	Temperatura	Temperatura	*12 stepeni, 12 C, dva stepena, ispod nule ...*
		KatTemperature	*najviša, jutarnja ...*
		OpisTemperature	*hladno,toplije, porast ...*
	Pojava	TipPojave	*magla, grad, oluja ...*
Lokacija	Teritorija	ImeTeritorije	*Srbija, Evropa, Beograd ...*
		DeoTeritorije	*severoistok, južni delovi ...*
	Lokalitet	Lokalitet	*na planinama, u kotlinama, lokalno ..*
Vreme	Dan	Datum	*15. januar*
		ImeDana	*ponedeljak, utorak ..*
		DeoDana	*ujutru, posle podne*
	Period	Period	*sledeće nedelje, tokom februara*

U primeru koji sledi prikazan je deo teksta pre i posle izdvajanja informacija.

Primer.

Izvorni tekst pre obrade:

```
U većem delu promenljivo oblačno, mestimično kratkotrajna kiša,
pljuskovi i grmljavina. Na Pirinejskom poluostrvu, na jugu Balkana
i jugoistoku kontinenta pretežno sunčano.
```

Isti segment teksta nakon obrade:

```
<lokalitet>U većem delu</lokalitet>
<obimOblacnosti>promenljivo</obimOblacnosti>
<prisustvoOblaka>oblačno</prisustvoOblaka>,
<lokalitet>mestimično</lokalitet>
<obimPadavina>kratkotrajna</obimPadavina> <tipPadavina>kiša
</tipPadavina>, <tipPojave>pljuskovi</tipPojave> i
```

```
<tipPojave>grmljavina</tipPojave>. Na <imeTeritorije>Pirinejskom
poluostrvu</imeTeritorije>, <deoTeritorije>na jugu</deoTeritorije>
<imeTeritorije>Balkana</imeTeritorije> i
<deoTeritorije>jugoistoku</deoTeritorije>
<imeTeritorije>kontinenta</imeTeritorije>
<obimOblacnosti>pretežno</obimOblacnosti>
<prisustvoOblaka>sunčano</prisustvoOblaka>.
                                ♦
```

5.5. Korišćeni elektronski resursi

Zbog već opisanih specifičnosti srpskog jezika, koje bitno utiču na efikasnost sistema za obradu tekstova, u procesu izdvajanja meteoroloških podataka korišćen je UNITEX, kao alat za kreiranje kolekcije transduktora kojima se opisuju pravila ekstrakcije. Pomoću UNITEX-ovih programa za obradu teksta, na tekstove o vremenskim prilikama je dodatno primenjen elektronski rečnik za srpski jezik (Krstev i Vitas, 2005; Vitas i sar., 2003).

Elektronski rečnik za srpski jezik napisan je u DELA formatu (opisan ranije u poglavlju 3.6.1). Sadrži reči srpskog jezika, zajedno sa vlastitim imenicama, grupisane kao proste reči ili kao složenice. Istraživači Grupe za jezičke tehnologije Matematičkog fakulteta, koji su kreirali ovaj rečnik, navode u (Krstev i sar. 2011) da rečnik sadrži ukupno 125,269 lema prostih reči i 4,378,245 oblika prostih reči, kao i 5,251 lemu složenica i 106,731 oblika složenica. Od toga, oko 35,000 lema se odnosi na vlastite imenice (imena geopolitičkih pojmova, srpska imena ljudi i strana imena ljudi).

Za svaki oblik reči koji se nalazi u rečniku navedena je njena lema, zatim kodovi koji označavaju različite gramatičke kategorije (vrstu reči, rod, broj i dr.), kao i različite oznake koje označavaju derivaciona, sintaksna ili neka druga obeležja leme. Sledi jedna od linija u rečniku:

```
Evropi,Evropa.N+NProp+Top:fs3q:fs7q
```

Kod `N` označava da se radi o imenici (engl. *noun*), kod `NProp` označava da se radi o vlastitoj imenici (engl. *proper noun*), a kod `Top` označava da se radi o toponimu. Kodovi `fs3q` i `fs7q` bliže označavaju svojstva oblika reči (`f` – ženski

rod, `s` – jednina, `3` – treći padež, `q` – neživa). Lista svih gramatičkih kategorija koje su korišćene elektronskom rečniku srpskog jezika, zajedno sa listom kodova kojima su označavane reči data je u (Krstev, 2008). Za obeležavanje vlastitih imenica, skup oznaka se oslanja na (Grass i sar. 2002).

Postojanje ovakvog tipa informacija u rečnicima omogućava da se u okviru UNITEX-a koriste leksičke maske koje se odnose na stavke u rečniku (poglavlje 3.6). Na primer, leksička maska `<N+NProp+Top>` bi odgovarala svim oblicima reči koje su označene navedenim kodovima, tj. svim imenicama (kod `N`), i to vlastitim (kod `NProp`), koje su označene kao toponimi (`Top`). Ovakve maske su korišćene u okviru procesa ekstrakcije podataka o vremenskim prilikama, kako bi se odredila lokacija na koju se odnosi pojedina pojava ili koja se pominje u tekstu. Bez upotrebe rečnika, efikasno izdvajanje takvog tipa podataka ne bi bilo moguće.

5.6. Transduktori – pravila ekstrakcije

Za svako obeležje koje označava po jednu semantičku klasu podataka, dato u tabeli 5 unutar poglavlja 5.4, kreiran je po jedan transduktor ili RTN koji opisuje pravilo izdvajanja informacije iz teksta, a koja odgovara datom obeležju. U prvoj fazi istraživanja, pravila su primenjivana kroz softverski sistem UNITEX, pri čemu je strukturiranje podatka vršeno obeležavanjem segmenata u tekstu koji su nosioci informacija. Za oznake su korišćeni nazivi obeležja dati u tabeli 5. Za sada nije vršeno njihovo spajanje u informacije višeg nivoa niti razrešenje koreferenci. Kreirani transduktori mogu jednostavno da se modifikuju kako bi vršili označavanje na neki drugi način, u zavisnosti od potrebe.

Primena transduktora je vršena sekvencijalno, jedan po jedan. Za većinu kreiranih transduktora je bilo svejedno u kom redosledu će biti primenjeni, mada je moguće proces ekstrakcije koncipirati tako da se uzastopnom primenom transduktora poboljša efikasnost procesa (kaskadna primena transduktora pri čemu jedan transduktor koristi rezultate prethodno primenjenih (Friburger i Maurel, 2004)). U okviru ovog poglavlja biće predstavljeni neki od korišćenih transduktora, pri čemu je u poglavlju 5.6.3. demonstriran i slučaj kada je važan redosled primene transduktora.

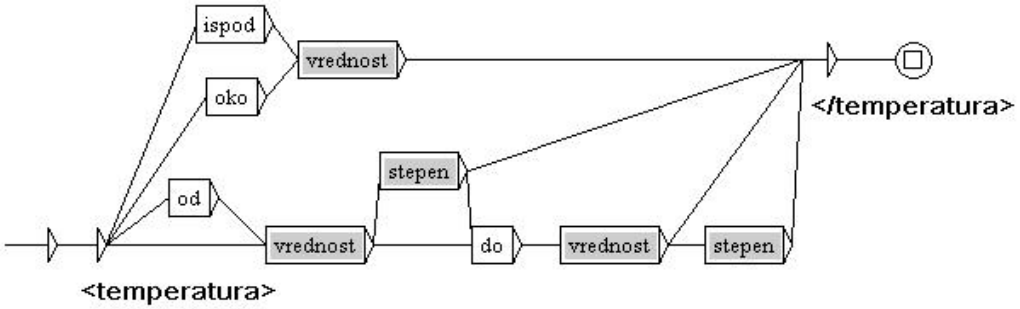
5.6.1. Transduktori za izdvajanje informacija o temperaturi

Podaci o temperaturi su u tekstu bili predstavljani na dva različita načina:

- vrednosno (*12 stepeni, 12 C, dva stepena, ispod nule, minus 5 ...)*

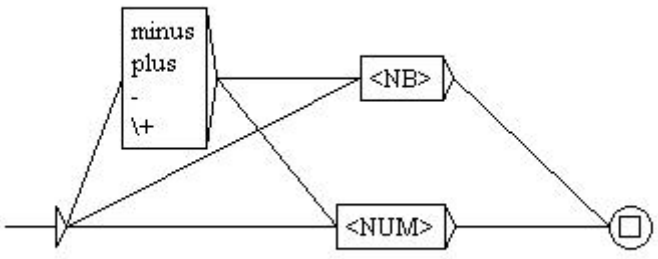
- opisno (*hladno, hladnije, toplo, toplije, pad temperature, temperatura u porastu ...*)

Za svaki od načina predstavljanja temperature kreirano je posebno pravilo ekstrakcije. Ovde će biti opisana pravila (transduktori) za izdvajanje vrednosnog predstavljanja temperature. Na slici 38. prikazan je glavni transduktor (*temperatura.grf*) u okviru RTN za izdvajanje informacije o temperaturi koja je predstavljena vrednosno.



Slika 38. Glavni transduktor *temperatura.grf* u okviru RTN za izdvajanje informacije o temperaturi

Sivom bojom su označeni pozivi podgrafova. Podgraf *vrednost* prepoznaje različite izraze kojima se prikazuje konkretna vrednost (broj stepeni) temperature. Ovaj pogdgraf je prikazan na slici 39.



Slika 39. Podgraf *vrednost.grf* koji prepoznaje numeričke vrednosti zapisane brojevima ili tekstualno

Oznaka <NB> prepoznaje neprekidni niz cifara. Leksička maska <NUM> prepoznaje sve reči koje su u rečniku označene kodom NUM (*jedan, dva, tri,...*). Tako ovaj podgraf prepoznaje, između ostalih sledeće izraze:

- *10, minus dva, +5, jedanaest ...*

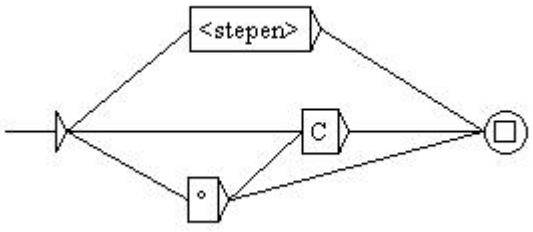
NAPOMENA. Graf sa slike 39. je prepoznavao i token „*C*“ kao broj, s obzirom da u elektronskom rečniku postoji linija koja se odnosi na rimsku cifru C:

```
C,.NUM+Roman
```

S obzirom da su rimske cifre u elektronskom rečniku označene kodom Roman, tada bi bilo moguće koristiti masku <NUM~Roman>, koja bi vratila sve reči obeležene kodom NUM, a koje nisu obeležene kodom Roman. Operator ~ se koristi u UNITEX-u kao negacija nekog gramatičkog koda počevši od verzije 2.1. Prethodne verzije su za negaciju koristile znak minus (-). U novoj verziji UNITEX-a moguće je koristiti stare grafove bez izmene ovog operatora jedino ako se direktno poziva program Locate sa opcijom –g minus.

Sa druge strane, kada se graf *vrednost.grf* upotrebljava u okviru grafa *temperatura.grf*, a s obzirom da se tada zahteva i pojavljivanje brojeva u određenom kontekstu, rimski brojevi se ne pojavljuju među rezultatima, pa je zbog toga graf *vrednost.grf* ipak korišćen u obliku predstavljenom na slici 39:

Glavni transduktor *temperatura.grf* (slika 38) sadrži i poziv podgrafa *stepen.grf*. Ovaj graf je namenjen za prepoznavanje izraza kojima se opisuje stepen Celzijusove skale, kao uobičajene jedinice mere temperature u tekstovima na srpskom jeziku. Prikazan je na slici 40.



Slika 40. Podgraf *stepen.grf* koji prepoznaje izraze za označavanje stepena Celzijusove skale

Upotreba leksičke maske koja se odnosi na reč iz rečnika (`<stepen>`) omogućava da bude prepoznat bilo koji oblik reči *stepen* (*stepena*, *stepeni*, *stepenima* i sl.).

Graf *temperatura.grf* je napravljen tako da izdvaja celu sekvencu koju je prepoznao, pa se na taj način jezik prepoznatih izraza (L_{prep}) i jezik izdvojenih izraza (L_{izdv}) u slučaju ovog grafa poklapaju. Neke od fraza koje pripadaju ovim jezicima, a koje su se nalazile u tekstu su:

oko +8 °C
- 1C
- 30 °C
- 4 stepena
-1 C do 1 C
-1 do +3 stepena
-12 do -8
11 do 15 stepeni
11 stepeni
od 15 do 18
od 15 do 19 °C
od pet do devet stepeni
od sedam do 10 stepeni
od tri do osam stepeni
oko +2
oko četiri
oko minus 12
oko plus tri
ispod 0
ispod deset

Prilikom kreiranja grafa *temperatura.grf* uzeto je u obzir da se u velikom broju slučajeva ne navodi reč „*stepen*“ niti neka druga oznaka jedinice za meru temperature, već se ona podrazumeva. Posebno u rečenicama u kojima se navodi nekoliko vrednosti temperature, česte su situacije da se uz prvu vrednost navede jedinica mere, a uz sledeću ne (na primer, „*jutarnja temperatura oko 4 stepena, maksimalna dnevna oko 15.*“). Ova činjenica otežava proces ekstrakcije, jer je potrebno napraviti kompromis između toga da deo informacija ostane neprepoznat (ukoliko se zahteva eksplicitno pominjanje jedinice mere da bi informacija bila ekstrahovana) ili da se prepoznaju izrazi koji ne predstavljaju temperaturu (na primer, „*vidljivost je oko 200 m*“).

Inicijalno, transduktor *temperatura.grf* je bio kreiran tako da prepoznaje i izraze oblika „*preko 30*“. Međutim, povećanje odziva koje je ostvareno ovakvim proširenjem je bilo malo u odnosu na smanjenje preciznosti. Naime, pojavila su se svega tri rezultata koja počinju rečju „*preko*“, a odnose se na temperaturu, a veoma veliki broj (preko 200) rezultata koji su pogrešno izdvojeni i uglavnom su se ticali visine snežnog pokrivača („*preko 1 m*“ i sl.) ili brzine vetra („*preko 20 m/s*“). Zbog toga je grana grafa koja počinje rečju „*preko*“ uklonjena. Sa druge strane, grana koja počinje rečju „*oko*“ je zadržana, jer je vraćala 9564 rezultata, od kojih svega 16 pogrešno izdvojenih („*oko 17 m/s*“, „*oko 30 l/m^2*“ i sl.).

Ovakva podešavanja transduktora su moguća na osnovu analize teksta koji se obrađuje. Takođe, moguće je da tako podešeni transduktori ne budu na isti način efikasni kada se primene na neke druge tekstove ili korpus.

5.6.2. Transduktori za izdvajanje informacija o vetru

Obeležja *PravacVetra* i *JacinaVetra* su izdvajana jednim trasnduktorom, kako bi bio uzet u obzir kontekst u kome se pojavljuju ove informacije. Naime, analizom tekstova o vremenskim prilikama je utvrđeno da se skoro uvek ove informacije nalaze u jednoj rečenici, i to veoma blizu jedna drugoj. Najčešći oblik izjava koje su sadržale ove informacije bio je:

`vetar <JacinaVetra> <PravacVetra>` - „*Vetar slab, jugozapadni.*“,

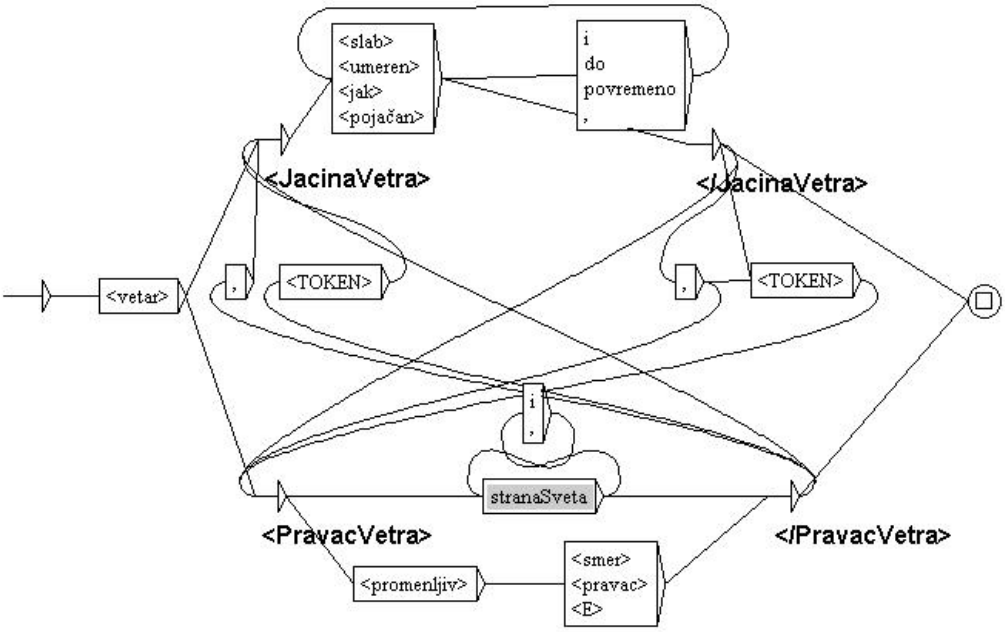
`vetar <PravacVetra> <JacinaVetra>` - „*Vetar severni, slab.*“,

`<JacinaVetra> <PravacVetra> vetar` – „*Jak severozapadni vetar..*“,

`<PravacVetra> <JacinaVetra> vetar` - „*Severni, jak vetar ..*“.

Pri tom su se reči unutar izjava nalazile u različitim morfološkim oblicima. Od posebnog značaja je bila obrada reči kao što su *severni, jugozapadni* i slično, tj. njihovo prepoznavanje kao pravca duvanja vetra, a ne kao strane sveta koje bi mogle da se odnose na lokaciju na kojoj je primećena neka pojava (kao u „*..u istočnim krajevima kiša..*“). Zbog toga je ovaj graf kreiran tako da se zahteva postojanje reči *vetar* u blizini izdvojenog segmenta teksta. Na slici 41. prikazan je transduktor koji izdvaja informacije o jačini i pravcu duvanja vetra, i to kada se

reč „*vetar*“ nalazi na početku segmenta koji nosi informaciju. Transduktor za slučajeve kada se reč „*vetar*“ nalazi na kraju segmenta nosioca informacije je veoma sličan i ovde će biti izostavljen.



Slika 41. Transduktor za izdvajanje jačine i pravca vetra

Ovaj transduktor je prepoznavao, između ostalih, i sledeće izraze:

Vetar jak zapadni.
Vetar slab i umeren, jugoistočni.
Vetar umeren i povremeno jak ..
Vetar jak, uglavnom severni i severoistočni
vetar jugozapadni, jak

Nakon primene transduktora sa slike 41, gore navedeni izrazi su poprimali sledeće oblike, pri čemu su izdvojeni delovi teksta prikazani podebljanim slovima:

*Vetar <JacinaVetra>**jak**</JacinaVetra>*
*<PravacVetra>**zapadni**</PravacVetra>*

*Vetar <JacinaVetra>**slab i umeren**</JacinaVetra>,*
*<PravacVetra>**jugoistočni**</PravacVetra>*

*Vetar <JacinaVetra>**umeren i povremeno jak**</JacinaVetra>*

*Vetar <JacinaVetra>**jak**</JacinaVetra>, uglavnom <PravacVetra>**severni i severoistočni**</PravacVetra>*

*vetar <PravacVetra>**jugozapadni**</PravacVetra>,*
*<JacinaVetra>**jak**</JacinaVetra>*

5.6.3. Transduktori za izdvajanje informacija o lokaciji

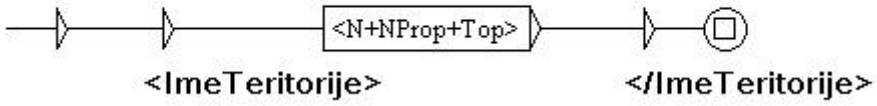
Lokacija na kojoj je zabeležena ili na kojoj se očekuje neka pojava je u tekstu bila predstavljana na veliki broj različitih načina. Informacije o lokaciji su podeljene u tri semantičke klase (tri obeležja):

- *ImeTeritorije* („*na Balkanskom poluostrvu*“, „*u Beogradu*“, „*Srbija*“ i sl.)

- *DeoTeritorije* („*na istoku..*“, „*u severozapadnim krajevima*“ i sl.)

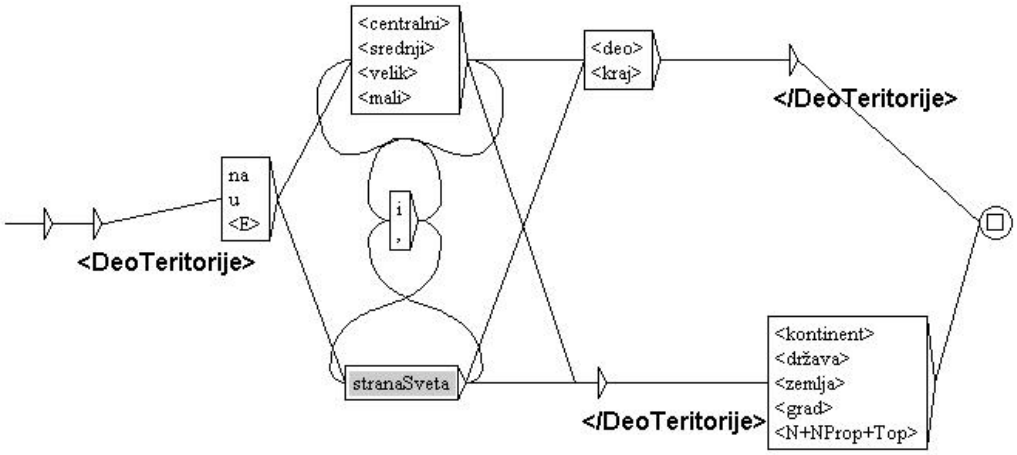
- *Lokaltet* („*u kotlinama*“, „*na planinama*“ i sl.)

Obeležje *ImeTeritorije* u stvari predstavlja neki toponim, pa je za njegovo izdvajanje neophodna upotreba rečnika koji u sebi sadrže informaciju o rečima koje označavaju toponime. Glavni graf koji je korišćen za izdvajanje toponima je prikazan na slici 42.



Slika 42. Transduktor za obeležavanje imena teritorija

Izdvajanje dela teritorije je izvršeno grafom prikazanim na slici 43.



Slika 43. Transduktor *deoTeritorije.grf* za obeležavanje dela teritorije

Transduktor sa slike 43. bi u sledećim niskama teksta izdvojio delove označene podebljanim slovima:

u centralnim delovima *kontinenta*
na severozapadu *kontienenta*
u južnim i jugoistočnim delovima *Evrope*
na severu *Evrope*
u istočnim krajevima *Srbije*
iznad ***većeg dela*** *poluostrva*

Graf *deoTeritorije.grf* zahteva pojavljivanje neke od reči *deo*, *kraj*, *kontinent*, *država*, *zemlja*, *grad* ili nekog toponima nakon strane sveta, da bi bio prepoznat kao deo teritorije (*u južnim i jugoistočnim krajevima*), jer se u protivnom prepoznata strana sveta najverovatnije odnosi na pravac duvanja vetra (*vetar slab, južni i jugoistočni)*. Da bi tako kreiran graf zaista bio efikasan, bitno je da se pre njega ne vrši primena grafa za *ImeTeritorije*, jer bi u tom slučaju bile umetnute oznake ispred toponima. Naravno, moguće je promenitu redosled primene ova dva transduktora, ali je tada potrebno prilagoditi pravila ekstrakcije. Ovo je jedan od primera kada je važan redosled primene, tj. kada redosled primene grafova utiče i na dizajn grafa.

5.7. Analiza izdvojenih podataka i efikasnosti procesa

Izdvajanje informacija iz tekstova o vremenskim prilikama je u početnoj fazi, tokom koje se vrši analiza tekstova sadržanih u korpusu i kreiranje tranduktora za izdvajanje prostih obeležja. S obzirom da se pravila ekstrakcije još uvek razvijaju, kao i da je korpus nad kojim se vrši ekstrakcija prilično velik (45862 rečenice sa preko milion tokena), evaluacija efikasnosti sistema kojom bi se tačno procenila preciznost i odziv trenutno nije moguća. Ipak, moguća je određena analiza rada transduktora, koja bi usmerila i odredila pravce daljeg razvoja sistema i korišćenih transduktora.

U tabeli 6 navedeni su transduktori koji su korišćeni za izdvajanje informacija, i to onim redom kojim su primenjivani. U drugoj koloni tabele za svaki transduktor prikazan je broj prepoznatih segmenata teksta, a u trećoj

procena preciznosti rezultata. S obzirom da je u UNITEX-u moguće sortirati rezultate, tj. prepoznate izraze, po abecednom redu otkrivanje pogrešno prepoznatih izraza je u nekim slučajevima relativno jednostavno. Ipak, ovu procenu treba shvatiti paušalno, više kao pokazatelj nad kojim transduktorima je potrebno vršiti dodatna podešavanja i modifikacije nego kao tačnu vrednost preciznosti transduktora.

Tabela 6. Ocena uspešnosti grafova korišćenih za ekstrakciju informacija

Red. primene	**Transduktor**	**Obelezja izdvojena transduktorom**	**Broj izdvojenih izraza**	**Procena preciznosti**
1.	opisTemp	*OpisTemperature*	11518	100%
2.	temperature	*Temperatura*	25618	99.6%
3.	katTemp	*KatTemperature*	14817	100%
4.	vetarPre	*JacinaVetra* i *PravacVetra*	7720	100%
5.	vetarPost	*JacinaVetra* i *PravacVetra*	1559	100%
6.	padavine	*TipPadavina* i *ObimPadavina*	18878	100%
7.	oblacnost	*ObimOblacnosti* i *PrisustvoOblaka*	18875	98%
8.	deoTeritorije	*DeoTeritorije*	4918	99,8%
9.	imeTeritorije	*ImeTeritorije*	6036	95%
10.	lokalitet	*Lokalitet*	7623	98%
11.	pojava	*Pojava*	3737	100%

Visoka preciznost transduktora je očekivana, s obzirom da se radi o ranoj fazi dizajna sistema i pravila ekstrakcije. Tek analizom odziva i podešavanjem transduktora kako bi izdvojili što veći broj pojedinačnih informacija, svakako će doći do narušavanja preciznosti. Međutim, očekuje se da će transduktori ipak zadržati visoku efikasnost.

Takođe, treba imati u vidu i da je nakon izdvajanja prostih obeležja planirano njihovo objedinjavanje u klase višeg semantičkog nivoa, tokom čega je moguće dalje poboljšanje efikasnosti, u smislu razrešenja nekih višeznačnih izraza ili pogrešno protumačenih segmenata teksta.

Glava 6

6 ZAKLJUČAK

Konačni modeli, u računarstvu veoma korišćeni zbog niza osobina koje ih čine pogodnim za modeliranje različitih procesa i algoritama, našli su svoje mesto i u oblasti kakva je ekstrakcija informacija. Tokom godina, njihova upotreba u ovoj oblasti se menjala, od toga da su sistemi za ekstrakciju informacija bili potpuno bazirani na konačnim modelima, pa do toga da se koriste samo u ograničenom skupu koraka unutar procesa ekstrakcije, obično u fazi predprocesiranja tekssta. Pa ipak, njihov značaj je izuzetno velik, jer su nezamenjivi u situacijama kada se traži visoka preciznost procesa ekstrakcije informacije ili kada je potrebno obraditi neki tekst na efikasan način.

Akcenat ove doktorske disertacije je bio na konačnim modelima (na prvom mestu konačnim automatima i transduktorima, ali i na rekurzivnim mrežama prelaza automata i transduktora) upravo zbog njihovog značaja, kao i zbog činjenjice da su u poslednje vreme potisnuti metodama baziranim na verovatnoći i statistici.

Oblast ekstrakcije informacije je u velikoj ekspanziji, i problemi koji se u njoj javljaju, kao i problemi koji nastaju u drugim oblastima, a koji se rešavaju metodama ekstrakcije informacije, veoma su aktuelni. Međutim, u literaturi se retko nalaze sveobuhvatni pregledi upotrebe konačnih modela u ekstrakciji informacija, sem onih nastalih devedesetih godina kada je ova oblast nastajala. Novija literatura uglavnom daje prikaz metoda baziranih na verovatnoći i statistici. Zbog toga je u okviru ove disertacije dat osvrt na oblast ekstrakcije informacije, ali sa aspekta korišćenja konačnih modela i to u svim fazama procesa ekstrakcije.

U okviru prvog dela disertacije (poglavlja 1 i 2) dat je pregled obalsti ekstrakcije informacije, njenog predmeta istraživanja, definicije problema koji pokušava da reši. Dat je takođe i pregled postojeće teorije formalnih jezika koja je usko povezana sa ekstrakcijom informacija, kao i veza ovih oblasti sa konačnim modelima.

Na kraju poglavlja 2 pojedini pojmovi ekstrakcije informacija predstavljeni su iz ugla teorije formalnih jezika kako bi ova veza bila još jasnija i time dat svojevrstan doprinos teoriji ekstrakcije informacija. Formalno su opisani jezik relevantnih informacija, jezik izdvojenih informacija i jezik konteksta informacije. Takođe, data je definicija pravila ekstrakcije i formulisano i dokazano osnovno svojstvo relacije transdukcije za zadato pravilo ekstrakcije. Pokazana je i regularnost jezika konteksta informacija.

Poglavlje 3 posvećeno je pregledu sistema za ekstrakciju informacije. Detaljno je opisana arhitektura ovih sistema, a posebno moduli koji su im zajednički, bez obzira na druge razlike koje postojeći sistemi imaju među sobom. Ukratko su prikazani postojeći sistemi za ekstrakciju, grupisani po načinu na koji generišu pravila ekstrakcije. Zatim je dat detaljan prikaz programskog paketa UNITEX, koji je u okviru ovog istraživanja i bio korišćen. Ovaj sistem je zasnovan na konačnim modelima, te je kao takav bio od izuzetne koristi u ekstrakciji informacija.

Glavni doprinos disertacije - razvoj novog, *Dvofaznog metoda za ekstrakciju informacija baziranog na konačnim transduktorima,* prikazan je u poglavlju 4. Metod predviđa razdvajanje procesa identifikacije slogova podataka u tekstu od procesa izdvajanja konkretnih vrednosti atributa pojedinih entiteta. Na ovaj način omogućena je upotreba različitih softvera i tehnika u jednoj ili u drugoj fazi, čime je moguće poboljšati efikasnost procesa ekstrakcije. Implementacija metoda je demonstrirana projektovanjem konkretnog sistema za ekstrakciju i njegovom primenom na enciklopedijski tekst o mikrorganizmima. Sva pravila ekstrakcije su predstavljena konačnim transduktorima i rekurzivnim mrežama prelaza. Izvršena je i evaluacija sistema i ustanovljena je izuzetno dobra preciznost izdvojenih podataka, kao i mogućnost povećanja odziva daljim podešavanjem transduktora. Kao rezultat primene ovog metoda na enciklopedijski

tekst o mikroorganizmima nastala je baza podataka namenjena za dalja istraživanja iz oblasti genetike i bioinformatike. Ovi rezultati disertacije objavljeni su u nekoliko radova publikovanih u časopisima ili izlaganih na međunarodnim konferencijama (Pajić, 2011; Pajić i sar. 2011a; Pajić i sar. 2011b) i predstavljaju osnov za dalja istraživanja, jedna u pravcu razvoja dvofaznog metoda, a druga u pravcu upotrebe dobijene baze podataka za istraživanje podataka iz oblasti biologije i bioinformatike.

U poglavlju 5 prikazana je još jedna primena modela konačnih stanja u ekstrakciji informacija, ovoga puta na tekstove na srpskom jeziku. Cilj je bio da se pokažu specifičnosti morfološki bogatih jezika kakav je i srpski jezik i problemi koji nastaju prilikom ekstrakcija informacija iz tekstova na tim jezicima. Postojeći metodi su uglavnom razvijani za engleski jezik, koji ima relativno jednostavnu morfologiju, pa su samim tim često neadekvatni kada se primene na tekstove na npr. srpskom jeziku. Kod jezika kakav je srpski, u literaturi nazivanim i *less resourced languages*, kod kojih često nisu dostupni različiti leksički resursi, konačni modeli dobijaju poseban značaj. Nepostojanje anotiranih korpusa tekstova za ovakve jezike čini probabilističke metode neupotrebljive za ekstraciju informacija, pa su konačni modeli jedini izbor. Zato je u poglavlju 5 prikazan pristup problemu ekstrakcije informacija iz teksta na srpskom jeziku, tačnije korpusa tekstova o vremenskim prilikama preuzetih iz više izvora. Izdvajanje informacija je vršeno transduktorima i mrežama prelaza pomoću softverskog sistema UNITEX. Prepoznavanje i izdvajanje niski iz teksta vršeno je primenom elektronskog rečnika i upotrebom leksičkih maski. Svi kreirani transduktori su još uvek u fazi razvoja i za sada imaju veoma visoku preciznost (od 95% do 100%). Iako je istraživanje opisano u poglavlju 5. tek u začetku, ipak odlično ilustruje probleme koji postoje u obradi jezika sa bogatom morfologijom, kakav je i srpski jezik.

REFERENCE

Abolhassani, M., Fuhr, N. i Gövert, N. (2003) Information Extraction and Automatic Markup for XML documents, in Blanken et al, *Intelligent Search on XML Data. Applications, Languages, Models, Implementations, and Benchmarks*, Volume 2818, pp 159-174, Springer

Agichtein, E. i Ganti, V. (2004) Mining reference tables for automatic text segmentation, in *Proceedings of the Tenth ACM, SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, USA.

Agichtein, E. i Gravano, L. (2000) Snowball: Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

Allen J. (1995) *Natural Language Understanding*. Benjamin/Cummings.

Appelt, D. E. (1999) Introduction to information extraction. *AI Communications*, 12:161– 172.

Appelt, D. E., Hobbs, J., Bear, J., Israel, D. i Tyson. M. (1993) FASTUS: A finite-state processor for Information Extraction from real world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1172–1178.

Ayuso, D., Boisen, S., Fox, H., Gish, H., Ingria, R. i Weischedel, R. (1992) BBN: Description of the PLUM system as used for MUC-4. In *Proceedings of the Fourth MessageUnderstanding Conference (MUC-4)*, pages 169–176.

Bangalore, S., Ricardi, G. (2000) Stochastic finite-state models for spoken language machine translation, in: Workshop on Embedded Machine Translation Systems, 2000.

Bangalore, S., Ricardi, G. (2000) Finite-state models for lexical reordering in spoken language translation, in: Proceedings of the ICSLP, 2000.

Banko, M., Cafarella, M., Soderland, S., Broadhead, M., i Etzioni, O. (2007). Open Information Extraction from the Web. In *Procs. of IJCAI*.

Baumgartner, R., Flesca, S. i Gottlob, G. (2001) Visual Web Information Extraction with Lixto. In *Proceedings of the Conference on Very Large Databases (VLDB).*

Berry, M.J.A. i Linoff, G.S. (2004) *Data mining techniques, second edition - for marketing, sales, and customer relationship management*, Wiley

Berstel, J. (1979) *Transductions and Context-free Languages*, B.G. Teubner, Stuttgart, 1979.

Bilisoly, R. (2008). Practical Text Mining with Perl, New York: John Wiley & Sons. ISBN 978-0470176436

Bilofsky, H.S. i Christian, B. (1988) The GenBank® genetic sequence data bank, *Nucl. Acids Res.* 16(5): 1861-1863 doi:10.1093/nar/16.5.1861

Borkar, V. R., Deshmukh, K. i Sarawagi, S. (2001) Automatic text segmentation for extracting structured records, in *Proceedings of ACM SIGMOD International Conference on Management of Data*, Santa Barabara, USA

Borthwick, A. , Sterling, J., Agichtein, E. i Grishman, R. (1998) Exploiting diverse knowledge sources via maximum entropy in named entity recognition, in *Sixth Workshop on Very Large Corpora New Brunswick*, New Jersey, Association for Computational Linguistics.

Brandt, S.R. (2000) Writing a Java Beautifier for Jbuilder, *JBuilder Journal*, 2000.

Brin, S. (1998) Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extended Database Technology, EDBT'98*, 1998.

Brkić, M., i Matetić, M. (2007) Modeling Natural Language Dialogue for Croatian Weather Forecast System. *Proceedings of the 18th International Conference on Information and Intelligent Systems* (pp. 391-396). Croatia, Varaždin.

Burns, G., Feng, D. and Hovy, E. (2008) Intelligent Approaches to Mining the Primary Research Literature: Techniques, Systems, and Examples, *Computational Intelligence in Medical Informatics, Studies in Computational Intelligence*, Springer Berlin / Heidelberg, p. 17-50

Cafarella, M. J., Downey, D., Soderland, S. i Etzioni, O. (2005) KnowItNow: Fast, scalable information extraction from the web, in *Conference on Human Language Technologies* (HLT/EMNLP), 2005.

Califf, M. E. i Mooney, R. J. (1999) Relational learning of pattern-match rules for Information Extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 328–334, Orlando, FL, July 1999.

Califf, M. i Mooney, R. (2003). Bottom-up Relational Learning of Pattern Matching rules for Information Extraction, *Journal of Machine Learning Research* 4, 177/210.

Cardie, C. i Wagstaff, K. (1999) Noun phrase coreference as clustering. In *Proceedings of Joint Conference on Empirical Methods in NLP and Very Large Corpora* (pp. 82-89), USA, 1999.

Casacuberta, F., Vidal, E. i Picó, D. (2005) Inference of finite-state transducers from regular languages, *Pattern Recognition*, Volume 38, Issue 9, pp.1431-1443

Chang, R., Shoemaker, R. i Wang, W. (2011) A Novel Knowledge-Driven Systems Biology Approach for Phenotype Prediction upon Genetic Intervention, *IEEE/ACM Transactions on*

Computational Biology and Bioinformatics, vol. 8, no. 5, pp. 1170-1182, doi:10.1109/TCBB.2011.18

Chen, C., DeClerck, G., Casstevens, T., Youens-Clark, K., Zhang, J., Ware, D., Jaiswal, P., McCouch, S. i Buckler, E. (2010) The Gramene Genetic Diversity Module: a resource for genotype-phenotype association analysis in grass species. Available from *Nature Precedings* http://hdl.handle.net/10101/npre.2010.4645.1

Chidlovskii, B. (2001) Automatic repairing of web wrappers. In *Proceeding of the Third International Workshop on Web Information and Data Management*, pages 24–30, Atlanta, Georgia, USA.

Chinchor, N.A. (1998) Overview of MUC-7/MET-2

Chrobot, A., Courtois, B., Hammani-Mc Carthy, M., Gross, M. i Zellagui. K. (1999) Dictionnaire electronique DELAC anglais: noms composés. *Technical Report 59*, LADL, Université Paris 7

Ciravegna, F. (2001) (LP)2, an Adaptive Algorithm for Information Extraction, in *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining* to be held in conjunction with the 17th International Conference on Artificial Intelligence (IJCAI-01), Seattle, August, 2001

Courtois, B. (1996) Formes ambiguës de la langue française, *Lingvisticæ Investigationes*, 20(1) Amsterdam-Philadelphia: John Benjamins Publishing Company, pp. 167 -202

Courtois, B. i Silberztein, M. (1990) *Les dictionnaires électroniques du français*, Larousse, Langue française, vol. 87.

Cowie, J. i Lehnert, W. (1996) Information Extraction. *Communications of the ACM*, 39 (1), 80-91.

Cunningham, H., Maynard, D. Bontcheva, K. i Tablan, V. (2002) GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, July 2002.

Ćavar, D., Jazbec, I.P. i Runjaić, S. (2009) Efficient Morphological Parsing with a Weighted Finite State Transducer, *Informatica* (0350-2325) 33 (2009), 1; 107-113

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S. i Weischedel, R. (2004) The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. *In Proc. Conference on Language Resources and Evaluation.*

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. i Yates, A. (2005) Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. iUthurusamy, R. (1996) *Advances in Knowledge Discovery and Data Mining*, American Association for Artificial Intelligence

Feldman, R., i Sanger, J. (2006). The Text Mining Handbook, New York: Cambridge University Press. ISBN 9780521836579

Feng, D., Burns, G. i Hovy, E. (2007) Extracting Data Records from Unstructured Biomedical Full Text, in *Proceedings of the EMNLP conference*, Prague, Czech Republic.

Finkel, J., T. Grenager, i C. Manning (2005) Incorporating Nonlocal Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pp. 363-370.

Freitag, D. i Mccallum, A. (2000) Information Extraction with HMM Structures Learned by Stochastic Optimization, in *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, Texas, USA, AAAI Press, pp. 584—589

Friburger, N. i Maurel, D. (2004) Finite-state transducer cascades to extract named entities in texts, *Theoretical Computer Science* 313, 93 – 104

Gaizauskas, R., Davis, N., Demetriou, G.,Guo, Y. i Roberts, I. (2004) Integrating Text Mining Services into Distributed Bioinformatics Workflows: A Web Services Implementation, In Proceedings of the IEEE International Conference on Services Computing (SCC 2004)

Garrity, G. (2005) *Bergey's Manual of Systematic Bacteriology, Volume 2 : The Proteobacteria*, 2005, ISBN 978-0-387-95040-2

Garrity, G., Don, J., Krieg, N.R. i Staley, J.T. (2005) *Bergey's Manual of Systematic Bacteriology, Volume Two: The Proteobacteria (Part C)* , ISBN 978-0-387-24145-6

Goh, C.S., Gianoulis, T.A., Liu, Y., Li, J., Paccanaro, A., Lussier, Y.A. i Gerstein, M. (2006) Integration of curated databases to identify genotype-phenotype associations, *BMC Genomics*, 7, pp. 257-257.

Gopalacharyulu, P.V., Lindfors, E., Miettinen, J., Bounsaythip, C.K. i Oresic, M. (2008) An integrative approach for biological data mining and visualisation, *International Journal of Data Mining and Bioinformatics* 2008 - Vol. 2, No.1 pp. 54 – 77

Grass, T., Maurel, D. i Piton, O. (2002) Description of a multilingual database of proper names. In Elisabete Ranchod and Nuno J. Mamede, editors, *PorTAL, volume 2389 of Lecture Notes in Computer Science*, pages 137–140. Springer

Grishman, R. i Sundheim, B. (1996) Message Understanding Conference 6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics* (pp 466-471), San Mateo, CA, 1996.

Gross, M. and Perrin, D. (1987) “Electronic Dictionaries and Automata in Computational Linguistics”, in *Proceedings of LITP Spring School on Theoretical Computer Science*, Saint-Pierre d’Oleron, France

Grosz, B. i C. Sidner (1986) Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3):175-204.

Gu, Z. i N. Cercone (2006) Segment-Based Hidden Markov Models for Information Extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 481-488.

Halliday, M. i R. Hasan, (1976) *Cohesion in English*. Longman, London.

Han, J. i Kamber, M. (2006) *Data Mining: Concepts and Techniques*, Elsevier Inc.

Hirschman, L. (1992) An Adjunct Test for Discourse Processing in MUC-4. *Proceedings of MUC-4*, 67-77, Morgan Kaufmann, 1992.

Hobbs, J. R., Appelt, D., Tyson, M., Bear, J. i Islael, D. (1992) SRI International: Description of the FASTUS system used for MUC-4. In *Proceedings fo the 4th Message Understanding Conference (MUC-4)*, pages 268–275.

Hobbs, J. R., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M. i Tyson, M. (1997) FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, In Roche E. and Y. Schabes, eds., *Finite-State Language Processing*, The MIT Press, Cambridge, MA, pages 383-406.

Huffman, S. B. (1996) Learning information extraction patterns from examples. In *Lecture Notes in Computer Science. Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040, pages 246–260, London, UK, 1996. Springer Verlag.

Jayram, T. S., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S. i Zhu, H. (2006) Avatar information extraction system, *IEEE Data Engineering Bulletin*, vol. 29, pp. 40–48, 2006.

Jim, K., Parmar, K., Singh, M. i Tavazoie, S. (2004) A cross-genomic approach for systematic mapping of phenotypic traits to genes, *Genome Res*, 14(1), pp. 109-115.

Jimeno-Yepes, A., Berlanga-Llavori, R. i Rebholz-Schuhmann, D. (2009) Exploitation of ontological resources for scientific literature analysis: Searching genes and related diseases, *Engineering in Medicine and Biology Society*, EMBC 2009. Annual International Conference of the IEEE, pp 7073 – 7078

Jurafsky, D. i Martin, J. H. (2008) *Speech and language processing*, 2nd edition, Prentice-Hall Inc.

Kaiser, K. i Miksch, S. (2005) *Information Extraction. A Survey*. Vienna University of Technology, Institute of Software Technology and Interactive Systems, Vienna, Technical Report, Asgaard-TR-2005-6, May 2005.

Kim, J.-T. i Moldovan, D. I. (1995) Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):713–724, October 1995.

Kiparsky, P. (2002) On the Architecture of Panini's Grammar, in *Proceedings of Hyderabad Conference on the Architecture of Grammar*, 2002

Klarsfled, G. i Hammani-Mc Carthy, M. (1991) Dictionnaire électronique du ladl pour les mots simples de l'anglais (DELASa). *Technical report*, LADL, Université Paris 7

Knight, K. i Al-Onaizan, Y. (1998) Translation with finite-state devices, in: *Proceedings of the Fourth ANSTA Conference*, 1998.

Koehn, P. (2005) Europarl: A Parallel Corpus for Statistical Machine Translation, *MT Summit* 2005.

Kononenko I., Popov I., Zagorulko Yu. (1999) Approach to Understanding Weather Forecast Telegrams with Agent-Based Technique. In: *A. Ershov Third International Conference «Perspectives of System Informatics»*, 1999, pp.295-298.

Kononenko I., Kononenko S., Popov I., Zagorulko Yu. (2000) Information extraction from non-segmented text (on the material of weather forecast telegrams). *RIAO* 2000: 1069-1088

Korbel, J., Doerks, T., Jensen, L.J., Perez-Iratxeta, C., Kaczanowski, S., Hooper, S.D., Andrade, M.A. i Bork, P (2005) Systematic association of genes to phenotypes by genome and literature mining, *PLoS Biol*, 3, pp. 134-134.

Kornai, A. (1999) *Extended finite state models of langu*age, Cambridge University Press

Krieg, N.R., Ludwig, W., Whitman, W.B., Hedlund, B.P., Paster, B.J., Staley, J.T., Ward, N., Brown, D. i Parte, A. (2010) *Bergey's Manual of Systematic Bacteriology, Volume 4: The Bacteroidetes, Spirochaetes, Tenericutes (Mollicutes), Acidobacteria, Fibrobacteres, Fusobacteria, Dictyoglomi, Gemmatimonadetes, Lentisphaerae, Verrucomicrobia, Chlamydiae, and Planctomycetes*, ISBN 978-0-387-95042-6

Krstev C. (2008) *Processing of Serbian, Automata, texts and electronic dictionaries*, Faculty of Philology, University of Belgrade, Belgrade 2008.

Krstev, C. i Vitas, D. (2005) Corpus and Lexicon - Mutual Incompletness, in *Proceedings of the Corpus Linguistics Conference*, Birmingham

Krstev, C., Vitas, D., Obradović, I. i Utvić, M. (2011) E-Dictionaries and Finite-State Automata for the Recognition of Named Entities, *Proceedings of the 9th International Workshop on*

Finite State Methods and Natural Language Processing, pages 48–56, Blois (France), July 12-15, 2011.

Krupka, G., Jacobs, P., Rau, L., Childs, L. i Sider, I. (1992) GE NLTOOLSET: Description of the system as used for MUC-4. In *Proceedings of the 4th Message Understanding Conference (MUC-4)*, pages 177–185.

Kushmerick, N. (1997) *Wrapper Induction for Information Extraction*. PhD thesis, University of Washington.

Kushmerick, N. (2000) Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

Kushmerick, N., Weld, D. S. i Doorenbos, R. (1997) Wrapper Induction for Information Extraction. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya.

Labelle, J. (1995) Le traitement automatique des variantes linguistiques en français: l'exemple des concrets, *Lingvisticæ Investigationes*, 19(1), Amsterdam - Philadelphia: John Benjamins Publishing Company, pp.137–152

Labsky, M., Nekvasil, M., i Svatek, V. (2007) Towards web information extraction using extraction ontologies and (indirectly) domain ontologies, *K-CAP '07 Proceedings of the 4th international conference on Knowledge capture*, ACM New York, NY, USA 2007

Lange, D, Böhm, C., i Naumann, F. (2010) Extracting structured information from Wikipedia articles to populate infoboxes, Universitätsverlag Potsdam, 2010

Lawson, M. V. (2004) *Finite Automata*, Chapman & Hall/CRC, USA, 2004.

Lehnert, W., Cardie, C., Fisher,D., McCarthy, J., Riloff, E. i Soderland, S. (1994) Evaluating an Information Extraction system. *Journal of Integrated Computer-Aided Engineering*, 1(6).

Lewis, D. i Jones, K. S. (1996) Natural Language Processing for Information Retrieval. *Communications of the ACM* 39(1): 92-101

Li, J., Zhu, X. i Chen, J.Y. (2010) "Discovering breast cancer drug candidates from biomedical literature", *International Journal of Data Mining and Bioinformatics* 2010 - Vol. 4, No.3 pp. 241 - 255

Liu, L., Pu, C., i Han, W. (2000) XWRAP: An XML-enabledWrapper Construction System for Web Information Sources. In *Intern. Conference on Data Engineering (ICDE)*, pages 611–621.

MacDonald, N.J. i Beiko, R.G. (2010) Efficient learning of microbial genotype-phenotype association rules, *Bioinformatics*, 26, pp. 1834-1840.

Madarasz,R.S i Crvenković, S. (1995) *Uvod u teoriju automata i formalnih jezika*, Univerzitet u Novom Sadu, Novi Sad, Srbija

Mahdavi, M.A. (2010) Medical informatics: transition from data acquisition to data analysis by means of bioinformatics tools and resources, *International Journal of Data Mining and Bioinformatics*, 2010 - Vol. 4, No.2 pp. 158 - 174

Maier, M., B. Taylor, H. Oktay i D. Jensen (2010). Learning Causal Models of Relational Domains. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Mani, I. i Maybury, M.T. (1999) *Advances in automatic text summarization*, MIT Press, USA

Manning, C.D., Raghavan, P. i Schutze, H. (2008) *Introduction to Information Retrieval*, Cambridge University Press, 2008

Maslennikov, M. i T. Chua (2007) A Multi-Resolution Framework for Information Extraction from Free Text. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. 28 Handbook of Natural Language Processing

Maslennikov, M., H.K. Goh i T.S. Chua (2006) ARE: Instance Splitting Strategies for Dependency Relation-based Information Extraction. In *Proc of ACL*-2006.

Mäkinen, E. (1999) Inferring finite transducers, Tech. Rep. A-1999-3, University of Tampere, 1999.

Mccallum, A., Freitag, D. i Pereira, F. (2000) Maximum entropy markov models for information extraction and segmentation, *Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 591—598

McCarthy, J. i Lehnert, W. (1995) Using decision trees for coreference resolution. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1050–1055.

Michelson, M. i Knoblock, C.A. (2008) Creating relational data from unstructured and ungrammatical data sources, *Journal of Artificial Intelligence Research* (*JAIR*), vol. 31, pp. 543–590, 2008.

Miltsakaki, E. (2003) *The Syntax-Discourse Interface: Effects of the Main-Subordinate Distinction on Attention Structure*, PhD thesis.

Mirhaji, P., Byrne, S., Kunapareddy, N. i Casscells, SW. (2006) "Semantic approach for text understanding of chief complaints data", in *AMIA Annual Symposium Proceedings*, Washington, p.1033

Mohri, M. (1996) On some applications of finite-state automata theory to natural language processing, *Natural Language Engineering*, Volume 2 Issue 1, March 1996, Cambridge University Press New York, NY, USA

Moens, M.F. (2006) *Information extraction: algorithms and prospects in a retrieval context*, Springer, 2006.

Moens, M.F. i R. De Busser (2002) First steps in building a model for the retrieval of court decisions. *International Journal of Human-Computer Studies*, 57(5):429-446.

Monceaux, A. (1995) Le dictionnaire des mots simples anglais: mots nouveaux et variantes orthographiques, *Technical Report 15*, IGM, Université de Marne-la-Vallée, France.

Muslea, I., Minton, S. i Knoblock, C. (1999) A hierarchical approach to wrapper induction. In O. Etzioni, J. P. M¨uller, and J. M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 190–197, Seattle, WA, USA, 1999. ACM Press.

Nelson, M. (1997) A Fresh Cup of Zip, *Dr. Dobb's Journal*, Dec 1997.

Oncina, J., Garcia, P. i Vidal, E. (1993) Learning subsequential transducers for pattern recognition interpretation tasks, IEEE Trans. Pattern Anal. Machine Intel. 15 (5) (1993) 448–458.

Pajić, V. (2011) Putting Encyclopaedia Knowledge into Structural Form: Finite State Transducers Approach, *Journal of Integrative Bioinformatics, Informationsmanagement in der Biotechnologie e.V.*, Germany, 8(2):164, ISSN 1613-4516.

Pajić, V., Pavlović-Lažetić, G. i Pajić, M. (2011a) Information Extraction from Semi-structured Resources: A Two-Phase Finite State Transducers Approach, *Implementation and Application of Automata: Proceedings of 16th International Conference CIAA, Lecture Notes in Computer Science*, Springer Berlin / Heidelberg 282-289, ISBN 3642222552, 9783642222559.

Pajić, V., Pavlović-Lažetić, G., Beljanski, M., Brandt, B. i Pajić, M. (2011b) Towards a Database for Genotype-Phenotype Association Research: Mining Data from Encyclopedia, *International Journal of Data Mining and Bioinformatics*, Inderscience publishers**,** ISSN (Online): 1748-5681, ISSN (Print): 1748-5673, http://www.inderscience.com/browse/index.php?journalID=189&action=coming.

Palakal, M. i Pavithra, N (2009) An on demand data integration model for biological databases, *International Journal of Data Mining and Bioinformatics*, 2009 - Vol. 3, No.1 pp. 40 - 54

Patwardhan, S. i E. Riloff (2007) Effective Information Extraction with Semantic Afinity Patterns and Relevant Regions. In *Proceedings of 2007 29the Conference on Empirical Methods in Natural Language Processing* (EMNLP-2007).

Paumier, S. (2011) *Unitex 2.1 User Manual*, Universit´e de Marne-la-Vall´ee. http://www-igm.univ-mlv.fr/˜unitex/UnitexManual2.1.pdf

Pavlović-Lažetić G. (2006) Electronic Resources of Serbian: Serbian WORDNET, *36th International Slavic Conference*, MSC, Belgrade, Serbia, september 2006.

Peng, F. i McCallum, A. (2006) Information extraction from research papers using conditional random fields, *Information Processing & Management*, Volume 42, Issue 4, pp. 963-979

Peng, F. i McCallum, A. (2004) Accurate information extraction from research papers using conditional random fields, in *HLT-NAACL*, pp. 329–336, 2004.

Porter, M.F. (1980) An algorithm for suffix stripping, *Program*, 14(3) pp 130−137.

Ray, S. (2001) Representing Sentence Structure in Hidden Markov Models for Information Extraction, in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1273—1279

Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H. i Vaithyanathan, S. (2008) An algebraic approach to rule-based information extraction, in *ICDE*, 2008.

Riloff E i Philips W. (2004) *An introduction to the Sundance and AutoSlog Systems*. University of Utah School of Computing; 2004.

Riloff, E. (1993) Automatically constructing a dictionary for information extraction tasks. In *Proc. of the 11th National conference on Artificial Intelligence*, pages 811–816.

Riloff, E. (1996) Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligende (AAAI-96)*, pages 1044–1049.

Riloff, E. i Jones, R. (1999) Learning dictionaries for information extraction by multilevel bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 474–479. AAAI Press/MIT Press, 1999.

Riloff, E. i Lorenzen, J. (1999) Extraction-based Text Categorization: Generating Domain-specific Role Relationships Automatically, In *Natural Language Information Retrieval* , Tomek Strzalkowski, ed., Kluwer Academic Publishers.

Riloff, E. i Phillips, W. (2004) An Introduction to the Sundance and AutoSlog Systems, Technical Report UUCS-04-015, School of Computing, University of Utah.

Roberts, A., Gaizauskas, R., Hepple, M., Demetriou, G., Guo, Y., Roberts, I., i Setzer, A. (2009) Building a semantically annotated corpus of clinical texts, Journal of Biomedical Informatics 42(5), 950-966

Roche, E. (1999) Finite state transducers: parsing free and frozen sentences, *Extended finite state models of language*, Cambridge University Press, pp. 108.-120.

Roche, E. i Schabes, Y. (1997) *Finite-state language processing*, The MIT Press

Root, T.L. i sar. (2011) Association of Candidate Genes with Phenotypic Traits Relevant to Anorexia Nervosa, *European Eating Disorders Review*, doi: 10.1002/erv.1138.

Röske, K., Foecking, M.F., Yooseph, S., Glass, J.I., Calcutt, M.J. i Wise, K.S. (2010) A versatile palindromic amphipathic repeat coding sequence horizontally distributed among diverse bacterial and eucaryotic microbes, *BMC Genomics*. 11:430

Salton, G. i Buckley, C. (1988) Term-weighting approaches in automatic text retrieval. In *Information Processing & Management*, 24(5): 513-523.

Sang, E. i Meulder, F. D. (2003) Introduction to the conll-2003 shared task: Language-independent named entity recognition, in *Seventh Conference on Natural Language Learning (CoNLL-03)*, (W. Daelemans and M. Osborne, eds.), pp. 142–147, Edmonton, Alberta, Canada: Association for Computational Linguistics, May 31–June 1, 2003. (In association with HLTNAACL, 2003).

Sarawagi, S. (2008) Information extraction. *Foundations and Trends in Databases*, Vol. 1, No. 3 261–377

Sastre, J. M. (2009) Efficient Parsing Using Filtered-Popping Recursive Transition Networks, *Lecture Notes in Computer Science*. vol. 5642, pp. 241–244

Sastre, J. M. i Forcada, M. (2007) Efficient parsing using recursive transition networks with output, In Zygmunt Vetulani, editors, *Proceedings of 3rd Language & Technology Conference* (LTC'07) pp. 280–284

Savary, A. (2000) *Recensement et description des mots composés - méthodes et applications*, Thèse de doctorat. Université de Marne-la-Vallée, France.

Sayers, E.W. i sar. (2009) Database resources of the National Center for Biotechnology Information,. *Nucleic Acids Research.* 2009;37:D5-D15.

Seki, K. i Mostafa, J. (2009) Discovering implicit associations among critical biological entities, International Journal of Data Mining and Bioinformatics 2009 - Vol. 3, No.2 pp. 105 – 123

Seymore, K., McCallum, A. i Rosenfeld, R. (1999) Learning Hidden Markov Model structure for information extraction," in *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, pp. 37–42, 1999.

Shen, D., Zhang, J., Su, J., Zhou, G. i Tan, C. (2004) Multi-Criteria-based Active Learning for Named Entity Recognition, in *Proceedings of the ACL 2004*.

Shen, W., Doan, A., Naughton, J. F. i Ramakrishnan, R. (2007) Declarative information extraction using datalog with embedded extraction predicates, in *VLDB*, pp. 1033–1044, 2007.

Shinyama, Y. i Sekine, S. (2006) Preemptive information extraction using unrestricted relation discovery, in *HLT-NAACL*, 2006.

Silberztein, M. (1993) *Dictionnaires électroniques et analyse automatique de textes: le système INTEX*, Edition Masson, Paris, 1993.

Slocum J. (1985) A Survey of Machine Translation: its History, Current Status, and Future Prospects. In: *Computational Linguistics*, Vol. 11, No 1, 1985, pp. 1-17.

Soderland, S. (1999) Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272.

Soderland, S., Fisher, D., Aseltine, J. i Lehnert, W. (1995) CRYSTAL: inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1314–1319.

Sprague, J. i sar. (2008) The Zebrafish Information Network: the zebrafish model organism database provides expanded support for genotypes and phenotypes. *Nucleic Acids Research* 36:D768.

Steinberger R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D., i Varga, D. (2006) The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. *Proceedings of the 5th International Conference on Language Resources and Evaluation* (LREC'2006). Genoa, Italy, 24-26 May 2006.

Stevenson, M. i M. Greenwood (2005). A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, MI, pp. 379{ 386.

Sudo, K. (2004) *Unsupervised Discovery of Extraction Patterns for Information Extraction*. PhD thesis, New York University, New York, September 2004.

Sudo, K., S. Sekine, i R. Grishman (2003). An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (ACL-03).

Surdeanu, M. i Harabagiu, S. (2002) Infrastructure for Open-Domain Information Extraction. *Proceedings of HLT 2002*, San Diego, California.

Surdeanu, M., S. Harabagiu, J. Williams, i P. Aarseth (2003). Using predicate-argument structures for information extraction. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*.

Szabo, Z. G. (2004) Compositionality. In Edward N. Zalta (Ed.) *The Standford Encyclopedia of Philosophy*, (Fall 2004 Edition),.

Taboada, M. i Mann, W. (2005) *Applications of Rhetorical Structure Theory*. Discourse studies, 8(4).

Tamura, M. i D'haeseleer, P. (2008) Microbial genotype-phenotype mapping by class association rule mining, *Bioinformatics*, 24, pp. 1523-1529.

Thelen, M. i E. Riloff (2002). A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pa ttern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pp. 214-221

Thorisson, G.A., Muilu, J. i Brookes, A.J. (2009) Genotype-phenotype databases: challenges and solutions for the post-genomic era, *Nature reviews. Genetics*, Vol. 10, No. 1., pp. 9-18. doi:10.1038/nrg2483

Vidal, E., García, P. i Segarra, E. (1989) Inductive learning of finite-state transducers for the interpretation of unidimensional objects, in: R. Mohr, Th. Pavlidis, A. Sanfeliu (Eds.), Structural Pattern Analysis, World Scientific, Singapore, 1989, pp. 17–35.

Vilar, J.M. (2000) Improve the learning of subsequential transducers by using alignments and dictionaries, in: Grammatical Inference: Algorithms and Applications, vol. 1891 of Lecture Notes in Artificial Intelligence, Springer, Berlin, 2000, pp. 298–312.

Vitas, D. (2006) *Prevodioci i interpretatori: Uvod u teoriju i metode kompilacije programskih jezika*, Matematički fakultet, Belgrade, Republic of Serbia

Vitas, D. (2007) O problemu ne(pre)poznate reči. *Zbornik Matice srpske za filologiju i lingvistiku*. Matica srpska, Novi Sad, Knj. 50, str. [111]-120, 2007.

Vitas, D., Krstev, C., Obradović, I., Popović, Lj. i Pavlović-Lažetić, G. (2003) Processing Serbian Written Texts: An Overview of Resources and Basic Tools, in *Workshop on Balkan Language Resources and Tools*, Thessaloniki, Greece, pp. 97-104

Vitas D., Pavlović - Lažetić G. (2007) Extraction of named entities in Serbian using INTEX, *Formaliser les langues avec l'ordinateur: De INTEX a Nooj*, eds. Koeva, S, Maurel, D, Silberztein, M, Presses Universitaires de Franche-Comte, serie Archive, Bases, Corpus, n03, 2007, ISBN: 978-2-84867-189-5, ISSN: 1771-8996, 281-302.

Vos, P., Garrity, G., Jones, D., Krieg, N.R., Ludwig, W., Rainey, F.A., Schleifer, K.H. i Whitman W.B. (2009) *Bergey's Manual of Systematic Bacteriology, Volume 3: The Firmicutes*, ISBN 978-0-387-95041-9

Woods, W. (1970) Transition network grammars for natural language analysis, *Communications of the ACM*, pp. 591-602.

Xiao, J., Chua, T.S. i Cui, H. (2004) Cascading Use of Soft and Hard Matching Pattern Rules for Weakly Supervised Information Extraction. In *Proc of COLING-2004*.

Xu, F. i Krieger, H.U. (2003) Integrating Shallow and Deep NLP for Information Extraction. In *Proceedings of RANLP 2003*, Borovets, Bulgaria.

Yangarber R (2003) Acquisition of domain knowledge. In lecture Notes in Computer Science, Vol. 2700, pp. 1-28. Springer-Verlag Berlin

Yangarber, R, Grishman, R., Tapanainen, P. i Huttunen, S. (2000) Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbr¨ucken, Germany, August 2000.

Yangarber, R. i Grishman, R. (1998) NYU: Description of the Proteus/PET system as used for MUC-7 ST. In *Proceedings of the 7th Message Understanding Conference: MUC-7*, Washington, DC.

Yatsko V. A., Starikov M. S., Butakov A.V. (2010) Automatic genre recognition and adaptive text summarization. In: *Automatic Documentation and Mathematical Linguistics*, 2010, Volume 44, Number 3, pp.111-120.

Zhang, M., Zhang, J., Su, J. i Zhou, G. (2006) A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. In *Proc of ACL*-2006.

Zhong, P., Chen, J. i Cook, T. (2007) Web Information Extraction Using Generalized Hidden Markov Model, *Hot Topics in Web Systems and Technologies*, HOTWEB '06. 1st IEEE Workshop, p. 1-8

Zhu, J., Nie, Z., Wen, J.R., Zhang, B. i Ma, W.Y. (2005) 2D Conditional Random Fields for Web information extraction, *ACM International Conference Proceeding Series; Proceedings of the 22nd international conference on Machine learning*, Vol. 119, Bonn, Germany, pp. 1044 – 1051

PRILOG 1.

Java programski kod klase *EkstrakcijaInformacije*. U drugoj fazi dvofaznog metoda ekstrakcije ova klasa je korišćena za izdvajanje pojedinačnih atributa mikroorganizama. Konkretno, kod koji sledi preuzima slog po slog tabele *Genus*, kreira po jedan objekat klase *EkstrakcijaInformacija* i na opise koji se nalaze u polju *GenusDesc* primenjuje 9 transduktora. Izlaz koji transduktori generišu se ubacuje nazad u bazu podataka, u polja koja odgovaraju određenim osobinama mikroorganizama.

```
package bakterije;
import java.io.*;
import java.sql.*;
import java.net.*;
import fr.umlv.unitex.io.*;

/**
 * @author Vesna Pajic, 2010.
 */
public class EkstrakcijaInformacija {
   File originalText = null;
   File sntText = null;
   File sntDir = null;
   File workingDir = null;
   File graph = null;
   File alphabet = null;
   File dictionary = null;

   public EkstrakcijaInformacija(int ime) {
      String filename = Integer.toString(ime);
      this.originalText = new File(filename);
      this.sntDir = new File(filename + "_snt");
      if (!sntDir.exists()) sntDir.mkdir();
      this.workingDir = new File(System.getProperty("user.dir"));
      this.alphabet = new File(workingDir + File.separator + "Alphabet.txt");
      this.dictionary = new File(workingDir + File.separator +
         "dela-en-public.bin");
   }

   public static void main (String[] args) {
      Connection con = null;
      String opis;

      try {
         //konekcija sa bazom
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         con = DriverManager.getConnection("jdbc:odbc:bakterije");
         Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE);
         ResultSet rs = st.executeQuery("SELECT * FROM Genus");

         // prolazak kroz bazu i preuzimanje opisa
         int brojac=0;
         while (rs.next()){
            int ID = rs.getInt("ID");
            EkstrakcijaInformacija ei = new EkstrakcijaInformacija(ID);
            System.out.println("Pocinje upis za " + ID);

            opis=rs.getString("GenusDesc");
```

```
if (opis==null||opis.equals("")) continue;
FileOutputStream bw =
  UnicodeIO.openUnicodeLittleEndianFileOutputStream(ei.originalText);
UnicodeIO.writeString(bw, opis);
bw.close();

//pred-procesiranje
ei.sntText = UnitexObrada.preProcessText(ei.originalText,
    ei.workingDir, ei.alphabet, ei.dictionary);

// grafovi koji treba da se obrade
String[] grafovi = {"Size.fst2", "CellSize.fst2", "GC.fst2" ,
    "pHRange.fst2","Gram.fst2","GenBank2.fst2",
   "TypeStrain.fst2","Habitat.fst2", "TempCat.fst2"};
String match = "l"; //longest match za prva cetiri grafa
String[] data = new String[9];
String sql="";

for (int i=1; i<= grafovi.length; i++){
   File graph = new File(ei.workingDir + File.separator +
       "MicrobiologyGraphs" + File.separator +grafovi[i-1]);
   //pretrazuje fajl
   // rezultat se nalazi u fajlu concord.ind i concord.n
   File exe = null;
   Process p = null;
   //locate
   exe = new File(ei.workingDir + File.separator + "App"
      + File.separator + "Locate.exe");
   if (i>4) match = "s";
   p = Runtime.getRuntime().exec(exe.getAbsolutePath() + " \"" +
      ei.sntText.getAbsolutePath() + "\" \"" +
       graph.getAbsolutePath() + "\" \"" +
       ei.alphabet.getAbsolutePath() + "\" " + match + " r all");
   p.waitFor();
   //napravljen je conocrd.ind, treba ga procitati
  File concord = new File(ei.sntDir + File.separator + "concord.ind");
   FileInputStream konkord =
       UnicodeIO.openUnicodeLittleEndianFileInputStream(concord);

   try {
      String line = UnicodeIO.readLine(konkord);
      //preskacemo prvu liniju
      line = UnicodeIO.readLine(konkord);
      data[i-1] = line.substring(line.indexOf(" ")+1);
      //uzimamo od drugog praznog mesta
      data[i-1] = data[i-1].substring(line.indexOf(" ")+1);
   } catch (Exception e)
   {
         e.printStackTrace();
         break;
   }
   //zatvori konkord
   konkord.close();
}

// UPDATE RS
rs.updateString("Size", data[0]);
rs.updateString("CellSize", data[1]);
rs.updateString("GC", data[2]);
rs.updateString("pH", data[3]);
rs.updateString("Gram", data[4]);
rs.updateString("GenBankNmbr", data[5]);
rs.updateString("TypeStrain", data[6]);
rs.updateString("Habitat", data[7]);
rs.updateString("Temperature", data[8]);
rs.updateRow();

System.out.println("Upisani su opisi za " + rs.getString("ID"));

// brise sve fajlove
File[] listaZaBrisanje = ei.sntDir.listFiles();
// brise sadrzaj direktorijuma
for (int i=1; i<listaZaBrisanje.length;i++)
   listaZaBrisanje[i].delete();
```

```
            //brise direktorijum i fajlove sa originalnim tekstom
            ei.sntDir.delete();
            ei.sntText.delete();
            ei.originalText.delete();
         }
         System.out.println("Ukupno upisano: " + brojac);
      } catch (Exception e) {
         e.printStackTrace();
      }
      finally {
         try{
            con.close();
         } catch(Exception e){}
      }
   }
}
```

BIOGRAFIJA KANDIDATA

Vesna Pajić (rođ. Stojković) rođena je u Beogradu, 1974. godine. Nakon završene osnovne škole, 1988. upisuje X beogradsku gimnaziju „Mihajlo Pupin“, a 1992. Matematički fakultet Univerziteta u Beogradu. Diplomirala je 1998. godine, na smeru Numerička matematika i optimizacija, sa prosečnom ocenom 8,96.

Nakon završenog fakulteta neko vreme radi kao programer u nekoliko kompanija u Beogradu. 2002. godine upisuje magistarske studije na Matematičkom fakultetu u Beogradu, smer Računarstvo. Magistrirala je 2010. godine sa tezom „Konačni transduktori u nadgledanju veba“.

Od 2003. godine zaposlena je na Poljoprivrednom fakultetu Univerziteta u Beogradu, kao asistent pripravnik za predmete Matematika i Informatika. 2011. godine izbarana je u zvanje asistenta. Pored rada na Poljoprivrednom fakultetu, od 2006. godine angažovana je i kao nastavnik na Visokoj školi strukovnih studija za informacione tehnologije – ITS na kojoj je držala veći broj kurseva iz oblasti računarstva i programiranja, a trenutno vodi dva premeta (Programerski alati i Internet programerski alati). Honorarno je anagažovana i u Regionalnom centru za talente u Zemunu.

Učestvovala je na nekoliko naučnih projekata Ministarstva nauke Republike Srbije. Kao rezultat naučnog rada, objavila je veći broj naučnih radova i učestvovala na nekoliko međunarodnih konferencija.

Udata je i ima tri sina.

Прилог 1.

Изјава о ауторству

Потписани-а ВЕСНА С. ПАЈИЋ

број уписа _____

Изјављујем

да је докторска дисертација под насловом

МОДЕЛИ КОНАЧНИХ СТАЊА У ЕКСТРАКЦИЈИ ИНФОРМАЦИЈА

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 4.6.2012.

Весна Пајић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора ВЕСНА ПАЈИЋ

Број уписа _____

Студијски програм _____

Наслов рада МОДЕЛИ КОНАЧНИХ СТАЊА У ЕКСТРАКЦИЈИ ИНФОРМАЦИЈА

Ментор ЂОРЂАНА ПАВЛОВИЋ-ЛАЖЕТИЋ

Потписани ВЕСНА ПАЈИЋ

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

У Београду, 4.6.2012.

Потпис докторанда

Весна Пајић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

МОДЕЛИ КОНАЧНИХ СТАЊА У ЕКСТРАКЦИЈИ ИНФОРМАЦИЈА

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 4.6.2012.

Весна Дарић