An Online Syntactic and Semantic Framework for Lexical Relations Extraction Using Natural Language Deterministic Model

Orešković, Marko

Doctoral thesis / Disertacija

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

https://doi.org/10.13140/RG.2.2.31092.19849

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:639527

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI



University of Zagreb

FACULTY OF ORGANIZATION AND INFORMATICS

Marko Orešković

AN ONLINE SYNTACTIC AND SEMANTIC FRAMEWORK FOR LEXICAL RELATIONS EXTRACTION USING NATURAL LANGUAGE DETERMINISTIC MODEL

DOCTORAL THESIS

Varaždin, 2019



Sveučilište u Zagrebu

FAKULTET ORGANIZACIJE I INFORMATIKE

Marko Orešković

MREŽNI SINTAKSNO-SEMANTIČKI OKVIR ZA IZVLAČENJE LEKSIČKIH RELACIJA DETERMINISTRIČKIM MODELOM PRIRODNOGA JEZIKA

DOKTORSKA DISERTACIJA

Varaždin, 2019.

DOCTORAL THESIS INFORMATION

I. Author

Name and surname	**Marko Orešković**
Place and date of birth	**Požega, January 23rd 1984**
Faculty name and graduation date for level VII	**University of Zagreb, Faculty of Organization and Informatics, April 3rd 2009**
Current employment	**National and University Library in Zagreb**

II. Doctoral thesis

Title	**An Online Syntactic and Semantic Framework for Lexical Relations Extraction Using Natural Language Deterministic Model**
Number of pages, figures, tables, appendixes, bibliographic information	**237 pages, 80 figures, 46 tables, 6 appendices and 179 bibliographic information**
Scientific area and field in which the title has been awarded	**Social Sciences, Information and Communication Sciences**
Supervisors	**Full Prof. Mirko Čubrilo, PhD** **Full Prof. Mario Essert, PhD**
Faculty where the thesis was defended	**University of Zagreb, Faculty of Organization and Informatics**
Thesis mark and ordinal number	**152**

III. Grade and defence

Date of doctoral thesis topic acceptance	**October 17th 2017**
Date of doctoral thesis submission	**November 19th 2018**
Date of doctoral thesis positive grade	**February 19th 2019**
Grading committee members	**Assoc. Prof. Jasminka Dobša, PhD** **Full Prof. Sanja Seljan, PhD** **Full Prof. Mirko Maleković, PhD**
Date of doctoral thesis defence	**March 15th 2019**
Defence committee members	**Assoc. Prof. Jasminka Dobša, PhD** **Full Prof. Sanja Seljan, PhD** **Assoc. Prof. Markus Schatten, PhD**
Date of promotion	



University of Zagreb

FACULTY OF ORGANIZATION AND INFORMATICS

Marko Orešković

AN ONLINE SYNTACTIC AND SEMANTIC FRAMEWORK FOR LEXICAL RELATIONS EXTRACTION USING NATURAL LANGUAGE DETERMINISTIC MODEL

DOCTORAL THESIS

Research supervisors: Prof. Mirko Čubrilo, PhD, Prof. Mario Essert, PhD

Varaždin, 2019

to my wife, Željka
and our angels Nika and Iva,
with love

A C K N O W L E D G E M E N T S

I would like to begin by expressing my gratitude to all those who made this thesis possible. First and foremost, my supervisors: Professor Mirko Čubrilo, who was of great help during my graduate studies as well as this postgraduate research project, and Professor Mario Essert, who played an important role in selection of this research project topic as well as in its conduction.

Special thanks goes to my colleague Mrs Tamara Krajna for her help and support at the beginning of my doctoral study.

In the name of all the readers of this thesis, I would specially like to thank Mrs Željka Mihljević for all the energy and effort she put into proofreading this text.

To all my friends who helped me stay focused when I was exhausted, thank you for never tiring of my stories about lexical relations and semantic domains.

I would also like to thank my parents, who believed in me and encouraged me throughout the entirety of my graduate and postgraduate studies.

Last but not least, I am deeply grateful to my family, especially my wife Željka, who stood beside me, constantly encouraged me, and cared for our children while I was working on this research. Without their patience, understanding and support, this thesis would have never been completed.

Abstract

Given the extraordinary growth in online documents, methods for automated extraction of semantic relations became popular, and shortly after, became necessary. This thesis proposes a new deterministic language model, with the associated artifact, which acts as an online Syntactic and Semantic Framework (SSF) for the extraction of morphosyntactic and semantic relations. The model covers all fundamental linguistic fields: Morphology (formation, composition, and word paradigms), Lexicography (storing words and their features in network lexicons), Syntax (the composition of words in meaningful parts: phrases, sentences, and pragmatics), and Semantics (determining the meaning of phrases). To achieve this, a new tagging system with more complex structures was developed. Instead of the commonly used vectored systems, this new tagging system uses tree-like T-structures with hierarchical, grammatical Word of Speech (WOS), and Semantic of Word (SOW) tags. For relations extraction, it was necessary to develop a syntactic (sub)model of language, which ultimately is the foundation for performing semantic analysis. This was achieved by introducing a new 'O-structure', which represents the union of WOS/SOW features from T-structures of words and enables the creation of syntagmatic patterns. Such patterns are a powerful mechanism for the extraction of conceptual structures (e.g., metonymies, similes, or metaphors), breaking sentences into main and subordinate clauses, or detection of a sentence's main construction parts (subject, predicate, and object). Since all program modules are developed as general and generative entities, SSF can be used for any of the Indo-European languages, although validation and network lexicons have been developed for the Croatian language only. The SSF has three types of lexicons (morphs/syllables, words, and multi-word expressions), and the main words lexicon is included in the Global Linguistic Linked Open Data (LLOD) Cloud, allowing interoperability with all other world languages. The SSF model and its artifact represent a complete natural language model which can be used to extract the lexical relations from single sentences, paragraphs, and also from large collections of documents.

Keywords: syntax analysis, semantic analysis, lexical relations extraction, new lexicon types, hierarchical tagset structure, linked open data

Prošireni sažetak

Prirodni jezik predstavlja proces tvorbe i povezivanja (izgovorenih ili napisanih) riječi, s ciljem prenošenja informacije i znanja, koje riječi kao leksičke jedinice u sebi i među sobom nose. S obzirom da je broj leksičkih jedinica, a pogotovo njihovih kombinacija u rečenicama nekog jezika ogroman, strojna obrada u smislu prepoznavanja informacije u digitalnim dokumentima zahtjevan je znanstveni izazov. Izvlačenje leksičkih relacija računalom znači oponašati procese razmišljanja ljudskog uma, tj. baviti se problemima umjetne inteligencije. U ovoj radnji načinjen je deterministički model prirodnoga jezika i realiziran u obliku mrežnog artefakta: Sintaktičko-semantičkog mrežnog okvira (eng. *Syntactic and Semantic Framework - SSF*). Apstraktni model prirodnoga jezika osmišljen je kroz novi matematički model, na tragu semantičkih modela Igora Mel'čuka [117] i Vladimira A. Fomichov-a [64], a koji je onda implementiran u relacijskoj bazi s 40 tablica, 250 atributa i oko 200 indeksa. Model pokriva sva temeljna područja jezikoslovlja: morfologiju (tvorbu, sastav i paradigme riječi) s leksikografijom (spremanjem riječi i njihovih značenja u mrežne leksikone), sintaksu (tj. skladnju riječi u cjeline: sintagme, rečenice i pragmatiku) i semantiku (određivanje značenja sintagmi). Da bi se to ostvarilo, bilo je nužno označiti riječ složenijom strukturom, umjesto do sada korištenih vektoriziranih gramatičkih obilježja poput MULTEXT-East standarda. Taj novi pristup temelji se na stablenoj T-strukturi koja ima rekurzivna, gramatička (WOS) i semantička (SOW) obilježja. Grane T-strukture mogu biti višerazinske i imati svoja tekstualna, numerička i poveznička obilježja, što predstavlja rudimentarnu ontologijsku strukturu. Ona omogućuje stvaranje novih vrsta rječnika, na tragu 'generativnog leksikona' Jamesa Pustejovskoga [136] u kojem svaka riječ ima ne samo svoje gramatičko značenje (npr. vrstu riječi i gramatičke kategorije), nego i okoliš u kojem se može pojaviti, tzv. semantičke domene.

Tako označene riječi spremljene su kao tri međusobno povezana mrežna rječnika: LEX – koji sadrži oko 800.000 riječi (svih pojavnica), MSY – rječnik subatomarnih dijelova riječi (slogova, morfema i silabomorfema) s oko 10.000 sastavnica, te MWE – rječnik višerječnica (kolokacija, frazema, termina i sl.) s oko 130.000 primjeraka. Zato ne čudi što računalna baza modela zaprema oko 5 GB prostora. Model LEX rječnika uključuje polje s naglaskom riječi, može sadržavati sliku koju riječ predstavlja i njen izgovor (zvuk). Posebnost ovog leksikona je također njegova povezanost s mrežnim znanjem drugih domaćih (Leksikografski zavod Miroslav Krleža - LZMK, Hrvatski jezični portal - HJP, Croatian WordNet - CroWN) i stranih (BabelNet, WordNet) mrežnih repozitorija. SSF je uključen u lingvistički oblak povezanih podataka (LLOD), što znači da je ostvarena interoperabilnost s ostalim svjetskim jezicima. Te i takve povezanosti imaju izravan učinak na jezikoslovne analize koje SSF nudi, što je i postavljeni cilj ove radnje – izvlačenje leksičkih (sintaktičko-semantičkih)

relacija u nekom nestrukturiranom tekstu.

Da bi se relacije mogle pronalaziti bilo je potrebno osmisliti sintaktički (pod)model jezika, na kojem će se u konačnici graditi i semantička analiza. To je postignuto uvođenjem nove, tzv. O-strukture, koja predstavlja uniju WOS/SOW obilježja iz T-struktura pojedinih riječi i omogućuje stvaranje sintagmatskih uzoraka. Ti uzorci spremaju se kao poseban rječnik (O-rječnik) nekog prirodnog jezika i čine osnovne elemente rečeničnih struktura. O-strukture i pripadni im uzorci služe kod ulazne raščlambe (*parsiranja*) novog teksta, tj. stvaranja SSF korpusa, a također i za sintaktičku analizu rečenice. SSF okvir omogućuje interaktivnu vizualnu, programsku i mrežno dohvatljivu analizu. Prva omogućuje pretraživanja riječi sa zadanim obilježjima i susjedstvo s riječima koje ih kontekstualno okružuju. Za drugu vrstu analize služi poseban funkcijski programski modul FPJ u kojem se mogu pisati i izvoditi programske funkcije u poznatim programskim jezicima (Python, R, Haskell, SPARQL). Za taj modul napisano je više od 40 često korištenih programskih funkcija, koje prijavljeni korisnik može pozivati iz svojih specifično pisanih programa. Sam program, pisan u PHP kodu, ima oko 30.000 programskih linija. Važno je napomenuti da se u FPJ-u mogu izvoditi i programski moduli drugih autora, kao što su popularni lingvistički alati: NLTK, TextBlob, CliPS, Scikit-learn i drugi. To znači da se mnogobrojne gramatičke teorije (npr. Chomsky-eva generativna gramatika [30], Halliday/Caplan/Bresnan funkcionalna gramatika [75] i sl.) mogu ispitivati i usavršavati unutar SSF artefakta. Kao primjer, pokazana je analiza zavisnih rečenica hrvatskoga jezika, koje stroj, na temelju 35 ispravno napisanih deterministčkih uzoraka O-struktura, ispravno prepoznaje i svrstava u 14 tipova poznatih zavisnih rečenica tradicijske hrvatske gramatike. Dakako, napisani su i uzorci za otkrivanje službe riječi u rečenici (SPO - subjekta, predikata, objekta i dr.), kao i provjere gramatičkih podudaranja (npr. imenice i pridjeva u rodu, broju i padežu i sl.). Nužan preduvjet izvlačenju semantičkih relacija je baš ovo rješenje izvlačenja sintaktičkih relacija uz pomoć sintaktičkih uzoraka. Rječnik uzoraka načinjen je u potpunosti kao model i realiziran kao (pod)modul artefakta, ali dakako nije napunjen svim instancama jezika (kojih može biti i više desetaka tisuća). To će se, kao i u slučaju LEX/MSY/MWE rječnika, prepustiti jezikoslovnim stručnjacima, te graditi i provjeravati u idućem razdoblju razvoja SSF okvira. Na koncu, treća, mrežno dohvatljiva analiza, temelji se na razvijenim API modulima, kako bi se SSF artefakt mogao iskoristiti i za primjene u drugim područjima, npr. edukaciji i prijevodima. Radi se o samostalnim programskim rješenjima koja rješavaju specifičnu zadaću, npr. učenje jezika, a za provjeru rezultata preko API komunikacije koriste SSF model.

Leksičke relacije sadrže u sebi i semantičku komponentu koja je strojno teže prepoznatljiva, budući da riječi, a pogotovo njihove kombinacije, daju mnoštvo višeznačnosti, tj. konceptualni objekti se teško strojno identificiraju. Ipak, SSF model zahvaljujući

povezanosti svih svojih internih modula: MSY + MWE + T-struktura + O-struktura (uzorci) i posebnog obilježja s imenom 'Domena' u LEXu uspješno rješava i taj problem, što se i kroz artefakt s nekoliko primjera može pokazati (otkrivanje usporedbi, metonimija i metafora). Semantička domena je obilježje riječi koje predstavlja skup riječi među kojima se dotična riječ može pojaviti. Jedna domena daje jedno značenje riječi, a druga neko drugo, ovisno o kontekstu. SSF stvara automatski domene na temelju informacije koju dohvaća na mreži (npr. LZMK ili BabelNet enciklopedija) i potom pravi skupove riječi, npr. one s vrstama riječi (imenica, glagol, pridjev, prilog) ili SPO službom u rečenicama koje se nalaze u definicijama neke enciklopedijske natuknice. Tako se na primjer, za riječ 'Jezik' u LEX leksikonu otkriva čak 17 (semantičkih) domena. Neke od njih se mogu (to se mora načiniti ručno) preko T-strukture označiti jezgrenima (realnima), dok će ostale pritom postati metaforične ili metonimijske. Semantička analiza neke, dotad nepoznate, rečenice, je moguća jer stroj može usporediti obilježja svake njene riječi s domenama u kojima se ona pojavljuje. Taj složeni postupak riješen je originalnim pristupom obogaćivanja riječi u rečenici s onim ontologijskim parametrima koji su zanimljivi u željenom traženju, tj. riječi u rečenici se ne obogaćuju svim obilježjima koje pojedina riječ u T-strukturi ima, nego samo onim granama stabla koje mogu dovesti do točnog rezultata. Takva optimizacija traženja uporabom regularnih izraza znatno povećava brzinu izvedbe samog otkrivanja i izvlačenja semantičkih relacija iz teksta i predstavlja ostvareni cilj ove radnje.

Za sve korisnike koji nisu vješti u pisanju regularnih izraza načinjen je interaktivni generator uzoraka u kojem se na intuitivan način zadaju riječi i/ili njihovi WOS/SOW uzorci koji onda daju filtre regularnih izraza s kojima se iz teksta izvlači sintaktičko-semantička informacija. Budući da je u SSFu složen i MSY rječnik, u tom traženju informacije često pomaže i njegova subatomarna struktura riječi, npr. afiks '-ica' za umanjenice ili neki prefiksi riječi za neko stanovito značenje. Ta povezanost između dubinske semantičke i površinske sintaktičke reprezentacije na tragu je Melčuk-ove MTT (Meaning Text Theory) teorije, a tri vrste SSF rječnika na tragu njegovog, teorijski predloženog, i samo djelomično realiziranog kombinatorijskog ECD rječnika (Explanatory Combinatorial Dictionary) u Rusiji, Francuskoj i Kanadi [117].

Prirodni jezik osim svoje sintagmatske (horizontalne) ima i paradigmatsku (vertikalnu) komponentu koja se intenzivno proučava u semantici. Radi se o pozicioniranju pojedine riječi unutar sinonimskih skupova, pojmovnih relacija kao što su sinonimi i antonimi, hiponimi i hipernimi, meronimi i holonimi i sl. Za njihovu analizu i obradu SSF ima više vlastito razvijenih funkcija, npr. otkrivanje antonima s uvjetnim argumentima (starica je antonim od djevojke po dobi, dok je mladić antonim po spolu), traženje podskupova semantičkih domena za hiponime i dr. Dakako, WordNet i njegova hrvatska inačica CroWN također je uključena u ovaj SSF model, ali i sinonimski skupovi uvažene hrvatske

jezikoslovkinje Ljiljane Šarić. Ovaj rad svojim novim pristupom O-struktura (sintaktičko-semantičkih uzoraka), koje djeluju nad riječima kao rudimentarnim ontologijama, otvara novi pogled na središnju temu jezikoslovnog trenda poznatog pod pojmom 'linguistic valence', kojom se naglašava međusobna povezanost sintakse i smisla. Po uzoru na kemijske elemente, već se od 60-tih godina prošlog stoljeća (Tesniere [159]), proučavaju valencije riječi, u početku samo glagola. U svijetu je načinjeno već nekoliko valencijskih rječnika (postoje pokušaji i kod nas), a cilj im je poslužiti u identifikaciji značenja preko sintaktičkih relacija leksičkih jedinica (Jackendoff [82]). Upravo je O-struktura, izgrađena kao uzorak WOS/SOW obilježja, načinila tu jedinstvenu uniju valencijski povezanih riječi i njihovoga (SOW i/ili MWE) značenja, uz formalizirane semantičke domene. Za SSF model sasvim je svejedno o kojoj se vrsti riječi radi – rječnik uzoraka sprema glagole i s njima svezane druge riječi, kao i za prijedloge ili imenice. Na primjer, zahvaljujući globalnom uzorku 'prijedlog imenica', stroj neće iz teksta 'iz hrama' zaključiti da se radi o prijedlogu iza kojeg dolazi aorist glagola 'hramati – hramah, hramaš, hrama...', nego će znati da se radi o imenici (za koju ovaj jednostavan uzorak postoji u O-rječniku). Slično je i s drugim uzorcima u kojima je, zahvaljujući SOW oznakama, spremjeno i značenje (npr. kolokacija u MWE rječniku, sinonima u O-rječniku, onomastike iz SOW obilježja i sl.).

Na koncu, obrada jezika, pogotovo u vrijeme ogromnog broja digitalnih dokumenata na globalnoj mreži, zahtijeva i statističku obradu. SSF je za to predvidio Python i R statističke alate i njihovu izravnu povezanost s učitanim ili mrežno dohvaćenim korpusom. To znači da su u SSFu omogućena i statistička istraživanja i rudarenja podataka u kojima riječi gube gramatička, a poprimaju statistička obilježja (frekvencije ili učestalost pojavljivanja, n-gramske pojavke, distribucije i sl.), pa se primjerima pokazuju analize tekstova preko Markovljevih lanaca, Analize glavnih komponenti (PCA) i sl. U radu je pokazano nekoliko stohastičko-statističkih analiza odabranog korpusa s klasičnim algoritmima obrade sa i bez uključivanja SSF spremljene informacije. Pokazuje se da je moguće unaprijediti rezultate stohastičkih metoda korištenjem SSF-a. To je posve razumljivo u fleksijskim jezicima, kao što je hrvatski, gdje stroj svaku različitu pojavnicu riječi u klasičnim algoritmima smatra različitom, dok uz pomoć SSF informacije isto obilježje pridjeljuje svim riječima koje imaju istu lemu. Zahvaljujući O-rječniku sintaktičko-semantičkih uzoraka čak i one pojavnice koje su iste, ali imaju različitu lemu (npr. 'usta' kao imenica i 'usta' kao 3. lice aorista od glagola 'ustati') i bez navedenog naglaska riječi (po kojem se sigurno razlikuju), stroj može razlikovati jedno od drugoga, jer je to zapisano u jednom od općih O-uzoraka.

S obzirom da su svi programski moduli mrežnog okvira razvijeni kao opći i generativni entiteti, ne postoji nikakav problem korištenja SSF-a za bilo koji od indoeuropskih jezika, premda su provjera njegovog rada i mrežni leksikoni izvedeni za sada samo za hrvatski jezik. S ovako osmišljenim i realiziranim načinom, SSF model i njegov realizirani artefakt,

predstavljaju potpuni model prirodnoga jezika s kojim se mogu izvlačiti leksičke relacije iz pojedinačne rečenice, odlomka, ali i velikog korpusa (eng. *big data*) podataka.

Extended abstract

Natural language represents the process of forming and connecting words (spoken or written), with the aim of transmitting information and knowledge, which words carry as lexical units within and among themselves. Since the number of lexical units is enormous, especially their combination in sentences of a language, machine processing in terms of identifying information in digital documents is a highly demanding scientific challenge. Computer-based lexical relation extraction is founded on the principles of human thinking (i.e. the principles of artificial intelligence). This thesis proposes a new, deterministic language model with the associated artifact, Syntactic and Semantic Framework (SSF), which acts as an online framework for the extraction of morphosyntactic and semantic relations. An abstract model of natural language is conceived through a new mathematical model, which is inspired by the semantic models of Igor Mel'čuk [117] and Vladimir A. Formichov [64] and implemented in a relational database containing 40 tables, 250 attributes, and over 200 indices. The model covers all fundamental linguistic fields: Morphology (formation, composition, and word paradigms) with Lexicography (storing words and their features in network lexicons), Syntax (the composition of words in meaningful parts: phrases, sentences, and pragmatics), and Semantics (determining the meaning of phrases). To achieve all of this, it was necessary to develop a tagging system with more complex structures, instead of the commonly used vectored systems like MULTEXT-East standard. Newly developed tagging system is based on the usage of tree-like T-structures with recursive, grammatical (WOS), and semantic (SOW) tags. Branches of such T-structures can be multileveled and contain textual, numeric, and linking features, which represent rudimentary ontological structure. Such structure enables the building of new lexicons, such as the 'generative lexicon' from James Pustejovsky [136], in which every word, along with its grammatical meaning (e.g. open word class and grammatical categories), also has the environment in which it can exist (the so called 'semantic domains').

Words that are tagged in that way are stored in three interconnected network lexicons: LEX, which contains over 800,000 words (in all forms); MSY, the lexicon of subatomic word parts (syllables, morphs and syllablemorphs) with over 10,000 parts; and MWE, the lexicon of multiword expressions (collocations, phrases, terms, etc.) with over 130,000 entries. The database containing these lexicons uses 5 GB of storage space. The LEX model, along with words, also contains the word's accent, image, or sound (spoken variant of the word). The specialty of this lexicon is also its link to other local network resources (e.g. The Miroslav Krleža Institute of Lexicography - LZMK, Croatian Language Portal - HJP, Croatian WordNet - CroWN), as well as foreign ones (e.g. BabelNet and WordNet).

The SSF is included in the global Linguistic Linked Open Data (LLOD) cloud, which means that interoperability with all other world languages is achieved. These links produce a direct impact on language analysis that the SSF is able to conduct, such as the extraction of lexical (syntactic and semantic) relations from an unstructured text, which is also one of the goals of this thesis.

For relations to be extracted, it was necessary to develop a syntactic (sub)model of language, which ultimately would be the foundation for conduction of semantic analysis. This was achieved by introducing the new, so-called 'O-structure', which represents the union of WOS/SOW features from T-structures of words and enables the creation of syntagmatic patterns. These patterns are stored in the form of a new lexicon (O-lexicon) of a natural language and make up basic elements of a sentence's structures. O-structures and their patterns are used in the process of parsing new documents, e.g. building the SSF corpus, and for syntactic analysis of sentences as well. The SSF framework provides an interactive, visual program and network-accessible analysis. The first feature makes it possible to search for words with specific features as well as the words that contextually surround them. For the second type of analysis, there is a special programmatic module called Natural Language Functions (NLF), which supports different programming functions from popular programming languages (Python, R, Haskell, SPARQL). There are over 40 most commonly used functions prepared for usage within the NLF module, which the signed user can execute within his own scripts. The SSF itself is written in the PHP programming language and contains over 30,000 lines of code. It is important to mention that NLF can also execute programming modules of other authors, such as popular linguistic tools: NLTK, TextBloc, CLiPS, Scikit-learn, etc. This means that many grammatical theories (e.g. Chomsky's generative grammar [30] or Halliday/Caplan/Bresnan's functional grammar [75]) can be examined and improved within the SSF artifact. As an example, an analysis of compound sentences in the Croatian language is shown. The machine, based on the 35 correctly written, deterministic O-structure patterns, correctly recognizes and categorizes the sentences in the 14 distinct types of complex sentences of traditional Croatian grammar. Of course, the patterns for detection of word roles within the sentence (SPO - subject, predicate, object, etc.) are also defined, as well as validation of grammatical matching (e.g. nouns and adjectives in gender, number, and case, etc.). The prerequisite for the extraction of semantic relations is the extraction of syntactic relations by using syntactic patterns. The lexicon of patterns is developed completely as a model and realized as a sub(module) of the artifact, but is not loaded with all language instances (there could be tens of thousands and more). This will, as in the case of the LEX/MSY/MWE vocabulary, be passed on to the linguistic experts, and constructed and verified in the next period of development of the SSF framework. In the end, the third, network-accessible analysis is

based on API modules, which enables the usage of the SSF in other fields as well (e.g. education and translation). These third-party program solutions are designed to solve specific tasks (e.g. foreign language learning, which use the SSF model for validation of results).

Lexical relations within themselves also contain a semantic component that is more difficult to recognize by the machine, since the words, especially their combinations, may result in many ambiguities (i.e. it is more difficult to identify conceptual objects). However, the SSF model, due to the interconnectivity of its internal modules (MSY + MWE + T-structures + O-structures (patterns) and special 'Domain' tags in the LEX), successfully solves that problem, which can also be demonstrated in the artifact (e.g. detection of similes, metonymies, and metaphors). The semantic domain is a special feature of words which is a set of other words that are related to the observed word. One domain can give a word one meaning, whereas some other domain can give a word a completely different meaning based on the context. The SSF automatically generates domains based on the network available definitions (e.g. LZMK, BabelNet, etc.), and, in the process, creates sets of words, e.g. ones with open word classes (nouns, verbs, adjectives, adverbs) or SPO roles in the sentence. For example, the word cro. '*jezik*' is associated with 17 different (semantic) domains. Some of them can be tagged as core (real) domains (this needs to be done manually), whereas other can be tagged as metaphorical or metonymical. The semantic analysis of some, previously unknown, sentence is possible since the machine can compare tags from each of the sentence's word use with domains in which the word is contained. This complex procedure is based on a new, original approach to word enrichment in the sentence with ontological parameters that are related to a specific search, i.e. words in the sentences are not enriched with all the tags that they have, but only with those that can lead to the correct result. Such optimization of matching using regular expressions drastically increases the speed of execution and extraction of semantic relations in textual documents and represents the achieved goal of this thesis. For users who are not experienced in writing regular expressions, there is a special interactive generator to aid them in the process of defining WOS/SOW patterns and their integration in regular expressions for the extraction of syntactic and semantic information. Since the SSF also contains complex MSY lexicon, in such extractions, the usage of words' subatomic components may usually help (e.g. affix '-ica' for diminutives or some word's prefixes for some specific meanings). That interconnectivity between deep semantic and surface syntactic representations is inspired by Mulčuk's MTT (Meaning Text Theory); three types of SSF's lexicon are inspired by his theoretically proposed, and only partly realized, Explanatory Combinatorial Dictionary (ECD) in Russia, France, and Canada [117].

The natural language, beside its syntagmatic (horizontal) component, also has a paradigmatic (vertical) component that is researched extensively in semantics. Such research deals with a word's position within synonymic sets, and terminological relations like synonyms and antonyms, hyponyms and hypernyms, meronyms and holonyms, etc. For the analysis of such relations, the SSF has its own set of functions, e.g. for the detection of antonyms with conditional arguments (the 'old lady' is an antonym of the word 'girl' based on age, but if gender is taken as a criterion of antonymity, the result may be a 'young man'). Of course, the WordNet and its Croatian version (CroWN) are also included in the SSF, as well as synonymic sets of the distinguished Croatian linguist Ljiljana Šarić. This thesis, with its original approach to O-structures (syntactic and semantic patterns), which treat words as rudimentary ontologies, opens a new view of the central theme of the linguistic trend known as 'linguistic valence', emphasizing the interconnectedness of syntax and meaning. Inspired by chemical elements, the valency of words (in the beginning only for verbs) has been examined since the '60s of the last century (Tesnière [159]). Several valency dictionaries have been made in the world (there are also attempts in Croatia to produce them), with the main goal being to help in the process of a word's meaning detection through the syntactic relations of lexical units (Jackendoff [82]). The O-structure, constructed as a pattern of WOS/SOW features, created a unique union of words' valences and their (SOW and/or MWE) meanings with formalized semantic domains. When it comes to the SSF model, it does not matter which word class it is about; the lexicon of patterns contains verbs and other words related to them, as well as adverbs or nouns. For example, based on the global pattern called 'adverb noun', analyzing the phrase cro. '*iz hrama*' will not conclude that there is an adverb followed by the aorist of the verb, cro. '*hramati*', but will rather conclude that it is a noun (for which this simple pattern is stored in the O-lexicon). It is similar for other patterns, too, for which the meaning of the word is also stored (e.g. collocation in the MWE lexicon, synonyms in the O-lexicon, onomastic from SOW tags, etc.) due to the SOW tags.

In the end, natural language processing, especially at a time of an enormous number of digital documents on the global network, requires statistical processing too. The SSF embeds Python and R statistical tools and their direct connection with a loaded or network fetched corpus. This means that the SSF offers tools for statistical analysis of textual data and data mining in which words lose their grammatical features, and receive their statistical features (frequencies, n-gram sets, distributions, etc.); therefore, different examples like Markov chains, Principal component analysis (PCA), etc. are shown. This thesis presents several stochastic-statistical analyses of the selected corpus using classical processing algorithms with and without the inclusion of SSF stored information. It is shown that the same algorithms may be improved if the information from the SSF is used.

This is understandable since in flexible languages, such as Croatian, the machine considers each different occurrence of the word differently, whereas with the help of SSF, the same information is assigned to all the words that have the same lemma. With the O-lexicon of syntactic and semantic patterns, even words which are written the same but have different lemmas (e.g. the word cro. '*usta*' as a noun and cro. '*usta*' as a third person aorist of the verb cro. '*ustati*') can be recognized one from another by the machine because such patterns are also stored in the global O-lexicon.

Since all program modules are developed as general and generative entities, there is no problem using the SSF for any of the Indo-European languages, although its work validation and network lexicons have been done only for the Croatian language. So designed and realized, the SSF model and its artifact represent a complete natural language model which can be used to extract the lexical relations from single sentences, paragraphs, but also from the large corpus.

C O N T E N T S

Contents I

List of Figures IV

List of Tables VII

Abbreviations IX

1. Introduction 1
- 1.1. Objective 2
- 1.2. Hypotheses and research questions 4
- 1.3. Related works 6
- 1.4. Research methodology 9
- 1.5. An outline of the thesis 11

2. Deterministic language model 12
- 2.1. Abstract model 13
- 2.2. Conceptual model 20
- 2.3. Model realization 23
- 2.4. Statistical methods 25

3. Tagging 41
- 3.1. Types of tagsets 46
- 3.2. T-structures 49
- 3.3. Word tagging 51
- 3.4. MWE tagging 56
- 3.5. Lemma tagging 59

4. Lexicography 62
- 4.1. The word grammar 63
- 4.2. Different types of lexicons 67
- 4.3. Generative Lexicon requirements 75

5. Syntax . . . 83
5.1. Syntax model . . . 83
5.2. Word as a syntactic unit . . . 84
5.3. Sentences . . . 87
5.4. Natural Language Functions . . . 97
5.5. Regular expressions . . . 99
5.6. O-structures . . . 101

6. Semantics . . . 103
6.1. Semantic model . . . 103
6.2. Lexical functions . . . 106
6.3. Semantic domains . . . 108
6.4. Integration of external resources . . . 112
6.5. Sentiment analysis . . . 114

7. Extraction of lexical relations . . . 119
7.1. Corpora . . . 119
7.2. Extraction of word's environment in the SSF . . . 123
7.3. Conceptual structures . . . 128
7.4. Extraction of relations using O-structures . . . 132
7.5. Artifact API functions . . . 137

8. Semantic Web integration . . . 140
8.1. LOD wrapper . . . 143
8.2. Virtuoso triplestore . . . 146
8.3. SPARQL queries . . . 148
8.4. Croatian word in the Linguistic Linked Open Data Cloud . . . 152

9. Conclusion . . . 155

References . . . 158

Appendixes . . . 174

A. Creation of static domains from SOW definitions . . . 175

B. Creation of RDF triples . . . 180

C. Vocal changes in Croatian . . . 187

D. Python Natural Language Functions 191

E. WOS/SOW marks 219

F. List of tags in different tagging systems 226

LIST OF FIGURES

No. **Figure name** **Page**

Figure 1. Conceptual Model of the Syntactic and Semantic Framework 21
Figure 2. The SSF's main screen 23
Figure 3. Homonym 'usta' as a noun and a verb 24
Figure 4. Network lexicon - a source of a lexical information 25
Figure 5. Visualized results from the SSF function in the R 29
Figure 6. Correlations of variables for PCA 31
Figure 7. Components contributions to PCA 32
Figure 8. PCA Plot 33
Figure 9. A sample cluster dendrogram of Croatian intensifiers divided into three groups 36
Figure 10. A sample cluster dendrogram of Croatian intensifiers divided into two groups 37
Figure 11. Conceptual model of a tagging subsystem 41
Figure 12. T-structures in relation to the commonly used annotation models . . 48
Figure 13. Generation of MULTEXT-East tags from T-structures 49
Figure 14. T-structures 50
Figure 15. Database model of word tagging 51
Figure 16. Lexicon entry with associated word tags 52
Figure 17. Lexicon entry with associated image tags 54
Figure 18. Lexicon entry with associated sound tags 55
Figure 19. Database model of the MWE subsystem 56
Figure 20. MWE Lexicon entry 57
Figure 21. Database model of a lemma subsystem 59
Figure 22. Lemma lexicon entry 60
Figure 23. Conceptual model of the lexicon subsystem 62
Figure 24. Croatian morphology generator 67
Figure 25. Database model of the LEX subsystem 68
Figure 26. Screenshot of SSF's Lexicon setup screen 68
Figure 27. Screenshot of SSF's Lexicon output 70

Figure 28. Screenshot of SSF's Lexicon tags . . . 71
Figure 29. Lexical data validation . . . 72
Figure 30. Screenshot of SSF's MSY Lexicon setup screen . . . 73
Figure 31. Screenshot of SSF's MSY Lexicon output for syllables . . . 73
Figure 32. Screenshot of SSF's MWE Lexicon setup screen . . . 74
Figure 33. Screenshot of SSF's MWE Lexicon output . . . 75
Figure 34. Connection of grammatical and semantic relations [167] . . . 80
Figure 35. Conceptual model of the syntactic subsystem . . . 83
Figure 36. Generative grammar parsing in the SSF . . . 91
Figure 37. Dependency grammar parsing in the SSF . . . 91
Figure 38. Sentence splitting to MC and SC using O-structures . . . 94
Figure 39. Sentence splitting to MC and SC using NLTK . . . 96
Figure 40. Tree representation of the split sentence . . . 97
Figure 41. Conceptual model of the Natural Language Functions subsystem . . . 97
Figure 42. The SSF's Interface for the Natural Language Functions execution . . . 98
Figure 43. Conceptual model of the semantic subsystem . . . 103
Figure 44. Conceptual model of the semantic domains subsystem . . . 108
Figure 45. Semantic domains types . . . 108
Figure 46. Database model of the domains subsystem . . . 109
Figure 47. Automatic creation of a semantic domain . . . 110
Figure 48. Example of a semantic domain in SSF's lexicon . . . 110
Figure 49. Domains editor . . . 112
Figure 50. External lexicographic resources in the SSF . . . 114
Figure 51. The sentiment lexicons . . . 115
Figure 52. Sentiment analysis with SenticNet in the SSF . . . 117
Figure 53. Conceptual model of the parsing subsystem . . . 120
Figure 54. Flowchart diagram of the Parser component . . . 121
Figure 55. Database model of the corpora subsystem . . . 123
Figure 56. Screenshot of SSF's word search . . . 123
Figure 57. Word searching filters (WOS) . . . 124
Figure 58. Word searching results . . . 125
Figure 59. Word searching results with WOS/SOW info . . . 125
Figure 60. Triples crawling . . . 126
Figure 61. Visualisation of sentence segments . . . 127
Figure 62. Conceptual structures . . . 129
Figure 63. Metonymy detection in the SSF . . . 130
Figure 64. Metaphor detection in the SSF . . . 131

Figure 65. Example of sentence tagging patterns 132
Figure 66. Future tense O-structure pattern 132
Figure 67. Screenshot of O-structures editor 133
Figure 68. Division of a simple sentence into SPO 134
Figure 69. Screenshot of the web form for pattern search 135
Figure 70. Screenshot of the results of pattern searching 136
Figure 71. Process of extraction of sentences from the corpus 136
Figure 72. API call diagram 139
Figure 73. Conceptual model of the LOD subsystem 141
Figure 74. Building Linguistic Ontology in Protégé 142
Figure 75. Graph representation of one lexical entry in RDF 142
Figure 76. The architecture of D2RQ platform [17] 144
Figure 77. W3C Validator output for SSF ontology 147
Figure 78. Running SPARQL queries in the SSF 151
Figure 79. SSF's Lexicon as a dataset in the Datahub 152
Figure 80. The SSF in the LOD Cloud 154

LIST OF TABLES

No.	Table name	Page
Table 1.	Part of speech tags	14
Table 2.	Some open word class categories	14
Table 3.	Flexion phonology	16
Table 4.	Flexion morphology	16
Table 5.	Nouns and Verbs categories	17
Table 6.	Accentual declination of the reflexive pronoun 'ja'	18
Table 7.	An overview of syntactic phrases	19
Table 8.	Some commonly-used thematic roles with their definitions, adapted from [86]	20
Table 9.	Co-occurrence frequencies for association measurements	38
Table 10.	Croatian words by classes	55
Table 11.	Croatian words by sentiment polarity	56
Table 12.	Qualia theory roles [136]	79
Table 13.	State verbs	81
Table 14.	Activity verbs	81
Table 15.	Syntactical patterns for decomposition of complex sentences	93
Table 16.	Metacharacters in regular expressions	99
Table 17.	Regular expressions predefined character sets	100
Table 18.	Regular expressions modifiers	100
Table 19.	Decomposition results of an adverbial of place in sentences [35]	101
Table 20.	The problem of multiplicity and MWE	104
Table 21.	Examples of lexical functions	107
Table 22.	O-structures for predicate extraction	134
Table 23.	List of WOS and SOW marks	219
Table 24.	Alphabetical list of Penn Treebank tags [100]	226
Table 25.	MULTEXT-East Croatian categories	227
Table 26.	MULTEXT-East Croatian Specification for Nouns	227
Table 27.	MULTEXT-East Croatian Specification for Verbs	228
Table 28.	MULTEXT-East Croatian Specification for Adverbs	228
Table 29.	MULTEXT-East Croatian Specification for Adpositions	228

Table 30. MULTEXT-East Croatian Specification for Adjectives 229
Table 31. MULTEXT-East Croatian Specification for Conjunctions 229
Table 32. MULTEXT-East Croatian Specification for Particles 229
Table 33. MULTEXT-East Croatian Specification for Pronouns 230
Table 34. MULTEXT-East Croatian Specification for Numerals 231
Table 35. MULTEXT-East Croatian Specification for Residuals 231
Table 36. SWETWOL Part of Speech tags 232
Table 37. SWETWOL Verbal inflection tags 232
Table 38. SWETWOL Nominal inflection tags 233
Table 39. SWETWOL Derivational tags 233
Table 40. SWETWOL Governmental definiteness tags for determiners 234
Table 41. SWETWOL Other tags specifically for pronouns and determiners . . 234
Table 42. SWETWOL Other additional tags 235
Table 43. SWETWOL Miscellaneous tags 235
Table 44. UD POS Tags for open class words 236
Table 45. UD POS Tags for closed class words 236
Table 46. UD POS Tags for other words 236

ABBREVIATIONS

AC Agglomerative clustering. 33
API Application Programming Interface. 2, 11, 22

BNC British National Corpus. 30, 119

CCA Corpus of Contemporary American English. 120
CFG Context Free Grammar. 43
CG Cognitive grammar. 90
CroWN Croatian WordNet. 8, 84, 113
CSF Croatian Science Foundation. 131

DG Dependency grammar. 5, 89, 92
DM Derivational morphology. 65
DT Dependency Tree. 7

ECD Explanatory Combinatorial Dictionary. 10
ER Entity–relationship model. 12

FA Factor Analysis. 30
FG Functional grammar. 90

GL Generative Lexicon. 5, 75, 82
GUI Graphical user interface. 51, 97

HJP Hrvatski jezični portal. 24
HTML HyperText Markup Language. 152

IM Inflectional morphology. 65

JSON JavaScript Object Notation. 11

KBP Knowledge Base Population. 7

LF Lexical Function. 106
LFG Lexical functional grammar. 5, 90, 92
LLOD Linguistic Linked Open Data. 52, 104
LOB The Lancaster-Oslo-Bergen Corpus. 47
LOD Linked Open Data. 6, 10, 104, 141
LSA Latent Semantic Analysis. 3
LZMK Leksikografski zavod Miroslav Krleža. 24, 47

MaxEnt The Maximum Entropy. 45
MBT Memory Based Tagging. 45
MC Main Clause. 92
MLP Multilayer Perceptron Network. 45
MTT Meaning Text Theory. 5, 106
MVC Model-view-controller. 12
MWE Multiword expression. 47, 67

NLF Natural Language Functions. 24, 97, 98
NLP Natural Language Processing. 42
NLPS Natural Language Processing Systems. 1
NLTK Natural Language Toolkit. 11, 94, 98
NP Noun Phrase. 83

OIE Open Information Extraction. 4, 7
OMCS Open Mind Common Sense. 115
OWL Web Ontology Language. 140, 153

PAS Predicate Argument Structure. 7
PCA Principal Components Analysis. 12, 30

POS Part of Speech. 7, 42, 43, 50, 116
PP Preposition Phrase. 83
PWN Princeton Wordnet. 84, 112

RDF Resource Description Framework. 6, 52, 104, 140, 141, 152, 153
REGEX Regular expression. 99
REST Representational state transfer. 137
RRG Role and Reference Grammar. 43

SC Subordinate Clause. 92
SG Stochastic grammar. 90
SGML Standard Generalized Markup Language. 43
SOW Semantic of Word. 47, 49–51
SPARQL SPARQL Protocol and RDF Query Language. 10, 22, 148
SQL Structured Query Language. 10, 143
SSF Syntactic and Semantic Framework. 20, 22, 49, 50, 97, 98, 134
SVD Singular Value Decomposition. 3
SWECG Swedish Constraint Grammar. 232

TBL Transformation-based Error-driven Learning. 44
TF-IDF Term Frequency–Inverse Document Frequency. 3
TnT Trigrams'n'Tags. 44

UG Universal grammar. 89
ULO Unidentified Linguistic Object. 22
UML Unified Modelling Language. 12
URI Uniform Resource Identifier. 140, 152
UTF-8 Unicode Transformation Format 8. 122

VP Verb Phrase. 83

WOS Word of Speech. 47, 49–51, 58, 87

XML eXtensible Markup Language. 43

1. Introduction

Since the early works of Montague [119] in the 1970s, the development of natural language formal models has been highly influenced by mathematical logic and computational systems. Under the term 'Natural Language', modern science understands all languages as a primary means of communication. Systems dealing with textual documents in which words are associated with some meanings are called semantics-oriented natural language processing systems (NLPS). Formichov refers to this new field of studies as Mathematical Linguocybernetics [64]. Semantic-oriented NLPSs have become a main component in the systems dealing with artificial intelligence. Such systems still have some acute (scientific and technical) design problems that need to be addressed so a computer can 'understand' the meaning of textual data; one such problem is the extraction of information from unstructured textual data to build knowledge databases and integrate them in the Semantic Web environment.

The aim of this thesis is to propose a new, general model of natural language as well as a realized artifact that can detect and extract lexical relationships from textual data. The lexical relations include grammatical (morphosyntactic) and semantic relationships, that is the totality of relationships that are transmitted in a statement or sentence. It is possible to do this by using a deterministic system that does not need any large corpus to conclude about the content based on the word frequencies. The deterministic system considers every element of the system and its interconnection with other elements, and this is based on the classical linguistic laws of a natural language. The natural language system, in which the information is in three layers (morphological, syntactical and semantic), requires knowledge of the linguistic rules within each layer, as well as rules that act between the layers. These layers and laws must be projected in a mathematical model and then in a computational model to build a corresponding artifact for the user (graphical interface) and machine Application Programming Interface (API) modules.

The deterministic model of the natural language in its foundation must have a new structure of the online lexicon in which each lexical unit has a sufficient number of its grammatical and semantic characteristics so that the syntax of sentences can be uniquely described by syntactic patterns. Only a good connection between the lexical and semantic structure of the model allows the implementation of a new approach to information and knowledge extraction from the given corpus and any independent sentence or statement. Additionally, the system should be included into a global network of similar systems, utilizing their and its own interoperability within the Linked Open Data (LOD) cloud. This complex path represents a new and demanding scientific challenge.

1.1. Objective

Research goals should be both scientific and social, and the research should be useful for both the academic and broader community. Therefore, the aim of a computer artifact based on a new, deterministic language model is twofold: its usefulness in education (network lexicon/thesaurus, teaching modules via the SSF API's, grammar counselling, etc.) and its usability in future scientific research (proving authorship, plagiarism detection, machine translation, etc.) Data resources used in the artifact can be primary or secondary. Primary sources are used in the development of computer modules (Lexicon [128] with about 100,000 grammar units). Secondary data can be either internal (built by the user) or external (taken from other relevant institutions). The goal of the research is to build an artifact that will be able to fetch the data (documents) from internal/local and remote/network resources and will also be able to distinguish between private (owned by a private user) and public (accessible to all users) data. The artifact is, therefore, a multi-purpose system that can use dependent and independent sources. Retrieving secondary network sources such as WordNet or its partial Croatian version (CroWN), then the data from the network corpus of the Croatian DBpedia, WIKi.hr, the digitized newspaper Večernji or Jutarnji list etc., to extend the usability of the artifact to even more widespread humanistic areas. The terms and their definitions for semantic processing are harvested through a network of online encyclopaedias (LZMK), and the lexicon database is initially formed from the local repository 'Croatian Word' [128].

The researchers focused on quantitative language processing (word frequencies, splitting of texts into smaller entities, categorization of results, trends, etc.) to have the possibility of statistical processing due to the R language module that is embedded into the SSF network framework.

A special type of data is the one generated dynamically in RDF format, which makes a tree-like LOD structure to achieve interoperability with similar resources in the world and inclusion into a global data cloud.

The lexical relation is a culturally recognizable or linguistically defined pattern of correlation between lexical units (words) in some language. The lexical unit is a compound of a form and meaning in which the form represents a phonological (pronounced) set of sounds or a written lexical representation of a series of characters (symbols) whose meaning (to be properly understood) the listener or reader has already stored in his memory. The assignment of the form and content (real or abstract meaning) is acquired by the language learning process.

The goal of this thesis is to describe a general model of the natural language that is designed in such a way that its computer-based realization (artifact) finds and extracts

lexical relations. The lexical relations include morphosyntactic and semantic relationships, that is, the totality of relationships that are transmitted in a statement or sentence.

Unlike the stochastic processes that hold the elements of disorder/coincidence, and which are therefore statistically processed, the deterministic system treats processes as events with a connection of causes and consequences, which is conditioned by the necessity of knowing the internal laws that form the basis of the process. A deterministic model in which the words of the natural language have many or even all lexical features shall solve the problem of rough granulation of the information [143], but due to a large number of linguistic laws that must be taken into consideration, it represents a significantly greater challenge and complexity of the artifact [82].

Natural language is also a process, the process of forming and linking words, with the aim of transmitting the information and knowledge that they carry. Since the number of lexical units, and especially their combinations in sentences of a given language is vast, it is not surprising that it is commonly analysed using statistical methods. The actual forms and meanings of fundamental elements (words) are often overlooked, and only the number of their appearances (frequencies), (e.g., in the TF-IDF paradigms) [1] or are based on their interconnection inside documents (e.g., in LSA) [41]. These methods can be used for document or image classification in which numbers are assigned to the dots of the graphical objects with information about the positions and shades of colours, and then matrix decomposition (SVD) can be used to obtain results.

The deterministic system, however, considers every element of the system and its relation to other elements in order to determine the laws that exist between them. Development of such model is, of course, much more difficult, especially in the system of the natural language, in which the information is segmented into at least three levels (morphological, syntactic and semantic), which requires the study of linguistic laws within each layer and between each layer. Thus, for example, adding a morph (the smallest meaningful part of a word) or its change within a word can change the meaning of the word or its grammatical property (e.g., *lov+ac*, *lov+cu*) or its semantic property (e.g., *lov+čić* - diminutive and/or pejorative). An even more complicated situation appears in the formation (syntax) of sentences, where it is often not easy to determine the meaning of a sentence. For example, the popular example of a doubt: "*I see a man with binoculars in the garden*" is semantically unclear: Does the observer watch a man in the garden with his binoculars, or does he see a man who has binoculars and is in the garden?

A deterministic model of the natural language will introduce a new lexicon structure in which each lexical unit will have a sufficient number of grammatical and semantic tags so that the structure of sentences can be uniquely described by syntactic patterns that would lead to a more precise semantic analysis in the final step. Although directed to the

open information extraction (OIE), this artifact also represents a new approach to the network lexicography and online syntactic analysis [127], [125], the linkage of semantic repositories into a semantic network [123] and even a discovery of tropes (metonymy and metaphor) [124].

1.2. Hypotheses and research questions

Due to the large amount of the documents available online, information retrieval and text processing (such as text classification, semantic relations extraction, machine translation, etc.) are most often based on the statistical processing [179] of the vector space model [144] and its computer realization in the sense of the so-called *bag of words*. Bag of words is a model which represents collection of documents as a matrix where columns represent documents and rows represent terms. The problems (synonymy and polysemy) occur in the first step of document processing:

- Multiple words have the same or similar meaning (but are differently written) (*synonyms*);
- Words that have the same written form, but do not always have the same meaning (*homonymy*);
- Position of the words in the sentence can affect its meaning.

In the next step problem occur due to the morphology of the language. Namely, for different grammatical categories (type of word, gender, number, case, time, etc.) the meaning or concept is dispersed into multiple morphological variants (e.g. the machine cannot know that the word cro. '*misao*' and cro. '*mišlju*' are the same). For this reason, many morphological normalization methods are being developed for conflation or clustering terms into a unique form. The goal of this approach is to present all grammatical forms of words and their derivatives of the same meaning with a unique number in the set of N and to reduce the dimension of the word vector in the dictionary [41].

Morphological complexity is associated with affixing (parts of words or morphs that precede, i.e. prefixes or that extends words, i.e. suffixes). The morphological change of a word that does not change its meaning, but only its form in terms of declination (nouns, adjectives) or conjugation (verbs) is called flexion. Flexion is mostly caused by the change of suffixes. Changing prefixes often leads to a creation of words with a new meaning. This change is called derivation. How to encompass it within an algorithm and solve with a computer artifact?

There are two main approaches for normalization. One is to obtain the root of the word that is common to all forms, and the second is to obtain the linguistically correct lemma.

The first procedure is known as *stemming*, and the second is *lemmatization*. A large number of algorithms (e.g. Lovins, Porters, Paice-Husks, KSTEM ...) have been developed for stemming and lemmatization, along with the methods of supervised / unsupervised machine learning for automatic induction of the lemmatization rules. One such tool was made in Slovenia for the lemmatization of the Slovenian language [118]. Its accuracy is about 60% to 75% if sequential modelling or other paradigms, (e.g. ripple-down type rules or inductive logic programming) are used.

The question is whether it is possible to treat these (fundamental) problems in a different way? Is it possible to replace a stochastic model (which deals with statistical methods or machine learning) with a deterministic model which would incorporate all language laws prescribed by its standardization? It is obvious that extracting of semantic relations as a first phase of semantic language processing will be far more accurate if the model incorporates all linguistic aspects and eliminates any machine based normalization. In that case, a number of ambiguity words will significantly decrease, but the amount of work necessary to project and implement such model with a suitable artifact will increase.

This research is motivated by practical problems of the natural language processing whose ultimate goal is to create a general model of the language together with its associated innovative artifact which covers the core areas of linguistics and the functions for extracting lexical relations. Literature review and the efforts of many scientists to offer different approaches to address this problem [4], [33], [32] indicate the value of the goal, which, following the trends of the research community [18], [54], [56] can be formally expressed in a following way:

Design a general syntactic and semantic model of the language and its associated innovative artifact as a network framework which will integrate morphological, syntactic and semantic features of natural language and develop functions for extracting lexical relationships from digital documents (corpus).

The hypotheses of this thesis are as follows:

H1: The developed network framework of the natural language represents a new, deterministic language model which generalizes and extends the well known language models: Meaning Text Theory (MTT) [117], Generative Lexicon (GL) [136], Lexical Functional Grammar (LFG) [87] and Dependency Grammar (DG) [159].

H2: A deterministic linguistic model is applicable for implementation of the network framework for syntactic and semantic analysis of the natural language and the extraction of lexical relations from the corpora.

H3: The network framework provides network interoperability with a linguistic semantic cloud.

The following questions will be answered:

Q1: What kind of morphological structure and data types the network framework must be able to implement, so that the most popular annotation models (MULTEXT-East, Penn Treebank POS tags, SWETWOL, UD POS tags etc.) that are used today can be implemented?

Q2: Is the development of a network framework that integrates morphological and syntactic, and then also semantic features feasible (because similar solutions are not available worldwide)?

Q3: How to integrate network lexicon/thesaurus of such framework with an existing online knowledge (lexicographic, encyclopedic, linguistic)?

Q4: Is it possible to conduct a semantic analysis of the sentence beside the morpho-syntactic (e.g. detection of a sentiment) ?

Q5: How to dynamically convert relational data into LOD triples and publish them in the semantic cloud?

1.3. Related works

Traditional approaches to information extraction (e.g. WHISK [150] or Snowball [2]) relied on handwritten rules of a small set of predefined, specific relationships and large sets of hand-marked patterns for identification and retrieval purposes. Significant enhancements of the extractors, primarily in the addition of relational n-tuples extracted from the text and their independence from the predefined dictionary, appeared with software tools called KnowItAll [52], [53], [54] and TextRunner [10]. New, supervisory learning methods automatically generate new rules, which are more complex than those originally written and which are then used in sample training methods. Such learned samples are finally used in Naïve Bayes [107] type classificators, which are based on the POS-part of speech dependencies and neighboring words. In addition to TextRunner, two significant Open Information Extraction computer systems [32] are being developed: ReVerb [56] and WOE [175], which include heuristic methods of retrieving information from Wikipedia's info frames that are structurally well derived. These were used by DBpedia system developers several years later, for the purpose of linking information to all pages of the same categories from Wikipedia in different languages using LOD paradigm with RDF schema [89]. The weakness of these tools was the narrowing of information extraction provided only by the verb, and completely ignoring the context in which the information existed. This required a significant improvement of the concept and its implementation, resulting in

the emergence of a new system called OLLIE [115], in which the extraction was extended through names and adjectives, and the context was taken into account as well. In addition to this, the program is refactored and its speed boosted at least twice, as well as precision and quality of extracted information.

Since then the development of the Open IE systems has rendered four main development directions:

1. Usage of word tags (POS tags);
2. Relations in a dependency trees (DT);
3. Predicate-argument structure (PAS);
4. Mapping the relations to ontological structures [29, 177, 99, 91, 178].

Systems like TextRunner and StatSnowball [179] are based on the first, and systems like ReVerb, WOE, OLLIE and PROPMINER [4] on the second development direction. Christensen et al. [33], [32] in the task of extracting also include PAS predicate (the so called Chomsky's) language structure in which they examine semantically tagged arguments. In the similar way, the extractor SRL-IE [134] in which verbs are often matched with relational phrases of PAS works. Authors Soderland and Gilmer et al. [149] suggest a new approach based on the set of mapping rules OIE towards target ontology. For that purpose they defined a relative simple new programming language for defining mapping rules, extraction of relational triples and storing them into the database (KBP - Knowledge Base Population). According to Suredanu [157] for every KPB relation, there is a set of rules which define the final relation, while respecting lexical and semantic limitations. That rule language relies on semantic limitations of the observed language, in order to gain a higher precision of information extraction.

The proposed deterministic linguistic model and the associated instance of the artifact, partly uses all four directions of development, however, in a new way:

1. Instead of using simple POS tags, the new model introduces complex tree like structures (WOS/SOW) for morphosyntactic labels (professionally defined, but user-changeable), and in that way maps the complete ontological structure through numerous markers. Thus they bring about better morphologically syntactic and semantic properties with full respect for the linguistic laws of all linguistic levels, implicitly linking and extending the 1st and 4th development direction.
2. Syntactic patterns, marked in a new way, and associated with relevant information from network sources (encyclopaedias and repositories) combine the PAS and DT paradigms, thus implicitly linking and extending the 2nd and 3rd directions of the OIE systems development.

3. The model also provides parallel use of other tools - programs written in different programming languages (Python, Perl), statistical (R) or LOD-oriented (SPARQL), with the possibility to use their already written and published modules and programs (NLTK, Scikit-learn etc.). Linguistic knowledge generated from OIE systems is usually stored in relational databases which are suitable for fast information fetching. In order to ensure the reuse of generated knowledge in terms of semantic web, different approaches of mapping relational databases to the RDF format was discussed [5].

Regarding network artifacts in this field, there were many of them:

1. From the early 1980s until today, despite many lexical frameworks (e.g. Acquilex, Multilex, Eagles, Mile and others), only the most famous survived - WordNet, whose development is managed by the Global WordNet consortium (involving 60 countries). Croatian WordNet project under the acronym CroWN [138] [151] was launched in Croatia back in 2007, but unfortunately it has not been included in European dictionaries (EuroNet, BalkaNet), yet. Parallel to WordNet, network frames LMF (Lexical Markup Framework) and Lemon (Lexicon Model for Ontologies) are being developed.

2. Fillmore's framework grammar (from 1997) has developed a huge dependency lexical base, known as FrameNet [71]. In Croatia there were (unsuccessful) attempts to create dependence trees, as well as the attempt to build the Croatian FrameNet.

3. Extracting semantic categories is the most difficult problem in the Natural Language Processing, because semantics depends both on morphology and syntax at the same time. Words can have multiple meanings, depending on the context in which they occur. That is why the connection between the words is the central topic of linguistic research [38], [117]. In Croatia, the Croatian verb valence lexicon called CroVallex [131] was conducted in 2008. Apart from its online appearance, it has not been developed any further.

Although focused on information extracting, this artifact also represents a new approach to network lexicography and on-line syntactic analysis, linking semantic repository to semantic network, and even discovery of some types of tropes (metonymy and metaphor). The concept of the generative lexicon proposed by J. Pustejovsky [135] and the *Qualia* type of a dictionary word which, besides the grammatical characteristics, also relies on the verbs, in which the lemma can be used, is the basis for such detection. The good part of this approach is that such dictionary would be independent of the corpora (even in the natural language) and that it could be permanently updated and renewed with new words in accordance with the development of a language and Internet [65].

1.4. Research methodology

As a primary research approach to this type of research (innovative artifacts - the model and implementation of the network framework), design science methodology is chosen, which is a very often accepted research paradigm in engineering disciplines [130], especially in the area of information systems. Design science is a pragmatic approach in which actual problems (according to Hevner [80]) are solved by the development of innovative artifacts (constructs, models, methods and instances). The research is carried out systematically and evaluates the created artifact properly, which finally results in generation of the new knowledge created by making and using the artifact. Today there is a large number of methodological frameworks, guidelines and design patterns for the implementation of design science. The implementation of this research will be guided by the methodological framework for the design science proposed by Johannesson and Perjons [84], through the following five activities:

1. Explicate problem
2. Define requirements
3. Design and develop a model and artifact
4. Demonstrate the artifact
5. Evaluate the artifact

The first two activities (definition of the problem and requirements) are described in Chapter 1. The Model and the associated artifact are described in Chapter 2. The artifact demonstration and evaluation was conducted at two levels (by linguistic experts [51] and by computer scientists at the Department of Mathematics of the J. J. Strossmayer University of Osijek). However, as the research structure is regularly iterative and recursive, it should have in mind that the methodology of the previous research, and the following, in the written phases does not necessarily perform sequentially, but often returns to the previous stage and iteratively repeats it. The objectives of the defined evaluation strategy are in line with the objectives specified by the FEDS framework [170], which means that scientific rigor must be ensured. In the context of this research, that means that the use of a network linguistic framework (artifact) should show an improvement over existing solutions and similar scenarios. In addition to this, research has no particular ethical implications and can be feasible with respect to resources available to the researcher (time, money, participants, etc.).

The research plan will be carried out in several stages:

1. Adaptation of the 'Croatian word' [128] dictionary in a new, grammatical and semantic structure.

2. Building of the basic model that will cover both public and multiuser work (with administrator, workgroups and different types of permissions). The model will contain necessary data tables for corpora, lexicon and syntactic and semantic online processing in both modes (as a public user or exploratory).

3. Over the base model, software modules for the formation of dictionary described structure with grammatical and semantic features will be implemented, which will be able to associate the words from the dictionary in an automatic or manual mode. This will enable the creation of the first Croatian Explanatory Combinatorial Dictionary (ECD) [117] - a combinatorial vocabulary associated with network knowledge and semantic information [145].

4. In the syntax analysis, it is possible to extract samples from the corpora documents by any grammatical or semantic properties, or their combinations, which will allow the creation of syntactic patterns.

5. Document searching will be possible through word and their properties (with extension by using regular expressions), and through stored samples which were previously obtained in syntax analysis.

6. The central place of the Syntactic and Semantic Framework will be a new kind of thesaurus, which will, besides the linguistic information, collect and provide the encyclopaedic knowledge of the chosen terms automatically retrieved from the network.

7. All linguistic data, encyclopaedic knowledge and linguistic research results, will be stored in one of the LOD (Linked Open Data) formats to easily link them with other LOD repositories on the Internet [29].

8. It will be possible to process the research results (e.g. valence samples) by statistical methods.

The network framework itself will be built on the Linux platform in the PHP and Python programming languages, and for data storing relational database MySQL and the Virtuoso triple store will be used. On the client side, the user interface will be built with HTML5, CSS3, JavaScript technologies, and basic queries in SQL and SPARQL

languages. Some text processing algorithms will be realized using the NLTK (Natural language toolkit) tool in the Python programming language. Access to a computer base will be possible in two ways: through an Internet browser or through an API interface (as described in Section 7.5). The first approach offers a visually attractive representation of words, their syntactic and semantic characteristics, and the ability to define their own characteristics whereas the API approach offers the ability to link external sources or to download lexical structures in the form of JSON objects. The example of API usage is given in Section 7.5.

1.5. An outline of the thesis

The rest of the thesis is organized as follows. *Chapter 2 - Deterministic language model* introduces a new deterministic language model which is the core of the online Syntactic and Semantic Framework for extraction of lexical relations. *Chapter 3 - Tagging* presents different tagging systems, and describes a new hierarchical tree-like structures (T-structures) which are used in the SSF for a more detailed tagging of lexical units. *Chapter 4 - Lexicography* gives an overview of important lexicographic aspects, as well as three types of lexicons (LEX, MSY, MWE) which are used in the SSF. *Chapter 5 - Syntax* presents a part of the model which is important for dealing with syntactic components in the SSF (e.g. Natural Language Functions, regular expressions, different types of grammars, etc.) *Chapter 6 - Semantics* presents a part of the model which is important for dealing with semantic components in the SSF (e.g. innovative concepts for creation of semantic domains, usage of the SSF in the process of opinion mining, etc.). *Chapter 7 - Extraction of lexical relations* shows how lexical relations can be extracted from textual document using natural language deterministic model. It covers various aspects of collocations detection, recognition of metaphors and metonymies, usage of the SSF in third-party applications (over API interface), etc. *Chapter 8 - Semantic Web integration* presents different approaches on how relational data from the SSF can be transformed into the RDF form in order to be included into the global Linguistic Linked Data cloud. Finally, the *Chapter 9 - Conclusion* gives the overview of the thesis, and answers to all hypotheses and research questions.

2. Deterministic language model

A development of the computer artifact which will represent a language model implies these three key steps:

1. Defining of an abstract (mathematical) model of a language. Considering the existing related works and trends in the development of linguistics in the world, it is possible to focus either on statistical or deterministic models. Statistical models are more common and mainly involve corpus searching using statistical and machine learning methods (e.g. Naïve Bayes, PCA, etc.) and usually do not consider the morphosyntactic features of the language. In such models, the word is treated as a number that is then analyzed in correlation with other numbers. Deterministic models are more complex because they use the knowledge and rules of certain language. Apart from the general rules, they also define specific rules, which are unique for a given language. The abstract model of the deterministic language therefore works with multiple sets of data where some are related to the lexicons and thesauri, while others are related to the syntax and semantic rules of the processed language.

2. For a selected type of an abstract model, the computer model also needs to be developed. The core of the computer model usually disposes of the relational database with multiple interconnected entities (tables). The visual representation of such model is realized with Entity-Relation (ER) diagram, which can then be a foundation for database generation and the related program code. Interrelation of the database and other program modules, as well as the use case scenarios or activities in the system are represented in a UML notification.

3. From the UML model follows the development of a program artifact which in MVC paradigm, usually has three levels: the model, the viewer and the controller. The first level includes a well designed database and its tables, the second level is a modular user interface and the third level is program logic where based on the input data, the correct outputs are formed. The user interface can be graphical or programmatic, and the data may be stored in a special repository instead of relational database, such as a triplestore database, a network cloud, or a regular file.

Deterministic model is chosen as an abstract natural language within the scope of this thesis. Therefore, an ordered septuplet of four sets of data (letters/tags, subatomic linguistic parts, lexicon / natural language dictionaries and their multiword expressions), and three set of rules (phonetic-morphological, syntactic and semantic) are defined.

2.1. Abstract model

An abstract language L is a septuplet:

$$L \triangleq (\Lambda, \Psi, \Psi_r, \Upsilon, \Gamma_r, \Theta_r, \Xi)$$

in which the symbols denote the following:

Λ - An alphabet (ordered set of symbols and punctuation marks)
a<b<c<č<ć<d<đ<dž<e<f<g<h<i<j<k<lj<m<n<nj<o<p<r<s<š<t<u<v<z<ž<*interpunction*
interpunction = .<,<...

Ψ - A set of (sub)atom words obtained from all subgroups (power set) combination of symbols/alphabet letters Λ (e.g. the set for Croatian language, consists of ~7,500 syllables, ~10,000 morphs and their combinations (syllablemorphs)).

Ψ_r - A set of morphological rules (patterns) - the function for word creation from atoms. Containing 13 phonetic and morphological rules for two modes of action: flexibility and derivation

Υ - A non-final set of words formed of atoms from Ψ by applying rules from Ψ_r (i.e. a language dictionary which applies Ψ_r rules and has public usage (pragmatics)),

Γ_r - A set of syntactic rules (patterns) between words from Υ (i.e. grammatical rules for formation of multiword expression while respecting rules from Ψ_r).

Θ_r - A set of semantic rules (patterns) between words from Υ (i.e. semantic rules for formation of multiword expression while respecting rules from Γ_r)

Ξ - Multiword expressions which are formed by linking words from Υ and applying rules from Γ_r and Θ_r (i.e. meaningful and grammatically correct phrases and sentences through recursive structure).

For a computer to analyze the text it is necessary to assign a certain tag to each word. This is the task of linguistics. Although, the division of the word in different classes is not equal in every language. Since this thesis is focused on the Croatian language, it includes only the Croatian language examples and the division is made up of ten parts of speech (as shown in Table 1). However, it is possible to make the same sets for any other language in an equivalent way.

Table 1: Part of speech tags

Part of speech	Tag	Description
Noun	$\mathbb{N}$	Beings, things, abstract concept or behavior
Pronoun	$\mathbb{X}$	Presentation of nouns
Verb	$\mathbb{V}$	Action, state and occurrence
Adjective	$\mathbb{A}$	Noun's property
Number	$\mathbb{C}$	Quantity
Adverb	$\mathbb{D}$	Verb's property
Preposition	$\mathbb{P}$	Navigation
Conjunction	$\mathbb{J}$	Logic
Exclamation	$\mathbb{I}$	Emotion
Particle	$\mathbb{R}$	Appendix

Every word type has its one-letter tag and brief description. Apart from parts of speech classes each language also has its categories, which can then appear in one or more part of speech classes. The categories are related to the deep prototypes of human thought. They link spoken or written information (of language) to an abstract model of an object which the speaker/writer is trying to introduce to the listener/reader, to achieve a unique transfer of information. Table 2 shows categories with minimal descriptions of tags that they have:

Table 2: Some open word class categories

Category	Mark	Description
Gender	r	[m,f,n] - Masculine, Feminine, Neuter
Number	b	[s,p] - Singular, Plural
Case	p	[n,g,d,a,v,l,i] - Nominative, Genitive, Dative, Accusative, Vocative, Locative, Instrumental
Person	l	[1,2,3] - First, Second, Third
Stress	e	[y,n] - Stressed, Unstressed
Aspect	g	Perfective, Imperfective, Bi-aspectual
Transitivity	j	Transitive, Intransitive, Reflexive
Tense	v	Past [p,a,i], Present [p], Future [f, f2]

Some categories belong only to one part of speech (e.g. the aspect is a category which is related only to verbs), and other belong to many of them (e.g. gender, number or case for nouns, adjectives, some numbers and pronouns). The example of categories effects on a part of speech (without perfect accuracy):

$\mathbb{N}$ - r,b,p,a,t
$\mathbb{V}$ - v,n,g,j,l,r
$\mathbb{X}$ - r,b,p,l,n (not for all types)
$\mathbb{A}$ - d,s,r,b,p
$\mathbb{C}$ - r,b,p, (to 4)
$\mathbb{D}$ - s
$\mathbb{P}, \mathbb{J}, \mathbb{I}, \mathbb{R}$ - unchangeable

The specific features of the language are also manifested in the categories, which are not completely scattered. Thus, for some nouns, category number can have both values (singular and plural), while for some (pluralia tantum) it can have only the plural. All these specificities affect the complexity of the tagging structure which the computer must have, so that the computer model can implement the natural language model. The other kind of layering occurs at the part of speech level, because for some of them (e.g. pronouns) only one category is applicable, whereas for others (e.g. nouns), distinct categories may apply. Moreover, sometimes these are only some elements of a certain category (e.g. only one person, and not all). Categories, therefore, act on a part of speech as certain mathematical operators, giving subtypes of words with special names. In that way, among Croatian pronouns it can be distinguished as follows:

$\mathbb{X}_{os}$ - personal
$\mathbb{X}_{pv}$ - reflexive
$\mathbb{X}_{ps}$ - possessive
$\mathbb{X}_{pps}$ - return-possessive
$\mathbb{X}_{pk}$ - demonstrative
$\mathbb{X}_{uo}$ - interrogative and relative
$\mathbb{X}_{nd}$ - indefinite

The action of any particular category onto the word apart from the layering of the word type also affects the change of the word's form - derivation and flexion. The derivation (the most commonly by adding the prefix to the root of the word) forms the word of a new meaning, and with flexion (usually by changing the suffix of the word) the word contains the same meaning but a different form. Croatian is a flexion rich language, because it has eight phonological (Table 3) and three morphological (Table 4) changes which are related to the flexion process.

Table 3: Flexion phonology

	Sound change	Description
Phonology	Sound Assimilation	If a voiced and a devoiced consonant are in the immediate vicinity, they are equalized for easier pronunciation
	Articulation Assimilation	The obstruents *s, z, h* and the sonant *n* undergo changes of assimilation
	Consonant Elimination	If two consonants are found in the immediate vicinity, one of them is eliminated
	Insertion of the Sonant *j*	The sonant *j* is inserted between two vowels, at least one of which is *i* [except *io*]
	Insertion of the Vowel *a*	In many cases (see below)
	Replacement of Final *-l* with *-o*	Replacement of $l > o$ at the end of a syllable or a word
	Ablaut	Replacement of $o > e$ after palatal consonants and *c*, or after clusters *št, žd*
	Replacement of *yat*	Long *yat* syllable changes if it is shortened, long *ije*, short *je/e*

Vocal changes which occurs in flexion phonology and morphology are briefly described in Appendix C.

Table 4: Flexion morphology

	Sound change	Description
Morphology	Palatalization	Replacement of *k, g, h* and *c* preceding *i, e* with the palatals *č, ž, š*
	Sibilarization	Replacement of *k, g, h* preceding *i* with the sibilants *c, z, s*
	Iotation	Non-palatal consonants merge with *j* and change to the most similar palatals

A set of words Υ, implies all the words that were created using the phonological and morphological rules from a set of morphemes Ψ of a language. Usually, the lexicon of certain language consists only of words canonical forms - infinitives of verbs and nouns in singular nominative case. The SSF's lexicon stores all forms of words, not only the canonical form, even multiple words that are written in the same way but have different tags (meanings). Tags come from categories of specific word classes. These categories may be inherent, relational or an agreement [112]. Table 5 shows categories by word classes.

Table 5: Nouns and Verbs categories

Inherent	Relational	Agreement
	NOUNS	
Gender	Case	
Number		
Definiteness		
Size		
	VERBS	
Tense	Voice	
Aspect		
Mood		
Transitivity		

Each category consists of different elements, for example, the *case* in Croatian language has seven elements (in other languages that number may vary), the *number* has singular and plural (sometimes also dual, or in other languages even more). By the action of flexible (declination, conjugational) forms of some lexical morpheme in combination with grammatical morphs (e.g. by the change of case in nouns, adjectives or pronouns, time, personal changes of verbs, etc.), a paradigm is formed. The paradigm comes to the fore completely in multiword expressions, because they complement syntax and semantic rules. It would be a mistake to conclude that the case is a result of morphological rules, since it emerges in the interaction of syntax and semantic. That is the reason, why the paradigm of personal nouns in an abstract model of Croatian language (based on Table 2) can be formally noted as:

$$\mathbb{X}_{personal} = (\, l\,[1]\, b\, \&\, l\,[3]\, b\, r\,)\, p[-v]\, W \;\cup\; l\,[2]\, b\, W$$

which means that the first ($l[1]$) and (&) the third ($l[3]$) person of personal pronoun doesn't have word W in vocative ($p[-v]$), that the third person has all three genders (r), and that second person has number, but not the gender (like vocative). The union ($\cup$) connects these elements into the complete paradigm.

Table 6: Accentual declination of the reflexive pronoun 'ja'

			Singular		
N	jâ	tî	ôn	òno	òna
G	mȅne, me	tȅbe, te	njȅga, ga		njê, je
D	mȅni, mi	tȅbi, ti	njȅmu, mu		njôj, joj
A	mȅne, me	tȅbe, te	njȅga, ga, nj		njû, ju, je
V	-	= N	-	-	-
L	ò meni	ò tebi	ò njemu		ò njoj
I	mnôm, mnóme	tȍbom	njîm, njíme		njôm, njóme
			Plural		
N	mî	vî	òni	òne	òna
G	nâs, nas	vâs, vas		njîh, ih	
D	nȁma, nam	vȁma, vam		njȉma, im	
A	nâs, nas	vâs, vas		njîh, ih	
V	-	vi		-	
L	ò nama	ò vama		ò njima	
I	nȁma	vȁma		njȉma	

Categories for word compression or accentuation (*mene/me, njega/ga/nj*) - mark 'e', are included in the formalism for machine processing but due to the representation simplicity are left out. Similarly, the paradigm for reflexive pronoun cro. '*sebe*' can be written in the following way:

$$\mathbb{X}_{reflexive} = e\, p[g, d, a]\, W \cup p[l, i]\, W$$

Which refers to three compressed and two uncompressed forms, along with other two (nominative and vocative) that aren't in the paradigm at all. Formalized classification of all other open class words by their categories in the model is done in the similar way. In that way supporting information becomes a feature, and the word itself (with its sequence of characters) only one of the features. This implies many consequences in terms of closeness of words which now can be measured based on the number of tags they have. It is important to note that all tags are organized in hierarchical structures (as discussed in Chapter 3), which means that similarities and differences can be observed in scope of ontological frames, and not only as numerical values. Once the words arranged in a

computer lexicon got their tags (part of speech, categories, etc.), for a deterministic model of natural language to be able to execute a syntactic analysis of sentences, the syntactic patterns must be created. One of common sentence tagging systems is by phrases. Table 7 shows some basic syntactic phrases. The algorithm for detection of syntactic patterns and their storage into the patterns database of Croatian language is more briefly described in Section 5.3.

Table 7: An overview of syntactic phrases

Type	Mark	Description
Noun phrase	NP	Composition of a noun and an article or an adjective /phrase/
Verb phrase	VP	Composition of a verb and a noun (f.) or an adverb (f.)
Adjective phrase	AP	Adjective with a graded word or a noun
Adverb phrase	DP	Adverb with an article or a verb (f.)
Preposition phrase	PP	Preposition with an article or a noun (f.)
Function complement	CP	Composition of an AP, NP or DP and J or O and the predicate (VP)

In the end, for a model to be complete, it is not enough to detect syntactically correct sentences by the computer, but also to check whether they are semantically correct. In that segment it is important to define and detect the meaning of the syntactically correct sentence by computer model (as well as program artifact). It is common to solve this through the well-known thematic roles, which are listed in Table 8.

Table 8: Some commonly-used thematic roles with their definitions, adapted from [86]

Thematic role	Definition
Agent	The volitional causer of an event Example: *The waiter* spilled the soup.
Experiencer	Event experiencer Example: *John* has a headache.
Force	The non-volitional causer of the event Example: *The wind* blows debris from the mall into our yards.
Theme	The participant most directly affected by an event Example: Only after Benjamin Franklin broke *the ice*...
Result	The end product of an event Example: The French government has built a *regulation-size baseball diamond*...
Content	The proposition or content of a propositional event Example: Mona asked *"Did you met Mary Ann at a supermarket?"*
Instrument	An instrument used in an event Example: He turned to poaching catfish, stunning them with *a shocking device*
Beneficiary	The beneficiary of an event Example: Whenever Ann Callahan makes hotel reservations *for her boss*...
Source	The origin of the object of a transfer event Example: I flew in *from Boston*
Goal	The destination of an object of a transfer event Example: I drove *to Portland*

2.2. Conceptual model

The artifact of language model is realized as a modular network framework. The backend is implemented mainly in the PHP programming language (the parser component is made in Python), while the database management system that drives the SSF is the relational database MariaDB[1]. The frontend of the SSF (Figure 2) is developed using the Bootstrap[2] framework combined with the jQuery[3] JavaScript library and served over the

[1] `https://mariadb.org/`
[2] `https://getbootstrap.com/`
[3] `https://jquery.com/`

nginx[4] web server. Figure 1 shows the conceptual model of the program artifact, along with main components and their mutual relations.

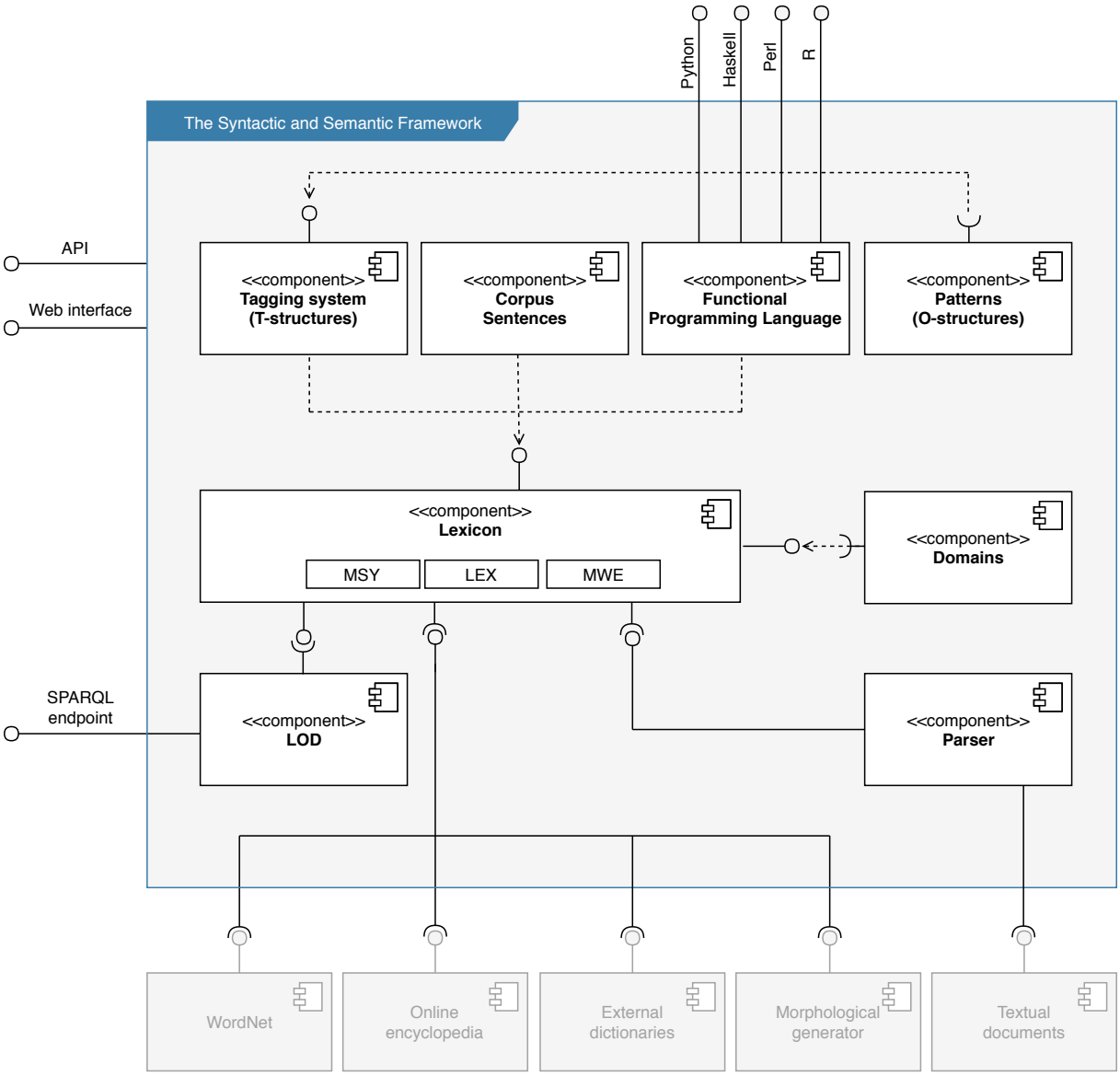


Figure 1: Conceptual Model of the Syntactic and Semantic Framework

The *Lexicon* is the crucial point of the SSF. Almost every process in the SSF uses it in some way. It consists of three main sub-lexicons: *MSY (morphosyllables)*, *LEX (lexicon of single words)* and *MWE (multiword expressions lexicon)*. Each of them plays a significant role in the process of data analysis and lexical relation extraction. The lexicon is initially loaded with data from external resources such as Princestones WordNet database, various online encyclopedias, and dictionaries, in combination with the morphological generator (described in Section 4.1).

[4] `https://nginx.org/en/`

The *Tagging subsystem* directly contributes to the quality of the lexicon by providing the set of syntactical and semantic features (WOS/SOW) which are the source of the knowledge about words, their parts, and multiword expressions. In addition to the system-wide WOS/SOW marks that are initially populated, every user can create their own tags and can use these tags later in the process of data analysis. The tagging subsystem is described further in Chapter 3.

Special types of tagging structures, called *O-structures*, are developed on the top of T-structures and are used for marking parts of words in sentences and their correlations. These structures are stored in a database in the form of unique sentence patterns. O-structures are a core mechanism for extraction of lexical relations and are used at both syntactic (Section 5.6) and semantic level (Section 7.4).

Another vital component of the SSF is the *Corpus*. Every registered user can upload their own textual documents into the SSF. Each uploaded file is then divided into sentences, and every sentence is divided into words, which are then linked to the lexicon. Since every word in the parsed sentence must be contained in the lexicon and has a unique identification number associated with it, the sentence is stored in the form of a linked list. If the processed word is not already a part of the lexicon, it will be automatically inserted and marked as an *Unidentified Linguistic Object (ULO)*. The administrator of the system can later associate appropriate tags with such words. The whole processing of textual files is done by the *Parser* component. Standard language functions can be used through a modern web interface, but SSF enables the usage of advanced language functions within the *Functional Programming Language (FPL)* component. In this way an advanced user can develop their own functions to process previously loaded textual data and synergistically use the strength of the well-known programming languages (e.g., Python, Haskell, Perl, or R) in combination with other specific SSF natural language functions (NLF). An overview of NLFs that can be used within the FPL appears in Section 5.4. The *Domains* component is responsible for handling and maintaining information about static or dynamic language domains. Static domains are domains made of a finite number of words, whereas dynamic (virtual) domains are defined by a set of rules which, when applied, generate a group of elements. Finally, to ensure interoperability of the SSF and its inclusion into the global Linguistic Linked Open Data cloud, the *LOD* component transforms the linguistic data from the relational database to RDF triples and serves that through the SPARQL endpoint in real time. There are two main approaches to transforming the relational data to the RDF form, one is by direct mapping described in Section 8.1, and the other is by periodical synchronization of relational database and a triplestore as described in Section 8.2.

The SSF can be used either through a modern web interface or over the Application Programming Interface (API). The first scenario is commonly used for exploring the system

and testing its functionalities, while the API interface is used when powerful SSF functions need to be integrated into other external systems. An example of API integration is described in Section 7.5.

2.3. Model realization

Network framework has general and specific functions. General functions are publicly available whereas specific ones require login with a username and a password. Below the login form, the user can select either Croatian or English language. A selected language doesn't affect only user interface, but also the whole lexicon along with structures and features that the selected language contains. Currently, the framework is developed only for the Croatian language, but due to the well structured database backend it is possibly to easily include other languages as well. The framework is publicly available at the `http://www.ss-framework.com` and looks as shown in Figure 2.

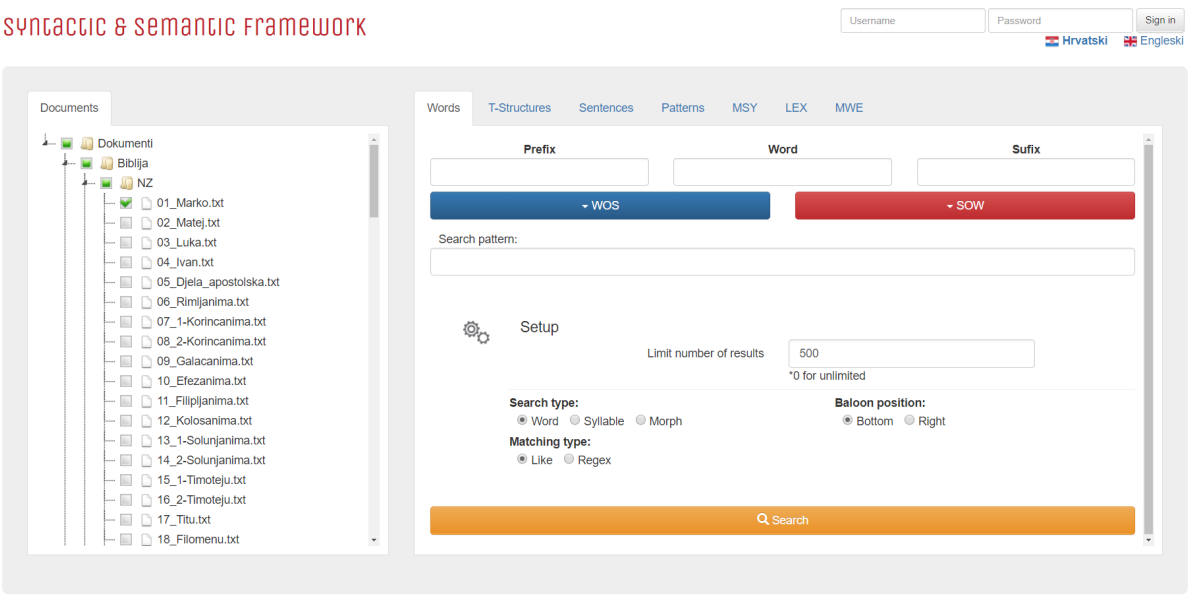


Figure 2: The SSF's main screen

Registered users can upload or create their own documents, structures and features, which other users cannot see or use. After the document has been uploaded the automatic processing of the document is started (which can take a while according to the document size and new words that needs to be added to the lexicon). Once the document is processed it will be visible in a corpora tree of the registered user that uploaded it.

The SSF has four main modules:

1. Words search engine
2. Sentences search engine
3. Word of Speech (WOS) and Semantic of Word (SOW) trees (T-structures)
4. Lexicons (MSY, LEX, MWE)

Registered users have three additional modules:

1. Domains
2. Natural Language Functions (NLF)
3. O-structures



Figure 3: Homonym 'usta' as a noun and a verb

Figure 3 shows the difference between two identically written words but tagged with different WOS/SOW marks. Beside lemma, syllables/morphs and WOS/SOW marks lexicon also has word weights (frequencies) which are used in organization of syntactic patterns. The model's special strength is its interoperability with other network resources and repositories (HJP[5], LZMK[6], BebelNet[7], etc.) Descriptions and definitions from these

[5] http://hjp.znanje.hr
[6] http://www.enciklopedija.hr
[7] http://babelnet.org/

resources are interconnected (as shown in Figure 4). In that way, new semantic networks are built, providing better quality of information of lexical relations between words. Therefore, it was possible to develop functions for extraction of lexical relations, including even those for detection of tropes [124]



Figure 4: Network lexicon - a source of a lexical information

Figure 4 shows an example of the word which is tagged with SOW tag that uses the data from external online encyclopedias. Every word from the definition which exists in the SSF main lexicon is automatically linked to it (and displayed in blue color). In this way the user of the SSF can endlessly crawl through the lexicon. Below every external resource the link to original resource is displayed.

2.4. Statistical methods

The SSF is based on a deterministic model but does not circumvent the statistical side of natural language processing. In this field, especially in the last decade, when the digital processing of documents from the classical corpus was replaced by an enormous number of documents on the Internet (big data), numerous packages were made in different programming languages that besides classical also allow highly advanced natural language processing methods. This applies primarily to packages in programming language R, but also more and more to Python language. The SSF with its NLF module for online language programming enables execution of all these packages within the framework, and

also offers many of its own functions that link deterministically-formatted information with statistical processing, which in the end results with significant improvement of the results. Among many SSF statistical functions that perform statistical processing only one (`univar()`) will be shown briefly (with list of arguments, input and output), whereas advanced functions (e.g. from cluster analysis) will be explained in more details. `univar()` function is developed for demonstration how custom R functions within the SSF can be developed. As an input value, the function accepts a data set, and in the output displays some basic info about the data in textual and graphical form.

univar R function

Function definition:

```
string univar (data, hist=TRUE, plot=TRUE, ggplot=FALSE)
```

Parameters:

```
string data[] - dataframe or vector of numerical values
bool hist, plot and ggplot - graphical representation options
```

Example:

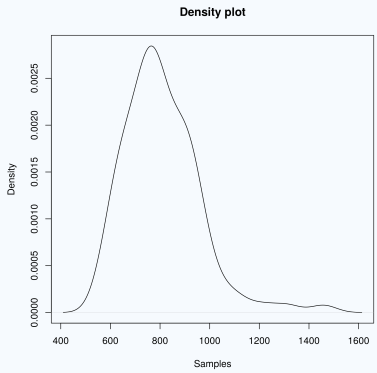
```
univar(ldt)
```

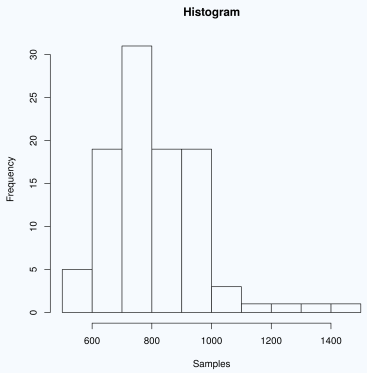
Output:

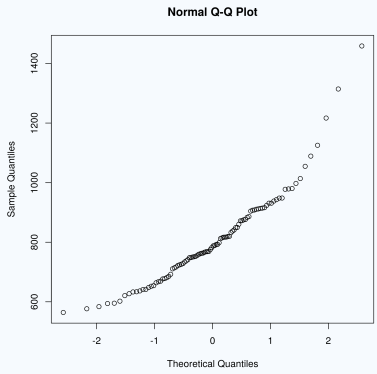
```
DATA SUMMARY
-----
Length Freq Mean_RT
Min. : 3.00 Min. : 0.0 Min. : 564.2
1st Qu.: 6.00 1st Qu.: 53.5 1st Qu.: 713.1
Median : 8.00 Median : 310.5 Median : 784.9
Mean : 8.23 Mean : 3350.3 Mean : 808.3
3rd Qu.:10.00 3rd Qu.: 2103.2 3rd Qu.: 905.2
Max. :15.00 Max. :75075.0 Max. :1458.8

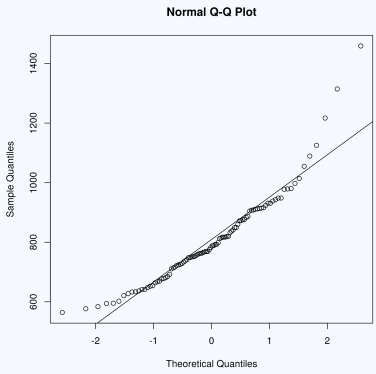
DATA SAMPLE
-----
Length Freq Mean_RT
marveled 8 131 819.19
persuaders 10 82 977.63
midmost 7 0 908.22
crutch 6 592 766.30
resuspension 12 2 1125.42
efflorescent 12 9 948.33
```

Graphical output (plot):

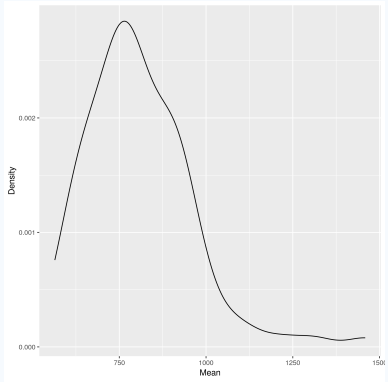


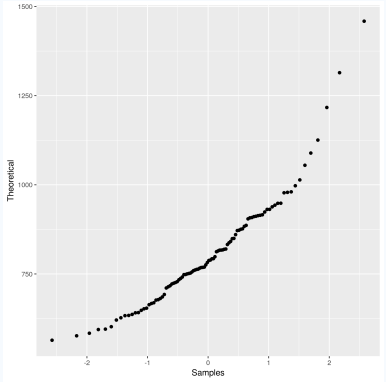


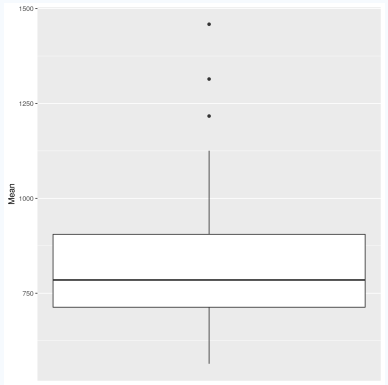


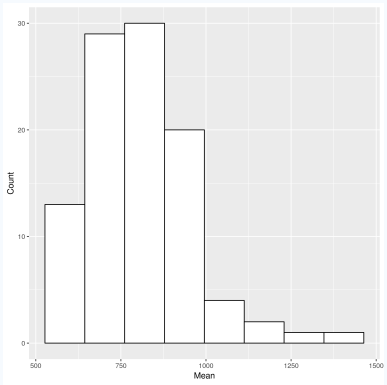


Graphical output (ggplot):









Any SSF specific R function can be made within the NLF module, and later used in statistic analysis of linguistic data, due to its direct connectivity to the relational database which lays in the backend of the SSF directly from the R environment. The connectivity between the NLF R module and the SSF's relational database is established using `RMySQL` package. In that way R can access all lexical data (words, multiwords, syllables and morphs, WOS/SOW tags, corpora, etc.) and use it in statistical processing. For example the function `WordsBySyllables()` retrieves the number of syllables in each word, and returns them as the R data set.

```
syllables <- WordsBySyllables()
print(syllables,row.names = FALSE)
```

will give the following output:

```
Syllables Num
1 3396
2 74543
3 237465
4 269522
5 148816
6 57784
7 14957
8 2952
9 458
10 62
11 5
```

The function `WordsBySyllables()` in R looks like:

```
WordsBySyllables <- function() {
   library(RMySQL)
   mydb = dbConnect(MySQL())
   rs = dbSendQuery(mydb, "SELECT Syllables, COUNT(Syllables) AS Num FROM
    (SELECT COUNT(syllableid) AS Syllables FROM word_has_syllables GROUP BY
    wordid) a GROUP BY a.Syllables")
   data <- dbFetch(rs)
   return(data)
}
```

Such data set is pulled from the live lexicon database and represents actual state of the lexicon within the SSF. The complete list of WOS/SOW marks is given in the Appendix E

and ER model segments are given throughout the thesis. Each change in the SSF's lexicon (insertion of new words or different WOS/SOW assignments) will immediately result in different results. Using other R functions this data can be further processed or visualized, for example:

```
library(ggplot2)
library(scales)
syllables <- WordsBySyllables()
positions <- c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11")
ggplot(data=syllables, aes(x=Syllables, y=Num)) +
    geom_bar(stat="identity", fill = "white", color="black") +
    scale_x_discrete(limits = positions) + scale_y_log10() +
    scale_y_continuous(labels = comma) + labs(x = "Number of syllables") +
    labs(y = "Number of words")
```

will produce the bar plot as shown in Figure 5:

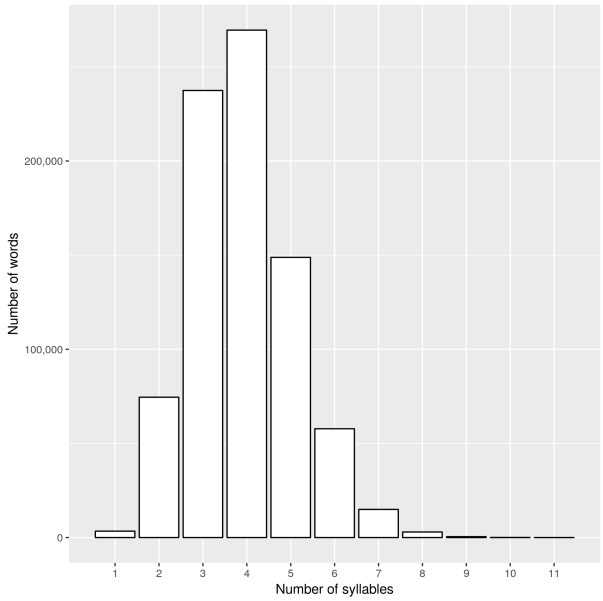


Figure 5: Visualized results from the SSF function in the R

Although, the focus of this thesis is not on statistic processing, some of commonly used analysis will be shown in this chapter in order to demonstrate how even statistically based linguistic researches can be conducted using the SSF. Since there is still no systematic data processing for Croatian language (e.g. computational corpus linguistic), these function can be demonstrated on simulated data or on the data provided by foreign authors for other languages. The SSF can thus become a sort of instigator of computer corpus linguistics in Croatia. Among these functions, multivariate analysis is commonly used. Multivariate

analysis covers different techniques for description, simplification and analysis of data which consists of several variables which are measured on the same sampling units. There are two main approaches (supervised and unsupervised) which are commonly used for description of data sets in multivariate analysis. Supervised methods are those in which the groups are known in advance from the data analysis whereas in unsupervised methods these groups are not known. Supervised methods are usually referred to as methods for classification. In the classification process, one of main problems which arises is how to organize the observed data into logical sets with a goal of building taxonomies. One of the commonly used methods is cluster analysis which can be used to discover a structure in unstructured data. One important concept in terms of cluster analysis of textual data are registers. Registers are language varieties which are specific for some language form (e.g. emails, SMS messages, textbooks, face-to-face conversation, etc.). These register can further be distinguished by various criteria such as form (spoken or written), the purpose of communication (education, entertainment, etc.). Each register has some of its own specifics, for example, face-to-face conversation is known to have more first and second person pronouns, and less nouns and adjectives [13]. This type of analysis is usually done by Factor Analysis (FA) and Principal Components Analysis (PCA). If the variables are strongly correlated and the number of variables is large these two methods usually give similar results [60]. Since there is no data for the Croatian language, for demonstration purposes, the data from British National Corpus (BNC) will be used. The data set `reg_bnc` is loaded from the Rling package[8] [98].

```
library(Rling); data(reg_bnc); str(reg_bnc)
```

```
'data.frame':  69 obs. of 12 variables:
$ Reg : Factor w/ 6 levels "Acad","Fiction",..: 6 6 6 6 6 6 6 6 6 6 ...
$ Ncomm : num 0.17 0.205 0.206 0.136 0.133 ...
$ Nprop : num 0.02697 0.02498 0.0468 0.0112 0.00985 ...
$ Vpres : num 0.0355 0.0391 0.0366 0.0485 0.0452 ...
$ Vpast : num 0.0219 0.0298 0.0236 0.0189 0.0198 ...
$ P1 : num 0.0347 0.0208 0.018 0.0276 0.0455 ...
$ P2 : num 0.01832 0.01137 0.00775 0.03749 0.03703 ...
$ Adj : num 0.0536 0.0585 0.0596 0.0407 0.0446 ...
$ ConjCoord: num 0.0395 0.034 0.0335 0.0339 0.0384 ...
$ ConjSub : num 0.031 0.0276 0.0232 0.0315 0.0283 ...
$ Interject: num 0.00997 0.00414 0.00226 0.02173 0.04298 ...
$ Num : num 0.0206 0.0192 0.0277 0.0414 0.0164 ...
```

[8] https://benjamins.com/sites/z.195/content/package.html

The listing shows a sample of the `reg_bnc` dataset which will be used in the following examples. The variable `Ncomm` represents a numeric vector with relative frequencies of common nouns, `Nprop` - of proper nouns, `Vpres` - of verbs in the Present Tense form, 3rd person singular, `Vpast` - of verbs in the Past Tense form, variables `P1` and `P2` represent a numeric vector with relative frequencies of the first-person and second-person pronouns, the variable `Ajd` - of adjectives, the variable `ConjCoord` - of coordinating conjunctions, the variable `ConjSub` - of subordinating conjunctions, the variable `Interject` - of interjections and the variable `Num` - of numerals. Before the PCA or FA are performed it is useful to check if the data is appropriate for such analysis. The variables should be intercorrelated and the correlation shouldn't be too high (since high correlation may be a problem for FA [60]).

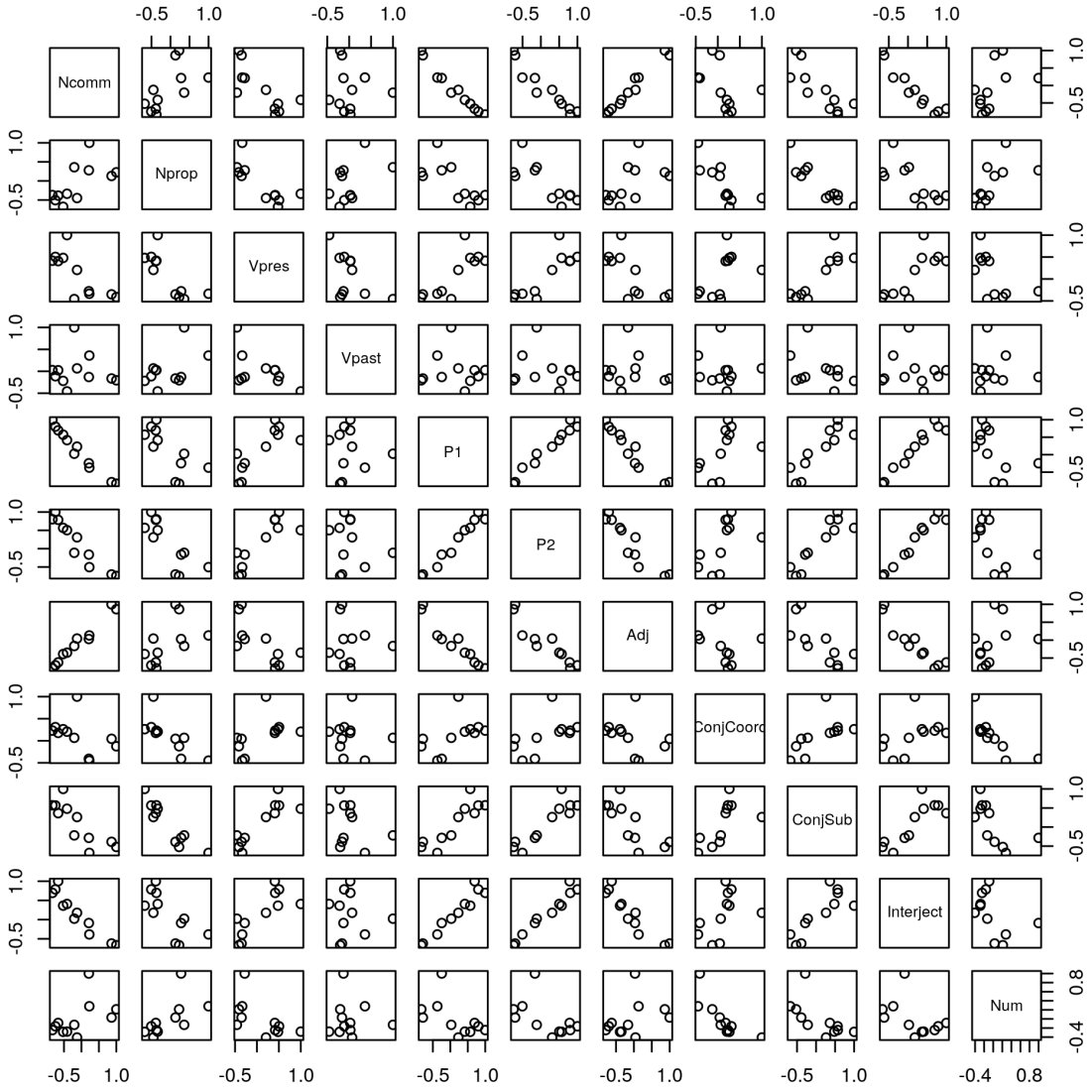


Figure 6: Correlations of variables for PCA

Figure 6 shows correlations between variables (i.e. for each two variables small correlogram is drawn). When the variable on Y axis tends to increase together with variable on X axis there is a positive correlation between them, and when the variable on Y axis tends to decrease as the variable on X axis increases, there is a negative correlation between them (e.g. it is visible that variables `P1` and `P2` are strongly correlated whereas variables `P1` and `Adj` have negative correlation. The PCA analysis in the R can be performed using `FactoMineR` package:

```
reg.pca <- PCA(reg_bnc, quali.sup = 1, graph = FALSE)
```

This function accepts three arguments, first the data set, second argument is `quali.sup = 1` which tells the R that the first variable `Reg` should be treated as a qualitative supplementary variable. Unlike active elements, supplementary elements are not taken into account in the PCA calculations. The third parameter `graph = FALSE` tells R to suppress generation of graphical output for the calculated PCA. In order to find out how many dimension are needed for register variation the eigenvalues are calculated. The output of `reg.pca$eig` shows how much of total variance is explained by each component. Higher eigen value means the higher correlation between components. Total sum of explained variance is always 100%. Figure 7 shows component contributions of eigenvalues for BNC data set. There are 11 components (because there are 11 rows in `reg.pca$eig`). Different statisticians define different rules when it comes to which of these components should be taken into account, but usually eigenvalues greater than 0.7 are considered.

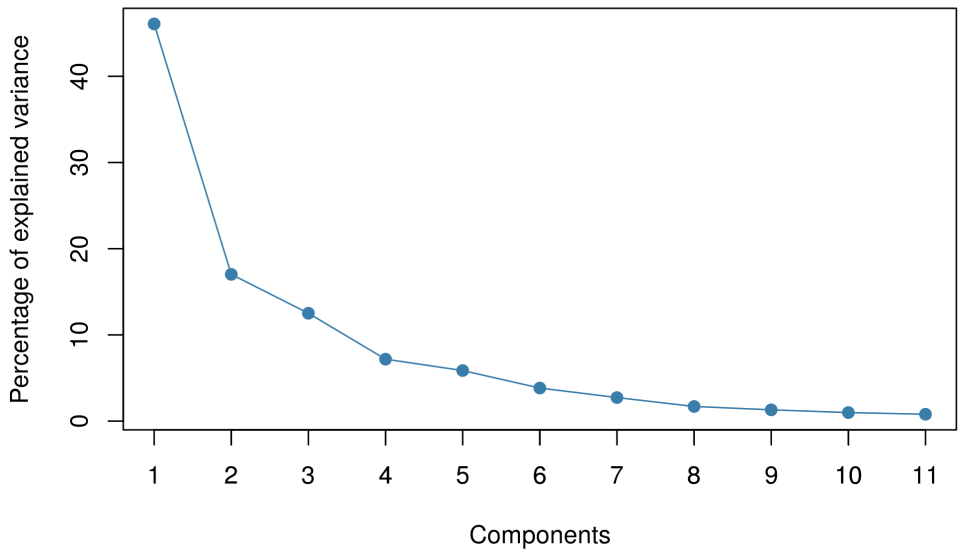


Figure 7: Components contributions to PCA

The `plot()` function can be used to visualize variable space with the following R command:

```
plot(reg.pca, choix = "var", cex = 0.8)
```

The result is shown in Figure 8:

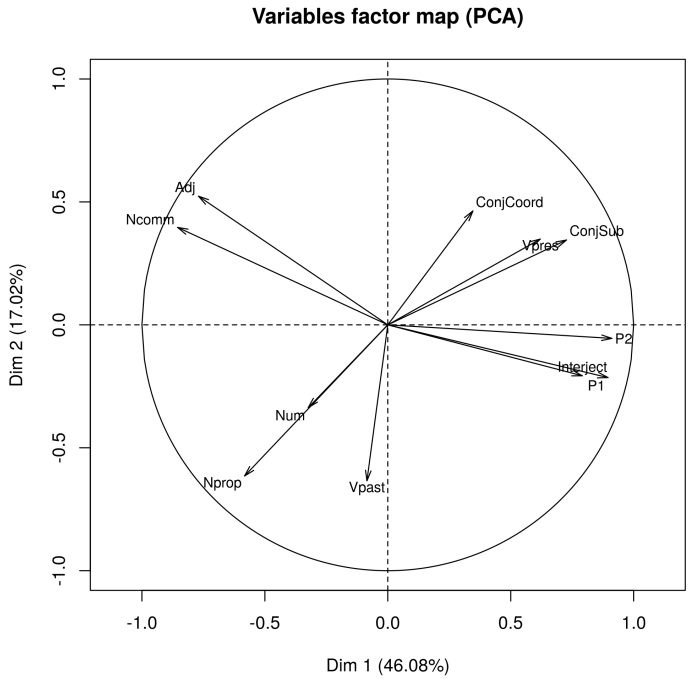


Figure 8: PCA Plot

The diagram shows the first two components as two axes and each vector which goes from the center of the diagram represents a variable. Angles between these vectors show how strongly the variables are correlated (a smaller angle represents stronger correlation). Vectors length defines a quantity of variation in the variable (maximum quantity is 1 and is represented by the circle on the plot). Vector orientation shows correlation of variables. For the given BNC corpus, it is visible that variables `P1` and `P2` (fist and second person pronouns) are opposed to common nouns (`Ncomm`) and adjectives (`Adj`).

Another example of statistical processing of natural language within the SSF is agglomerative clustering (AC). The AC is a bottom-up method for the creation of word groups which treats each word as singleton cluster and then merges it with other similar clusters until all words are merged into one single cluster. The R function which implements AC is called `hclust()`. The function can use different clustering methods such as: a) complete (which observes the farthest neighbors of clusters in relation to a singleton and merges it with those clusters whose farthest neighbors are the nearest), b) single (observes the nearest neighbors in relation to singleton and merges it to the cluster with the smallest distance), c) average (observes distance from the singleton to the average distance of each

cluster element, and merges the singleton to the cluster with the smallest distance values) and d) ward (this method is different from all other methods since it uses a difference in variance to calculate distances between clusters). The distance between clusters can be calculated in different ways. The R function `dist()` which is used for distance calculation can accept many methods of distance calculation but some of the most common are `euclidean`, `maximum`, `manhattan`, `canberra` or `minkowski`.
The euclidean distance is is calculated as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The maximum distance is calculated as:

$$d(x, y) = \max_{i=1}^{n} |x_i - y_i|$$

The Manhattan (city-block) distance is calculated as:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

The Canberra distance is calculated as:

$$d(x, y) = \sum_{i=1}^{n} \left| \frac{x_i - y_i}{x_i + y_i} \right|$$

To illustrate how AC can be conducted in the SSF, an example inspired by Desaguilier [44] is shown. For that purpose, the special demonstrational data set with some of Croatian intensifiers is prepared.

```
words <- c('malo', 'užasno', 'posve', 'potpuno', 'krajnje', 'ponešto',
    'savršeno', 'dosta', 'više', 'neznatno', 'nešto', 'sasvim')
data <- ACdata(words)
```

The `data` object was created using the special SSF's NLF R function called `ACdata()` which extracts words and their collocations from the SSF's corpus. In the above example a subset of 12 words was extracted. The `data` object is a two dimensional matrix which in each row has one of the intensifiers, whereas each column is a word from the SSF's corpora which followed it. The SSF's corpus is built from various online documents which are freely accessible and openly licensed texts. Values within the matrix are a number

of occurrences of such collocations. The function `rownames()` displays row names of the `data` object.

```
> rownames(data)
[1] "malo"     "užasno"   "posve"    "potpuno"  "krajnje"  "ponešto"
    "savršeno" "dosta"    "više"     "neznatno"
[11] "nešto"    "sasvim"
```

Similarly, the function `colnames()` would display names of columns, however, since there are 364 variables in the example data set, only the first 10 are shown.

```
> colnames(data[1:10])
[1] "prije"      "puta"       "rasrdih"    "riba"       "ribica"
    "riječi"     "smirila"    "stanovnika" "svijeta"
[10] "toga"
```

Such a large table must be converted into a distance object. The function `dist()` as an argument takes the `data` object, and calculates distances using one of the already mentioned methods. For this demonstration, the Canberra method is chosen.

```
> dist.object <- dist(data, method="canberra", diag=T, upper=T)
> print(dist.object)
```

```
            malo   užasno    posve  potpuno  krajnje  ponešto savršeno  ...
malo       0.0000 418.8591 430.9602 430.6604 431.4528 427.4396 432.0000
užasno   418.8591   0.0000 432.0000 432.0000 409.7387 419.3910 423.1490
posve    430.9602 432.0000   0.0000 365.1911 432.0000 423.2072 410.9155
potpuno  430.6604 432.0000 365.1911   0.0000 428.5023 402.8318 401.0934
krajnje  431.4528 409.7387 432.0000 428.5023   0.0000 402.6862 427.9836
ponešto  427.4396 419.3910 423.2072 402.8318 402.6862   0.0000 414.8180
savršeno 432.0000 423.1490 410.9155 401.0934 427.9836 414.8180   0.0000
...
```

Finally, the function `hclust()` defines how elements from `dist.object` are clustered. The first parameter is a distance object, and the second is a method which will be used in clustering process. The method `ward.D` was used in this demonstration.

```
clusters <- hclust(dist.object, method="ward.D")
plot(clusters, sub="Canberra, Ward method")
rect.hclust(clusters,3)
```

The function `plot()` creates the dendrogram (as shown in Figure 9), whereas function `rect.hclust()` assigns groups to it. The first parameter of the function is a `cluster` object and the second parameter is a number of groups that should be highlighted. In this example, three groups were created, from the left to the right: maximizers (cro. *posve* and *sasvim*), moderators (cro. *neznatno, malo* and *nešto*) and boosters (cro. *potpuno, savršeno, više, ponešto, dosta, užašno, krajnje*).

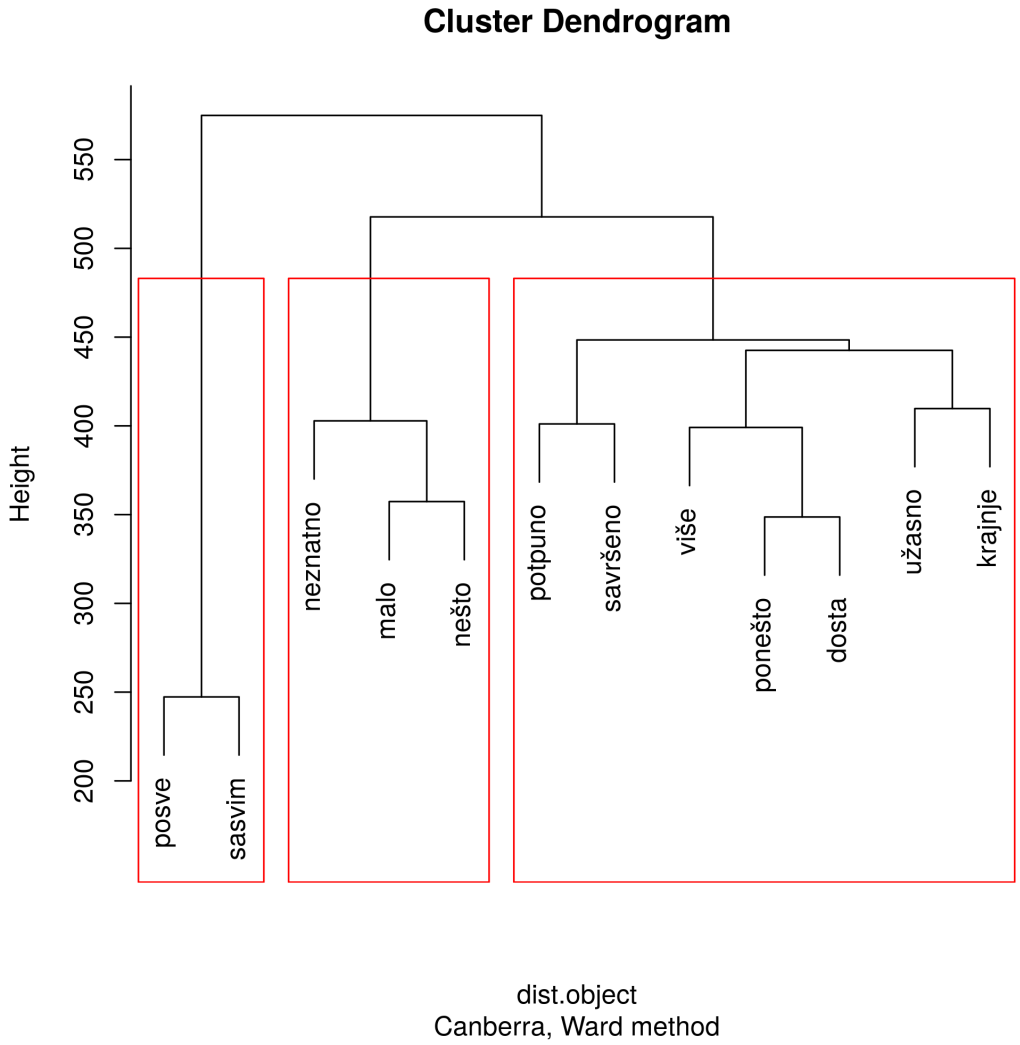


Figure 9: A sample cluster dendrogram of Croatian intensifiers divided into three groups

If the second parameter of the function `rect.hclust()` is for example 2, the dendrogram like shown in Figure 10 would be drawn. These examples were made using a very small subset of sentences from the SSF's corpora in order to demonstrate how agglomerative clustering can be conducted in the SSF. The output results depend on the corpus size and corpus types. A greater number of sentences will result in better word grouping. Every SSF user has an ability to upload his own textual documents and experiment with his own data.

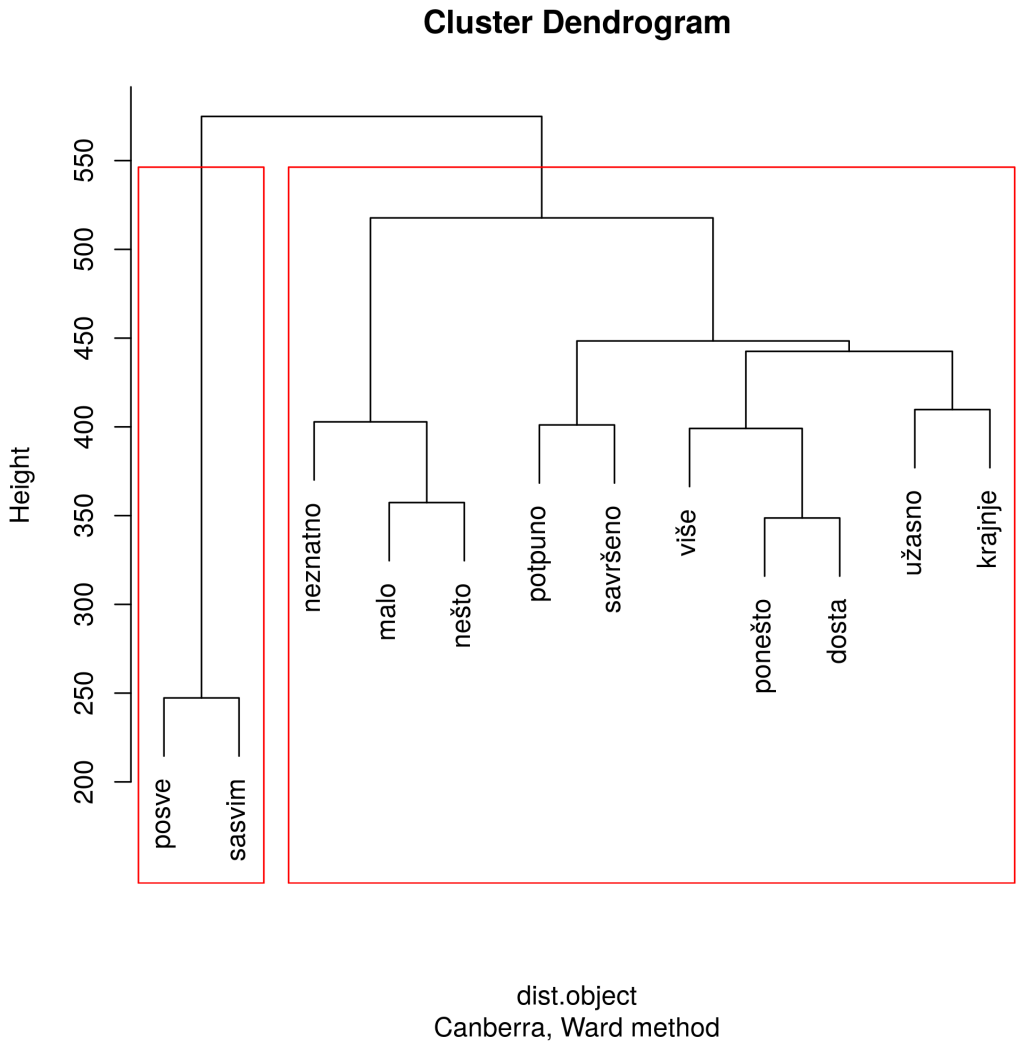


Figure 10: A sample cluster dendrogram of Croatian intensifiers divided into two groups

One more example of statistical processing within the SSF is calculation of association measures (i.e. detection of collocations, colligations and collostructions). All these linguistic terms are related to the measures of association between two or more words. Collocations are expressions which consist of words that correspond to some conventional way of saying things and in combination they mean more than each word would mean by itself (e.g. strong tea, USB port, etc.). Firth [63] states that "collocations of a given word are statements of the habitual or customary places of that word". Colligations are similar to collocations except that they also introduce a grammatical component of the word (i.e. the bonding is based on the syntactic structure of the words). These lexical terms are very similar since some multiword expressions can be at the same time both a collocation and a colligation. Sometimes they can even have other words inserted within them. Lexical constructions which are the interactions of words and gramatical constructions (i.e. the basic unit of linguistic organization: a pairing of form with meaning/use) are called

collostructions [154]. Detection of such constructions is extremely important in the process of sentences parsing in order to properly detect noun/verb phrases.

Association measures are usually calculated using frequencies of words within a corpus (like shown in the Table 9). The quadrant I corresponds to the number how many times the word W_1 appeared together with the word W_2, the quadrant II corresponds to the number of how many times the word W_1 occurred without the word W_2, the quadrant III corresponds to the sum of occurrences of all other words (which are not W_1), together with W_2, and the quadrant IV is the total number of words without W_1 and W_2.

Table 9: Co-occurrence frequencies for association measurements

	W_2	$\neg W_2$
W_1	I	II
$\neg W_1$	III	IV

These four values are used in various calculations for measuring associations between words. For that purpose, the SSF implements its own NLF R function called `WordCooccurrences(`W_1`,`W_2`)` which accepts two arguments (words W_1 and W_2) and as a result returns a dataset with values as described in Table 9. The function connects to the SSF relational database and extracts the data (described in Section 7.1) about word occurrences from the whole SSF's corpus. The following R example shows occurrences of words from the collocation cro. *crno vino* (eng. red wine).

```
> data <- WordCooccurrences("crno","vino")
> print(data)

       crno  !crno
vino   1     298
!vino  94    2563197
```

The probability of the word *crno* given the word *vino* can be calculated as the co-occurrence of *vino* and *crno* divided by a total number of occurrences of *crno*: $1/(1+94) = 0,01$ (which is 1%). The conditional probability of *vino* given *crno* is calculated as their co-occurrence divided by the total occurrences of *vino*: $1/(298+1) = 0,003$ (which is 0,3%), which means that the *crno* is a stronger clue for *vino* than the opposite. These numbers emerged from the relatively small Croatian corpus which is loaded into the SSF. The bigger the corpus is the more relevant results will be.

No association measure is perfect, but some of the most popular collocational and collostructional strength measures are: Student's t-test, Pearson's chi-square test, log-likelihood and the Fisher exact test [43].

Student's t-test is often used for discovery of collocations. The null hypothesis H_0, which says that two words are independent, is defined first. H_1 hypothesis says that words are not independent is stated at the same time. The t value is then calculated as:

$$t = \frac{\bar{\chi} - \mu}{\sqrt{\frac{s^2}{N}}}$$

where μ denotes the frequency with the assumption that the words are independent, $\bar{\chi}$ denotes an observed frequency, s^2 is an observed variance, and N is a total number of words in the corpus. If the t value is greater than some threshold value (2.576 for large N and 99.5% confidence) the hypothesis H_0 can be rejected. The main disadvantage of the t-test was the assumption that the data are normally distributed. Another commonly used method is Pearson's chi-square test (χ^2) which also assumes that the words are independent and is calculated as:

$$\chi^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where i and j denotes all rows and columns of the Table 9. The O_{ij} denotes the observed value from the table, whereas the E_{ij} denotes the expected value (if the words are really independent). Similarly, if the χ^2 value is above some threshold, the collocation is accepted. An often used alternative to the χ^2 is log-likelihood and is calculated as:

$$G^2 = 2 \sum_i O_{ij} \log(\frac{O_{ij}}{E_{ij}})$$

The G^2 is used to test two hypotheses (H_0 - which states that two words are independent, and the H_1 - which states that the W_2 depends on the W_1). Manning and Schütze [106] states that the G^2 is more interpretable than χ^2 because there is no need for the usage of the table with threshold values. G^2 value of 3.8415 or higher is significant at the level of $p < 0.05$, whereas a value of 10.8276 is significant at the level of $p < 0.001$. Therefore, with such values and with such significance levels the hypothesis H_0 can be rejected. One another association measure is Fisher's exact test. Similar to the Pearson's chi-square test, Fisher exact test uses the values from the matrix shown in Table 9. The null hypothesis (H_0) is that two words W_1 and W_2 are independent from each other. The R has a function called `fisher.test()` which as an input can take the matrix from the SSF's `WordCooccurrences()` function.

```
> data <- WordCooccurrences("crno","vino")
> fisher.test(matrix(unlist(data), 2))

Fisher's Exact Test for Count Data

data: matrix(unlist(data), 2)
p-value = 0.01102
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
2.285001 534.822027
sample estimates:
odds ratio
91.34794
```

In above example, the p-value is very small, therefore the null hypothesis can be rejected at a significance level of 0.05. The function also calculates the odds ratio value (θ) which quantifies the association between presence or absence of words (W_1 and W_2), and can be formally expressed as:

$$\theta = \frac{\Omega_{W_1}}{\Omega_{\neg W_1}}$$

where:

$$\Omega_{W_1} = \frac{P(W_1, W_2)}{P(\neg W_1, W_2)}$$

and:

$$\Omega_{\neg W_1} = \frac{P(W_1, \neg W_2)}{P(\neg W_1, \neg W_2)}$$

If the θ value is greater than 1, the relation between words is stronger [3]. In the example from the SSF's corpus, the θ value is 91.34794 which is high above 1, and indicates a strong relationship between words W_1 and W_2.

The decision about the method the one will use is usually conditioned with the type of data (e.g. for low frequency data, t-test is less reliable than the log-likelihood). There are also many other association measures (e.g. Attraction, Reliance, Pointwise Mutual Information, z-score, Minimum sensitivity, Jaccard coefficient, Dice coefficient, Log odd ratios, Liddell's difference of proportions, Geometric mean, etc.) [98, 55], and all of them can be used within the SSF's R NLF module. Every registered user has an ability to develop his own statistical functions using the corpus data from the SSF's database.

3. Tagging

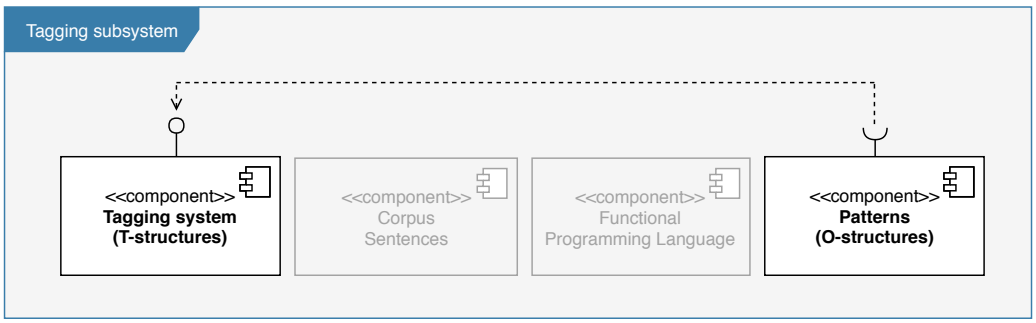


Figure 11: Conceptual model of a tagging subsystem

Tagging is simple mapping of $y = f(x)$, where each element in the set of X corresponds (is mapped) to one and only one element from the set of Y. This function is represented by ordered pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where $x_i \in X$, and $y_i \in Y$. Mathematicians consider a set of X to be a *domain*, and a set of Y *codomain* of the function $f()$. Philosophers consider these sets to be sense – reference pairings, linguists consider them to be lexeme – concept pairings, and the software developers in their scripts consider these to be key – value pairs. Functions can be chained to extend the domains in the so-called composition of functions – i.e. the *codomain* of one function is the domain of the other. It should be noted that by the definition of the function it is necessary that x_i is not mapped to more than one element from Y. This renders semantic problems with homonyms (even synonyms) or computational problems with elements that have the same keys. The importance of mathematical function is not only in associating individual concepts to their natural language tags; moreover, this functional mapping continues in groups and word interconnections, as well as in phrases, sentences, and finally in lexicographic definitions. In lexicography and encyclopedics, a lemma is an element from the set of Y, and its definition is an element from the set of X, which is often a result of multiple functions composition with elements from other, different starting domains. In the printed versions of lexicographic or encyclopedic articles, this is graphically represented by using different font styles, or different symbols, which made it easier for users to find all the elements in the set of X that are linked to the informative element Y. The whole lexicographic tradition was built on this simple foundation. However, this simplicity may be deceiving, for it resulted in many perplexing and demanding lexicographic issues through the history of dictionary use and production. Some of those involve decisions such as: which elements from the set of X to choose, which functions to choose, which compositions to make, how to make the work accessible to the user, easy to use, durable etc.

Previous NLP techniques of stochastic language models required a huge amount of information (so, the Internet became almost the only source), since the words in the artifact are treated as numbers, which are then observed in their interrelationships. If words are associated with a tag - mostly just a POS tag, (i.e. an open word class), before it becomes a vector number, the information is richer, so it gets a better recall, but still not good enough. Most of these methods (except those with machine learning) are completely ineffective if they work with a small set of documents or just individual sections of texts. Moving to a deterministic model in which words will have many or even all tags, would solve the problem of rough information granularity, but then another problem arises - building of such a lexicon as the basis of the model.

In other words, if a digital dictionary/lexicon whose words are tagged with a huge number of grammatical and semantic tags can be developed, it will be possible to construct a deterministic model of language, which will include other syntax-semantic laws of the given language. Only then, when the words are precisely defined by the written form (grammar, emphasis, morphology and syllable analysis) and by meaning (taxonomy of concepts), it is possible to work on the discovery of their bonds in sentences, i.e. discovering lexical relationships that include syntactic (agreement, government, assignment) and semantic (paradigmatic and syntagmatic) relationships. The lexical relations arise from the relations of words in their comprehensiveness within the lexicon, the sentence, and within semantic categories (domains).

Tagging can also be seen as a process in which metadata are assigned to real data and thus they create a new type of information. It is in the foundation of the natural language - the living being marks concepts in its brain by sounds or written marks. In this sense, the language itself (written or spoken word, phrase or a sentence) is tagged thought. In the machine processing of the natural language, the tagging process is even more complex, since it can be used to extract structures from an unstructured text. This can be carried out at several levels of which these are standard ones:

- Morphological tagging - by using word classes (in Croatian language there are ten different word classes) and their categories (gender, number, case, time, etc.). This type of tagging is not always unambiguous, because often it connects different language levels. For example, feminine nouns in Croatian will have the same written form in both dative and locative, but the case tag will be correctly assigned only in the context - at semantic level and based on the word's environment (neighboring words), since the case is a syntax and semantic category. This type of problem is common in flexion languages (such as Croatian) with rich affixing.

- Syntactic tagging - is carried out on basis of already performed lower (morphological) level analysis. The meta tags are treated as objects which are then bonded and

raised to a higher (syntactic) level by new tags. This procedure can have two possible paths. The first path in which new tags contain only tags from the lower level stay only as a formal representation of the middle (grammatical) level. This set of tags is used in tagging of complex verb forms (which usually consists of multiple words, e.g. in Croatian, Perfect Tense consists of - an auxiliary verb '*to be*' and an active participle, or Future Tense consists of - an auxiliary verb '*want*' and the Infinitive). Additionally, syntagmatic tags of neighboring words and their relations (e.g. adjectives and nouns congruity, noun or verb phrases, etc.). The second path of tagging refers to the word service in the sentence, and the roles of these words (this time, they are not related to their class), e.g. subject, predicate, object or adverbial. In this sense, the subject in the sentence can be at the same time the noun in the nominative, or the verb in infinitive, so the meta tag of the subject is above the word class tag. This path mainly intersects with the semantic tagging but is limited to the functional service of the word in the sentence.

– Semantic tagging - is carried out on basis of the word's meaning in the sentence. This is the hardest type of tagging, because the ambiguity of the words and their relations, sometimes conditioned by the word ordering, represents a combinatorically complex task. In contrast to simple grammars (e.g. CFG - Context Free Grammar or Chomsky's argument structures grammar) this type of tagging is used in grammars which connect all sentence levels and structures. For example, such grammar is RRG - Role and Reference Grammar proposed by professor Robert van Valin [169], in which the tags (core, nucleus, periphery, etc.) are related to both syntactic and semantic features (roles), which is already highlighted by the name of the grammar.

Although the process of marking the corpus is rather complicated (time consuming in terms of manual marking, and imprecise in terms of automated machine marking), still for over three decades the processes of automated methods have been continuously developed and improved. Among them, the most notable ones are those for POS (Part of Speech) tagging. The obtained information refers not only to frequency of each open word class, but also to the word's environment. Formally, tagging procedure (δ) can be expressed as:

$$\delta : X \to T,\ \delta(w_i) = t_i,\ t_i \in ST_{w_i},\ \forall i : 1 \leq i \leq |X|$$

where T is set of tags, for the input text X, and ST_{w_i} is the set of tags with meanings for the word w_i. There are many different marking conventions, and the most commonly used are SGML and XML. Both of them use similar notation (attribute-tag format) where attributes are placed within angled brackets (e.g. `<w>Word</w>` denotes the word) for an easier machine processing.

Automated machine marking can be either supervised or unsupervised. Supervised tagging is based on the usage of pre-tagged corpora which is used for training in the machine learning algorithms. In the next step the algorithm can, based on the training data, tag previously untagged corpora. On the other hand, unsupervised tagging does not require any training data, but instead uses sophisticated computational models for automatic word tagging. Each of these approaches have its pros and cons. In supervised models the output results are good when algorithms are used on the same genre of text as the training data, but that is usually not the case. Unsupervised models do not require any training data but word clustering and the results of such models are very coarse. The performance of a tagger is usually measured by the following measures:

$$Precision = \frac{Correctly\ Tagged\ Tokens}{Tokens\ generated}$$

$$Recall = \frac{Correctly\ Tagged\ Tokens}{Tokens\ in\ data}$$

$$F_1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall}$$

All these methods use the context of the word's environment (usually, two or three neighboring words) to determine a correct tag. Some of statistical models used in automated tagging processes are:

- n-gram taggers [20, 34, 174] which limit the class of models to n - 1st order Markov models. Since this tagger model is a base for many other variants of taggers, the SSF implements general n-gram function (described in Appendix D) which can execute many other methods of supervised learning in the context of tagging.
- Trigrams'n'Tags (TnT) is a statistical model for tagging based on the Markov model, proposed by Brants [20]. For explicitly defined arguments of n-gram function, the SSF model can execute TnT tagging.
- Transformation-based Error-driven Learning (TBL) [23] automatically creates linguistic easily understandable rules for classification from the training set of data. The supervised TBL can be performed on both, small tagged and large untagged texts. Transformations are defined by a set of *if-then* rules. Due to the T-structure within the SSF, it is possible for an unknown text to use this type of tagging, and in the same way apply rules like: "change the tagging of a word from noun to verb if the previous word is tagged as a modal".

– The Maximum Entropy (MaxEnt) tagger [141], uses a probabilistic model defined as:

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

where h denotes a context from the possible set of words and tags, t is a tag from the set of tags, π is normalization constant, $\{\mu, \alpha_1, \alpha_2, ..., \alpha_k\}$ are positive parameters and $\{f_1, f_2, ..., f_k\}$ is a set of yes/no features. For each parameter α_i (weight) there is one feature f_i. For the observed word, the context is limited to the word itself, two preceding words together with their tags, and the following two words. The positive model parameters are used for the maximization of the likelihood of the training data.

– The memory-based (MBT) tagging [39, 40] stores a set of example cases in the memory. The example case consists of an observed word with the preceding and the following context, together with the corresponding category for the observed word. The tagging process is done by observing the current state and comparing it to the test sets in the memory. If such a pattern is not found in the memory, the tags are determined based on the form and context. In order to give quality results, memory based tagging requires large sets of training data.

– Decision trees [147] (also known as TreeTagger) is a Markov model tagger which is based on decision trees for getting better results in estimating of contextual parameters. The binary tree is built from the training data, where every node corresponds to the question about one or two previous tags and branches are yes or no answers to these questions.

– Neural networks - built from the large amount of processing units which are interconnected by weighted links. Each unit has an activation value, which is propagated through the network to other units [146]. The neural network can have multiple layers (MLP network) where units are arranged in layers and connections are made only between the neighboring layers. Layers in between the input and output layer are called hidden layers. For the given text, the network is trained to activate the output unit (every tag from the tagset has its own output unit), which represents the correct tag for the observed word.

All of these approaches have something in common. For any analyzed word there is a limited number of possible tags, and between them the correct tag can be detected by observing the context of the word. Described models which are used in unsupervised tagging can only partly be applied in an unsupervised, since they rely on the lexicon

with all possible tags for a given word type. The SSF is based on the lexicon model, but NLF enables the usage of supervised learning which is aided by its specially designed O-structures (Section 5.6) for even more precise tagging. Another useful tool which can be used for tagging, and which is a part of the SSF is a statistical tool R. Part of speech tagging in R is implemented by using automatic taggers like CLAWS (the Constituent Likelihood Automatic Word-tagging System) [68], the Stanford Log-linear POS tagger [161] or earlier mentioned TreeTagger.

3.1. Types of tagsets

Tags are descriptive elements that are assigned to words to produce a better classification for later machine processing [171]. There are many paradigms like (MULTEXT-East, Penn Treebank POS tags, SWETWOL, UD POS tags etc.) that are used for word tagging. Tags are usually divided into two main types:

- Atomic [37] - tags are treated as atomic symbols without any deeper internal structure (e.g. the Penn TreeBank tagset [111]); and
- Structured - tags can be decomposed in the deep, where every sub-tag branch has its own features. Two main types of such tagsets are: compact tagsets (e.g. MULTEXT-East [49] or or Czech Compact tagsets [74]) and positional tagsets (e.g. Czech Positional tagset [74]).

For languages with rich inflection (such as Romance and Slavic) tagsets are consequently large, so in order to make them more manageable is to use a structured principle (positional and compact tagsets). Another common question is whether to use the same tagging principles across languages which have similar properties (harmonized tagsets). Such tagsets make development of multilingual applications much easier. One common example of a harmonized tagset is MULTEXT-East. Its tags are vectors of 12 elements, that describe each word. If for any reason, some tag element is not applicable for a particular word it is set to `null` (e.g. each pronoun will be tagged with vector of 12 elements no matter if some of them are applicable or not). Pronouns without a gender, number and case, will still have a placeholder for this element. In this way unnecessary space is allocated, and additional time is later used in algorithms to process this kind of tags. At the same time beside syntactic features, some of semantic features (like animate) are also stored in the same tag, but other semantic features are missing. Similar situation occurs in Romance languages, too (e.g. the Spanish and the Catalan CLiC-TALP [36] tagsets). There are certain disadvantages of standardization [133] (e.g. in the MULTEXT-East project, to ensure interoperability between nine languages, new category 'type' was introduced, and

its values vary along affected languages. Another problem is that it is not clear that different grammatical categories have the same interpretation in each language).

Some of the first tagsets was American, Penn Treebank Tagset with 36 tags and then the other: The Lancaster-Oslo-Bergen Corpus (LOB) with 132 tags and Czech with about 4,000 tags. In Czech, there have already been structural changes in comparison to the American and English variants, and position vectors were created which hold information of every feature. This practice was adopted by Croatian [104] as well as Slovenian [50] linguists, which, in accordance with the universal grammar trend, defined similar vectors that covered 13 languages, including Croatian. Vectors for flexion languages (such as Croatian) are regularly richer (with greater number of elements) than those for inflexion languages (such as English). They called that standard MULTEXT-East Tagset and for Croatian, currently version 5 is available. Vector based tagging proposed by MULTEXT-East has a set of tags (columns) for each word grammar and its subcategories, and if the given word doesn't have them, it is set to *null* (e.g. each pronoun will be tagged with vector of 12 elements, no matter if some of them are applicable or not). Pronouns which cannot have gender, number and case, will still have a placeholder for this element. In this way unnecessary space is allocated, and additional time is later used in algorithms to process this kind of tags. At the same time beside syntactic features, some of semantic features (like animate) are also stored in the same tag, but other semantic features are missing. In the newly conceived and derived tree-like T-structure (which is briefly described in next section), every word is tagged only with those tags (grammatical and/or semantic) that are applicable for it (without unnecessary redundancy). T-structures are projected as scalable data types which can easily cover all annotation properties of commonly used tagsets. To emphasize that the processing of such a tagging is the same, the palindromes WOS (Word of Speech) and the SOW (Semantics of Words) are used for the names of the tag sets. This means that word tagging is universal and that no distinction is made between syntactic and semantic characteristics [18]. Moreover, the same procedure is carried out for both of them. Every tag can have its own tags, to any depth. In order to implement it in the artifact it was necessary to develop MWE procedures, so that words can be bundled into closed groups based on previously defined criteria. In this way it is possible to extract complex tenses (e.g. Perfect = unstressed forms of the auxiliary verb to be + participle, or semantic structures (e.g. collocations, metonymies etc.). Each MWE can have one or more tags from T-structures. If network information (e.g. from LZMK[9] encyclopedia) is added, then rich information node of every lexical unit is formed, which makes basis for the syntax composition of statements or sentences. In this way the semantic reach of such node is significantly expanded, while until recently most commonly

[9] `http://enciklopedija.lzmk.hr/`

it referred to the WordNet structure [67] (sets of synonyms - *synsets*). In the proposed model, besides WordNet and encyclopedic information, it is possible to add any other either classic or network repositories. The overview MULTEXT-East, Penn Treebank POS tags, SWETWOL, UD POS tags is given in Appendix F.

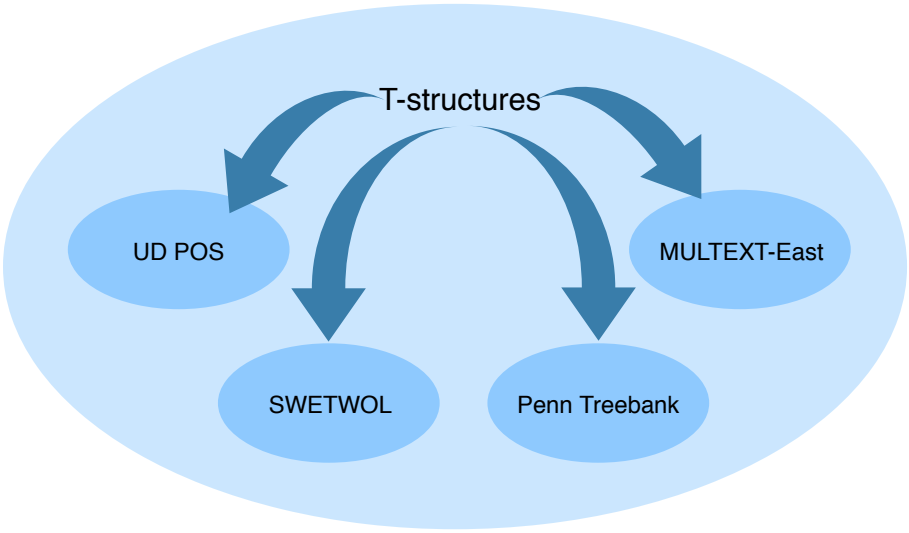


Figure 12: T-structures in relation to the commonly used annotation models

Since all annotation models are of linear type and also subsets of T-structures (Figure 12), it is possible to implement them within the network framework which is an answer to research question **Q1**. The example of how WOS/SOW marks (which are implemented using T-structures) are used for generation of MULTEXT-East tags is shown in Figure 13. Function `Word2MULTEXT` as input parameter takes a word for which tags are generated. For every lexical entry in the SSF's lexicon it checks WOS/SOW tags and using a set of rules (described in Appendix F) it generates MULTEXT-East tag. If the word has multiple occurrences (as show in in the example below), the word cro. *stol* (eng. desk) can at the same time be in nominative and accusative case, the list of possible tags is returned.

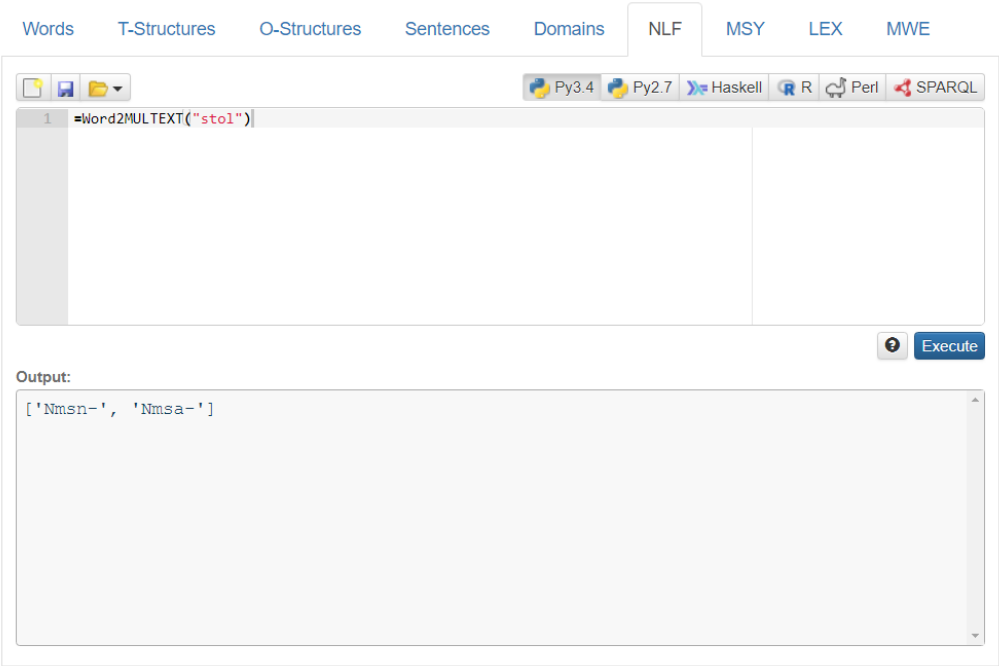


Figure 13: Generation of MULTEXT-East tags from T-structures

Similarly, to the `Word2MULTEXT` function, there is a function called `Wid2MULTEXT`. The difference is in the input parameter, which in the case of `Wid2MULTEXT` function is an integer (i.e. the ID of the word for which the tag is generated). Both functions are briefly described, with examples in Appendix D.

3.2. T-structures

Compared to MULTEXT-East, the main advantage of T-structures is that instead of using a limited set of tags, every SSF user has an ability to define its own sets of tags as well as their relations in a hierarchical tree structure. For example, users dealing with formation of words can extend WOS marks for morphs and deepen it with marks like prefix, base, suffix or extension. Tags created by regular users are by default marked as private tags and therefore are not visible to any other user of the system. The system administrator has an option to publish them publicly if needed. Each word has only these marks that are applicable to it, without unnecessary redundancy like in vector-based paradigms.

Language analysis functions at five main levels: phonological, morphological, syntactic, semantic and pragmatic. WOS markups cover the first three (phonological, morphological, syntactic) levels, while SOW markups cover the last two (semantic and pragmatic) levels. Since the word itself is a unity of form and meaning, it is not always possible to distinguish markups that relate to form from those related to the meaning. For example, in the Slavic

philology, pronouns in grammar are traditionally classified as personal, possessive, relative, etc. [42]. In the SSF this classification is done by applying WOS marks similar to POS marks in MULTEXT-East project.

Marks are organized into categories, taxonomically to infinite depth. The lower the mark is located in the tree, the more precisely it defines the word. For example, Figure 14 shows SOW markups for cro. *Ime* (eng. Name).

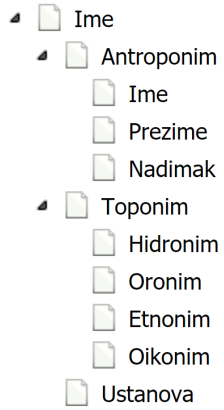


Figure 14: T-structures

If a word is tagged with the mark cro. *Oronim* (eng. Oronym), it is more specific than the mark cro. *Toponim* (eng. Toponym), and it is more specific than mark *Ime* (because *Ime* can be either a *Toponim* or an cro. *Antroponim* (eng. Anthroponyme). Extracting words with more marks and finer structure (more depth) brings more information (which can be processed). For example, to extract names of the rivers or last names of people from the text is not the same as extracting only some names (irrespective of the category). This applies to all information that can be transferred using language, not only for names. The complete list of WOS and SOW tags of the SSF is shown in Appendix E.

Each T-structure tag can also hold a value which can be of distinct types like integers, floats, strings, domains, etc. Most commonly they are strings (e.g. word definitions), but they can also be links to other words. Groups of words linked in this way, give special strength in forming of word nets. If T-structure value is a number (e.g. integer or float), then it is possible to weight words, for example in the process of sentiment analysis where a number can have a positive or a negative value. Another possible data type is a group of elements or a domain. For example, tag can hold a list of other words that are somehow related to the particular word (e.g. list of homonyms, hypernyms, synonyms etc.) Implementation of T-structures inside the SSF made writing of grammatical or semantic functions that deals with the words tagged by some of WOS or SOW marks a lot easier (e.g. to check grammatical congruity of two adjacent words, it's only necessary to compare WOS tags of case, gender and a number for them). The same applies to a

sentence construction (e.g. if the sentence is in singular and needs to be translated into plural, it is only necessary to find a word that has WOS tag of singular, find its lemma and for the same gender and case fetch the word with WOS tag of plural).

3.3. Word tagging

As each word has its own form (letters in writing and voices in speech) as well as meaning which that form implies, these two main features of words (syntactic and semantic) without which no valid identification is possible, are implemented using these WOS and SOW instances of T-structures.



Figure 15: Database model of word tagging

The database realization of T-structures subsystem for word tagging is shown in Figure 15. The central table `words`, stores all words that are part of the SSF's lexicon and their assigned unique ID's. Tables `wos` and `sow` are used to store information about T-structures itself. Beside the T-structure name they store there are information about the short form of the tag (`wos_mark` and `sow_mark`), data type, order of appearance in the GUI, username of the user who created the tag (for tags that are not publicly available), and the information about parent ID (because T-structures are hierarchical data type, as shown in Figure 14). Every word in the lexicon can have one or many tags assigned to it and at

the same time WOS/SOW tag can be associated to one or many words, in the so called many-to-many relationship which in terms of relational database models is usually solved by introducing a weak entity table(s) (`word_has_wos` and `word_has_sow`). These entities (tables) store information about the tagged word, tag itself, its value and the username of the editor which assigned the tag. In the same way, T-structures are assigned to other lexical parts (e.g. multiword expressions, syllables, morphs or lemmas). This innovative approach to word tagging plays a significant role in extraction of lexical relations since its extendable nature enables an unlimited number of specific tags to be created and assigned to words on the fly, which is important for a later extraction of sentence patterns. It also enables an easier transformation of lexicon data into an RDF structure for a later LLOD inclusion as described in Section 8.1. Figure 16 shows the word cro. '*majka*' (eng. mother) from the lexicon which is a good example on how different data types can be stored within T-structures.

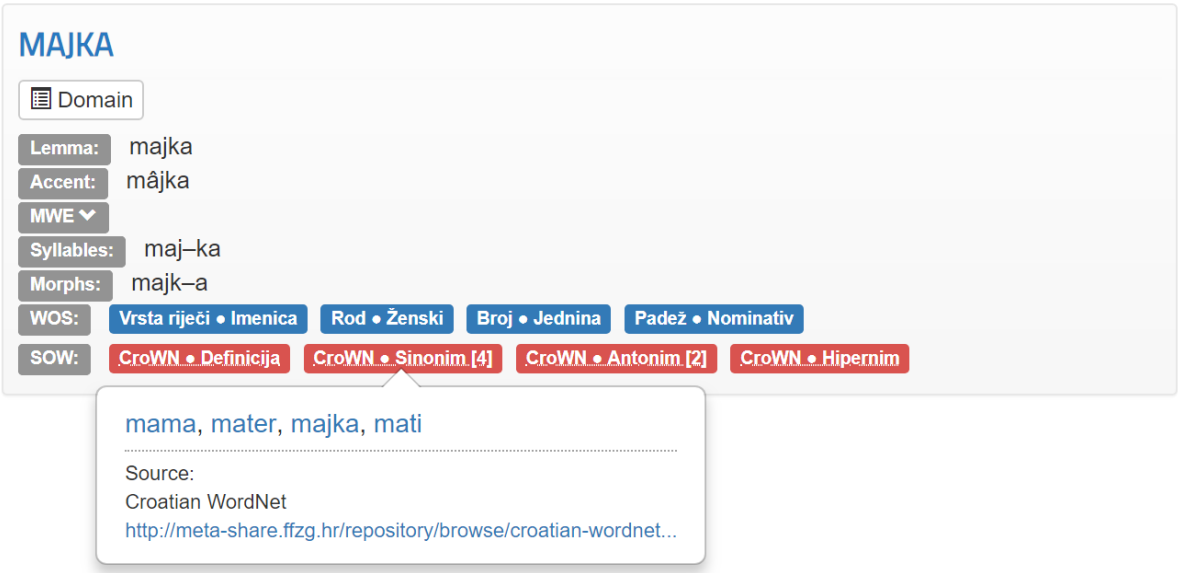


Figure 16: Lexicon entry with associated word tags

Blue boxes show WOS tags, while red boxes show SOW tags. The first SOW tag in Figure 16 holds the definition of the word as a string value, and the second SOW tag is a list of synonyms which are stored as integers (unique ID's of the words) and are directly linked to these words. At a database level it looks like the situation shown in an ER diagram in Figure 15, the word *'majka'* is stored in the table `words`. Executing the following SQL query:

```
SELECT wordid, word FROM words WHERE word='majka'
```

will result with the following:

wordid	word
18600	majka

meaning that the word '*majka*' is stored in the database with its unique ID number `18600`. If the SQL query which retrieves all SOW tags that are assigned to the word with the ID `18600` is executed:

```
SELECT sowid, sow_value, value_type FROM word_has_sow WHERE wordid=18600;
```

the result is the following output:

sowid	sow_value	value_type
97	žena koja je rodila jedno ili više djece	
102	9750	wid
104	18602	wid
104	18603	wid
104	18600	wid
104	18605	wid
110	18601	wid
110	18604	wid

The first line says that the word '*majka*' is tagged with the SOW tag (ID `97`) which has the value cro. "*žena koja je rodila jedno ili više djece*" (eng. a woman who gave birth to one or more children). Value_type is empty which means, it is a literal value (string) and will not be linked by identifying relation to any other entity. Other possible values are (`wid` - word ID, `mid` - multiword ID, `did` - domain ID, `txt` - strings, `img` - image, `snd` - sound, etc.). Behind these ID's in the SOW table there are definition, hypernym, synonym and antonym tags.

```
SELECT sowid, sow_name FROM sow WHERE sowid IN (97, 102, 104, 110);
```

sowid	sow_name
97	Definicija
102	Hipernim
104	Sinonim
110	Antonim

When data_type is `wid` then the data_value field stores the word's ID and not the word itself. To be able to see all the words that are behind these numerical ID values, it is possible to execute the following SQL query that joins all these tables and outputs an aggregated result:

```
SELECT whs.sowid, whs.sow_value, whs.value_type, s.sow_name, w.word
FROM word_has_sow whs LEFT JOIN sow s ON (whs.sowid=s.sowid)
LEFT JOIN words w ON (whs.sow_value=w.wordid) WHERE whs.wordid=18600;
```

sowid	sow_value	value_type	sow_name	word
97	žena koja je rodila jedno ili više djece		Definicija	
102	9750	wid	Hipernim	roditelj
104	18602	wid	Sinonim	mama
104	18603	wid	Sinonim	mater
104	18600	wid	Sinonim	majka
104	18605	wid	Sinonim	mati
110	18601	wid	Antonim	otac
110	18604	wid	Antonim	tata

Which is exactly the same what Figure 16 shows. The same logic applies also to the WOS tags. Along with the word ID's (integers) or definitions (strings), the WOS/SOW tag can hold images or sounds. Figure 17 shows such an example where the word cro. *žaba* (eng. frog) is tagged with an image, and a sound (Figure 18).

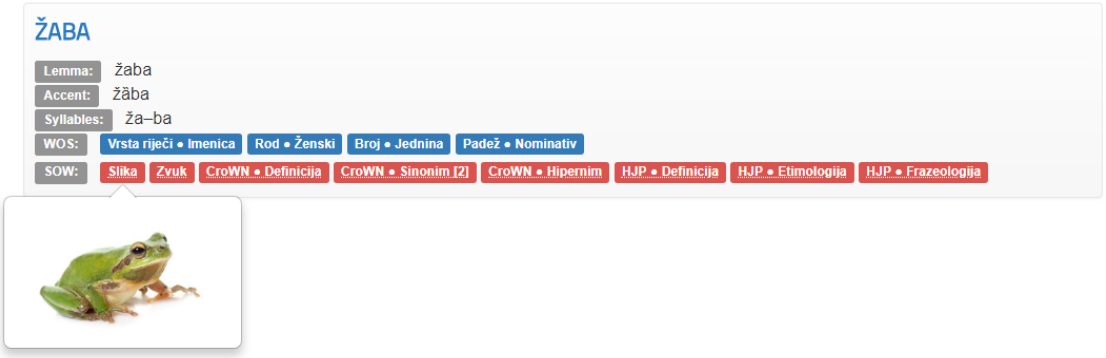


Figure 17: Lexicon entry with associated image tags

The image/sound file content is stored on a server's filesystem and the path to the file is stored as a textual value within the SOW tag.

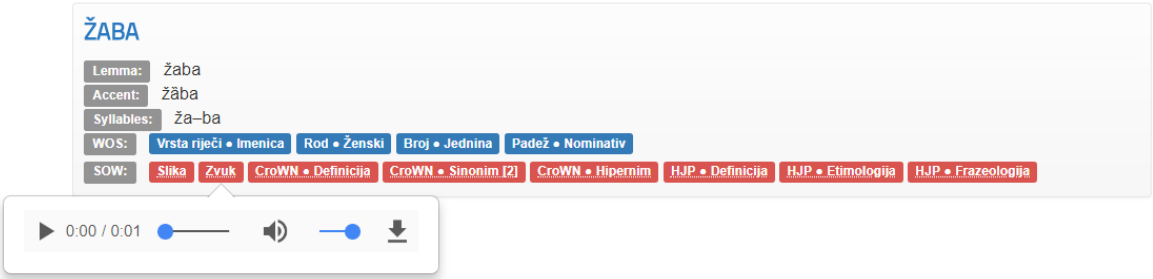


Figure 18: Lexicon entry with associated sound tags

Assigning tags to words within the SSF can be done in several ways. Most often it is done by an administrator through the lexicon module, but when it is necessary to do batch tagging the API NLF function for tagging is a better choice. As well as tagging, the NLF also enables some information retrieval from the lexicon, based on specific WOS/SOW tags. The functions `CountWOS()` and `CountSOW()` can be used for retrieving the number of lexical units in the lexicon which are tagged with specific WOS or SOW tag (Table 10 shows number of words in Croatian, in every open word class which were obtained by the `CountWOS()` function). The complete list of tagging related functions is listed in Appendix D.

Table 10: Croatian words by classes

WOS ID	Open word class	No.	%
2	Nouns	178,968	22,4%
3	Pronouns	2,310	0,3%
4	Adjectives	443,443	55,6%
5	Verbs	117,421	14,7%
6	Numerals	1,201	0,2%
7	Adverbs	1,035	0,1%
8	Prepositions	131	0%
9	Conjunctions	51	0%
10	Interjections	66	0%
	Other	52,713	6,6%

In the same way the SSF user can conduct different research works based on any other WOS/SOW marks (ID's are listed in Appendix E). For example, Table 11 shows the ratio of positive and negative words in Croatian language based on the SOW sentiment tags.

Table 11: Croatian words by sentiment polarity

SOW ID	Polarity	No.	%
285	Positive	82,755	10%
286	Negative	73,258	9%
	Unknown / inapplicable	641,359	81%

3.4. MWE tagging

Similar to the word tagging, multiword expressions can be tagged as well. Figure 19 shows an extract from the SSF's ER model which is related to multiwords and their tagging.

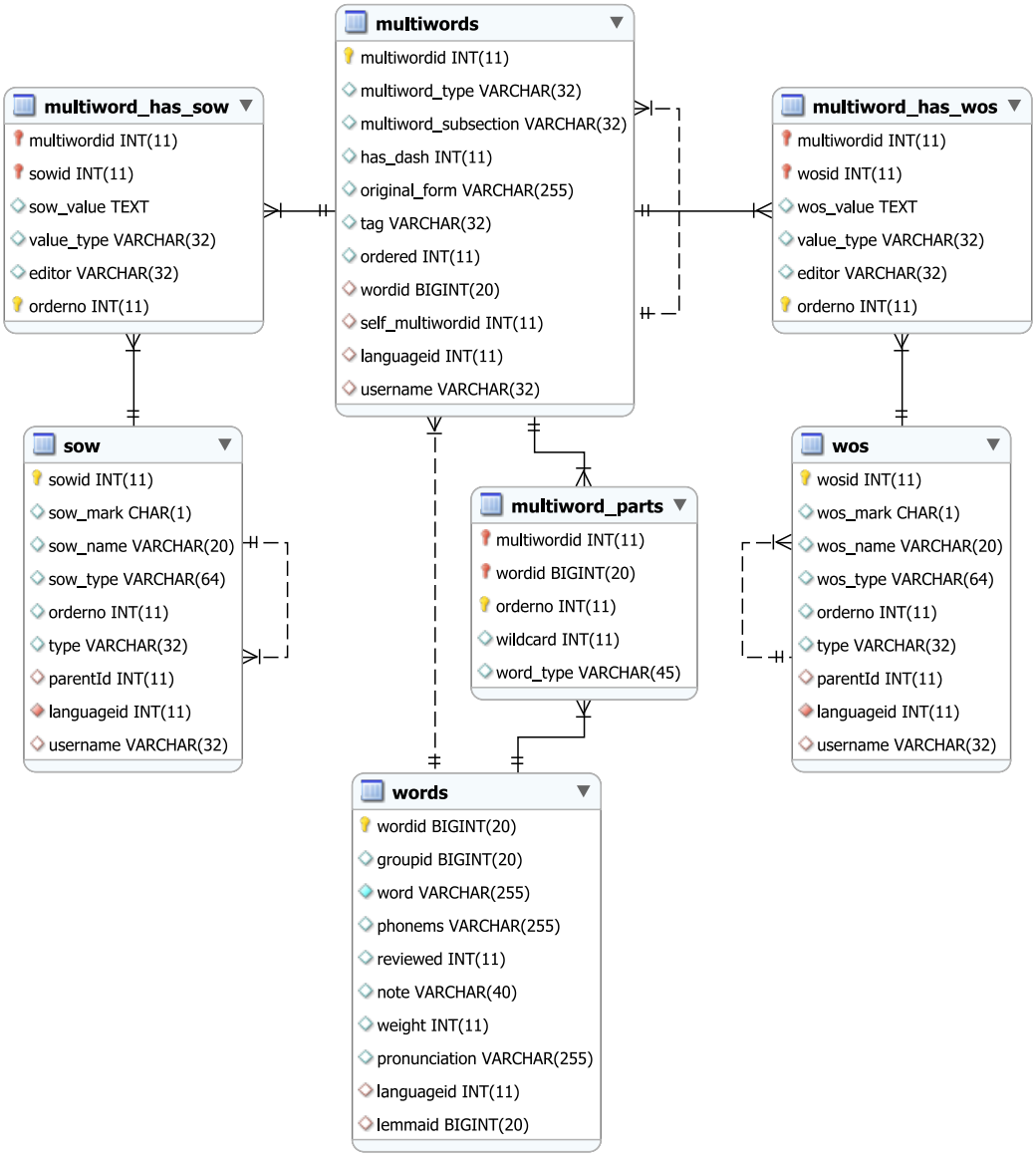


Figure 19: Database model of the MWE subsystem

The central table of a subsystem is the `multiwords` table, which stores multiwords in their original form. Table `multiword_parts` is a weak entity connecting multiwords to their elements in the `words` table. Similar to words tagging, multiwords tagging information is stored in tables `multiword_has_wos` and `multiword_has_sow`.

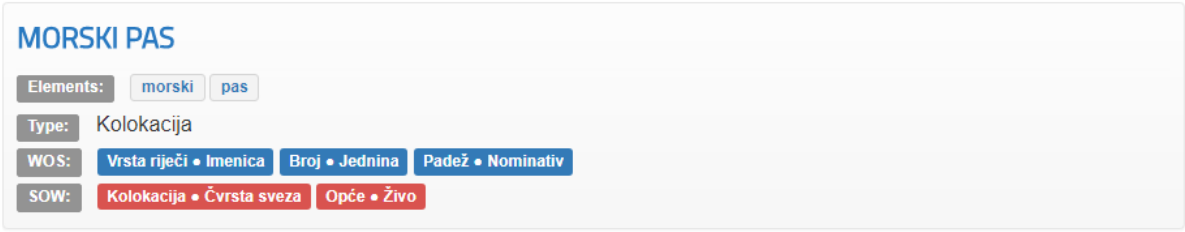


Figure 20: MWE Lexicon entry

Figure 20 shows an example of the Noun Phrase cro. *'morski pas'* (eng. shark) which is tagged with WOS tag as a noun, and SOW tag as a collocation. At a database level it looks as shown in an ER diagram in Figure 19, the multiword '*morski pas*' is stored in the table `multiwords`. Executing the following SQL query:

```
SELECT multiwordid, original_form FROM multiwords
WHERE original_form="morski pas";
```

will result with the following:

multiwordid	original_form
11201	morski pas

meaning that the multiword '*morski pas*' is stored in the database with its unique ID number `11201`. If the SQL query which retrieves all SOW tags that are assigned to the multiword with the ID `11201` is executed:

```
SELECT sowid, sow_value, value_type
FROM multiword_has_sow WHERE multiwordid=11201;
```

the result is the following output:

sowid	sow_value	value_type
290		

Which means that the multiword '*morski pas*' is tagged with SOW tag ID `290` (Kolokacija - čvrsta veza). It should be emphasized that tagging at the level of multiwords is not the same as tagging the words itself. In this example the multiword '*morski pas*' consists of

two elements (*morski* and *pas*). At the database level it is possible to execute the following query to see it:

```
SELECT mp.wordid, mp.orderno, w.word
FROM multiword_parts mp
LEFT JOIN words w ON (mp.wordid=w.wordid)
WHERE mp.multiwordid=11201;
```

which will produce the following:

wordid	orderno	word
7271	1	morski
17242	2	pas

If every word is checked separately for WOS marks it is visible that there are additional features that every word has, but the multiword itself doesn't have. For the word *morski* (ID 7271):

```
SELECT wo.word, whw.wosid, w.wos_name from word_has_wos whw
LEFT JOIN wos w ON (whw.wosid=w.wosid)
LEFT JOIN words wo ON (whw.wordid=wo.wordid)
WHERE whw.wordid=7271;
```

the outputs are:

word	wosid	wos_name
morski	4	Pridjev
morski	15	Nominativ
morski	23	Muški
morski	28	Množina
morski	32	Pozitiv
morski	36	Određen

and for the word *pas* (ID 17242):

```
SELECT wo.word, whw.wosid, w.wos_name from word_has_wos whw
LEFT JOIN wos w ON (whw.wosid=w.wosid)
LEFT JOIN words wo ON (whw.wordid=wo.wordid)
WHERE whw.wordid=17242;
```

the outputs are:

word	wosid	wos_name
pas	2	Imenica
pas	18	Akuzativ
pas	23	Muški
pas	27	Jednina

3.5. Lemma tagging

Along with words and multiword expressions, lemmas are the third entity in the SSF that can have WOS/SOW marks assigned to them. Figure 21 shows an extract of the ER model which describes how lemmas are stored in the database.

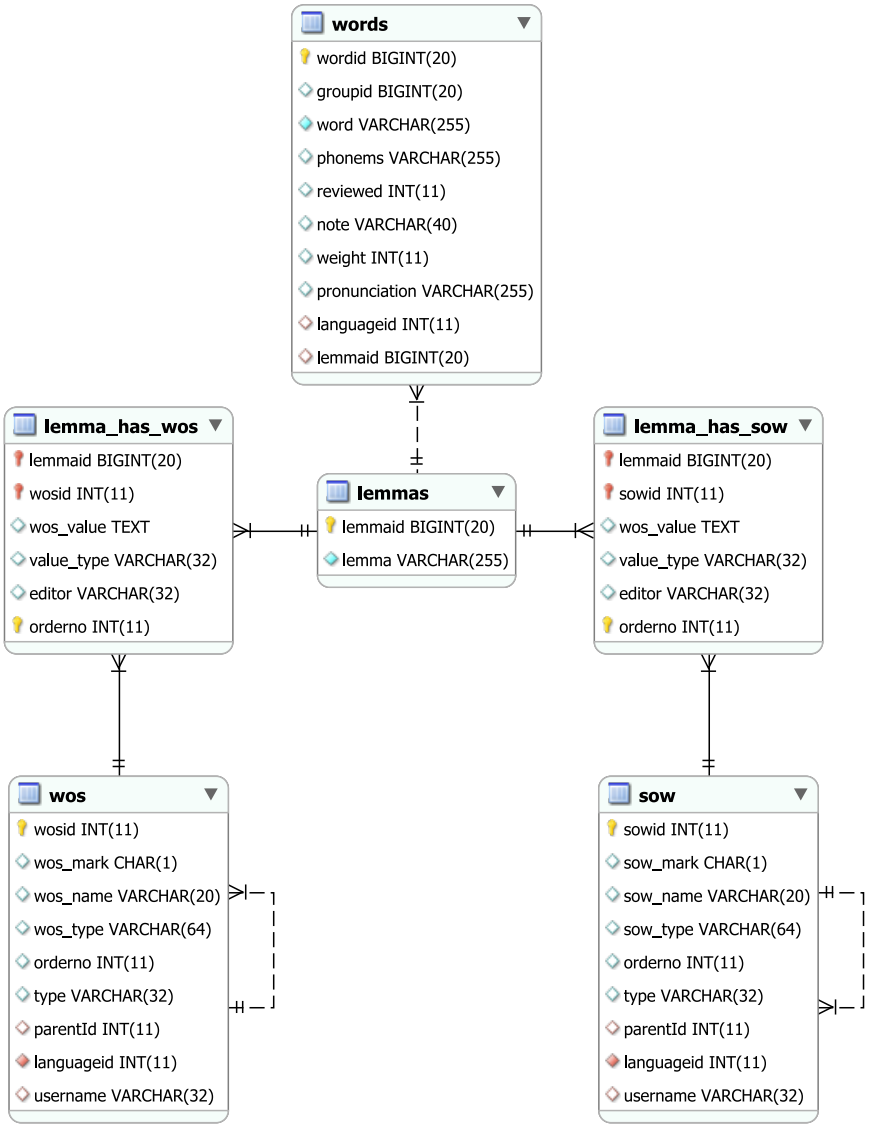


Figure 21: Database model of a lemma subsystem

Tagging at a lemma level is extremely important in cases when some syntactic and even more often semantic patterns need to be detected (e.g. in the algorithm of metonymy detection, SOW tag for *Symbol* is used at the lemma level in combination with the *Core verbs* SOW tag).

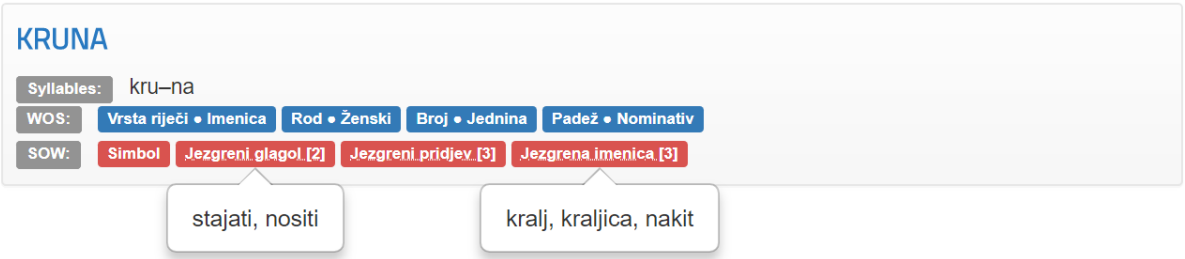


Figure 22: Lemma lexicon entry

Figure 22 shows the word cro. *kruna* (eng. crown) in the lemma lexicon. Similar to the examples above, if the following query is executed in the database:

```
SELECT lemmaid, lemma FROM lemmas WHERE lemma='kruna';
```

will output the following result:

lemmaid	lemma
9019	kruna

furthermore, to see which SOW tags are assigned to this lemma the following query can be executed:

```
SELECT lhs.lemmaid, lhs.sowid, s.sow_name, lhs.sow_value, l.lemma,
    lhs.value_type
FROM lemma_has_sow lhs
LEFT JOIN sow s ON (lhs.sowid=s.sowid)
LEFT JOIN lemmas l ON (lhs.sow_value=l.lemmaid)
WHERE lhs.lemmaid=9019;
```

and will result with the following output:

lemmaid	sowid	sow_name	sow_value	lemma	value_type
9019	303	Simbol			
9019	304	Jezgreni glagol	6541	stajati	lid
9019	304	Jezgreni glagol	975	nositi	lid
9019	313	Jezgrena imenica	9435	kralj	lid
9019	313	Jezgrena imenica	10814	kraljica	lid
9019	313	Jezgrena imenica	2444	nakit	lid
9019	314	Jezgreni pridjev	9994	kraljevski	lid
9019	314	Jezgreni pridjev	2439	zlatan	lid
9019	314	Jezgreni pridjev	8071	željezan	lid

The importance of such tagging in the process of metonymy and metaphor extraction is briefly discussed in Section 7.3.

4. Lexicography

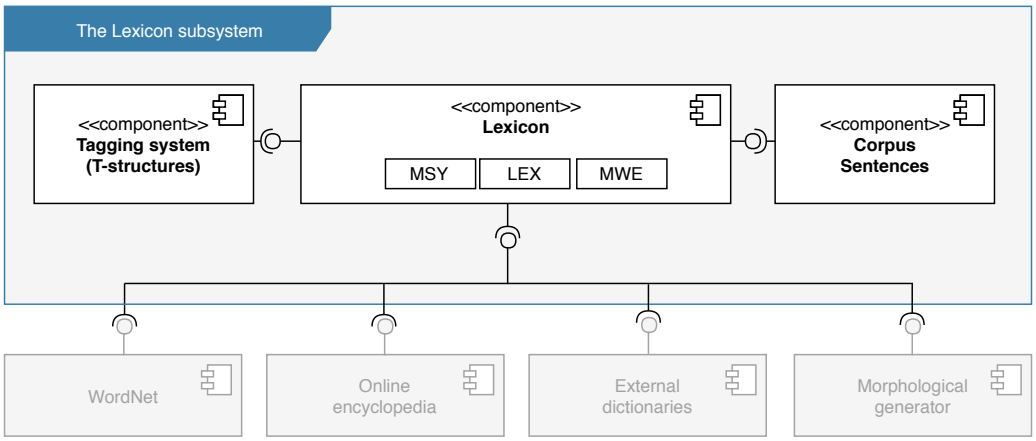


Figure 23: Conceptual model of the lexicon subsystem

Lexical units organized as a catalogue (abecedarium) form a dictionary of a given language. If a dictionary is enriched with more information about a word (e.g. word lemma), it becomes a lexicon, and furthermore if those lemmas are organized in a field categories, it becomes a thesaurus. Dictionary, lexicon and a thesaurus until recently, appeared regularly only in the printed form, but nowadays with the growth of digitalized materials they are becoming more and more accessible in digital form, too. Regardless of the form, lexicons or thesauruses store information that contain a variety of features:

- Form and word tags (lexical categories);
- Definition;
- Proper use of words and phrases, and the relation between them;
- Etymological tags about morphs origin; and
- Examples of sentences that describe the meanings.

If the same printed lexicon or digital repository combines and processes in parallel lexical units from multiple languages, it is called a multilingual dictionary. Digital lexical repository or lexical database was almost an exact copy of the printed edition (e.g. Croatian Language Portal dictionary [81]) while nowadays it is a common practice to build it around lexemes which include all language morphs, even if some of them never appear independently [137]. There are primary and secondary lexical categories. Each natural language has at least two primary lexical categories: nouns and verbs, and usually their modifiers (adjectives and adverbs). Secondary categories usually are consisted of unchangeable words: prepositions,

exclamations, conjunctions and particle, while the nouns are associated with the pronouns, and modifiers are associated with numbers. Primary categories can then be classified in different subcategories, e.g. nouns based on gender, number and case, and verbs based on tense, mood, person, etc. The question that arises is how to implement them in digital format? The goal is to provide as much information as possible on a particular word, so that information can be algorithmically quickly and efficiently processed. One way is to break down the lexical category to the smallest parts, then to assign each part (most commonly a mnemonics) with tags, and then to add these tags to the words that have them. Different tagging systems are briefly described in Section 3.1.

4.1. The word grammar

Word is a complex grammatical structure regarding internal components and relation to other words. Thus, instead of a regular lexicon (list of words in a given language) it was necessary to develop three types of lexicons. The first type is related to a lexicon of word's internal components, second type is related to word itself, while third type is lexicon of word groups which are conceptually different (multiword expressions). For such approach it is necessary to define basic terms of components and compounds.

According to Carstairs-McCarthy [28] and Booij [19] a lexeme is an abstract unit of vocabulary realized by words/word forms, 'having both form and meaning, but being neither, and existing outside of any particular syntactic context' [7], i.e. a lexeme is an abstract unit which is represented by a set of grammatical forms.
Lexeme appears:

- Language specific (every language has different sets of word forms);
- Word-class specific (word forms are specific for different word classes); and
- Inflection-class specific (different inflection classes defines different word forms (e.g. cro. *šetam, šećem*)).

Inflexional morphology consists of many grammatical forms (paradigms) (e.g. cro. *radim, radiš, radi..*), whereas derivational morphology consists only from derived form (e.g. cro. *predradnik*). The SSF's lexicon contains all morphological forms of words, not only lexemes and along with words for every word class the lemma is also stored (for verbs lemma in infinitive form, for nouns lemma in nominative singular and for adjectives lemma in nominative singular masculine in short form. If the adjective does not have a short form, then a long form is stored). A root is a fundamental part of a word which remains when all prefixes and suffixes are removed and cannot be further decomposed. The nature of human natural language that every expression (text) can be divided into a sequence of units at

two levels is one of fundamentals of Martinets functional linguistics and is called double articulation. By the first articulation, every expression is divided to a sequence of minimal language characters (i.e. units which has vocal form and meaningful content). The result of the first articulation are morphemes. The type of morpheme is morph. By the second articulation, the signifier or the vocal expression of each morpheme can be broken down into a series of successive distinctive units, i.e. those that do not have their own meanings, but contribute to the differentiation of the higher level meaning units. Double articulation contributes, on one hand, to the fact that the tagged and the tagger are related, and on the other hand, contributes to the economics of the linguistic organization: with only a few tens of phonemes it is possible to compose a thousands of different phonemes/morphemes, and by using a relatively limited number of morphs (few thousands) it is possible to express a very large number of general spoken or written terms of human experience. The place where the morphs are connected is called morphic boundary, link or junction. In Croatian there are four junction points between morphs [112]:

- Between prefix and the root: *na-uk, na-učiti*;
- Between root and suffix: *uč-iti (uk-i-ti), učenost (uk-je-n-ost-0)*;
- Between two bases/roots: *vjer-o-nauk, stran-putica*; and
- Between the base and grammatical sufix: *putnic-i (putnik-i), nauk-a, nauči-ti, nauči-m.*

while morphological structure of open multimorpheme words class in Croatian language is:

$$\mathrm{P} - \mathrm{R} - \mathrm{I} - \mathrm{R} - \mathrm{S} - \mathrm{S_{gr}}$$

where: P= prefix, R = root, S = sufix, I= interfix, $\mathrm{S_{gr}}$ = grammatical sufix.

An interfix appears when there are two roots, and $\mathrm{S_{gr}}$ appears in each flexional change which has suffix. Prefixes and non-grammatical suffixes may be more, for example:

$\mathbf{P_{0,1,2...}}$	**R**	**I**	**R**	$\mathbf{S_{0,1,2...}}$	$\mathbf{S_{gr}}$	**word form**
ne+na	∅	∅	uk	je+n+ost	ju	*nenaučenošću*

The boundaries between morphs do not have the same character. It is universal in languages that boundaries between prefix and root, as well as between two roots are prone to agglutination, bonding of words (cro. *sjeverozapad* (compound), *blagdan*) whereas a boundary between root and suffix is prone to voice fusion, i.e. word endings easier apply phonetic reduction than beginnings, which are prone to conserve morphological boundaries. There are also words made of only one morpheme (e.g. some conjunctions –

cro. *a, ali, ili..*), particles (cro. *da, ne*), numbers (cro. *pet, šest*), prepositions (cro. *na, o, po..*), adverbs (cro. *jučer, tik..*). One morpheme words are unchangeable, but not all multi morpheme words are changeable, e.g. derived adverbs (cro. *nabrzinu*) or derived prepositions (cro. *uoči*). There is also a general form of Slavic words division:

((PREF)-BASE-(DSUFF)-(TM)-(ISUFF))

according to Manova [109], where: Pref = prefix, Base = root, radix, Dsuff = derivational suffix, TM = thematic marker and ISUFF = inflectional suffix. Depending of which slot is filled, words can be in different conceptual category (e.g. derivation, cro. *ruka - narukvica*) or change the focus within the same term (e.g. deminutive cro. *ruka - ručica*).

There are two main approaches to morphology: grammatical and cognitive. Grammatical approaches are classical, generative [30], natural [48, 109], and cognitive is prototypical morphology [96, 95]. While generative grammar is focused on developing formalisms for word formation from morphs, natural morphology is focused on universal way for word formation using cognitive and semiotic principles [48], while taking into account diachronic (time) changes, functional analysis and language ontology. Cognitive Linguistics appeared in the mid 1980s and gives an overview of the basic ideas of the model in Ungerer and Schmid [164] or Tuggy [162], among others. Similar to natural morphology it is a reaction to the expansion of Chomskian linguistics. Both theories are similar but still are not connected yet, though there were attempts towards their integration through a prototype theory [47].

Prototype is a typical instance of a category whose elements are divided into categories based on their similarities [96]. Non-prototype instances match the same category but deviate from the prototype based on their distance from the prototype category. It is not necessary that every member of a category contains all of its attributes (e.g. category *birds* includes *robin*, *swallow*, *chicken* and *penguin*, based on zoology classification. However, as the *birds* prototype cognitive linguistics classifies only *robin* and *swallow*, whereas the *chicken* and *penguin* are classified as non-prototype. WOS/SOW structure enables easier definition of prototypes and non-prototypes of any type, due to its hierarchical nature and possibility to be extended with new tags. Furthermore, every word which is tagged with a tag of higher level has a direct link to those at the lower, atomic, level via breakdown to morphs, syllables and syllablemorphs (i.e. words tagged as diminutive will usually contain one of morphs like *-(č)ić, -ek, ak* or *–ica*, whereas words tagged as augmentative will have *-[e,u,č]ina*. In the morphological theory the idea of prototypes is applied in classification of derivation (DM) and inflection (IM) [47]. Linguists elaborate sets of many criteria for distinguishing IM from DM such as:

– Change of word class: Since IM involves smaller meaning changes than DM, it does not change word class, DM, however, does;

– IM serves syntax (e.g. grammatical agreement), whereas DM have a semantic function (e.g. augmentative or diminutive function);

– The meanings of IM are more abstract than those of DM;

– Inflectional rules tend to be general and are therefore more productive than those of DM, etc.; and

– Inflectional affixes have a more peripheral position in the word form than derivational affixes.

Due to its subatomic lexicon, the SSF model can control every IM and DM morphemes.

The SSF embeds Croatian morphological generator[10] as a part of the administrative module (as shown in Figure 24). The user selects open word class, enters the lemma of the word and based on which word class is selected enters some additional information (e.g. gender, number, etc.). After clicking on the button 'Generate' the output is shown in form of a table with all possible forms of a word. Due to a high complexity of the Croatian inflectional morphology, the algorithm for generating morphological forms (paradigms) for certain types of words gives a number of possible solutions. Therefore, it is necessary to perform a manual expert control in such cases to determine the correct form among offered solutions. The correct form is selected and permanently stored in a database (lexicon) with all WOS tags that are related to each word form. Editing of the lexicon can also be performed manually (as described in Section 4.2) by the administrator, but usage of such morphological generator rapidly speeds up the whole process since all word forms are automatically generated, inserted into the lexicon, and all tags are associated.

[10]The generator module is developed by Joško Markučič under the leadership of Professor Mario Essert

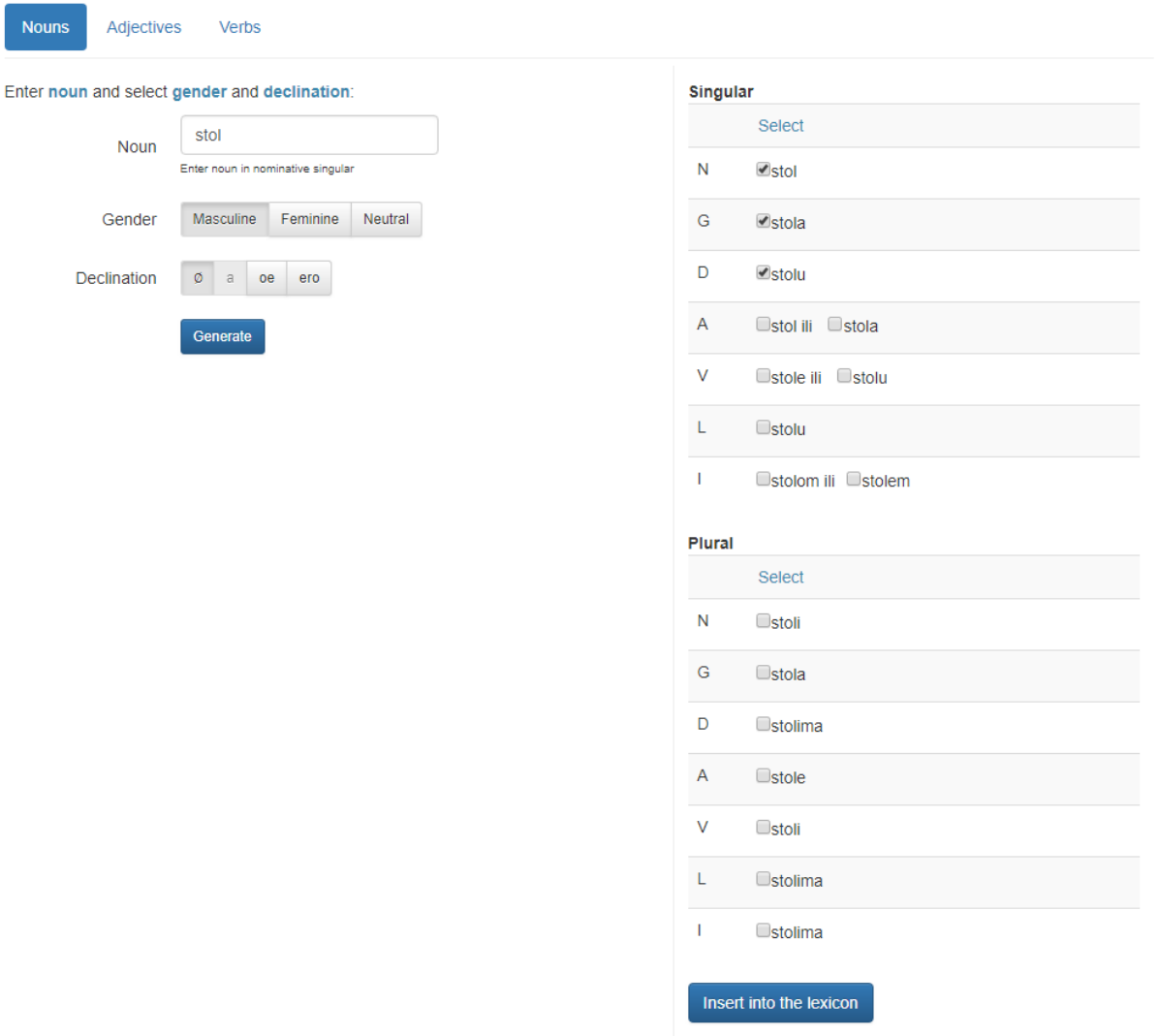


Figure 24: Croatian morphology generator

4.2. Different types of lexicons

The lexicon/thesaurus is built from the documents corpora or external resources [128]. The Parser component (described in Section 7.1) is responsible for populating all three types of Lexicons. Figure 25 shows part of the database model which is relevant for lexicons. The table `words` is a central place for storing words. Each word can be split to syllables or morphs which are stored in tables of the same name. Since every word can have one or more syllable/morph and at the same time syllable/morph can be a part of one or more words, the weak entity tables `word_has_syllables` and `word_has_morph` are introduced. They hold the information about which syllable/morph is a part of which word and its place within the word. The third type of a lexicon is the lexicon of Multiword expressions (MWE). Multiword expressions are expressions which are constructed of at least two words. The table `multiwords` stores information about such words. The table `multiword_parts`

holds the information about each single word which is a part of a multiword, and its place within the multiword.

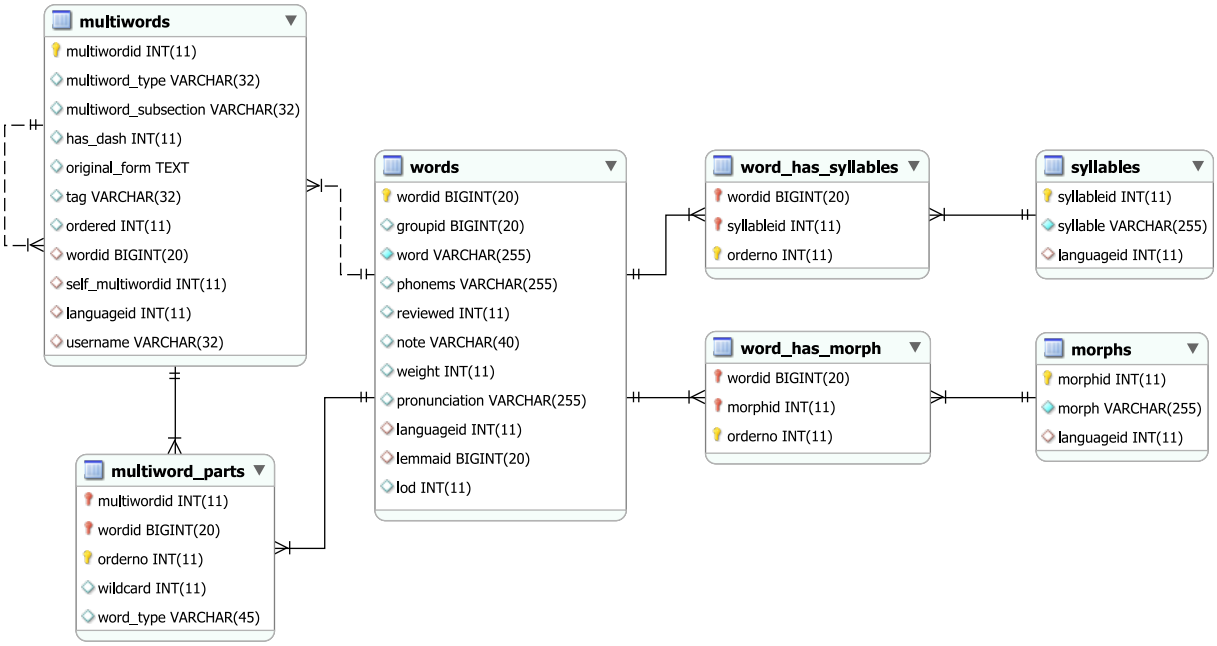


Figure 25: Database model of the LEX subsystem

The lexicon of all words, morphs, syllables and multiwords can be displayed and filtered based on a various criteria. For example, it is possible to easily display all nouns from some document, which start with letters 'ma', and are in a female gender in dative. A number of such combinations is huge since it involves all WOS/SOW features.

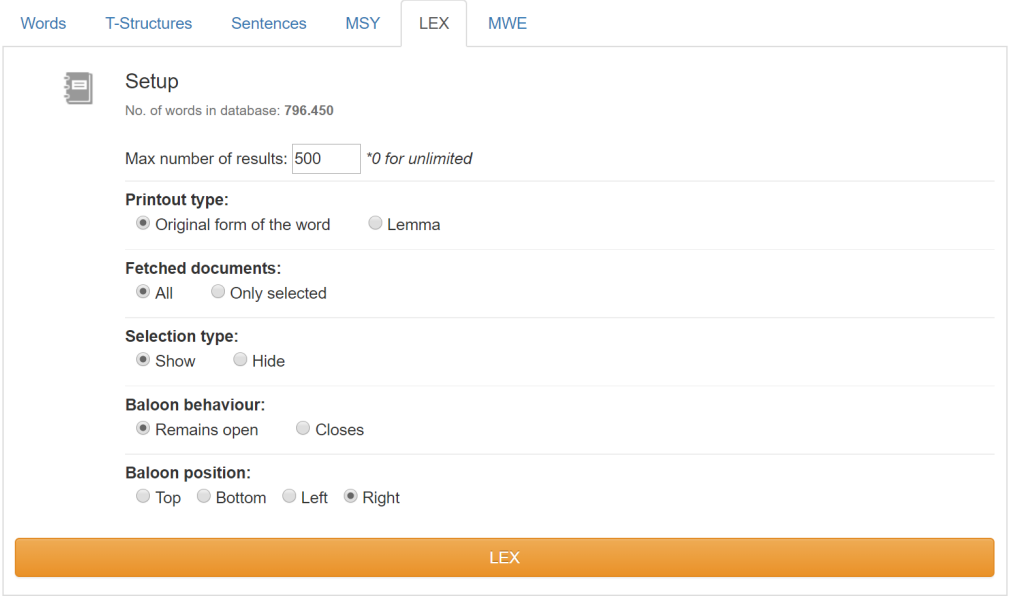


Figure 26: Screenshot of SSF's Lexicon setup screen

Before lexicon is displayed it is possible to set specific parameters which define a form of the output results (as in Figure 26):

- It is possible to define printout type to show either original form of the words or lemmas;
- If necessary, words in the output can be limited only to those which are related to some selected documents from the corpus or in another case the complete lexicon is displayed;
- Filtering of displayed words in the lexicon is possible based on WOS/SOW tags the word is tagged with. By default, all words are hidden in the output, and the user can click on filters and display only those words which meet filter criteria. If the selection type option is set to 'Hide', than all words are shown, and clicking on filters actually hides words from the output;
- Some words are tagged with WOS/SOW tags which along the tag itself also hold deeper information (e.g. definitions of words, list of synonyms, hypernyms etc.). That information is shown on click in a form of a balloon. By default, when the tag is clicked, and the balloon is opened it will stay opened as long as the user keeps clicking on it again. If the option of balloon behaviour is set to 'Closes', it will change its behaviour in a way that it automatically closes once the next balloon is opened;
- Balloons are shown next to the tag which is clicked. Default position of the balloon appearance is in the bottom of the tag. If it is necessary, balloon position option enables the user to set the position of balloon appearance to be on top, to the left or to the right from the clicked tag.

In the top of the lexicon setup screen, total number of words in the lexicon is displayed. Because of huge amount of words in the database it would be impractical to display them all on the same page, so the next parameter enables a user to limit the number of words in the output. By default it is set to 500, but the user can enter any number, or in a case when it is necessary to display the whole lexicon, the number value zero is used.

Once the lexicon is generated it looks as shown in Figure 27. In the central part is a list of words which meets search criteria. Left to the words are filters for WOS tags, and right to the words are filters for SOW tags. In the top of the page is a search box which is used when there is a need for quick lookup on a specific word. In Figure 27 it is shown the output of search for the word cro. *dijete* (eng. child). Along with the word, other useful information like lemma, accent and WOS/SOW tags are displayed. When

clicked on tags which hold deeper information, for example a tag with the definition of the word, the balloon with the definition pops up. Each word of the displayed definition (which exists in the lexicon) is linked to that particular word (in Figure 27, blue colored words are links), which results in increasing the semantic connectivity of words by at least one order of magnitude (about ten times more semantic relations).

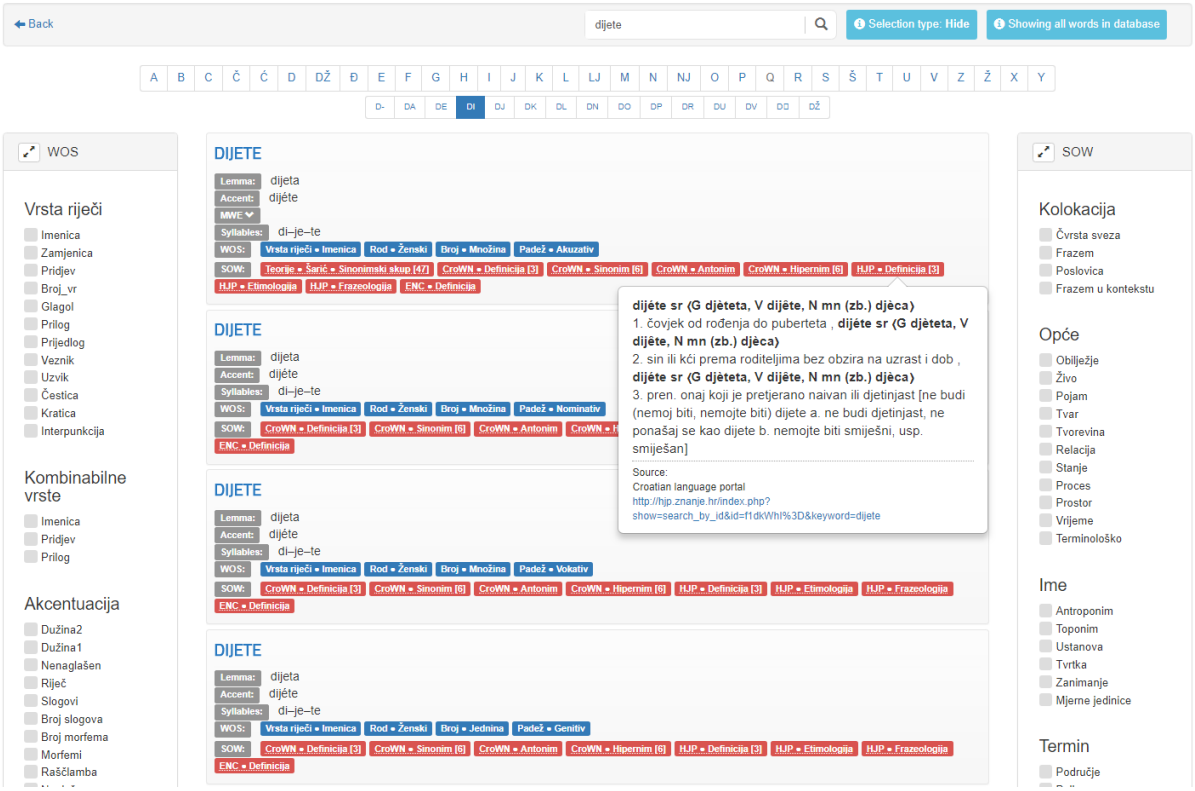


Figure 27: Screenshot of SSF's Lexicon output

Figure 28 shows the output of lexicon search for the word cro. *kraj* (eng. end). It is visible that the same word is displayed multiple times due to the different tags applied (e.g. in the first occurrence the word is in nominative, and in the second is in accusative). For tags that hold deeper information (e.g. antonyms) the popup shows list of words which are linked to their location within the lexicon). All external resources that are used within the SSF are properly credited and links to a original resource are displayed. In that way the original resource, which usually holds even more information than the SSF, is also promoted. This give an answer to research question **Q3** that the integration of external resources (e.g. lexicographic, encyclopedic, linguistic) is possible.

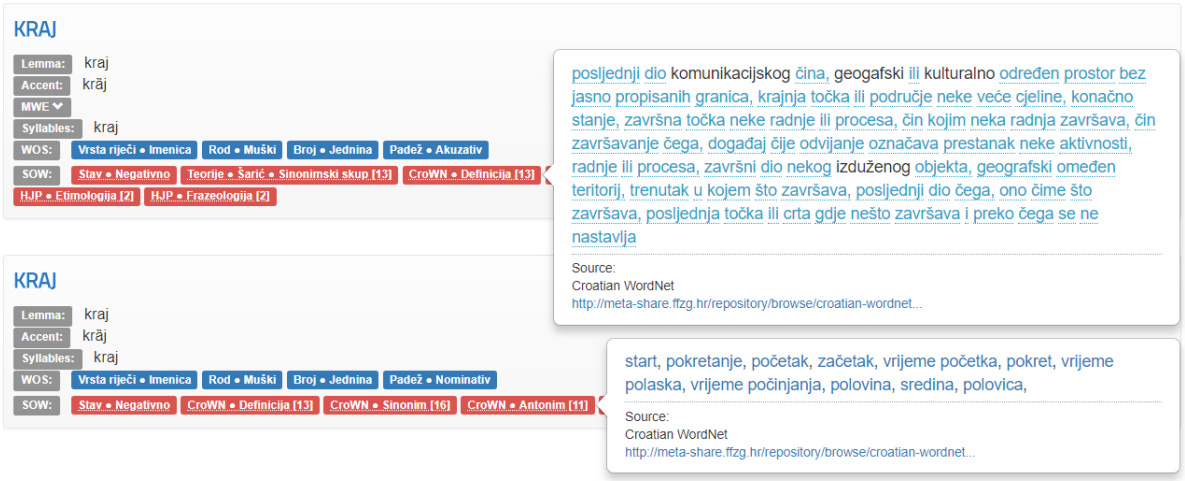


Figure 28: Screenshot of SSF's Lexicon tags

Signed in users additionally have options for lexicon administration:

- Adding new words to the lexicon;
- Adding new WOS/SOW tags to a words;
- Removing tags from words; and
- Setting a weight on a word (which is later used by the parser component in a process of corpus loading).

Along with manually editing lexicon entries the signed user can use morphological generator (as described in Section 4.1) and in a semi-automatic way generate all word forms for a given lemma, along with associated WOS tags. Validation of lexical data is conducted manually by linguistic experts as the part of the project *Croatian language in the global network cloud* (cro. Hrvatski jezik u mrežnom oblaku svjetskih jezika) at the Department of Mathematics at J. J. Strossmayer University of Osijek. The project was funded in 2018 by the Adris Foundation. Figure 29 shows special module for validation of lexical data within the SSF. The validator has insight into complete lexicon which was generated by the parser component in the stage of corpus loading. For each word he can manually assign WOS or SOW tags, as well as change lemma and morphs or syllables splitting. After the word is carefully checked, the validator confirms that the word is correct and as such it becomes a part of the SSF's global lexicon.

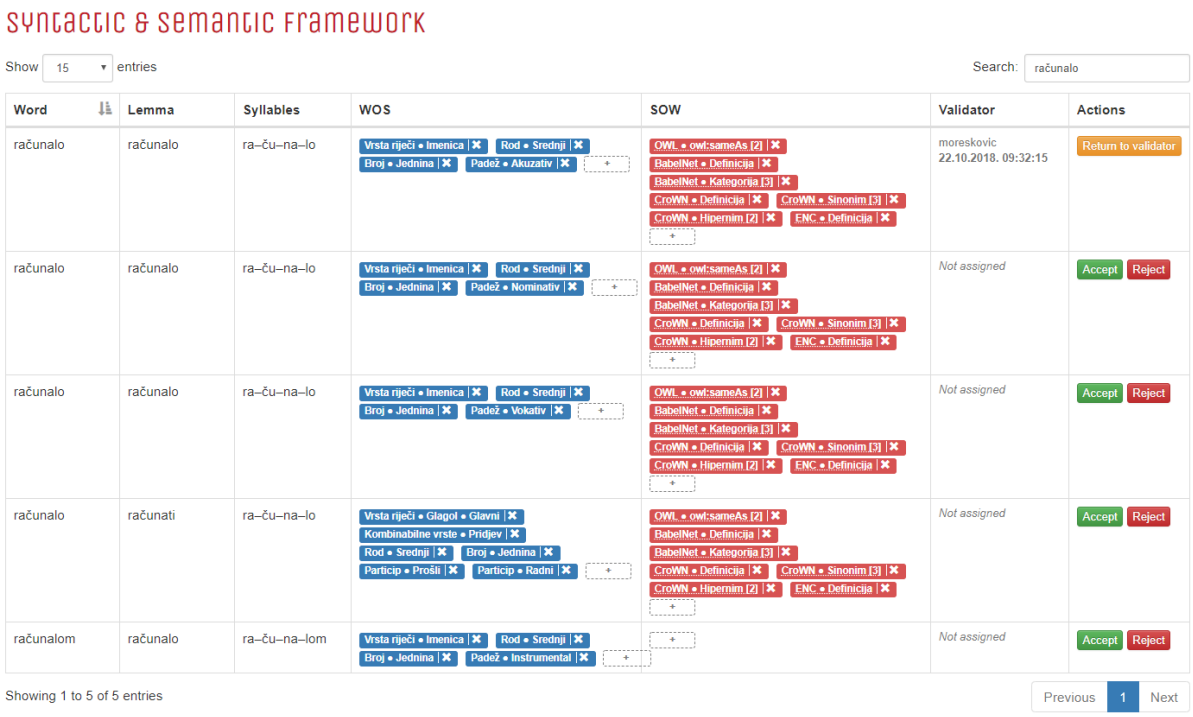


Figure 29: Lexical data validation

If the words are treated as atoms that form a certain language, then parts of a word (morphs and syllables) can be seen as subatomic particles that forms words. Morphs are relevant in morphological formation regardless of flexion or derivation, and syllables are relevant in phonetic sense. There are serious arguments to monitor relations of morphs and syllables through so-called syllabomorphemes [166]. The SSF supports all subatomic particles, but their formation, i.e. word degradation is solved only for syllables [113], and in a very small volume for morphs. Due to the well documented morphology of the Croatian language [112, 8], further research works on developing algorithms that could degrade words to morphs and syllables are expected. Similar to classical, the subatomic lexicon also has a setup screen which defines the output appearance (as in Figure 30).

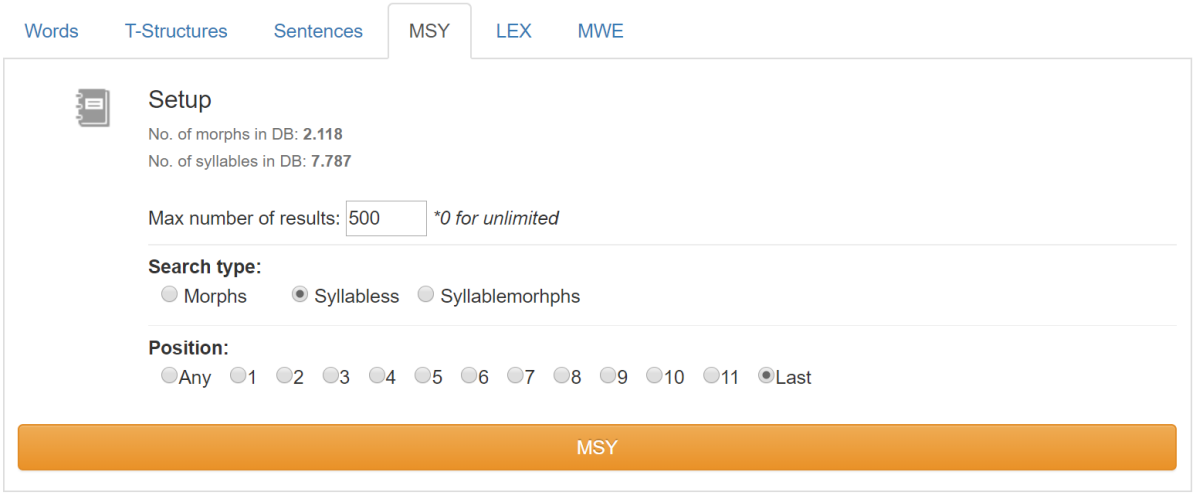


Figure 30: Screenshot of SSF's MSY Lexicon setup screen

In the top part of the screen, total number of morphs (2,118) and syllables (7,787) that are stored in the database is shown. Since the lexicon holds around 800,000 words (and the fact that the algorithm for syllables formation is completed), it is possible to conclude that Croatian language has around 8,000 syllables. In the case of morphs, there will probably be even more, once the algorithm for morphs formation is completed).

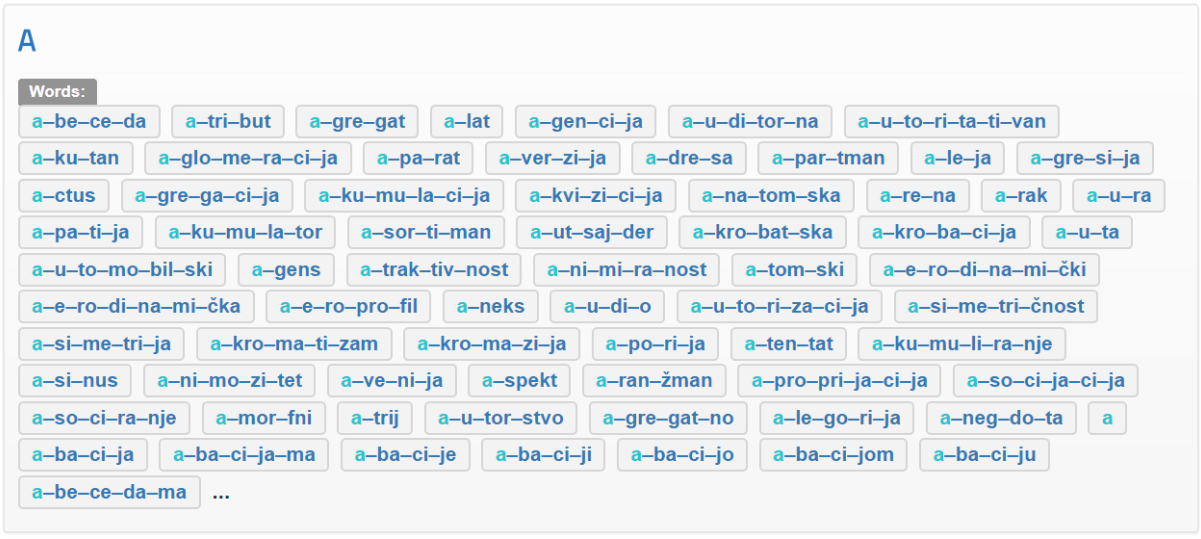


Figure 31: Screenshot of SSF's MSY Lexicon output for syllables

The lexicon of subatomic parts, and words which use them, is displayed in relation to the position of subatomic part within the word. If the user chooses position 1 in the setup screen the output will be ordered in ascending order of the first subatomic part (as in Figure 31). Since the longest word in the SSF's lexicon consists of 11 syllables that is the largest number for position setting in the setup screen. Similar to how molecules are formed from atoms, the multiword expressions are formed from words. They usually

hold different information type within (e.g. terminological: cro. *trgovački brod* (eng. merchant ship, cro. *ratni brod* (eng. war ship); sociological: cro. *labava carinska unija* (eng. loose customs union); metaphorical: cro. *sunce moje drago* (eng. my sweet sunshine); phrasematical, proverbial, etc. Having MWE lexicon is important for a language. The SSF also includes such 'molecular' lexicon, which is similar to already described lexicons of words and syllables/morphs.

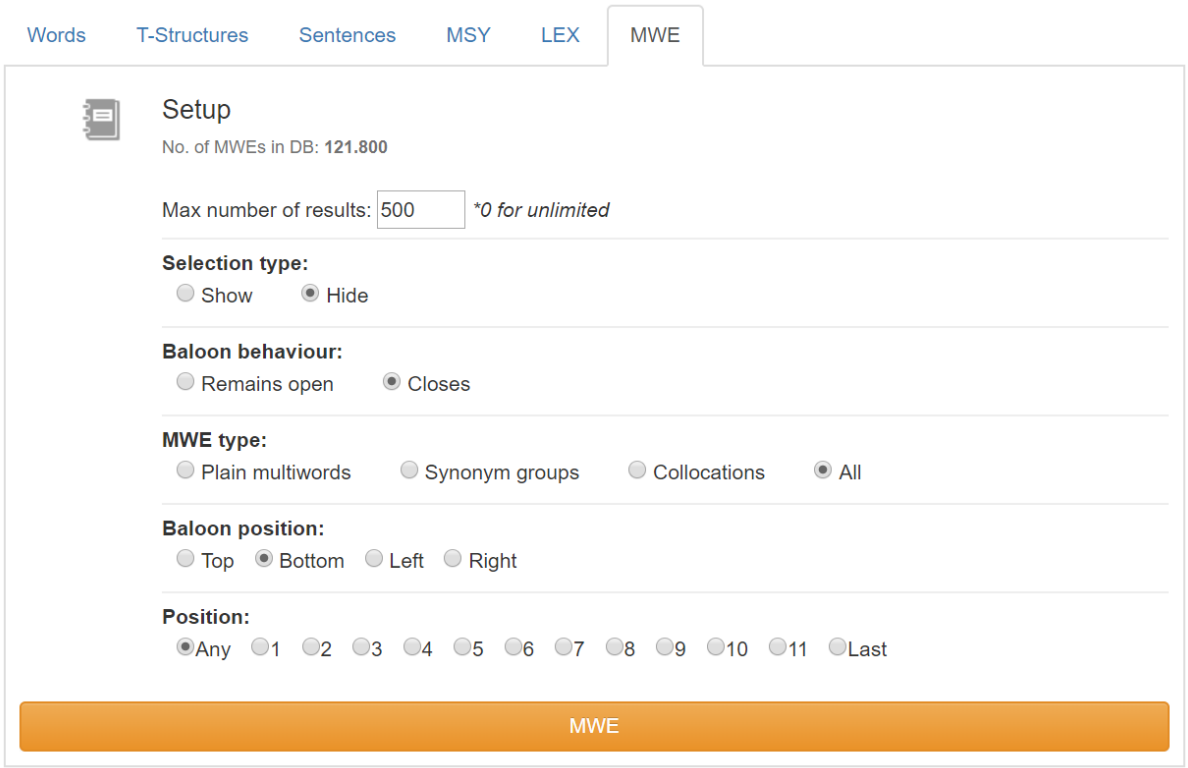


Figure 32: Screenshot of SSF's MWE Lexicon setup screen

Display settings are (as in Figure 32) also similar to previous lexicons. Currently there are over 120,000 multiword expressions which were automatically collected from external resources. Ordering of multiwords is by default in ascending order based on the words position within the multiword which can be additionally configured in the setup screen. Figure 33 show the output of multiwords lexicon which is limited to show only collocations. Near each multiword there is a list of words (atoms) from which the multiword is formed. Click on these words brings the popup balloon with WOS/SOW tags for each word.

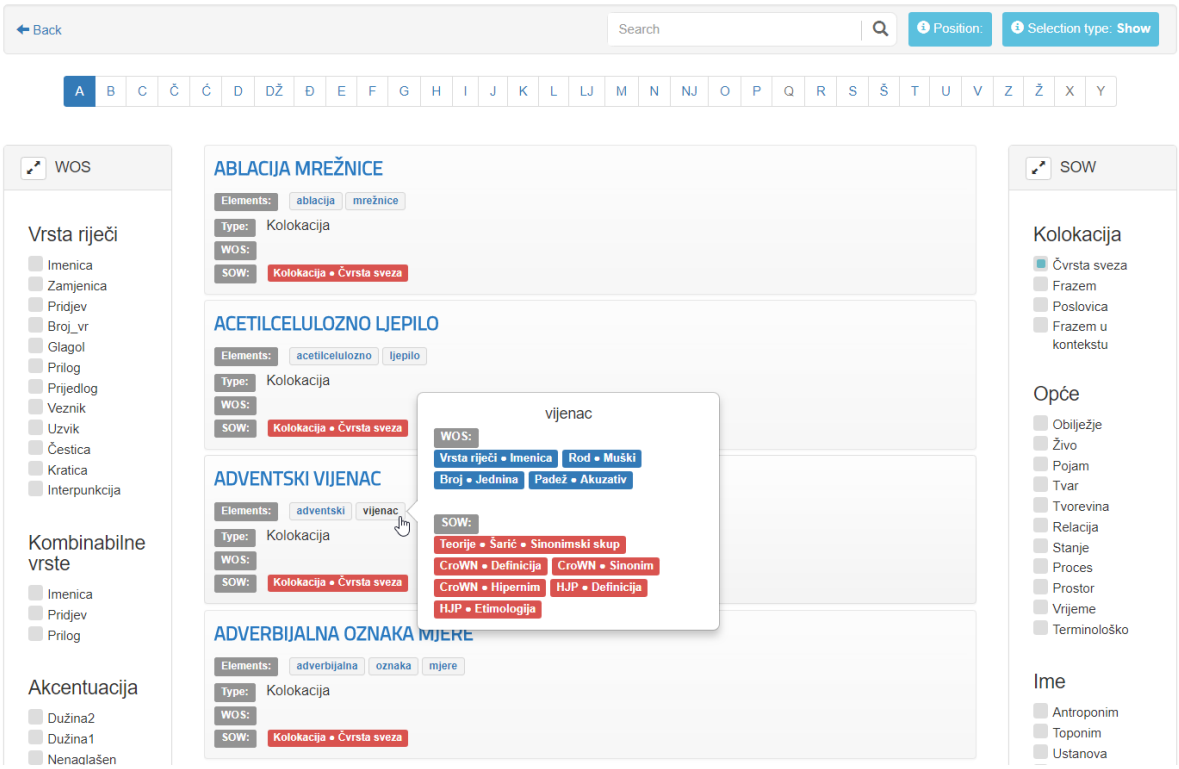


Figure 33: Screenshot of SSF's MWE Lexicon output

These three lexicons gives an answer to the hypothese **H1** because they cover all structures (e.g. qualia) of generative lexicon from J. Pustejovsky [135], and are briefly described in the next section.

4.3. Generative Lexicon requirements

A common question that might arise is the following: Is it possible to store, beside syntactic properties of words, also information about semantic properties and their relations to words environment within the lexicon? One of the answers was given by J. Pustejovsky in his Generative Lexicon (GL) [136]. It is an attempt to interpret the sentence: cro. "*Marko je počeo s novim poglavljem*" (eng. Marko has started a new chapter) by the machine, regardless of its ambiguity. Did Marko start writing a new chapter of his thesis or maybe he is only reading it and just reached a new chapter. Of course, its accurate understanding requires a broader context, pragmatics, in which the sentence exists, and also certain knowledge about each word in the sentence. It is not enough to know only the word class, but also its relation to other words - the so-called semantic roles that can be assigned to a particular word in the lexicon. So, the special focus will be on nouns and noun phrase structures in which modifiers plays an important role (e.g. adjectives and quantifiers), and especially to verbs which introduce functional and logical structure in the sentence, and pronouns which introduce additional properties (e.g. possessiveness

or indeterminacy), etc. The attempt is to make such a lexicon which will hold formal properties of each word that later on can be used in a semantic analysis of texts. In this sense each word is composed of complex concepts that can be decomposed in simpler notions. For example, the word cro. *stol* (eng. desk) is inanimate, concrete, having legs. The method which GL proposes is to concentrate on how a word meaning can be composed with other meanings and how it changes in different environments instead of concentrating on word decomposition. GL determines the meaning of a word by examining its contextual interpretations and introduces knowledge representation framework which renders rich vocabulary for lexical information. Unlike traditional lexicons, which are considered to be static, because words and their context are determined in advance, the GL attempts to offer a dynamic lexicon that is constantly evolving [137]. Computational resources which each lexical unit according to GL has are:

- **Lexical type structure**: defines an explicit type for the word positioned within a language type system;
- **Argument structure**: defines a number and types of arguments to a predicate;
- **Event structure**: defines the type of an event and sub-eventual structure it may have with subevents (state, process or transition); and
- **Qualia structure**: defines the essential attributes of objects, events, and relations for a lexical item.

While the 'event structure' is related to verbs (and their properties: time, mood, etc.), and 'argument structure' to verb's valences, the 'qualia structure' is focused on nouns and noun phrases with these four properties:

- **Formal**: the basic category which distinguishes the meaning of a word within a broader domain;
- **Constructive**: the relation between an object and its parts;
- **Telic**: the purpose or function of an object; and
- **Agentive**: the factors involved in the object's origins.

Generally, a lexical item in terms of GL can formally be expressed as:

$$\begin{bmatrix} \alpha & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG1} = x \\ \dots \end{bmatrix} \\ \text{EVENTSTR} = & \begin{bmatrix} \text{E1} = e_1 \\ \dots \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{CONST} = \textbf{what } x \textbf{ is made of} \\ \text{FORMAL} = \textbf{what } x \textbf{ is} \\ \text{TELIC} = \textbf{function of } x \\ \text{AGENTIVE} = \textbf{how } x \textbf{ came into being} \end{bmatrix} \end{bmatrix}$$

Where, for example, the event type structure may have: State: Nika loves her bicycle. Accomplishment: Iva built a castle. Achievement: Željka found a Euro coin on the floor. Process: John played in the kindergarten for an hour. Point: Jane knocked on the door (for 2 minutes).

Specifically, GL as a member of the lexical term 'book' or, for example, 'PhD thesis' means:

$$\begin{bmatrix} \textbf{book} & \\ \text{ARGSTR} = & \begin{bmatrix} \text{ARG1} = \texttt{y:information} \\ \text{ARG2} = \texttt{x:phys_obj} \end{bmatrix} \\ \text{QUALIA} = & \begin{bmatrix} \text{FORM} = \texttt{hold(x,y)} \\ \text{TELIC} = \texttt{read(e,w,x.y)} \\ \text{AGENT} = \texttt{write(e',w,x.y)} \end{bmatrix} \end{bmatrix}$$

Where arguments e and e' are derived from event type, and can mark tense (e.g. past) or mood (e.g. passive or active - Marko wrote a book). Also, they may have subevents related to logical conditions (if-then). For example, the event to *kill* can be decomposed to subevents in a following way:

$$\lambda y x e_1 e_2 [\text{act}(e_1, x, y) \wedge \neg \text{dead}(e_1, y) \wedge \text{dead}(e_2, y) \wedge e_1 < e_2]$$

For nouns which don't have any logical structure, the formal expression of properties can be event more complex. Some of these properties are listed in Table 12 which is made regarding Qualia theory. As shown in the table, nouns can have four roles: constructive (of which they are made), formal (describing an object within the domain), telic (purpose and function) and agentive (describing an object). For the previously mentioned example:

Marko started a new chapter, the word 'chapter' in the terms of GL can formally be expressed as:

$$\begin{bmatrix} \textbf{chapter} \\ \text{Constructive} : \textbf{part_}(y) \\ \text{Formal} : \textbf{book_}(y), \textbf{phd-thesis_}(y), \textbf{document_}(y) \\ \text{Telic} : \textbf{writing_}(x, [\textbf{reading_}(x, y)]) \\ \text{Agentive} : \textbf{artifact_}(y), \textbf{do_}(x, [\textbf{writing_}(x, y)]) \ \& \ \text{ingressive } \textbf{exists_}(y) \end{bmatrix}$$

These two multiple meanings of the words ('reading' and 'writing' a chapter), are derived from two different roles: one from 'telic', and the other from 'agentive'. Furthermore, it is necessary to observe the logical structure that is related to the verb, i.e. its functional arguments (described in Section 6.2).

$$\begin{bmatrix} \text{STARTS } \textbf{doing_} \ (\\ \quad [\textbf{Marko}(x), \{...\}], \\ \quad [\textbf{chapter}(y), \{..., \\ \quad\quad Q_T[\textbf{doing_}(x, [\textbf{reading_}(x, y)])], \\ \quad\quad Q_A[\textbf{doing_}(x, [\textbf{writing_}(x, y)])] \\ \quad \} \\ ])]) \end{bmatrix}$$

The logical structure of 'started' becomes 'STARTS doing_ (x,y)' where x denoted the noun (in this example, Marko, and y denotes the verb (which can be either 'reading' or 'writing' with its arguments). The arguments order is important, since any change in order will provide a completely different meaning. For example, have_(Marko, book) will mean that the Marko has the book, whereas have_(book, Marko) would mean that the Marko is with the book. Similarly, there is a functional behaviour with pronouns. The sentence "Marko saw himself" (in the mirror) require an argument of possessive pronoun (himself): see_(Marko, himself).

Table 12: Qualia theory roles [136]

Role	Description
Constructive Role	The relation between an object and its constituents, or proper parts (Material, Weight, Parts and component elements)
Formal Role	That which distinguishes the object within a larger domain (Orientation, Magnitude, Shape, Dimensionality, Color, Position)
Telic Role	Purpose and function of the object (Agent's purpose in performance of an act, a built-in function or aim that specifies certain activities)
Agentive Role	Factors involved in the origin or 'bringing about' of an object (Creator, Artifact, Natural kind, Causal chain)

Qualia relations are motivated by the purpose of nouns. For example, doors are for walking though, windows are for seeing through, bread is for eating, etc. Of course, the more complex semantic expressions are, another additional parameter such as: tense, aspect, modality and illocutionary force (e.g. questions and exclamations) is required. They act as operators on individual arguments, e.g. for the sentence: *Has Iva been crying?* - the expression could look like:

$$\langle_{IF} INT \langle_{TNS} PRES \langle_{ASP} PERFPROG \langle \text{do}'(\text{Iva}, [\text{cry}'(\text{Iva})]) \rangle \rangle \rangle \rangle$$

where TNS denotes tense, PRES denotes present, ASP denotes aspect, INT denotes intension and IF denotes illocutionary force. It is obvious that the bearing capacity of semantic relations is given by verbs and their arguments, which in GL are realized at three levels [168]:

1. 'Verb-specific' semantic roles, e.g. runner, killer, hearer, broken;
2. Thematic relations, which are generalizations across the verb-specific roles, e.g. agent, instrument, experiencer, theme, patient; and
3. Generalized semantic roles, the semantic macro-roles, actor and undergoer, which are generalizations across thematic relations. Actor is a generalization across agent, experiencer, instrument and other roles, while undergoer is a generalization subsuming patient, theme, recipient and other roles. Agent is the prototype for actor, and patient is the prototype for undergoer.

Their connections are shown in Figure 34. This information about arguments connectivity is the most important step in the machine processing of semantics, and needs to be done manually for all frameworks, as well as the SSF.

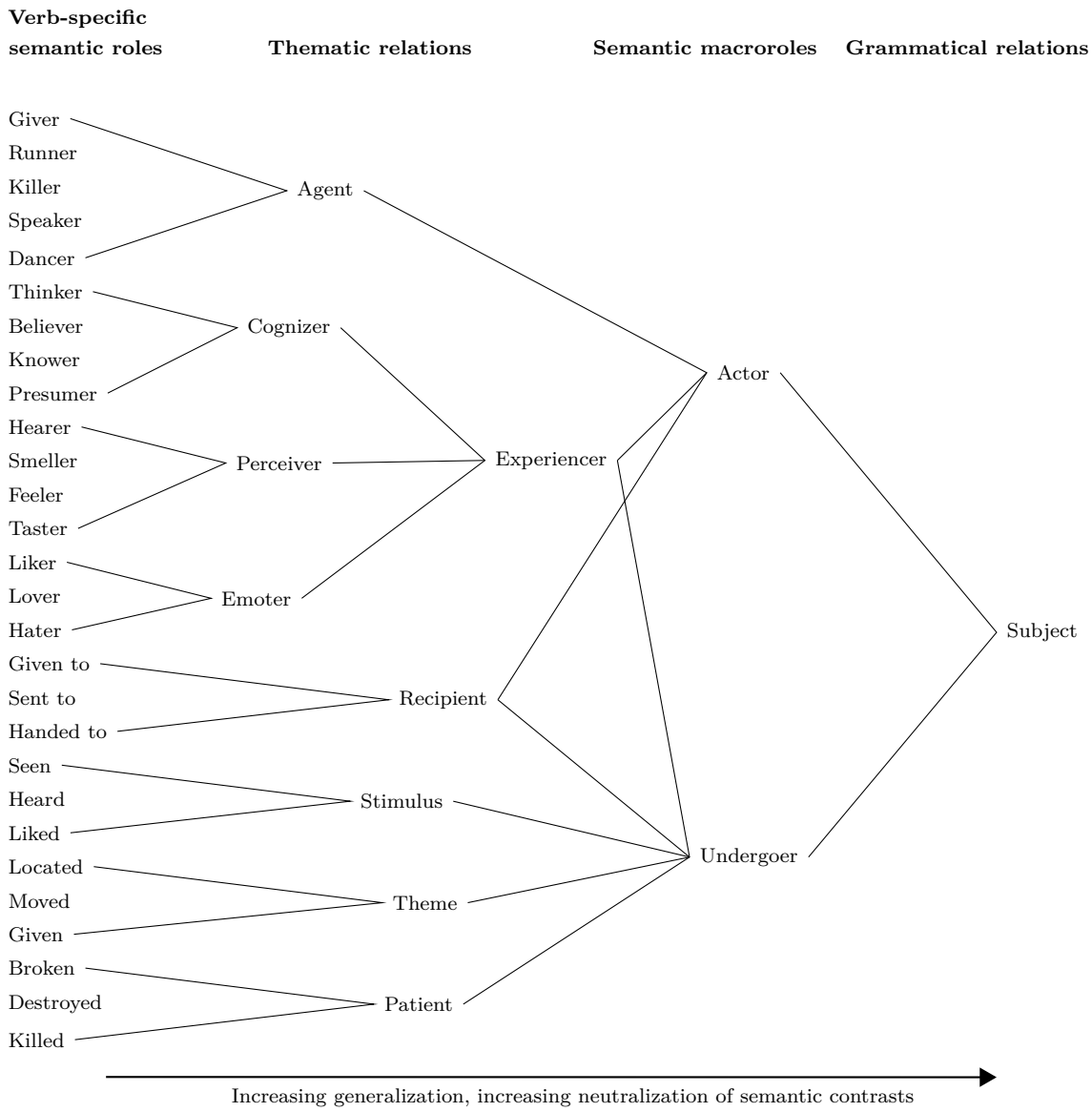


Figure 34: Connection of grammatical and semantic relations [167]

Thanks to the semantic domains, which can contain ordered elements in their information sets, the SSF has everything that is necessary for such usage, for all verb types. Since verbs describe action, an occurrence, or a state of being and can have one or more arguments, GL and the SSF must distinguish between multiple categories (as shown in Table 13 and Table 14). The table shows only a sample of verbs, but in the end it is possible to make the same for all other verbs (the Croatian language contains around 5,000 verbs).

Table 13: State verbs

	Single argument
State or condition	**broken**(patient)
Existence	**exists**(entity)
	Two arguments
Pure location	**be-LOC**(location, theme)
Perception	**hear**(perceiver, stimulus)
Cognition	**know**(cognizer, content)
Desire	**want**(wanter, desire)
Propositional Attitude	**consider**(judger, judgment)
Possession	**have**(possessor, possessed)
Internal Experience	**fell**(experiencer, sensation)
Emotion	**love**(emoter, target)
Attributive	**be**(attributer, attribute)
Identificational	**be**(identified, identity)
Specificational	**be**(variable, value)
Equational	**equate**(referent, referent)

Another type of verbs are Activity verbs, which tend to be single argument verbs, but there are some with two arguments (e.g. drink, eat, play, etc.).

Table 14: Activity verbs

	Single argument
Unspecificied action	**do**(effector, Ø)
Motion	**do**(mover, [**walk**(mover)])
Static motion	**do**(st-mover, [**spin**(st-mover)])
Light emission	**do**(l-emitter, [**shine**(l-emiter)])
Sound emission	**do**(s-emitter, [**gurgle**(s-emiter)])
	One or two arguments
Performance	**do**(performer, [**sing**(performer, (performance))])
Consumption	**do**(consumer, [**eat**(consumer, (consumed))])
Creation	**do**(creator, [**write**(creator, (creation))])
Direct perception	**do**(observer, [**hear**(observer, (stimulus))])
Use	**do**(user, [**use**(user, implement)])

Ever since theoretical assumptions have been defined until now, the GL in computational semantic systems has never been completely developed, but rather is used as a model for defining semantic information in computers. While Mulčuk's functions are developed completely for Russian and French, GL functions usually appear in modified versions. For example, for Croatian language, the valency dictionary CroVallex [131] was made, and e-Glava [16] which was inspired by the well-known valency dictionary e-VALBU [45]. In both cases, semantic roles of verbs with number of arguments they contain are listed. A multi-valency verb need not necessarily have the same semantic role when it comes with one or more arguments. The GL found application in the Role and Reference Grammar from Robert D. Van Valin [169]. In the end it must be mentioned, that the GL also has a lot of deniers, which explains its poor implementation in computer linguistic systems. There are also some authors which point out disadvantages of Generative Lexicon [140] such as: the lexicon does not need to be a structured module of grammar because it is not encapsulated, but it associates representations from radically different cognitive modules (conceptual, articulatory, formal). Lexical encyclopedic knowledge is of a piece with real world knowledge and does not give systematic compositional effects (the crucial distinguishing property of language). These disadvantages are solved in the SSF by integration of lexicographic knowledge and semantic domains which are derived from it. Domains in the SSF are briefly described in Section 6.3.

5. Syntax

The syntax studies rules on which sentences, and phrases are formed. Syntactic units are words, phrases (from one or more words) and sentences. The sentence is a phrase which contains a verb (predicate). There are many different theories which describe how phrases are composed into sentences, and among them there are three main paths: generative, functional and cognitive. While *generative* (Chomsky) [31] considers only formal organization of certain word types and their interrelationships (phrases), the *functional* gives more attention to words and their service (function) in the sentence (subject, object, predicate, etc.), and *cognitive* relies on semantic terms and their mutual interaction (agent, patient, topic, etc.). There are also theories that strive to merge these fundamental directions, such as Lexical Functional Syntax [22] which represents the bridge between formal and functional approaches [75].

5.1. Syntax model

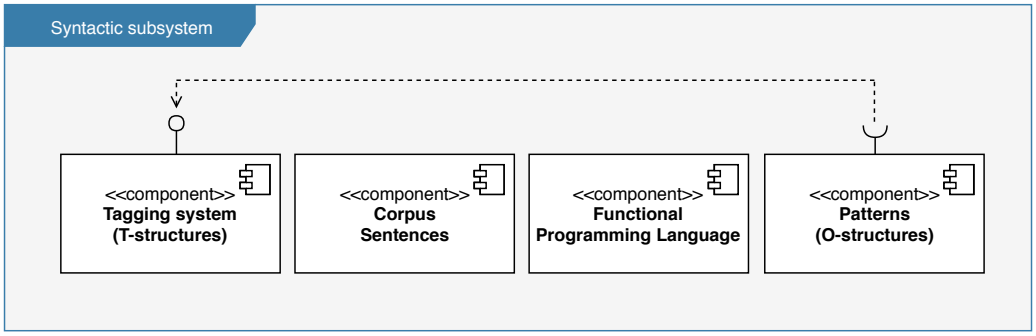


Figure 35: Conceptual model of the syntactic subsystem

Syntactic category or *syntactic class* is a collection of words and/or phrases of a language shared by a sizeable number of common features (characteristics). Classification is based on similar structure and distribution equivalence. In generative grammar, the syntactic category is represented by the node mark in the constituent treelike structure [78]. Main syntactic categories are all phrasal categories, like NP (noun phrase), VP (verb phrase), PP (preposition phrase) and syntactic categories which are used as a head of phrasal. Secondary categories are not merged to phrasal, e.g. questions with 'yes-no' answers or interpunction marks. Syntax is a grammar branch that studies the rules of merging lexical units into larger entities: phrases, statements (clauses) that have at least a predicate, explicit or implicit subject or express a proposition, then simple sentences, complex sentences, and finally text as a set of sentences and statements. There are two types of lexical relationships: paradigmatic (vertical) and syntagmatic (horizontal).

Vertical relationship connects words by its meaning and horizontal by its position in a sentence. For detection of horizontal relations syntactic patterns are necessary and for detection of vertical synonymous sets. For that purpose, syntactic patterns are developed, which can be generated and stored manually or semi-automatically by the user. These syntactic patterns enable better parsing of the input data. For the vertical (semantic) patterns in linguistic model, care is taken due to integration with CroWN [151] - Croatian version of WordNet, in its fundamental version (only 20% of information from PWN). To compensate for the lack of information in the model, the *Dictionary of synonyms of the Croatian language* [145] with over 12,000 lemmas and over 110,000 words in synonymous sets is also included.

Every word in a sentence serves a specific purpose. Based on grammar rules, sentence structure can be very complex but in terms of this thesis only basic parts will be discussed:

- Subject (S) - the word in a sentence which is closely related to the verb (predicate) and concurs with it in grammatical categories (gender, number);
- Predicate (P) - the base of the sentence (verb);
- Object (O) - words that give the result of S-P connectivity, object of predicate's activity or extension of information so that in the next sentence they can take the role of the subject. Traditional Croatian grammar differs between the real (direct) object (DO) and unreal (indirect) object (IO);
- Adverbial (A) - the word which gives an additional value to the subject or the object. There are several adverbials types (e.g. adverbials of place (AP), time (AT), etc.); and
- Complements (C) - words which extend meanings of S-P-O parts (e.g. predicate complement (PC) (cro. *išli su pjevajući*), attribute (AT) (cro. *osebujni stil*), apposition (AP) (cro. *starac Šimun*), etc.).

5.2. Word as a syntactic unit

Open word classes (nouns, adjectives, pronouns and numbers) and adverbs have complete content (lexical) and grammatical meaning, therefore they are called lexical words. Other four closed word types (conjunctions, exclamations, prepositions and particles) don't have any lexical meaning, but only grammatical. Nominal words belong to a type of open word classes which can be declined. In the SSF such words are tagged with WOS mark 'padež' (eng. *case*). Nouns are independent words which can serve as any

sentence part. They can be formed from verbs (e.g. cro. *Spavanje je zdravo*), adjectives (e.g. cro. *Mlada je lijepa*) or numbers (e.g. *Prva je zaspala*).

Examples of nouns as a word service

Augustin Ujević rodio se u Vrgorcu. Otac mu je bio učitelj. Ujević je hrvatsku
S AP S P S

književnost obogatio s pet zbirki pjesama.
DO IO A

Adjectives are dependent words usually in a role of an attribute or a part of (traditionally called) nominal predicate.

Examples of adjectives as a word service

Njegov osebujni stil nikoga ne ostavlja ravnodušnim. Život mu je bio neobičan
AT PC P

Pronouns have the same service as nouns and adjectives.

Examples of pronouns as a word service

On je umjetnik. Pokazali smo im sliku koja ih je oduševila.
S IO DO

Numbers are usually in a service of an attribute. They can also be a part of a nominal predicate (i.e. *floating object*) and sometimes even a subject.

Examples of numbers as a word service

Prvi govornik je malo oklijevao. On je prvi. Jedan je glavni broj.
S IO DO

The case (N - nominative, G - genitive, D - dative, A - accusative, V - vocative, L - locative, I - instrumental) is a syntactic and semantic category which is conditioned by the verb in the sentence. Nominative and vocative are standalone cases which act independently of other words in a sentence. Nominative serves as a subject (or traditionally a nominal predicate), and vocative is not a part of a sentence, therefore it is delimited by the comma. All other cases depend on other words in a sentence (usually on predicate). Such cases can have multiple services.

Service of a direct or indirect object (accusative and genitive)

Gledam nebo. Daj mi kruha.
A G

Service of an adverbial (applicable to all dependent cases)

Došli su do ruba doline. Vratili su se kući. Ušli su u stan.
G D A

Dugo nisu bili u gradu i nisu prošetali njegovim ulicama.
L S

Service of an attribute

Ušao je ravnatelj škole. Nisam imao smisla za glazbu. Pomogao mi je.
G A D

Service of a predicate part (rarely)

Osjećao se čovjekom.
I

Verbs in sentences serve as a predicate. They are a placeholders for other sentence parts by time, person, number and gender category. Due to the object they can be categorized as: transitive verbs (which are focused on a direct object, i.e. can have an object in accusative - e.g. cro. *brati cvijeće*, *gledati nebo*), intransitive verbs (which cannot have a direct object, but can have indirect i.e. object in some other case - e.g. cro. *pomagati prijatelju*, *razgovarati o jeziku*) and reflexive verbs (which are always followed by reflexive pronoun cro. *'se'* (e.g. cro. *umivati se*, *ljubiti se*, *bojati se*). In that case the subject and the object are the same person, and reflexive pronoun (cro. *sebe/se*) is in accusative. Based on their service in a sentence verbs can be categorized as:

- Infinitive - in some cases, when it gives the name of an action within the sentence; the subject (e.g. cro. *pušenje je štetno*); supplement to a modal verb (e.g. cro. *treba raditi*, *želim učiti*); supplement to a phrasal verb (e.g. cro. *početi graditi*, *prestati sanjati*); near the verbs which express feelings, thoughts, speech or will (e.g. cro. *voljeti pjevati*, *željeti putovati*) and rarely it can be a supplement to a nominal word (e.g. cro. *naše je poučavati*);

- Verbal adjective - can serve as an attribute (e.g. cro. *Otpalo lišće šušti pod nogama.*) or can stand alone in a service of optative (e.g. cro. *Živjeli!*); and

– Participle - can serve as a predicate complement (e.g. cro. *hodasmo mašući*).

Adverbs in a sentence usually express the adverbials (e.g. cro. *Govorila je glasno i brzo*). They can also be the supplement to other adverbs (e.g. cro. *Poznavali smo se jako dobro*) or adjectives (e.g. cro. *Ona ima jako lijep glas*). Rarely they have a reinforcement role (e.g. cro. *Bilo je baš zanimljivo*) or modification role (e.g. cro. *Naravno, on neće dobiti nagradu.*).

Along with verbs, prepositions define cases. They have no independent service, but they appear in propositional expressions as sentence members, as a part of an object (e.g. cro. *Sanjao je o putovanju*), adverbials (e.g. cro. *Prošetali smo uz more.*) or an unagreeable attribute (e.g. cro. *kula od karata*).

Conjunctions connect individual words within sentence members (e.g. cro. *Ivica i Marica*) or simple and complex sentences (e.g. cro. *Mislim i radim*).

Particles are words which take part in the modification of the sentence in a way that they make it either affirmative (e.g. cro. *Da, volim te*), negative (e.g. cro. *Ne razmišljam o prošlosti*), interrogative (e.g. cro. *Jesi li napisao zadaću?*) or imperative (e.g. cro. *Neka glumci uđu!*).

5.3. Sentences

Sentence is a statement which consists of subject, predicate, object and adverbial. According to the communication purpose sentences can be declarative, interrogative and exclamatory, which in a computational sense can be solved by introducing new WOS branch called *punctuation*. The predicate is a central part of the sentence and is also a placeholder for all other independent sentence parts (subject, object and adverbials) through predicate categories of person, number, time, tense and aspect. To express time (tense) the predicate can be used in an absolute or a relative way. An absolute way is used to express the tense: present (e.g. cro. *Pišem ovaj tekst*), perfect, pluperfect, aorist (e.g. cro. *On ne reće ništa*), imperfect (e.g. cro. *Jučer sam pisao*) and future tenses (e.g. cro. *Pisat ću ti.*). A relative way is used in expressing future (e.g. cro. *Sutra idemo na izlet.*), some recurring action (e.g. cro. *U svibnju cvjetaju ruže*), universality (e.g. cro. *Zemlja se okreće oko Sunca*) and relative future (e.g. cro. *Požuri, pobježe nam vlak!*). These examples cannot be automatically tagged with WOS tags but have to be marked with additional marks in multiword expressions or O-structures. The tense can also be expressed with adverbs and transgressives (e.g. cro. *Sutra je blagdan.*; *Napisavši zadaću, izjurio je iz kuće.*). To express the mood the predicate can be used in an indicative (e.g. cro. *Govorim*), conditional (e.g. cro. *Kad bi šutio, izgledao bi pametnije*), imperative (e.g. cro. *Šutite!*) or optative (e.g. cro. *Sretno ti bilo!*). There are three main types of predicate

based on the open word class they are expressed with. Verbal predicate is expressed with one verb which can have a simple and a complex form (e.g. simple form cro. *Ivan govori. Ivan povremeno nešto izgovori.*; complex form cro. *Ivan je stalno govorio. Ivan će govoriti na stručnom skupu. Kada Ivan bude govorio svi će šutjeti.*). Noun predicate is expressed with auxiliary verb *to be* when it is not possible to express it independently.

The second important part of a sentence is a subject which is used to express the topic of the sentence. It is always in nominative and is aligned with predicate in gender (when the predicate is complex verb form) and number. The subject can be expressed with nouns, pronouns, numbers and adjectives (e.g. noun cro. *Pogled na morsku pučinu odmara oči.*; pronoun cro. *Mi gledamo regatu jedrilica u daljini.*; number cro. *Jedna je naglo skrenula prema obali.* and adjective cro. *Pametan zna svoje prioritete.*). There are cases where sentences can exist without a subject (in cases when impersonal verb form is in the role of a predicate, e.g. cro. *Mislio sam da je lako upravljati tvrtkom.*)). Sometimes, the sentence can have more than one subject (e.g. cro. *Nika i Iva pjevaju u zboru*) because the predicate is in plural since the sentence was created by merging two sentences *Nika pjeva u zboru* and *Iva pjeva u zboru*. The subject in such sentences can be of different gender and number so they align with the predicate differently. Subjects of the neutral gender in singular are always aligned with the predicate of masculine gender in plural (e.g. cro. *Očarali su me nebo i more na otocima*), whereas subjects of different genders are aligned with the predicate of masculine gender in plural (or with the subject which is closer to the predicate).

The third important part of sentence is an object. It is a noun or a noun phrase governed by an active transitive verb or by a preposition. The object is never in the nominative and vocative case. There are two main types of objects: direct and indirect. Direct object is an addition to a predicate in accusative, and gives an answer to a question *whom?* and *what?* (e.g. cro. *pisati pismo*, *igrati košarku*...). Direct object can be expressed in Slavic genitive, in which genitive is replaced with accusative without changing the meaning and is characteristic for Slavic languages such as Croatian (so it is usually present in old art literature). Indirect object is an addition to the predicate whose place is opened by intransitive verbs, i.e. can answer to genitive, dative, locative or instrumental questions (e.g. genitive cro. *Sjećaš li se onog filma?*; dative cro. *Pomozi susjedu.*; locative cro. *Razmišljam o tebi.*; instrumental cro. *Služim se računalom.*). Exceptionally, indirect object can be in accusative only if it is near preposition (e.g. cro. *Mogu računati na Ivanu.*). Along with these three main sentence parts, there are also others like adverbials (of manner, cause, instrument, quantity, permission and condition), attributes and apposition.

An attribute is an addition to a noun, and most often is expressed as an adjective. Sometimes, it can be expressed by a pronoun or a number. It answers to questions *who?*,

whose?, *what?* and *how big?*. Apposition is such construction in which noun extends any other noun in a sentence and is aligned to it (e.g. cro. *grad Zagreb*). The nouns (regardless their function within the sentence) holds the place for attribute and apposition. A sentence can be only one word (e.g. cro. *'Upomoć!'*) with its incomplete part and verb (cro. *priskočite mi u pomoć*) or to understand it, it is necessary to know what was already said or written (whole discourse), for example cro. *'Pivo'* (eng. *'Beer'*) as a part of discourse, the answer to a question cro. *'Što želiš piti?'* (eng. *'What would you like to drink?'*). Every sentence conveys an action, an occurrence, or a state of being. Other sentence parts are add-ons that closely define it (like described in Table 8). These add-ons (in semantic analysis also known as *thematic roles*) are important in the process of extraction of lexical relation from sentence.

Sentences can be simple (when formed only from basic parts) or complex (formed from one or more simple sentences). Complex sentences can be dependent (subordinate clauses) or independent (main clauses). Subordinate clause cannot stand alone and based on the part it replaces in the main sentence it can be either a subject sentence, a predicate sentence, an object sentence, an adverbial sentence, etc. The best way to demonstrate the usage of O-structures and patterns is in the algorithm for the decomposition of independent sentences to its parts and recognizing dependent sentence types in cases where the sentence is made from the main clause and the subordinate clause. In this way all prerequisites for lexical information extraction are met.

Languages with the rich morphology usually abound in a rich syntax, too, whereas languages with the poor morphology tend to have more hierarchical (semantic) structure. There are many frameworks which are proposing scientific theory of the syntax grammar rules. The most popular frameworks, are based on the 'universal grammar' (UG) which is developed by Noam Chomsky [30]. The most prominent theories are:

- Generative grammar: algorithmic constituency aka 'phrase structure' relation (Noam Chomsky 1950) and its improved versions:
  - Transformational grammar (1960s),
  - Generalized phrase structure grammar (late 1970s),
  - Government and binding theory (1980s),
  - Head-driven phrase structure grammar (1985),
  - X-bar theory (1990),
  - Minimalist program-based grammar (1993).
- Dependency grammar (DG): dependency relation [159];

– Functional grammar (FG): Lexical FG (LFG) and usage-oriented (behaviorist);

– Cognitive grammar (CG) / Cognitive linguistics ; and

– Stochastic grammar (SG): probabilistic.

According to Chomsky, Universal grammar (UG) defines what language must have for successful acquisition:

– A set of features;

– Principles for assembling features into lexical items; and

– Operations that apply successively to form syntactic objects of greater complexity.

The computational system of a language, integrates lexical information and forms linguistic expressions at the interfaces where language interacts with other cognitive systems. The linguistic theory studies properties of such computational systems. The essential grammar goal is to render explicit language rules. In generative grammar language is a huge set of sentences, while grammar is a set of formal rules for generation of such sentences. Grammar rules when applied on lexical units result in meaningful sentences of a language. Such sentences production is recursive and formally correct but doesn't correspond to a conceptual structure. Another approach is to use the more abstract grammatical functions (e.g. subject, object, predicate) which are widely shared among languages. For example, sentence cro. *Štunja je zlato* and *Šutjeti je zlato* are identical at the conceptual level, but are different at grammatical level. In the first sentence, the subject (cro. *šutnja*) is a noun, whereas in the second sentence subject (cro. *šutjeti*) is verb; this means that the subject may be mapped to a noun or a verb. The fundamental element of such conceptual structure is the predicate which is linked to one or more arguments. The predicate contains verb which is linked to other sentence units. Number of links is called verb's valency. In Croatian language there are one, two and three valency verbs. These two approaches can be tested within the SSF NLF module as shown in Figure 36 and Figure 37, which show possibilities of usage of different grammar in the SSF.

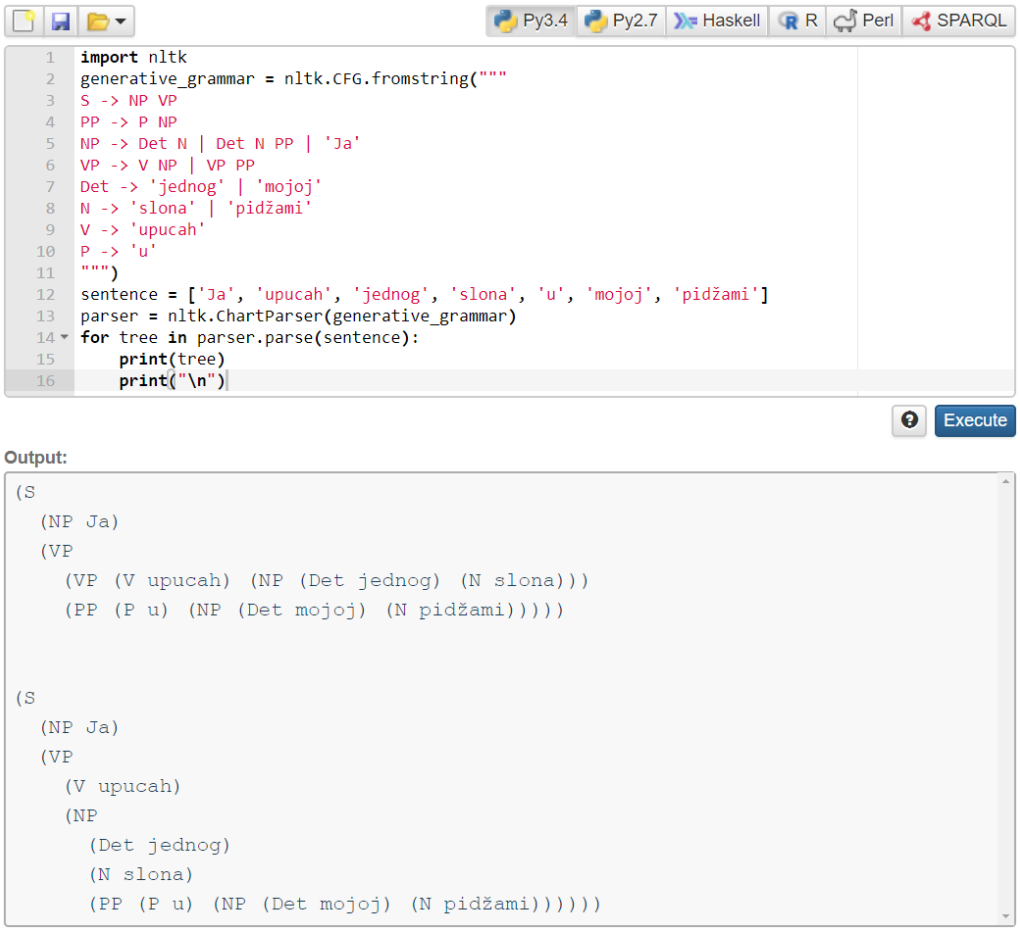


Figure 36: Generative grammar parsing in the SSF

Figure 36 shows the usage of generative grammar, where the well-known sentence cro. *Ja upucah jednog slona u mojoj pidžami* (eng. I shoot an elephant in my pajamas) is parsed with grammatical rules and lexical units of that sentence. The result of parsing are two syntactically and conceptually different sentences.



Figure 37: Dependency grammar parsing in the SSF

Figure 37 shows the same sentence parsed with dependency grammar which is based on S-P-O grammatical functions and gives a similar output as generative grammar. This shows an implementation of LFG and DG in the deterministic model of natural language, and partly confirms hypothesis **H1**.

To demonstrate how SSF can be used to extract the main clause (MC) and the subordinate clause (SC) two examples will be described. In the first example, the O-structures are used to split the sentences into MC and SC. As shown in Figure 38, the sentence cro.*'Tko traži prijatelja bez mane, ostat će bez prijatelja'* is split in two parts. Special type of O-structures which splits the sentence in two groups while respecting the rule that each part must contain the word which is tagged with WOS tag ID 5 (Verb) and have the comma (,) in between, formed the following regular expression:

```
(.+\[w:5\].+),(.+\[w:5\].+)
```

which, when matched against an enriched version of the sentence:

```
Tko␣traži[w:5]␣prijatelja␣bez␣mane␣,␣ostat[w:5]␣će␣bez␣prijatelja.
```

resulted in a list of two elements (MC and SC):

```
['Tko traži prijatelja bez mane', 'ostat će bez prijatelja']
```

In the same way other patterns (O-structures) for splitting more complex sentences can be developed. There are several differences between grammars, Težak and Babić [160] have cro. *poredbene* (eng. comparative) sentences, and do not have cro. *apozocijske* (eng. appositional), whereas Bičanić et al. [14] have opposite. Traditional Croatian grammar distinguishes these types of clauses: cro. *apozicijska* (eng. appositional), cro. *atributna* (eng. attribute), cro. *dopusna* (eng. permissible), cro. *mjesna* (eng. place), cro. *namjerna* (eng. intentional), cro. *načinska* (eng. modal), cro. *objektna* (eng. object), cro. *poredbena* (eng. compareable), cro. *posljedična* (eng. consequential), cro. *predikatna* (eng. predicative), cro. *subjektna* (eng. subjective), cro. *uvjetna* (eng. conditional), cro. *uzročna* (eng. causative) and cro. *vremenska* (eng. time). Possible pattern variants for such sentence types are shown in Table 15. Although, used in detection of compound sentences, such patterns have much more important role. Their strength is that they can take care of word ordering within the sentence and at the same time avoid problems with multiword expressions (collocations, idioms, terminology expressions, etc.). In this sense it is logical to think of permanent storage of such patterns and building of a special lexicon type (i.e. lexicon of patterns) which would have twofold purpose: a) it will be used in parsing procedure when corpora is processed for the first time, and b) will be used in syntactic and semantic researches.

Table 15: Syntactical patterns for decomposition of complex sentences

Pattern name	Variant	REGEX
apozicijska	1	(^.+\[w:.*2.*\].*,)(.+\[w:136\].+\[w:5\].+,) (.+\[w:5\].+)
apozicijska	2	(^.+\[w:5\].+\[w:2\])((?:gdje|što|koji|da.+\[w:136\]).+\[w:5\].+)
apozicijska	3	(^.+\[w:5\].+\[w:2\].*,) ((?:gdje|što|koji.+\[w:136\]).+\[w:.*5.*\].+)
atributna	1	(^.+\[w:.*5.*\].+\[w:.*2.*\]) ((?:gdje|što.+\[w:136\]).+\[w:.*5.*\].*)
atributna	2	(^.+\[w:2\] ,) (.+\[w:136\].+\[w:5\].+,) (.+\[w:5\].+)
dopusna	1	(^.+\[w:5\].*) ((?:iako|unatoč|premda|mada|makar).+\[w:5\].+)
dopusna	2	(.*(?:iako|unatoč|premda|mada|makar).*\[w:5\].+,) (.*\[w:5\].*)
mjesna	1	(^.+\[w:5\].*) ((?:kuda|kamo|gdje|odakle).+\[w:5\].+)
mjesna	2	(^(?:kuda|kamo|gdje|odakle).+\[w:5\] ,) (.+\[w:5\].+)
namjerna	1	(^.+\[w:5\].*) ((?:da bi|ne bi\[w:5\] li|da).+\[w:5\].+)
namjerna	2	(^.+\[w:5\].+\[w:2\]) (da.+\[w:5\].+)
namjerna	3	((?:da bi|ne bi\[w:5\] li|da).+\[w:5\].+,) (.+\[w:5\].*)
načinska	1	(^.+\[w:5\].*) ((?:kao što|kao da|kako).+\[w:5\].+)
objektna	1	(^.+\[w:5\]) ((?:što|da|kako) .+\[w:5\].+)
objektna	2	(^.+\[w:5\].+\[w:131\]) (da .+\[w:5\].+)
objektna	3	(^.+se.*\[w:5\]) (.+\[w:5\].+)
poredbena	1	(^.+\[w:5\].+) ((?:nego što|nego da).+\[w:5\].+)
poredbena	2	(^(?:što).+\[w:5\].+,) (.+\[w:5\].+)
poredbena	3	(^.+\[w:5\].+\[w:4\]) (nego.+\[w:5\].+)
posljedična	1	(^.+\[w:5\].+\[w:4\]) (da .+\[w:5\].+)
posljedična	2	(^.+\[w:5\].+\[w:2\].*) (tako da .+\[w:5\].+)
posljedična	3	(.*toliko.*\[w:5\].*?) ((?:tako|da).*)
posljedična	4	(^.+\[w:5\].*) (tako da .+\[w:5\].+)
predikatna	1	(^.+\[w:.*152.*\].*) ((?:kad|što|koji|da|.+\[w:136\]).+\[w:.*5.*\].+)
subjektna	1	(^S+\[w:136\].+\[w:5\].+),(.+\[w:5\].+)
subjektna	2	(^.+\[w:.*183\].+)(što.+\[w:5\].+)
subjektna	3	(^.+\[w:5\].+se)((?:kako|da) .+\[w:5\].+)
subjektna	4	(^.+\[w:152\]) (tko.+\[w:5\].+)
uvjetna	1	(^.+\[w:5\].+) ((?:ako|kad|da).+\[w:5\].+)
uvjetna	2	(^.*(?:ako|kad|kada|li|ukoliko|da).+\[w:5\].+,) (.+\[w:5\].+)
uvjetna	3	(^.*\[w:5\].+(?:ako|kad|kada|li|ukoliko|da).+,) (.+\[w:5\].+)
uzročna	1	(^.+\[w:5\].+) ((?:jer|zato što|zbog toga što) .+\[w:5\].+)
uzročna	2	(^budući\[w:5\] da.+\[w:5\].+,) (.+\[w:5\].+)
vremenska	1	(^.+\[w:5\].+)((?:dok|kad|prije nego|kada|čim).+\[w:5\].+)
vremenska	2	(^.*(?:dok|kad|prije nego|kada|čim).+\[w:5\].+,) (.+\[w:5\].+)

The Figure 38 shows the output of the `SplitSentences()` function in the NLF (described in Appendix D). In this example the function is called with only one parameter (sentence which needs to bi decomposed), but it can also accept additional parameter called `allmatches` which is a boolean type and controls whether all possible matches should be shown.



Figure 38: Sentence splitting to MC and SC using O-structures

The second example is also implemented within the SSF but relies on the NLTK module as backend of MC/SC extraction which is independent of the O-structures and has its own context free grammar rules which are built manually. For example, the rule for NP (in singular or plural), marked with NUM would look like this:

```
NP[NUM=?n] → N[NUM=?n] | N[NUM=?n] PP | ADV N[NUM=?n] | PRO | PRO N[NUM=?n]
             | ADV | PRO PP | ADV PRO
```

Where PP denotes Propositional Phrase, N denotes a Noun, ADJ denotes an Adverb which can be replaced by PRO which denotes the Pronoun. Depending on the syntax needs, these rules can become very complex which in the end may result in the longer processing time. For Verbal Phrase the syntax rules may look like this:

```
VP[NUM=?n] → V[NUM=?n] NP | V[NUM=?n] NP PP | V[NUM=?n] | ADV V[NUM=?n]
```

If categories in square brackets are still unknown, the following rules would explain them:

```
MC → NP[NUM=?n] VP[NUM=?n] | VP
SC → Conj NP[NUM=?n] VP[NUM=?n] | Conj VP
```

Where the `Conj` denotes the conjunction between the main and the subordinate clause. Since VP and NP rules both have the same content within the square brackets which means they need to be in the same number. For example, grammatically correct sentence for this category would be *'Lovac je došao kući'*, while the example of an incorrect sentence would be *'Lovac su došli kući'*.

The verbal phrase is a sequence of words that surround the verb. Since verbs can have different tense, person or number, and even can be negated, the production rules must take this fact into account. This is done by using categories. For example, the production rule for the verb cro. *ići* in the first person of Present tense, singular is:

```
V[TENSE=pres, NUM=sg, PER=1, NEG=0] →"idem"
```

And for the given Present, the negated version can be obtained by the following rule:

```
V[NEG=1] →"ne" V[NEG=0]
```

Since the number of words in a lexicon is huge, the rules should be as much as possible generalized. For example, the rules for Perfect tense could look like this:

```
V[TENSE=perf, NUM=?a, PER = ?b] →BE[NUM=?a, PER = ?b]
                                  V[TENSE=perf, NUM=?a, PER = ?b]
                                  | V[TENSE=perf, NUM=?a, PER = ?b]
                                  BE[NUM=?a, PER = ?b]
```

Where rules for Perfect tense looks like this:

```
V[TENSE=perf, NUM=sg, PER = 1] →"ostavio" | "ostavila"
```

and `BE` represents the conjugation of the present of the auxiliary verb '*to be*' (in which, for simplicity, the negation is included):

```
BE[NUM=sg, PER=1] →"sam" | "nisam"
BE[NUM=sg, PER=2] →"si" | "nisi"
...
```

The Future tense can be defined as:

```
V[TENSE=futur, NUM=?a, PER=?b] →WILL[NUM=?a, PER=?b] INF[-SHORT]
                                  | INF WILL[NUM=?a, PER=?b]
```

where `INF` denotes full and short infinitive form, which have rules in the following form:

```
INF[-SHORT] →"spavati"
```

```
INF[+SHORT] →"spavat"
```

and WILL present of an auxiliary verb cro. *htjeti* which is formed in following way:

```
WILL[NUM=sg, PER=1] →"ću"
WILL[NUM=sg, PER=2] →"ćeš"
...
```

To demonstrate how these rules work in real life, the sample Python code is shown in Figure 39.

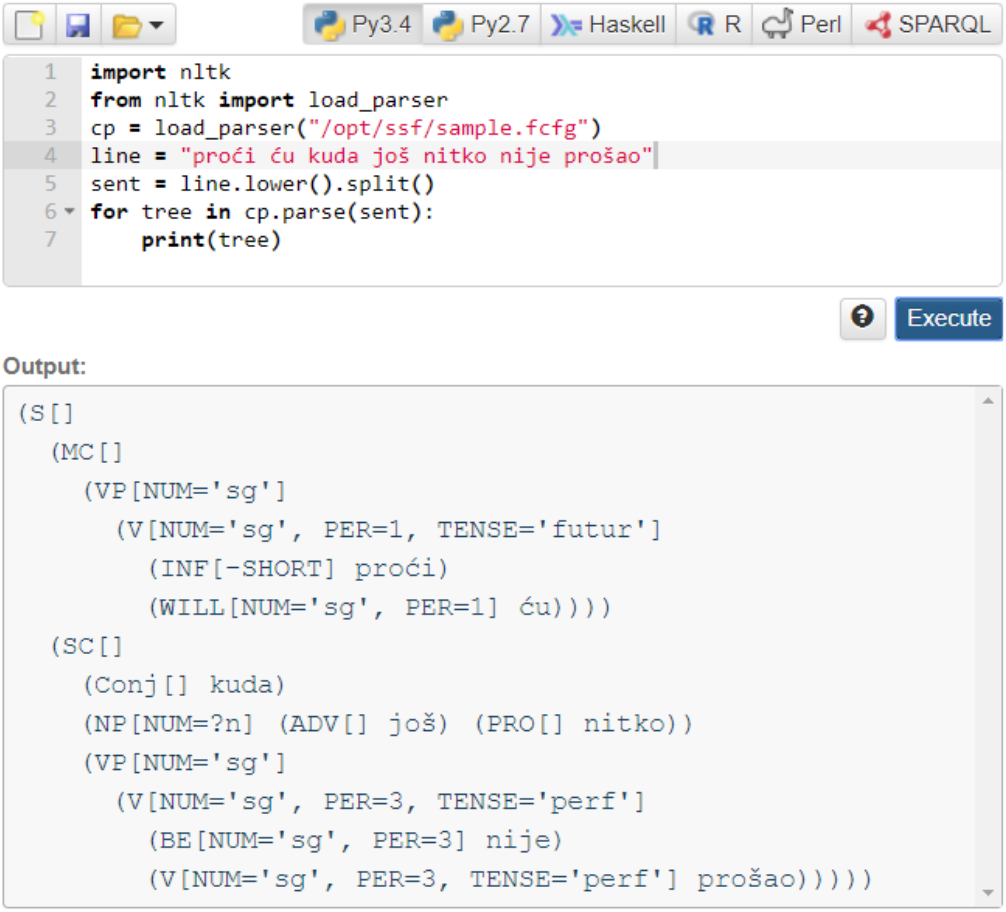


Figure 39: Sentence splitting to MC and SC using NLTK

Result from Figure 39 can also be displayed in a tree like structure as shown in Figure 40.

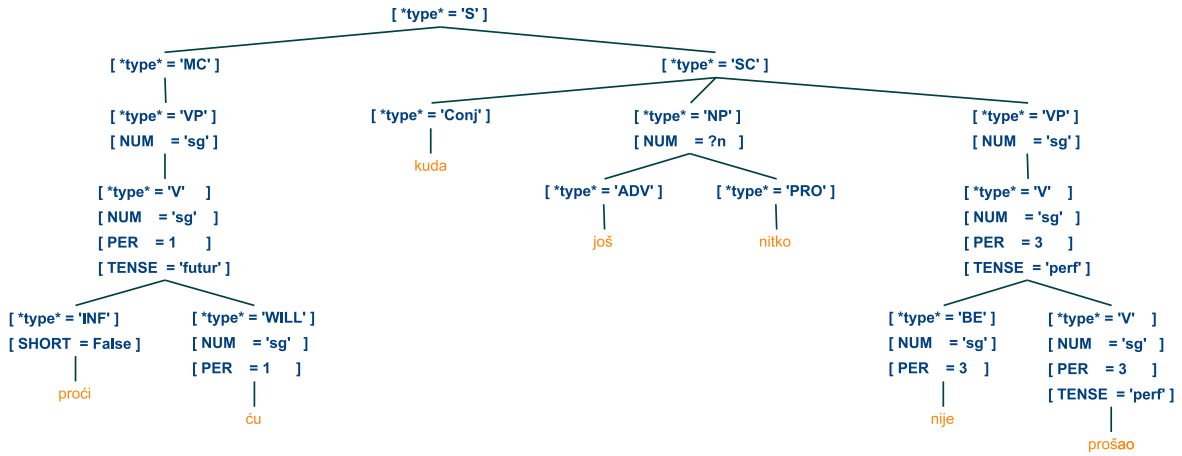


Figure 40: Tree representation of the split sentence

5.4. Natural Language Functions

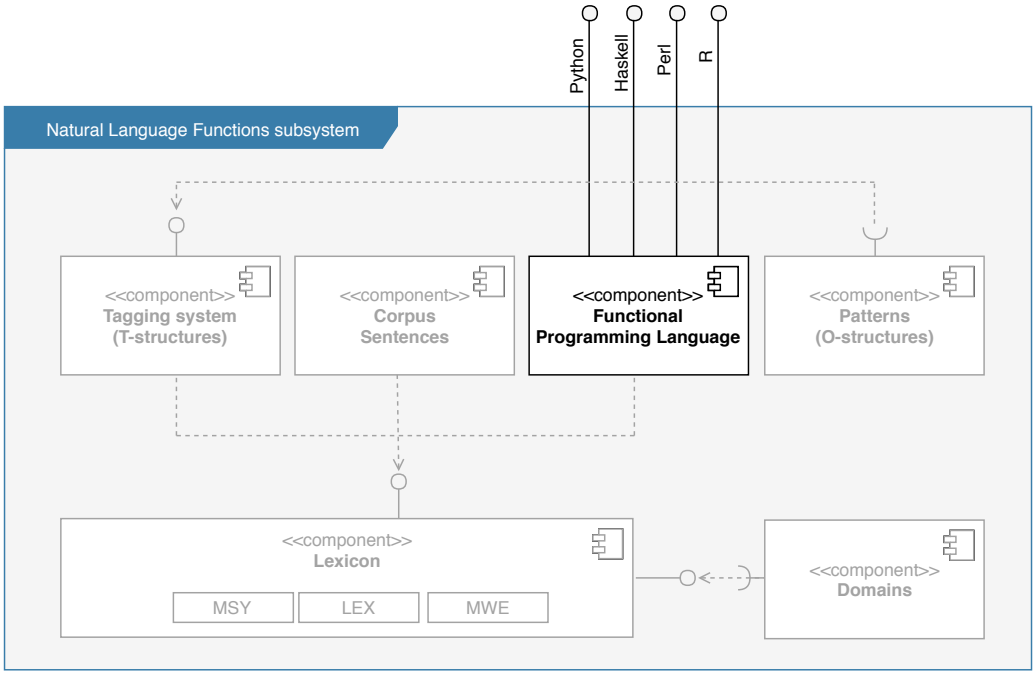


Figure 41: Conceptual model of the Natural Language Functions subsystem

One of the key features of the Syntactic and Semantic Framework are Natural Language Functions (NLF) which integrate some of the most powerful programming languages (e.g. Python, Haskell, Perl and statistics tool R) and the SSF. The NLF can be executed either through the API (described in Section 7.5) or using the GUI editor within the SSF interface (as shown in Figure 42).

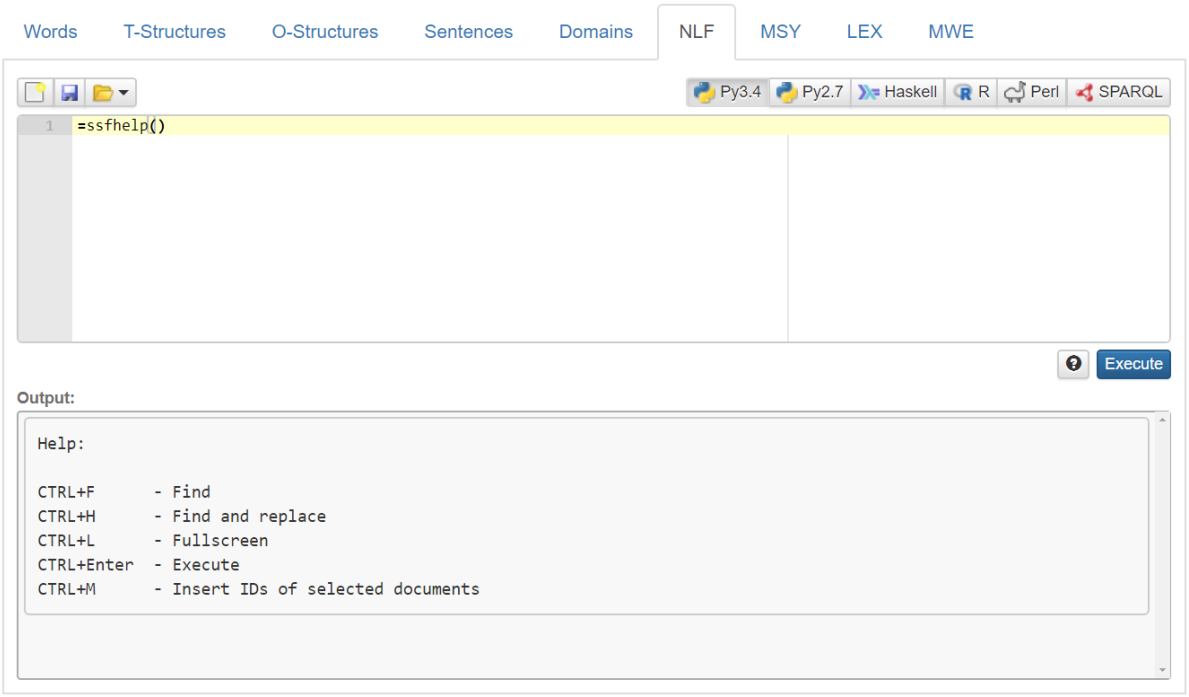


Figure 42: The SSF's Interface for the Natural Language Functions execution

The interface is divided in two main parts. The first part is an editor where a user can select which programming language will be used for executing scripts written in the editor below. After clicking the button 'Execute', the output is shown at the bottom of the screen. It is possible to store all scripts for a later usage by clicking on a 'Save' icon, or to open the previously prepared scripts by clicking on the 'Open' icon. For new users, who want to see SSF's specific functions there is a help function called `ssfhelp()`, which will display all possible functions along with their parameters, and usage examples. An additional strength to the NLF gives possibility to use all other linguistic frameworks within the SSF (e.g. Natural Language Toolkit, Scikit-learn, Pandas, CliPS, LanguageR, etc.).

Since Python has a rich repository of modules which deals with natural language processing (such as NLTK library) and is currently dominant in a field of computational linguistics, it was the first programming language that was implemented within the SSF. In Appendix D an overview of all NLF functions is given with a brief description of their definition, parameters and usage examples. Functions are classified in five distinct categories (functions for WOS/SOW tagging, Morphology based functions, functions for Syntax processing, Semantic functions and Statistics functions). For each function the category to which it belongs is written next to the function name.

5.5. Regular expressions

A regular expression (REGEX) is a sequence of characters that define a pattern used for finding (extracting) information from textual documents. The idea of regular expressions emerged in the 1950s when the American mathematician Stephen Cole Kleene formalized the description of a regular language. Few years later Ken Thompson implemented regular expressions in his QED (Quick Editor), and Unix editor `ed`. Since the 1980s, many different syntaxes for writing regular expressions appeared. The most popular among them are the POSIX standard and the Perl syntax (PCRE) [105].

Two main characteristics of regular expressions are: literals and a set of metacharacters that form regular expression (`.^$*+?{}[]\|()`). Literals are the simplest regular expressions. The match will be found whenever the literal is found. For example, if the regular expression `/fox/` is applied over the sentence `The quick brown fox jumps over the lazy dog`, one match will occur. However, if the regular expression `/be/` is applied over the sentence `To be or not to be`, two substrings will be matched. To use the real power of regular expressions, literals and metacharacters must be used in combination. Each metacharacter has its own meaning and usage, as shown in Table 16.

Table 16: Metacharacters in regular expressions

Metacharacter	Description
`[]`	Any character inside `[]`. If the metacharacter is found in `[]` it is treated as a literal.
`^`	Beginning of the text. If the set of literals within `[]` starts with `^`, it denotates the set complement.
`$`	End of the text.
`*`	Literal repeated `0` to `n` times. (Kleen's asterisk)
`+`	Literal repeated `1` to `n` times. (Kleen's plus)
`?`	Literal repeated `0` to `1` time.
`.`	One character.
`|`	OR operator. Uses values left or right to the `|` sign.
`()`	Group of characters with defined order of appearance.
`-`	Range of characters (e.g. numbers 0-9).
`\`	Escape character.
`{}`	Character is repeated number of times written within the brackets.

Along with metacharacters there are already defined character sets. Some of commonly used are shown in Table 17.

Table 17: Regular expressions predefined character sets

Set	Description
`\w`	Matches any character - a letter, a number or underline.
`\W`	Matches any character - but is NOT a letter, a number or underline.
`\d`	Matches only numbers. Same as `[0-9]`.
`\D`	Matches characters which are NOT numbers. Same as `[^0-9]`.
`\A`	Matches the beginning of the input.
`\Z`	Matches the end of the input.
`\t`	Matches a horizontal tab.
`\r`	Matches a carriage return.
`\n`	Matches a linefeed.

Finally, the last important component of regular expressions is - modifiers (flags) which regulate how regular expression matching is done. They are usually written (in PCRE notation) after the last slash in the regular expression. For example, the expression `/cat/i` has `i` modifier which denotes that the matching should be done in case insensitive mode. Therefore, the mentioned expression will match the words `cat, Cat, cAt, CAT`, etc. A list of the most commonly used modifiers is shown in Table 18.

Table 18: Regular expressions modifiers

Flag	Description
`i`	Letters in the pattern match both upper and lower case letters.
`u`	Match with full Unicode.
`U`	Ungreedy matching. Return of the smallest portion which is matched.
`s`	Whole text is treated as a single line.
`g`	Global matching. Find all matches rather than stopping after the first match.

In the terms of the SSF, regular expressions play a key role when extracting lexical relations and testing O-structure patterns against sentences. In the next chapters almost all information extraction is done by utilizing the power of regular expressions.

5.6. O-structures

In order to do analysis of syntactic and semantic patterns at a sentence level it was necessary to develop specialized tool which would in a simple and intuitive way extract sentences with defined properties in the corpus, and after that store them as patterns. The assumption is that languages differ themselves more at a syntax level than at vocabulary level, and that it is easier to prove similarity of certain languages by comparing their sentence structure than comparing them at ethimological level. The patterns aren't anything else than combinations of general objects of WOS/SOW tags that form a sentence. In each of them it is possible to put one or more (even thousands) words, which creates enormous combinatorical possibilities of expressioning - the richness of a natural language. The next step is usage of structures, in a similar way on how translation of words is done; it is possible to do translation of patterns within one language or more of them. The pattern structure of one language can be mapped to a pattern structure of another language. Usually it will map to one or more pattern structures of another language. After the structures are mapped, the mapping of vocabulary is the next step, which is actually real and effective machine translation. However, there are certain problems because the word may have synonyms, and there is a tendency (which sociolinguistics deals with) to attract only a specific synonym and not any of them.

One important role of O-structures is in typological researches in left-right asymmetry of the natural language (based on the word position to other nearby words, to the left or right from other word class (e.g. a noun or a verb) which was a popular research topic back in '60s [72], and in various versions appears nowadays [35].

Table 19: Decomposition results of an adverbial of place in sentences [35]

✓	a	Dem	Num	A	N	MANY	∅	m	Dem	A	Num	N	-
✓	b	Dem	Num	N	A	many	✓	n	Dem	A	N	Num	FEW
✓	c	Dem	N	Num	A	FEW	✓	o	Dem	N	A	Num	many
✓	d	N	Dem	Num	A	few	✓	p	N	Dem	A	Num	FEW
∅	e	Num	Dem	A	N	-	∅	q	Num	A	Dem	N	-
∅	f	Num	Dem	N	A	-	✓	r	Num	A	N	Dem	FEW
∅	g	Num	N	Dem	A	-	✓	s	Num	N	A	Dem	few
∅	h	N	Num	Dem	A	-	✓	t	N	Num	A	Dem	few
∅	i	A	Dem	Num	N	-	∅	u	A	Num	Dem	N	-
∅	j	A	Dem	N	Num	-	∅	v	A	Num	N	Dem	-
✓	k	A	N	Dem	Num	FEW	✓	w	A	N	Num	Dem	FEW
✓	l	N	A	Dem	Num	few	✓	x	N	A	Num	Dem	MANY

According to Greenbergs universalities (Table 19), it is visible that sentences (from the most Indo-european languages), the combination of **Dem - Num - A - N** or its inverse version **N - A - Num - Dem** is the most frequent. **Dem** denotes the demonstrative pronoun, **Num** denotes a number, **A** denotes adjective and **N** denotes a noun. Number of occurrences of such patterns is graded from few (few), very few (FEW), many (many) and very many (MANY). The SSF enables such researches for the whole corpora, as well as many other that are done worldwide (e.g. the order of attributive adjectives, order of adverbs, order of TAM (tense-aspect-mode) morphems, etc.). Similar researches are described in [73]. In order to do so, it is necessary to assign WOS/SOW tags to the words (and possibly to create whole new T-structure branches within WOS or SOW tree). Each registered user can create his/her own sets of tags, and use them in his/her own research projects or ask an administrator to publish them publicly. O-structures can also be used in the process of utterances extraction. Utterances are compositions of two or more lexical words which are bonded with grammatical rules that gives an utterance a complete meaning. They consist of the *main part* and one or more *dependent parts* (e.g. cro. *bijeli galeb*) and are categorized by the grammatical bond nature or by the service of the dependent part in relation to the main part. There are three types of grammatical bonds between utterance parts. The first type is called *Agreement* (Congruence) and represents matching of the parts in gender, number, person, and case. Agreements can be furthermore categorized as: *noun's agreement* (an attribute and apposition are placed before the main part (a noun), if they are placed behind, they are separated by commas. Attributes and appositions can be placed near any sentence part if it is a noun. A group of attributes near one noun is called an attribute/apposition set; *subject-predicate agreement* (subject and predicate are matched in gender, number and person (e.g. cro. *Marija je otišla*)). A special type of an agreement (concord) is when the noun which complements another noun doesn't match the case and it is called a disagreeable attribute. It is not necessary to be agreeable in gender or number either (e.g. cro. *boca vina*). The second type of grammatical bonds is government (rection). The government is grammatical relation in which the main part (the verb), defines the case in the dependent part. There are two types of government. The first type is called strong government (many transitional verbs require the object to be in accusative (e.g. cro. *čitati (što?) knjigu, voljeti (koga?) brata*). It is called 'strong' because the parts 'knjigu' and 'brata' must be in accusative. The second type is called a 'weak' government and represents such grammatical relation in which one form can be replaced by the other and to preserve the meaning (e.g. cro. *letjeti nebom* (instrumental) and *letjeti po nebu* (loccative)). The third type of grammatical bonds is called an association. It is such grammatical relation in which main part (verb) is associated to a closed word class (e.g. cro. *lijepo pjevati*).

6. Semantics

To distinguish between meaning and grammar is an extremely hard task since both concepts overlap one another. In this thesis whose main goal is the extraction of lexical relations from textual documents, semantic model which will be used and described for that purpose. After that, particular attention will be paid to each of its key components: functions, domains, external resources (i.e. knowledge from online encyclopaedias and lexical resources), and finally to lexical processing (i.e. sentiment analysis) which, thanks to these resources, can be implemented. In the next chapter, paradigmatic relationships between the words will be observed in order to analyse the key concept in terms of lexical relations. This chapter therefore serves to provide a strong foundation for creating a computing environment in which the machine can find the conceptual structure as the highest level of natural language (metonymy and metaphor).

6.1. Semantic model

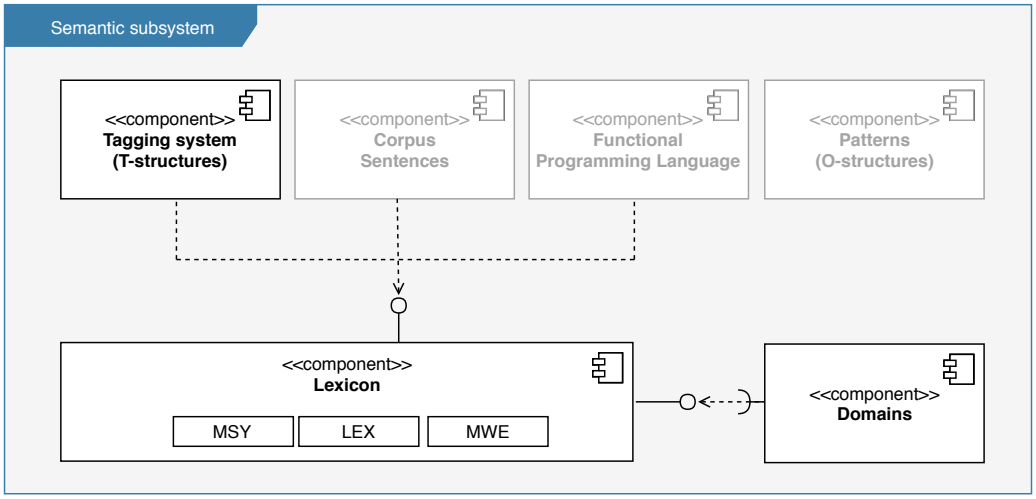


Figure 43: Conceptual model of the semantic subsystem

A model that solves the syntax-semantic relationships must create extremely complex algorithms [77] for solving the problem of multiplicity and obtaining multiword expressions, which is well noted in the following table:

Table 20: The problem of multiplicity and MWE

Semantics →	The content of the word		
↓ Morphosyntax	Monosemy	Polysemy	Semantic function
Expression: Single lexeme	Literal meaning	Homonym	Lemmatization
Expression: Multiple lexemes (MWE)	Synonym	Paronym of semantic domain	Arguments, Adjunct, Hypernym, Hyponym, Collocation, Thematic roles, etc.
Expression: Ordered lexemes	Synonymous sequence	Metaphors	Antonym, Meronym, Oxymoron

Finally, the model must be interoperable with other systems worldwide. In that sense LLOD is a logical choice. The solution can be twofold: either to convert information to LOD triples and store it into a Virtuoso server or to use a wrapper that will convert relational data to RDF format on the fly.

Paradigmatic lexical relations [38] are culturally determined between lexical units that:

- Share one or more fundamental semantic components (semantic characteristics);
- Belong to the same lexical category (part of speech);
- Fill the same syntactic position in the syntactic construction (statement or sentence); and
- Have the same semantic function [69].

Semantic components may be common or may be contrasting. For example, the component 'male' distinguishes a woman from a man or a young man from a girl, while the 'human' component belongs to both a man and a woman and a girl and a young man. Therefore, for semantic processing in the linguistic model, it is necessary to design and develop semantic categories, components and relationships and firmly associate them with syntactic patterns to make fine semantic granulation [82]. Although known since ancient times (hence bearing

Greek names), conceptual relations (lexical relations) are still actual, and their computer processing is a constant scientific challenge:

- Synonym - a word with the same or similar meaning with another word (e.g. cro. *majka-mama* (eng. mother-mom)),
- Antonym - a word with the opposite meaning with another word (e.g. cro. *pozitivno-negativno* (eng. positive-negative)),
- Hyponym - a word whose meaning is included in the meaning of another word (e.g. cro. *cvijet-ruža* (eng. flower-rose)),
- Hypernym - a word whose meaning also includes the meaning of another word (e.g. cro. *napitak-čaj* (eng. drink-tea)),
- Meronym - a word which is a constituent part of, or a member of something (e.g. cro. *prst-ruka* (eng. finger-hand)),
- Holonym - a word that consists of words that are its parts (e.g. cro. *tijelo-ruka* (eng. body-arm)),
- Homonym - a word that is written the same but has multiple meanings (e.g. cro. *pas* - a dog / a belt; (eng. rock - a genre of music / a stone)).

Automated detection of such structures within the SSF can be carried out using O-structures in the process of document parsing. Similar research was conducted by Sundbald [156] where method of extraction of hyponyms and meronyms from the question corpora was proposed. The method is based on the research of Hearst [79] who described how hyponym relations can be extracted using linguistic patterns. For instance, patterns like:

- `such NP as NP,* (or|and) NP`,
- `NP, NP*, or other NP`
- `NP, including NP, or|and NP`

can be used to extract such hyponymic relations. In terms of the SSF such patterns, are simple O-structures and can be defined in following way:

- `such (.*)\[w:2\] as (.*)\[w:2\],* (?:or|and) (.*)\[w:2\]`,
- `(.*)\[w:2\], (.*)\[w:2\] or other (.*)\[w:2\]`
- `(.*)\[w:2\], including (.*)\[w:2\], (?:or|and) (.*)\[w:2\]`

When tested over enriched version of sentences these O-structures would match nouns and their hyponyms. By observing sentences and their syntacic and semantic features similar patterns for extraction of other relations or features are also possible. For example, in the sentence: *In which country is X?*, the pattern will automatically assign the Toponym tag to the word X.

6.2. Lexical functions

Formally, a lexical function (LF) is such function which associates a given word (W) which is an argument of the function, with the textual value (V):

$$LF(W) = V$$

For example, lexical function `Antonym` will have value (V) `black` if the input parameter (W) is `white` (i.e. `Antonym(black) = white`). As a concept, lexical functions were introduced by Mel'čuk [117] in his Meaning Text Theory (MTT) in order to describe word's environment when expressing certain meanings. Nowadays, it is further developed by the Moscow semantic school [6]. MTT treats natural language as an advanced system with a complex set of rules which are responsible for transferring meanings into text and vice versa. Usually, it is used in classifications of collocations. Therefore, it is necessary first to explain institutionalized lexical relations, collocational relations and finally lexical functions. Some lexical relations are universal in all languages, but other are language specific and are called institutionalized lexical relations. Wanner [173] states that LF is a concept which can be used to systematically describe 'institutionalized' lexical relations, and clarifies that "a lexical relation is institutionalized if it holds between two lexical units L_1 and L_2 and has the following characteristics: if L_1 is chosen to express a particular meaning M, its choice is predetermined by the relation of M to L_2 to such an extent that in case M and L_2 is given, the choice of L_1 is a language specific automatism". In English language you may 'pay attention', whereas in Croatian you 'give' attention (cro. 'dati pažnju'). On the other hand, collocational relation as defined by Wanner [173]: "...holds between two lexemes L_1 and L_2 if the choice of L_1 for the expression of a given meaning is contingent on L_2, to which this meaning is applied. Thus, between the following pairs of lexical units collocational relations hold: *to do: a favor, to make: a mistake, close: shave, narrow: escape, at: a university, in: a hospital.*". As mentioned earlier, lexical functions are proposed by Mel'čuk within the frame of MTT to classify these lexical relations (collocations). There are two types of LF's, simple and complex. Table 21 shows examples of some simple LF's and their SSF equivalents. The names of Mel'čuk's functions are Latin abbreviations, whereas SSF's equivalents are named in English.

Table 21: Examples of lexical functions

LF	Name description	SSF equivalent	Gloss	Keyword	Value	Collocation
Syn	Lat. synonymum	`Synonym()`	word with the same meaning	to telephone	-	to phone
Imper	Lat. imperare	`Imperative()`	imperative expression meaning	shoot	-	Fire!
Anti	Lat. antonymum	`Antonym()`	word of opposite meaning	young woman	-	young man
Fact	Lat. factum	-	to accomplish itself	dream	come true	the dream came true
Real	Lat. realis	-	to fulfill the requirement contained in the argument	invitation	accept	to accept an invitation
Caus	Lat. causare	-	to cause to exist	association	found	to found an association
Magn	Lat. magnus	-	intense	temperature	high	high temperature

Mel'čuk proposed over 40 lexical functions, some of them are already implemented in the SSF, and others can be developed in the NLF module by advanced users. Since the list of LF's is made empirically, there is still possibility that in future there will be some new LF's defined. Simple LF's can be combined in order to form complex LF's. The output of one function is the input (argument) for another function. In terms of SSF this can be demonstrated by the following example: `Synonym(Antonym("mladić"))`. The function `Antonym()` executed first, and the argument is word cro. '*mladić*' (eng. young man). The result of that function call is the word cro. '*djevojka*' (eng. young woman) which is then used in the next step as an argument in the function `Synonym()`. The result of that is the list of words: *gospođica, mlada dama, cura.* Mel'čuk's lexical functions are also related to word's morphology and relation of morphs to semantic of the word. This partly confirms hypothesis **H1** because the proposed deterministic language model enables implementation of such functions. The NLF module implements some of these functions which are inspired by Mel'čuks theory and are based on the SSF lexicon and a new type of word tagging (described in Chapter 3). List of all functions in the SSF with usage examples is given in the Appendix D.

6.3. Semantic domains

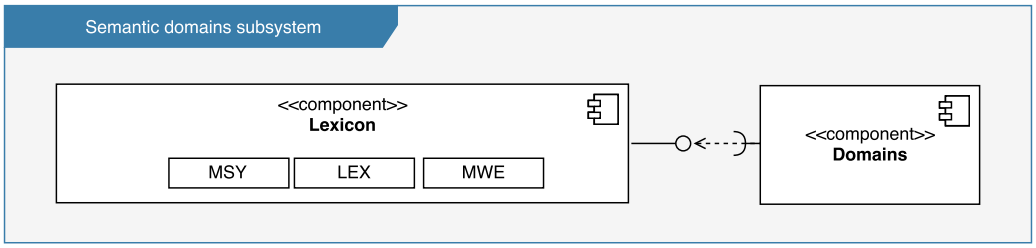


Figure 44: Conceptual model of the semantic domains subsystem

Semantic domains, in terms of SSF, are usually sets of words (ordered or unordered) that share some features in common (static domains) or sets of WOS/SOW features in combination with regular expressions that describe some set of words (dynamic domains). They are widely used within the system in various situations (e.g. patterns matching or advanced antonym algorithm). There are two main types of domains: static and dynamic (as shown in Figure 45).

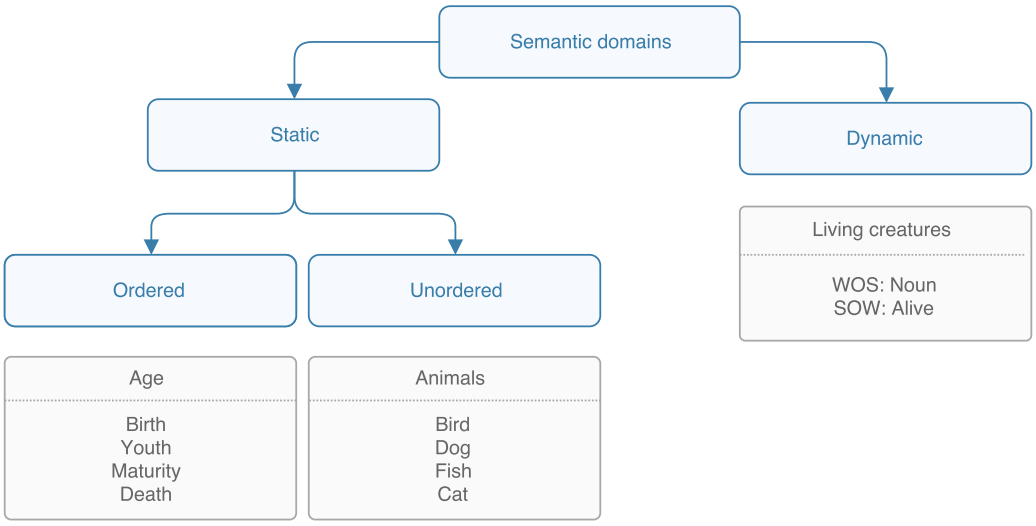


Figure 45: Semantic domains types

Static domains are made of a finite number of elements (words) and do not change in time. For example, the domain *Animals* can have members like: *bird*, *dog*, *fish*, *cat*, etc. The creation of a static domain is a difficult process, because it involves huge human effort if done manually.

On the other side, dynamic domains are defined by the set of rules, and as such do not have finite number of elements. The elements of dynamic domains are automatically generated in the time of domain usage. For example, one such domain can be domain of *Living creatures* which is defined by a rule which says that members of such domain are

all words which are tagged with a WOS tag *Noun* and a SOW tag *Alive*. The domain exists only as a set of rules, and therefore is calculated in the time of execution, which enables that domain is always up to date, no matter how lexicon has changed over time. If new words are inserted in the lexicon, and are tagged with SOW tag *Alive* they will automatically become members of such dynamic domain.

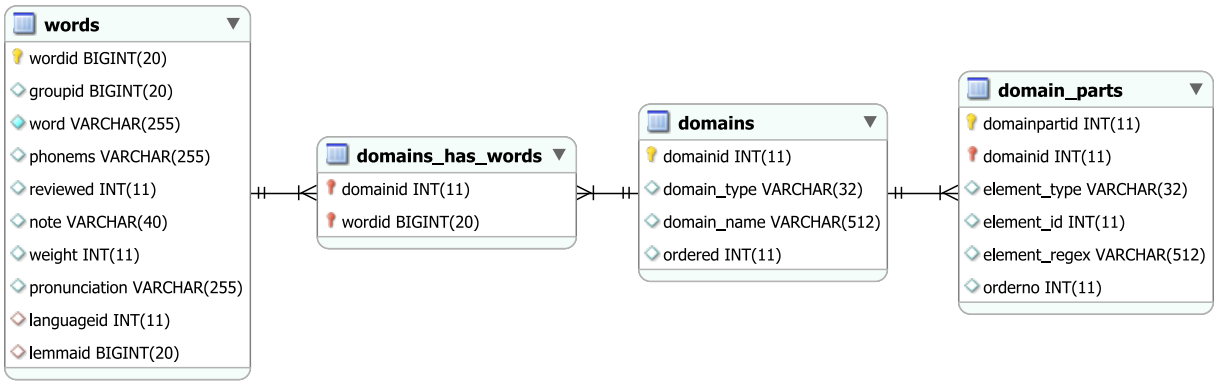


Figure 46: Database model of the domains subsystem

Figure 46 shows segment of a database model which is relevant for domains subsystem. Central table is called `domains` and holds the data about all domains that exist within the system. Every domain has a name (`domain_name`), a type (`domain_type`) which can be static or dynamic, and information whether the domain is ordered or unordered. Since domain can have one or more members (words in case of static and rules in case of dynamic domains) the table `domain_parts` holds information about domain members. The attribute `element_type` defines the type of a domain member. In static domains `element_type` will always be `wid` (wordid), due to the nature of static domains; only members that could form a domain are words. In dynamic domains `element_type` can be either `wos`, `sow`, `wid` or `regex`. Finally, a domain may or may not be associated to a specific word(s), and a word(s) from the lexicon (as shown in Figure 44) and the word(s) can be within one or more domains, so the weak entity table `domains_has_words` holds information about these relations.

To enable easier (automated) creation of static domains, methods of extraction information from publicly available encyclopaedic knowledge could be used [126]. Figure 47 shows two possible paths for static domain creation. The first relies on extraction based on word's syntactic tags (e.g. nouns, verbs, etc.), and the second is based on semantic roles within the sentence (e.g. subject, predicate, object). An example of subject, predicate and object extraction is given in Section 7.4.

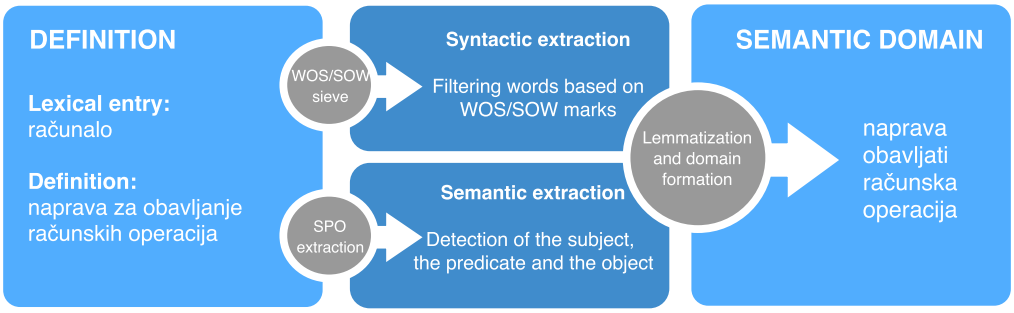


Figure 47: Automatic creation of a semantic domain

After a subset of words is extracted they are lemmatized in order to get a grammatically independent domain. An example of PHP script that creates such static domains from words definitions is shown in Appendix A. Every word with a domain associated to it in lexicon listing has an additional button called domains (as shown in Figure 48) which when clicked pops up a new window with all domains which are related to a selected word.

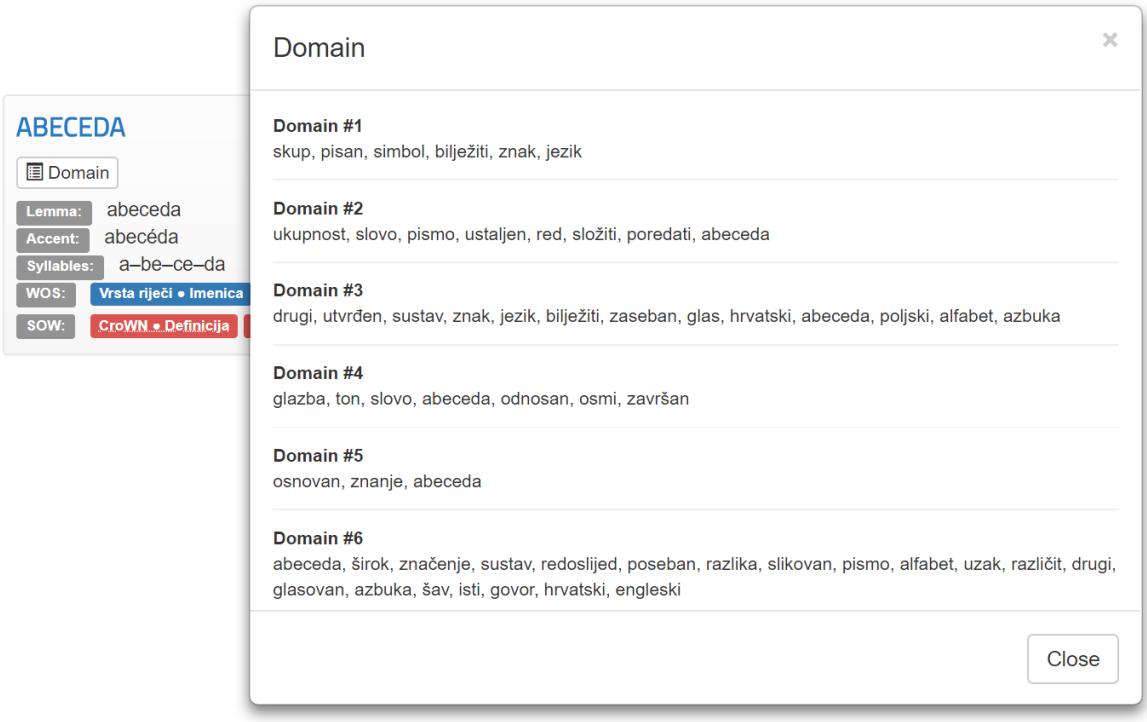


Figure 48: Example of a semantic domain in SSF's lexicon

In the above example, the word cro. *abeceda* (eng. alphabet) has six domains associated with it which were automatically created using the word's definitions from six different SOW tags.

Definitions of the word *abeceda*

(a) *"skup pisanih simbola kojima se bilježe znakovi nekog jezika"*

(b) *"ukupnost slova u latiničkom pismu poredanih po ustaljenom redu [složiti/poredati po abecedi]"*

(c) *"neki drugi utvrđeni sustav znakova kojima se u nekom jeziku ili sustavu bilježi svaki zaseban glas [hrvatska abeceda; poljska abeceda; Morseova abeceda], usp. alfabet, azbuka"*

(d) *"glazb. u glazbi, nazivanje tonova slovima iz abecede (u C-duru od c do h, odnosno do c, kao osmoga, završnog tona)"*

(e) *"pren. osnovno znanje o čemu [to je abeceda]"*

(f) *"abeceda, u širem značenju, svaki sustav i redoslijed grafema kojima se označavaju posebni fonemi (za razliku od slogovnoga i slikovnoga pisma; alfabet). U užem značenju, redoslijed grafema u latiničnom pismu različit od redoslijeda u drugim glasovnim pismima (alfabet, azbuka i sl.). Ni redoslijed u svim latiničnim pismima nije isti pa se govori o nacionalnim abecedama (hrvatskoj, češkoj, engleskoj i sl.)"*

Domain #1 was created from the Croatian WordNet definition (a), Domain #2, #3, #4 and #5 from the Croatian language portal definitions (b,c,d,e) and the Domain #6 from the Miroslav Krleža Institute of Lexicography definition (g). Underlined words are those which got through the WOS sieve (Nouns, Adjectives, Verbs and Adverbs) and after lemmatization process and elimination of duplicates became elements of the newly created domain. Using multiple definitions in the process of domain formation is extremely usefull for detection of lexical relations like metaphors. For example, the word cro. *srce* (eng. heart) will have at least two domains. One real domain which will contain words like cro. *organ* (eng. organ), cro. *tijelo* (eng. body), cro. *operacija* (eng. operation), cro. *krv* (eng. blood), etc., and other metaphorical which would contain words like cro. *zlatno* (eng. golden), cro. *lijepo* (eng. nice), cro. *dobro* (eng. good), etc.

The administrator of the system has an ability to edit this domain or to manually create new ones from the Domains tab within the SSF, as shown in Figure 49. Domain elements can be easily ordered by dragging and dropping elements of the domain in a specific place within the domain, and checking the '*Ordered domain*' checkbox. Ordered domains are relevant in some specific use cases like for example, advanced antonym algorithm described in Section 5.4.

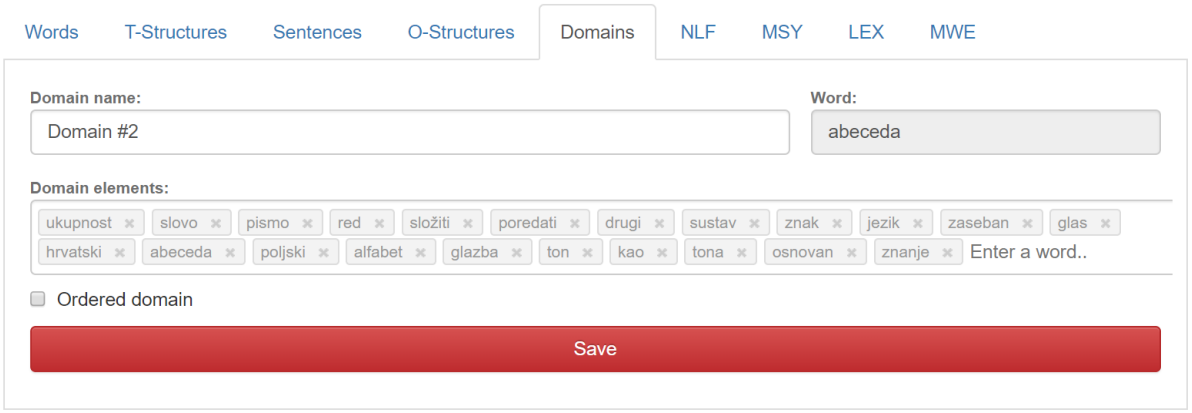


Figure 49: Domains editor

Beside editing it is possible to create new domains from the scratch and associate them to a certain word.

6.4. Integration of external resources

The natural language deterministic model, which is used for the extraction of lexical relations, relies on sentence patterns in which every word is tagged with as many tags as possible (both grammatical (WOS), and semantic (SOW)). In order to build a quality word lexicon that is enriched with WOS and SOW marks, the private lexical database 'Croatian word' [128] was adapted with the permission of the author. This initial lexicon was used as a base for the generation of different word forms using the Morphological Generator [113]. Each word form was assigned with belonging grammatical and semantic metadata. The richness of words' metadata increases the number of patterns that can be derived. In order to enrich the words lexicon, the SSF integrates some of the most popular lexicographic resources that are available online. In the SSF's lexicon, each WOS/SOW tag that originated from an external source is properly referenced, with a direct hyperlink to the source content. In this thesis, the focus is on the Croatian language; therefore, this section will cover only the resources that are relevant for this particular language.

The first, and, in scope of lexical resources, surely the most valuable, is WordNet [132]. WordNet is a lexical database that was first developed for the English language in the Cognitive Science Laboratory of Princeton University, also known as Princeton's WordNet (PWN). It is released under a BSD - license. Over the years, the WordNet database has continued to grow, and is currently in version 3.0, containing 155,327 words organized in 175,979 synsets (sets of synonyms). Each synset has information about synonyms, antonyms, hypernyms, hyponyms, and meronyms. The SSF has a slightly different organization of data than the WordNet. Instead of using synsets, the SSF tags

the word with all possible features, regardless of the synset from which the data originates. Building of the Croatian version of WordNet database (CroWN) started in 2004 at the Institute of Linguistics, Faculty of Humanities and Social Sciences at the University of Zagreb [139]. Currently, it comprises 31,300 literals in 10,040 synsets and is released under a CC-BY-NC-SA license. It is included in the SSF as an external resource.

In addition to the WordNet database, for sentiment analysis, the SSF integrates SentiWordNet. SentiWordNet is a lexical resource explicitly devised for supporting sentiment classification and opinion-mining applications [9]. Each synset of WordNet is assigned to three sentiment scores: positivity, negativity, objectivity. Although SentiWordNet is made for the English language, due to the alignment between CroWN and PWN, it is possible to transfer sentiments from English words to Croatian. Section 6.5 shows examples of sentiment analysis within the SSF.

The BabelNet is a multilingual semantic network and ontology which was developed at the Sapienza University of Rome, at the Department of Computer Science Linguistic Computing Laboratory [121, 120]. BabelNet was created by linking Wikipedia articles to WordNet; therefore, it retained the structure of grouping words in synsets. As of February 2018, it contains lexical information from 284 languages, including Croatian. The database contains about 53 million images with Babel synsets and provides Lemon RDF encoding of the resource.

The Croatian Language Portal [81] is a joint project of the publishing house Znanje and the University of Zagreb University Computing Centre (SRCE). It is the first online dictionary of the Croatian language and had been available for free since June 2006. The database contains 116,516 lexical entries and their definitions, about 60,000 examples, 18,000 syntagmatic expressions, and 10,000 phraseological expressions.

The Miroslav Krleža Institute of Lexicography network encyclopedia, which is available at `http://enciklopedija.hr/` is another valuable lexicographic resource of the Croatian language that is a part of the SSF. The encyclopedia is based on the printed edition, which was released in 11 issues from 1999 to 2009 (with 67,077 articles). It has been available as an online service since September 2013. As noted on the encyclopedia web page, it is permitted to use or quote individual articles in parts or as a whole with the indication of the source [97].

There are also many other lexical resources available worldwide, which can either be incorporated into the SSF in the form of WOS/SOW tags (by using `AssignWOS()` and `AssignSOW()` functions, which are described in Appendix D) or in the form of semantic relation to the external resources URI (by using the SOW tag *owl:sameAs*). This kind of relation enables linkage to any external semantic resource, which can be used in SPARQL queries or for harvesting external data and storing within the SSF. Even without 'hard'

semantic links, it is possible to extend the word's features (WOS/SOW tags) by utilizing these functions through the NLF/API. For example, an experienced Python developer could write their own script that fetches any network-available resource, and, using NLF functions, assigns tags and their values to specific words. Of course, these tags would be visible only to the developer as long as the administrator of the system doesn't make them public. In this way, there is no formal limitation on how enriched the lexicon can be. Figure 50 shows the word cro. '*majka*' (eng. mother) in the SSF lexicon, and links to external resources (BabelNet and CroWN).



Figure 50: External lexicographic resources in the SSF

Definitions that originated from external resources are used in the process of creating static domains (described in Section 6.3), whereas semantic links to external ontologies in the form of *owl:sameAs* SOW tags are used in federated SPARQL queries (described in Section 8.3). These semantic links can be assigned to any URI (e.g. DBpedia, LexInfo, GeoNames, etc.). In this way, an unlimited number of external resources can be integrated into the SSF, which is an answer to research question **Q3**.

6.5. Sentiment analysis

Sentiment analysis (also known as opinion mining) is a very active research area in the field of natural language processing, text mining and information extraction, which deals with computational analysis of people's sentiments, attitudes, emotions and moods. Sentiment analysis is not just determining the polarity of some sentence or a document but rather dealing with complex challenges such as issues of context sensitivity, sarcasm, irony, etc. These problems become even more complex when it is necessary to analyze idioms, implicit sentiments, ambiguity or compound sentences. Unlike other lexical information, sentiments have specific feature of being subjective. What someone considers as negative, other people may consider as positive. There are many strategies which can be used in the process of sentiment analysis based on the type of information which can be linguistic, statistical, implicit, etc. Linguistic information is usually linked to specific

grammatical features the word may have (i.e. POS tags). For instance, nouns usually denote 'tangible and intangible things', whereas prepositions are used to denote relations between 'things' [59]. Syntactic structure of the sentence defines the way in which words are bonded together [24]. Statistical sources are usually linked to term frequency (e.g. individual words (unigrams), multiwords expressions (bigrams, trigrams, n-grams) and their frequencies. Similarity information, for example, can be obtained from an affiliation to a given sentiment polarity. Implicit sentiment is not directly expressed but is extracted from the concept, so it is usually called 'concept-based sentiment analysis'. It is an approach to sentiment analysis proposed by Cambria et al. [25] at the MIT Media Laboratory in 2010. Since 2014, it has been further developed at the NTU School of Computer Science in Singapore. Concept level sentiment analysis is focused on the implicit sentiments associated with concepts, rather than statistical, linguistic or similarity analysis [27]. The output of that research was SenticNet, publicly available sentiment lexicon [26]. The idea behind SenticNet was to offer it as a resource for conduction of sentiment analysis. There are different versions of SenticNet available for download at `http://sentic.net/downloads/` in RDF/XML format. It contains almost 14,000 affective concepts and also offers API interface which is integrated in the SSF and can be used in the NLF module. Along with the words, SenticNet also provides sentiments for multiword expressions (in the same way the SSF's MEW lexicon does). Concepts in SenticNet are taken from ConceptNet [103]. The ConceptNet is a commonsense knowledge base and natural-language-processing tool-kit developed at the Massachusetts Institute of Technology (MIT). The concepts in ConceptNet rely on the knowledge from Open Mind Common Sense (OMCS) corpus. The current version, ConceptNet 5 has been enriched with additional data from external resources like DBpedia and WordNet [152]. Figure 51 shows diagram of sentiment lexicons and their origins.

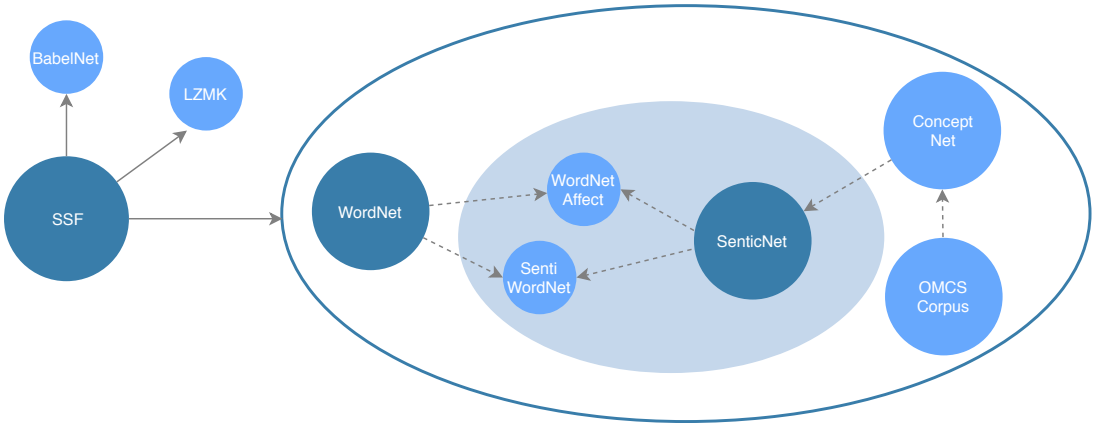


Figure 51: The sentiment lexicons

Sentiment analysis can be performed at several levels: documents, sentences and aspects.

Document level sentiment analysis deals with the whole document and classifies it as either positive or negative [163, 70, 46]. Liu [102] suggested that this kind of analysis treats each document as a single entity with a single sentiment that is not applicable in documents with multiple entities.

Sentence sentiment analysis processes single sentences and classifies them as either positive or negative [12, 88]. The problem with sentiment analysis at the sentence level is when there are more than one opinions within the sentence (e.g. compound sentences where multiple opinions may be incorporated within one sentence). For example, the sentence 'Although the service is not that great, I still love this restaurant', contains both positive opinion (concerning 'restaurant') and at the same time - negative opinion (concerning 'service'). The goal of this type of analysis is to discover sentiments in all sentence components [176, 148]. The SSF's O-structures (described in Section 5.3) show how SSF can decompose compound sentences into parts, which can be useful for sentiment extraction of each such sentence.

Aspect based sentiment analysis is focused towards various aspects of entities (in the above example, different aspect of a restaurant may be food quality, service, etc.). Its goal is to identify various aspects and then detect the sentiment towards each of them. Based on the performance type, sentiment analysis can be either supervised or unsupervised.

Supervised sentiment analysis is a type of analysis where algorithms try to determine the sentiment based on the large set of training samples. Such training samples are manually tagged and are domain specific. For each domain, algorithm analyzes test data which is used later in the process of decision making. Supervised methods usually implement machine learning algorithms (e.g. Naïve Bayes classifier, Maximum Entropy and Support Vector Machines).

Unsupervised sentiment analysis, however, is a type of analysis where algorithms try to determine sentiment without using training data or creating models. Consequently, they do not require any labelled data and are domain independent. Unsupervised methods are based on the sentiment value (polarity) of each word or phrase within a sentence or a document. Two main unsupervised approaches are linguistic-based [163] and lexicon-based [158, 58].

Linguistic based approach aims to find specific POS patterns which are most likely to express an opinion. The SSF implements O-structures which can be used in such extractions. For example, patterns for extraction of word sequences such as an adjective which is followed by a noun, or even more complex phrase structures or sentences [142]. Liu [102] notices that adjectives are usually very important when it comes to detection of opinions. Such lexicons, which are annotated with sentiment orientation are called sentiment lexicons [122]. The most commonly used sentiment lexicons are SentiWordNet

and WordNet-Affect [129]. Both are limited to single-word concepts. The SSF implements SentiWordNet in its SOW tags and is also linked to the SenticNet via API.

Sentiment analysis within the SSF is possible at multiple levels due to the embedded programming tools for processing of words, multiword expressions, various lexicons and their links to other network resources:

- Words - especially adjectives can be tagged with special SOW tags which contain sentiments;
- Word groups (phrases) may be processed with special `Ngrams()` function which parses the source sentence into n-grams (the function is part of NLF and briefly described in Appendix D);
- MWE lexicon which contains phrases and idioms can have special SOW tags with sentiments for later usage in analysis of sentences and documents;
- Network integration with SenticNet, BabelNet, WordNet and encyclopedias enable the usage of the already prepared information resources, and usage of their analyzers;
- The SSF inclusion in global LLOD cloud enables working with the so called 'sentic patterns' in a 'big data' environment. Furthermore, O-structures have all features which are necessary for such patterns to be included in the SSF in near future.

In this way the SSF becomes a part of global network infrastructure of other resources, but at the same time holds its own.

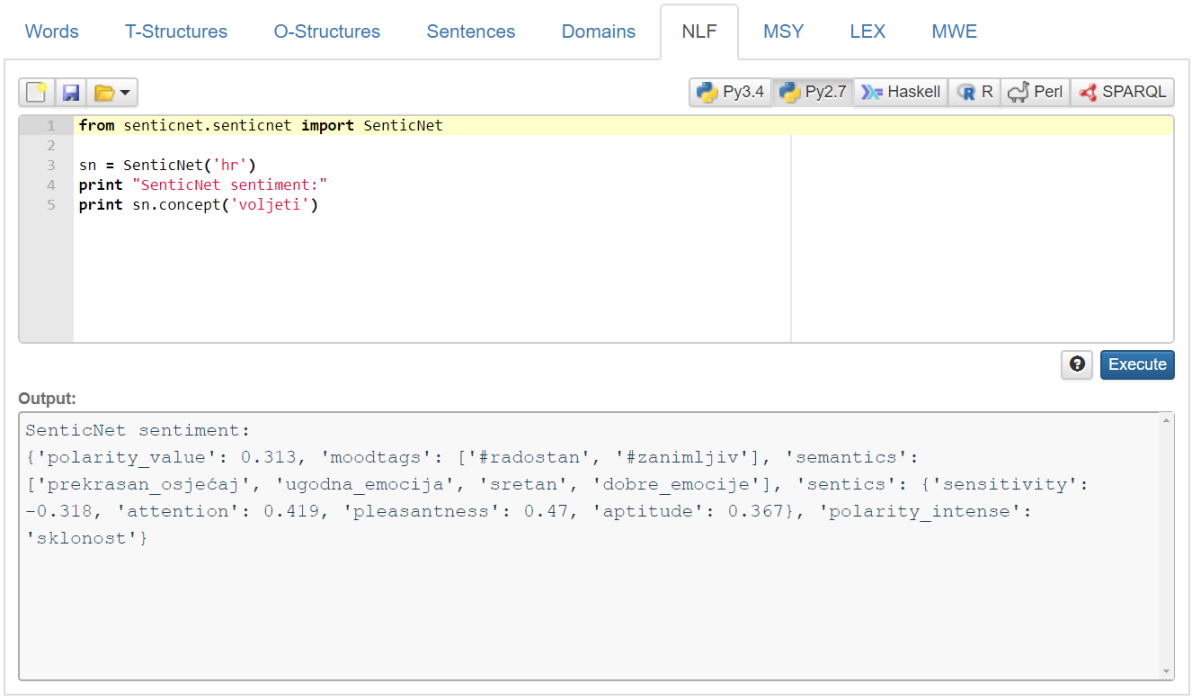


Figure 52: Sentiment analysis with SenticNet in the SSF

Figure 52 shows the screenshot of simple sentiment analysis in the SSF's NLF module, which gives an answer to research question **Q4** that it is possible to conduct the sentence semantic analysis besides the morphosyntactic one.

7. Extraction of lexical relations

Machine aided extraction of lexical relations requires a digitized document. Lexical relation is by definition: "... a culturally recognized pattern of association that exists between lexical units in a language" [110]. Usually, documents are stored in larger entities called corpora (plural lat. corpus). In the SSF; each document is divided into smaller units (first into sentences, and then each sentence is tokenized to words) and as such stored as a chained list of words within the SSF's database. This chapter deals with the processing of corpora (which is stored in the SSF), as a source of words which are extracted from sentences. Words between themselves are bonded with both syntactic and semantic relationships. Section 7.1 describes how corpora is processed and stored into the SSF relational database. The parser component which deals with complex problems of word's ambiguity and their solutions is also shown. Section 7.2 describes extraction of words, their syntactic features and relationships to other neighboring words, i.e. words which are to the left or to the right from the observed word. However, lexical relations have paradigmatic component too, which is embedded in the meaning of the word (the so called concept). Therefore, Section 7.3 describes conceptual structures (such as metonymy, simile, metaphors, etc.) which can be extracted as described in Section 7.4. In order to achieve this, it was necessary to develop a new type of syntactic patterns (the so called O-structures) which at the same time include both WOS and SOW tags (as described in Chapter 6: synonyms, antonyms, hyponyms, hypernyms, meronyms, holonyms, etc.). The paradigmatic information is extracted from static or dynamic semantic domains (as described in Section 6.3). Finally, at the end of this chapter, API module which is used to connect external applications to the SSF is described. This important feature of the SSF ensures that the research data from the framework are integrated into any software component which is RESTful API capable.

7.1. Corpora

In linguistics, corpora is a large set of texts which are used in the process of NLP. Desagulier [43] defines it as a finite sample of genuine linguistic productions by native speakers. Basically, a corpora is a set of documents that should represent some specific language, genre or period. Textbooks usually present Lancaster-Oslo-Bergen corpus [85] for British English, and Brown Corpus [66] as a representatives of balanced English texts from early 1960s. The size of corpora is a relative term, but it is considered that the larger corpuses are better foundation for various linguistic researches. In 1990 the British National Corpus (BNC) with ~100 million words was considered to be very large, compared

to Corpus of Contemporary American English (CCA)(~450 million words), but nowadays in the time of the Internet and digitalised materials, these corpuses does not seem that large anymore.

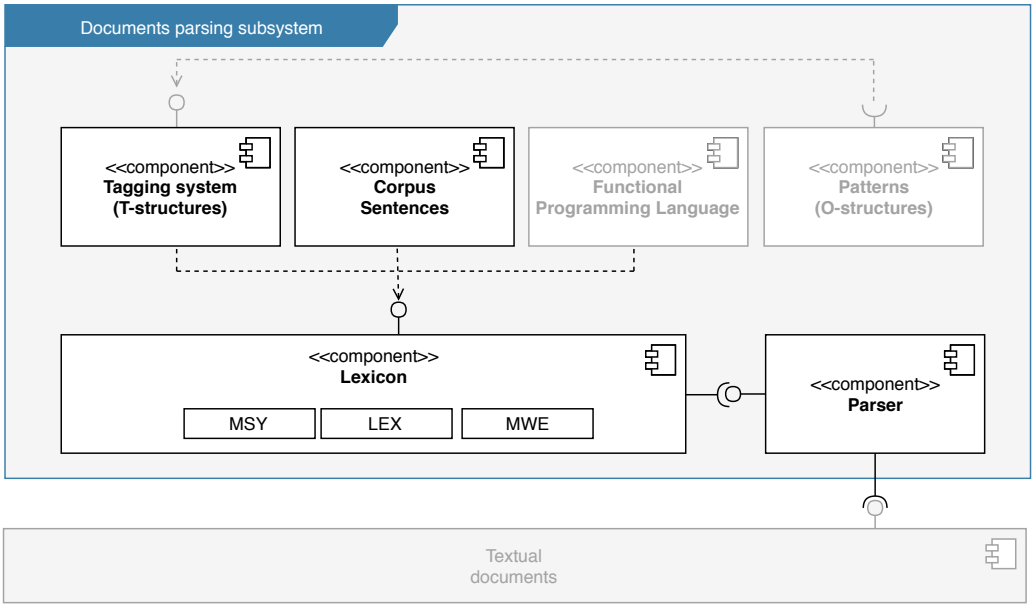


Figure 53: Conceptual model of the parsing subsystem

In order to make the corpora suitable for linguistic research, they are often subjected to a process known as annotation. In the SSF annotation of corpora is performed in the process of parsing (as shown in Figure 54). Parsing is a process of analyzing a string of symbols, conforming to the rules of formal grammar. The term parsing comes from Latin pars (*orationis*), meaning 'part' (of speech) [76]. Instead of commonly used tagsets, the SSF introduces a new T-structure tagset (described in Section 3.1), which is more suitable for text mining analyses since it has much more tags which are organized in a hierarchical data structure. Figure 53 shows conceptual model of the document parsing subsystem in the SSF which is responsible for the building up of corpora. In order for corpora to be representative for textual analysis it is necessary to ensure that its sampling scheme characterizes the target language. For example, if the research goal is to study spoken language of Croatian children, then in the process of the corpus design, the text which will be a part of the corpora, beside children conversations with peers, must also include their conversations with parents, teachers and other people. In the SSF, every user can upload its own sets of documents and organize them in different corpora. Each set of lexical functions can then be used over the whole corpus or partially only on some documents.

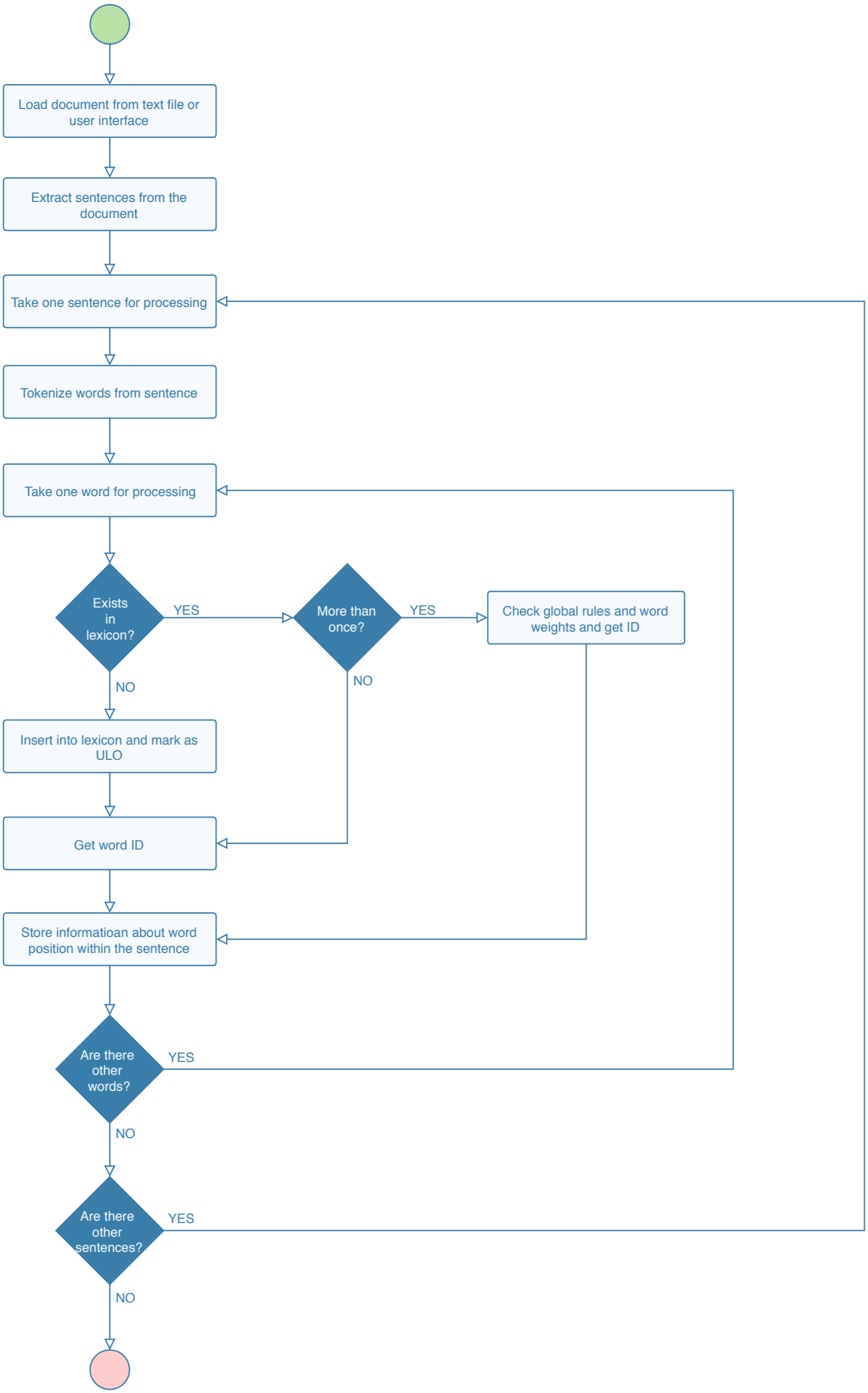


Figure 54: Flowchart diagram of the Parser component

The Parser component's main task is to load textual documents into the database and in that way build corpora as well as expand the lexicon. As shown in Figure 54 workflow of the Parser begins with the loading of textual document in an UTF-8 format, which can be either stored on the filesystem or entered directly via GUI. In the next step, loaded document is split into sentences, taking into account that the dot (.) is not always sentence delimiter. There are special cases (e.g. abbreviations) which can contain dot, but do not mark sentence ending. The Parser takes into account such cases and recognizes in the case of Croatian language. Every sentence is further split into words and analyzed at a word level. When a word is isolated, the Parser looks into the Lexicon in order to find the matching word. In most cases the word is already in the Lexicon (usually even multiple times, due to the homonimy) and Parser needs to determine which form of the word is right to be used in the observed sentence. This is done by using sentence patterns. For example, the sentence: cro. *Marija usta i pođe u gorje* (eng. Mary stands up and goes to the woods) and cro. *Marijina usta i oči* (eng. Mary's mouth and eyes), both contain the word *usta* which can depending of the case, be either a noun or a word. Only by observing of neighbouring words the Parser can conclude that, based on the previous word (e.g. Mary, which is a noun), the word Mary can be only followed by a verb, and not by another noun. Using such sentence patterns enables the parser to correctly determine which homonym word from the lexicon is right to be used in observed sentence. The richness of the SSF's lexicon in which words are tagged with as many WOS/SOW tags as possible enables the parsing component to be more precise when determining the service of the word within the observed sentence. Another example is: if the observed word is tagged with WOS tag as pronoun, the parser knows that there are pronouns which come in pair with nouns in genitive (e.g. cro. *bez, blizu, do, duž, ispod, ispred, iz, iza, između...*), dative (e.g. cro. *k(a), nasuprot, unatoč, usprkos*), accusative (e.g. cro. *na, o, po, u, kroz(a), među, nad(a), pod(a), pred, uz(a), niz(a), za*), locative (e.g. cro. *na, o, po, prema, pri, u*) or instrumental (e.g. cro. *među, nad, pod, pred, s(a), za*). In some scenarios, even patterns are useless, so the parser has to rely on the `weight` attribute of the word. As described in Section 4.2 each word can have a weight which is then used in cases where the algorithm is uncertain which word from the Lexicon to chose. The word with a higher weight value is chosen. For example, in terms of the word `usta` in Croatian language, it is more common for it to be a noun than a verb. In cases where the observed word is not in the lexicon, and cannot be analyzed, it is automatically inserted into the lexicon and tagged with an ULO tag (as described in Chapter 3). Since every word in the lexicon has its unique ID, it is retrieved in the next step, and used in building of a sentence chain. Figure 55 shows a segment of the SSF's ER model, which is relevant for corpora storing. The table `words_in_sentence` is filled in this step of Parsing procedure. The process of

word identification is then iterated for every word in the sentence, and the same applies for every sentence in the document. Once the document is fully processed it is stored in the table `documents`.

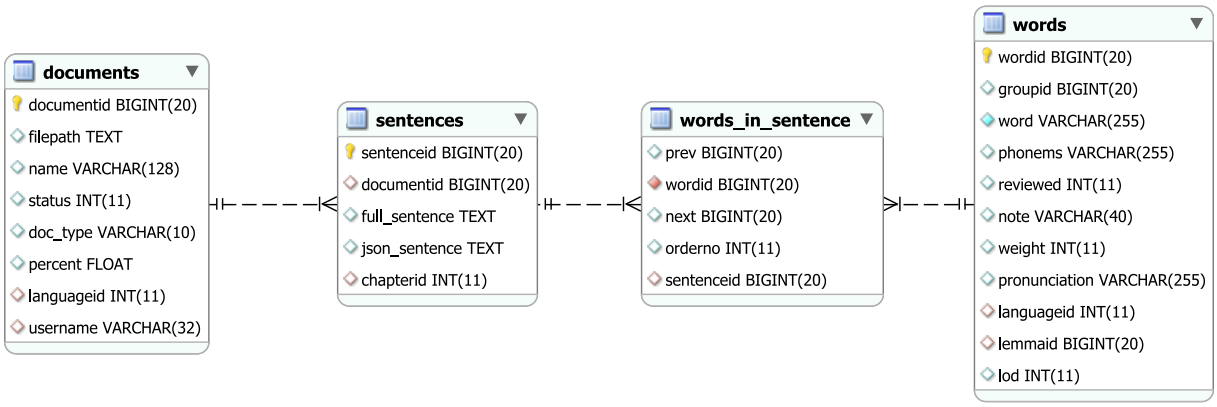


Figure 55: Database model of the corpora subsystem

7.2. Extraction of word's environment in the SSF

Word from the corpus can be searched by letters, WOS/SOW tags or their combinations. The main goal of this module is to find words in selected documents and their nearest neighbourhood (previous and following word), which forms a triple of minimal information for syntactic analysis. One of classical examples of such research are collocations.

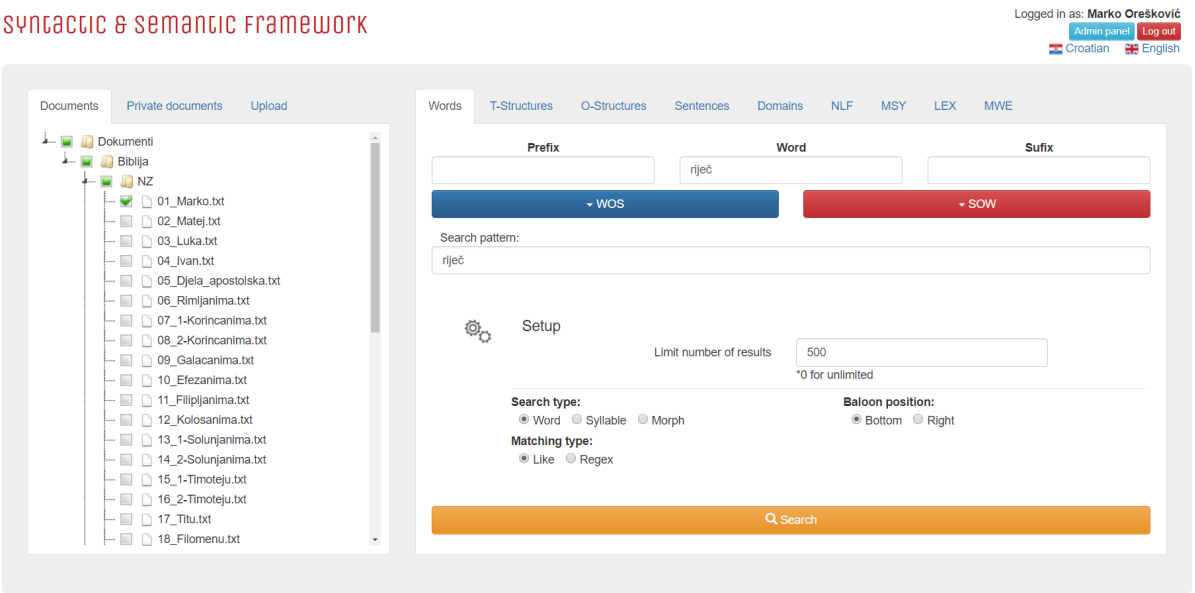


Figure 56: Screenshot of SSF's word search

However, compared to classical search in this case words are enriched with their WOS/SOW features which gives a researcher an opportunity to observe there triples as syntactic patterns (similar to `http://www.patternbank.uni-erlangen.de/cgi-bin/patternbank`.

cgi) which is nowadays a need when it comes to analysis of digitalised content of any natural language. The user interface for word searching (shown in Figure 56) consists of a form for entering prefix, word base and suffix. For example, if letters 'ma' are entered in prefix field, and 'i' in a suffix field, the result will be all words from the selected documents which start with letters 'ma' and end with letter 'i' (e.g. *mati*, *mariji*, *mariti*, etc.). Furthermore, if the search is limited by WOS filter, only to verbs, the output from previous example would contain only word *mariti* since other words are nouns). It is possible to use regular expressions (described in Section 5.5) in formation of search queries or for inexperienced users, a simplified version in a SQL like syntax can also be used.

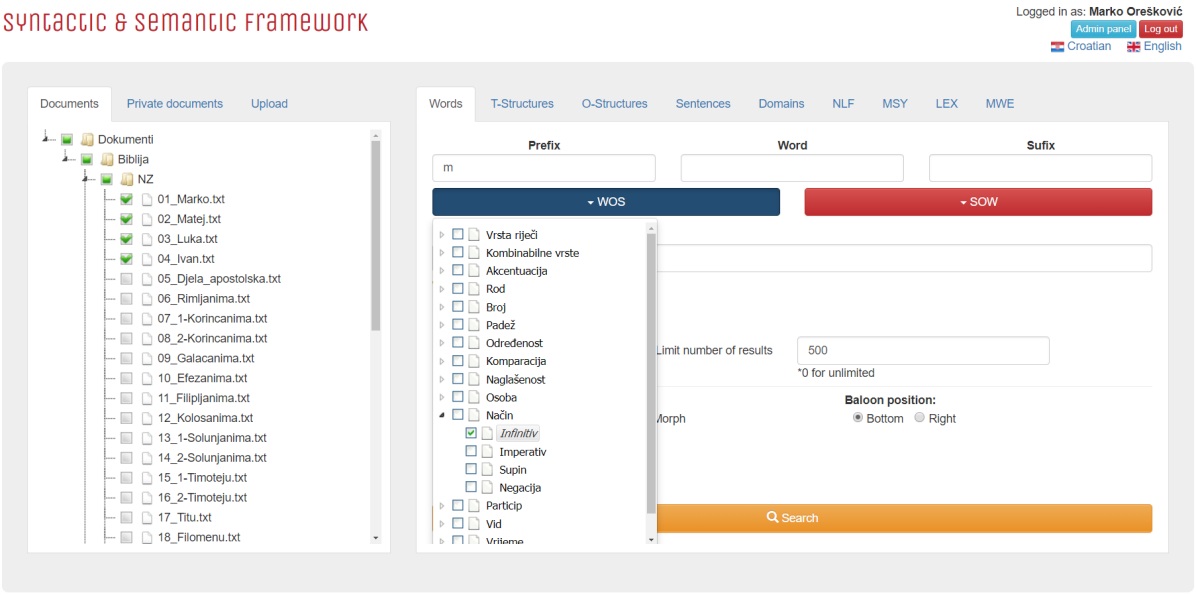


Figure 57: Word searching filters (WOS)

Figure 57 shows an example of a search query for words starting with a letter 'm' and WOS tag 'Infinitive'. The result of such query is shown in Figure 58.

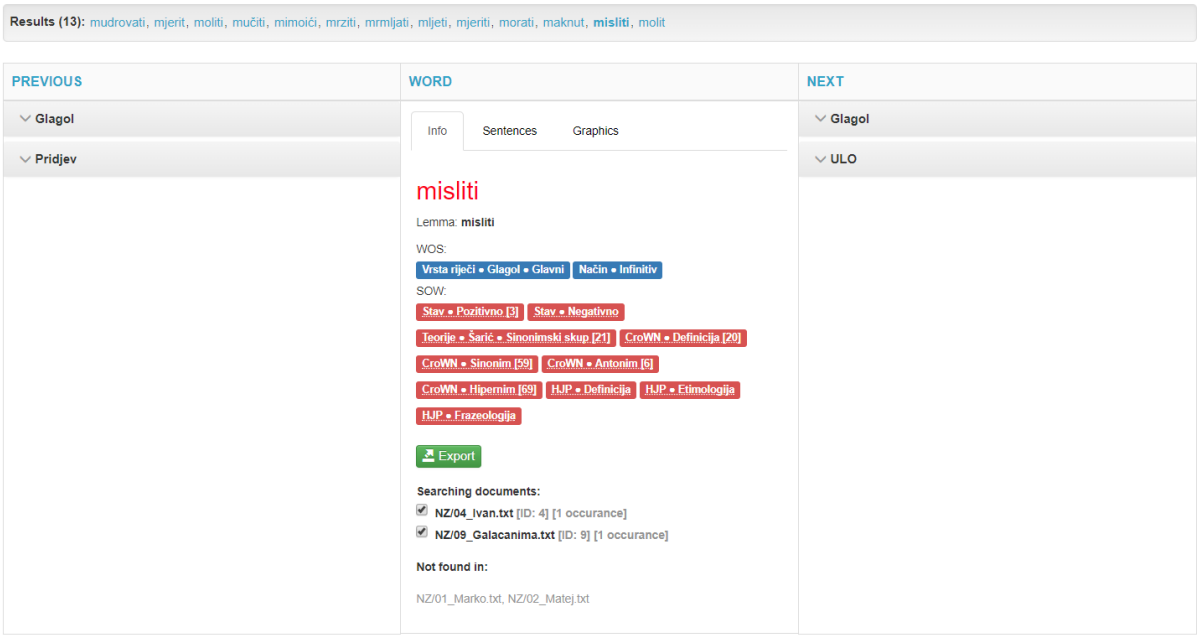


Figure 58: Word searching results

In situations where many words meet the search criteria, they are shown in the top part of the screen (as shown in Figure 59). When hovering mouse over each word the popup balloon with WOS/SOW tags is shown. Clicking on the word opens it in the central part of the screen. By default, number of results is limited to 500 but it can be increased in a settings window.

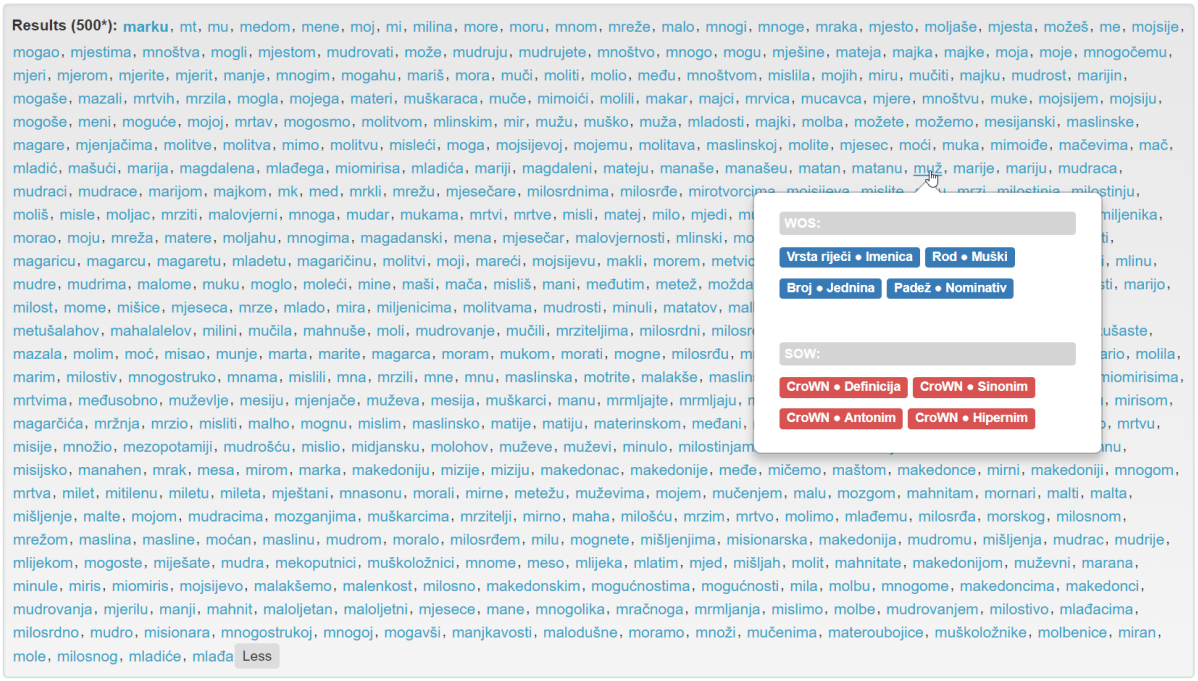


Figure 59: Word searching results with WOS/SOW info

To the left of the word are words that are found in documents prior to the observed word. They are grouped by their WOS features in a form of an accordion. The same logic applies to the word that follows the observed word and is located in the right part of the screen (as shown in Figure 60). By clicking on the word in the left or the right part of the screen the segment of the document where the observed triple is found is shown along with all WOS/SOW features. It is also possible to crawl through the document by clicking on the left or the right arrow, near the sentence segment.

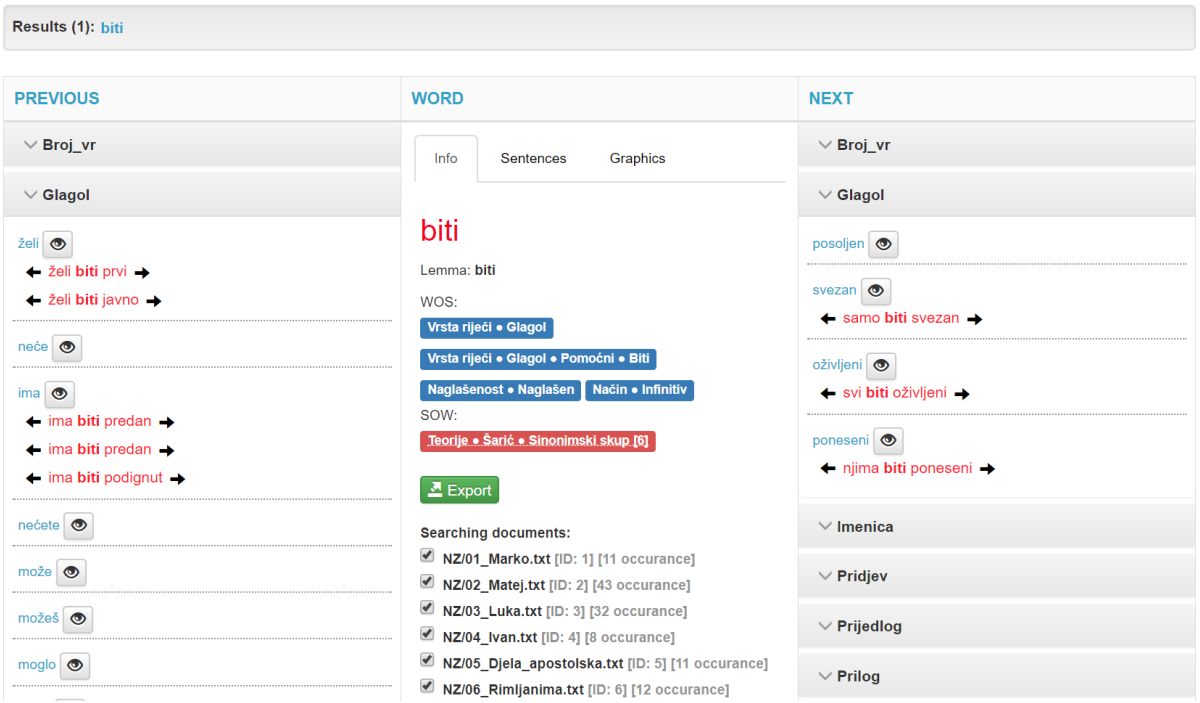


Figure 60: Triples crawling

It is possible to create a group of patterns for all WOS/SOW features for the observed triples (by clicking on the button 'Export'), which is an important upgrade to the collocation concept, i.e. it is a foundation of deep syntax of Melčuk's functions.

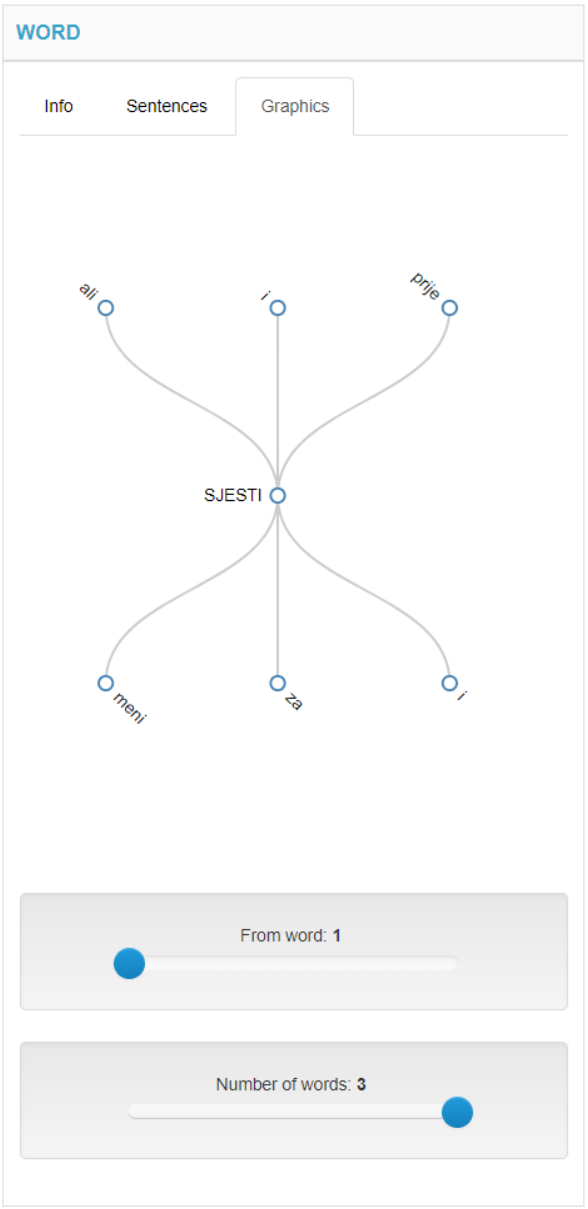


Figure 61: Visualisation of sentence segments

Triples can also be visualized like a graph where the observed word is located in the center of the graph (as shown in Figure 61) while predecessors and followers are located relatively to the central word. Number of words in the output graph can be adjusted using slidebars under the graph image.

7.3. Conceptual structures

In semantics, conceptual structure stands for a single level of cognitive (mental) representation postulated by Ray Jackendoff [83]. Conceptual structure represents concepts in terms of a small number of conceptual elements. The conceptual metaphors are cognitive processes for construction of meaning [93]. Since the meaning has its source in knowledge (which can be either empirical or cognitive), it implies that the construction of a new meaning necessarily has its source there too, [94, 90] which enables the creation of new knowledge. The knowledge source is also called a semantic domain. The conceptual metaphor, therefore, would be mapping of one semantic domain to another. The consequence of such mapping is that the target domain is more understandable based on the knowledge of the source domain [92]. The mapping itself is realized through metaphorical language expression, i.e. using terms from the source one for the target semantic domain. The constant usage of links between the same semantic (conceptual) domains leads to their establishment, therefore (from linguistics point of view) it is possible to treat them like the established paradigmatic knowledge, which (unconsciously) is used in construction of language expressions – metaphors are becoming phrases and idioms [153]. Namely, 'conceptual integration theory' [57], which is also known as 'blending theory' introduces a new, abstract layer of a conceptual domain, the so called 'mental space'. Mental space is a small conceptual package constructed as we think and talk, for purposes of local understanding and can be used generally for modelling of dynamical mappings in thought and language. One mental space may induce at least one more space, like 'opening the space' in the same way as verbs are doing in the sentence syntax. Since 'blending theory' is more general than 'conceptual theory of metaphors', its implementation in machine form is much more complicated. Back in 1998, the attempt was made through the cognitive network [57], and later also with some other realizations [172], with a goal of building, and not detecting metaphors. This thesis proposes a new approach which generalizes mental space to complete mathematical abstraction, and then works with mathematical sets of data, whose elements are structures of integers (or floating point) numbers. Although, written as a number structure, information is assigned to words in the natural language lexicon. Each word in that lexicon has numerous features, grammatical and semantic at the same time and both are interconnected which makes an ontology.

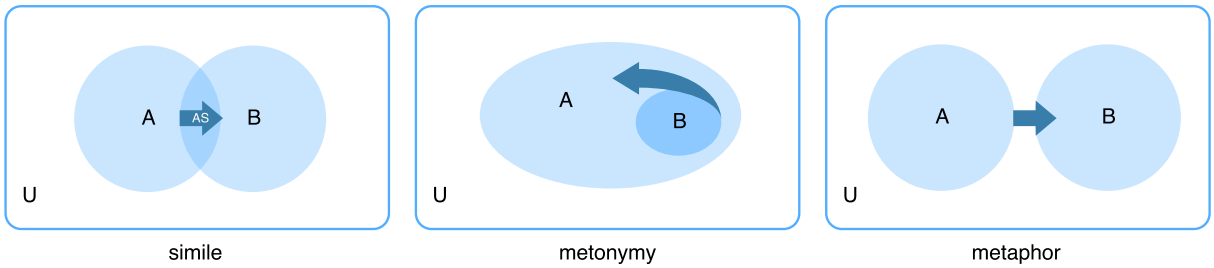


Figure 62: Conceptual structures

In this way, the connectivity of words turns into a connectivity of their features, from individual to the complete set. Figure 62 shows three common types of conceptual structures and their mappings in terms of semantic domains. If the mapping is only partial, we deal with similes. The similes are types of conceptual structures that compare two different things, from different semantic domains (e.g. expression 'busy as a bee'), and if mapping is complete then we deal with metaphors (all elements from one set are mapped to another set). Between the set interception (simile) and partition mapping (metaphor) is the so called 'real subset' which can be used to detect metonymy [21]. By using only some features (only parts of set elements), it is possible to describe the complete set (e.g. car wheel, person's face, etc.). Why some features get considered, and other don't, becomes clear if we look at the hierarchical structure which every element holds – ones are dominant (core), while others aren't. Dominant are usually these which give greater diversity, e.g. face instead of a leg or an arm. Such approach also enables the machine processing of other semantic variables, e.g. synonyms. The words will be more similar, as more dominant features they have in common, and at least one dominant feature that differs them. To use the same approach to detect e.g. antonyms, it is necessary to have an ordered set, which is determined by the core of the number – for every two integers it is possible to tell which one is larger. This solves the problem of metaphors type 'More is up', because sets of amounts and heights are only sets of numbers, e.g. for three levels we will have a set of 1-2-3. If both terms take the same (numerical) level, then the metaphor is detected. It is important to have features that have a hierarchical structure, and the rest will be done by the numbers on which the computer work is based. The machine cannot know if the word has any metaphorical meaning or not. Therefore, it is necessary to manually tag such words with appropriate SOW tags (e.g. Symbol). The SSF has a special tag branch called cro. '*Jezgreni*' (eng. Core) containing individual words or domains in which it is placed. For example, the word cro. '*kruna*' (eng. crown) could contain core verbs: cro. '*stajati*' (eng. stand) or cro. '*nositi*' (eng. wear) (because these are its core verbs), as well as cro. '*željezni*' (eng. iron), cro. '*zlatni*' (eng. golden), cro. '*kraljevski*' (eng. king's) as core adjectives or cro. '*nakit*' (eng. jewlery), cro. '*kralj*' (eng. king), cro. '*kraljica*' (eng. queen), cro. '*princ*' (eng. prince), cro. '*princeza*' (eng. princess), as a core noun. As an

example, we can take the sentence cro. '*Kruna je dala odobrenje*' (eng. The crown gave the approval). Since the crown cannot give anything to anyone (the verb *to give* is not its core verb), in the given example the crown is a reference to the Queen of England. The lemma of the word crown inside the SSF's lexicon is tagged with a SOW mark as a Symbol (SOW ID 303). When such word is detected, the algorithm finds core verbs which correspond to the observed word. For example, in case of the word *crown*, core verbs could be: *to wear*, *to place*, *to make*, etc... When the word which is marked with a SOW tag as a symbol is followed by a verb which is at the same time not the part of the core verbs domain, metonymy is detected. In above example, the symbol *crown* is followed by the verb *to give*, which is not the core verb of the symbol *crown*.

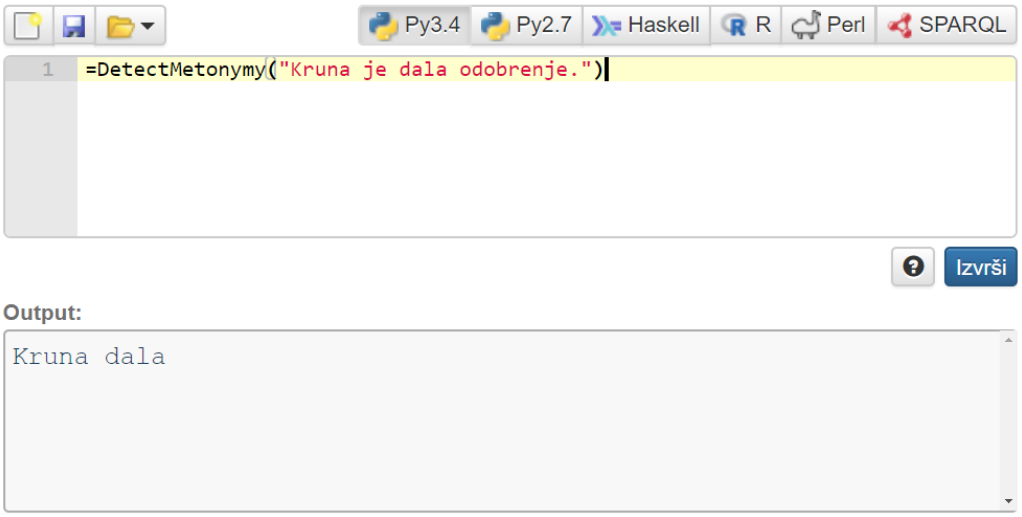


Figure 63: Metonymy detection in the SSF

Figure 63 shows an example of metonymy detection through the NLF function `DetectMetonymy()`. The live example was demonstrated at the Third International Symposium on Figurative Thought and Language in Osijek [124]. Similarly, the word which is enriched with core domains (which contains four types of lemmatized words: nouns, verbs, adjectives and pronouns as described in Section 6.3) when surrounded with words from some of domains can be recognized as a metaphor. The algorithm for extraction of metonymy and metaphors from the text is as follows:

1. Extract sentences from the corpus (problem of abbreviations).
2. Split complex and compound sentences into two or more simple phrases.
3. Detect functional parts from each and every simple sentence (Subject, Predicate, Object).

4. Enrich sentence with WOS/SOW tags (for metonymy WOS tags for verbs, and SOW tags for symbols and core verbs, and for metaphors core domains);

5. Detect sentences that match the search criteria for metonymy or metaphor (as described in Section 7.4).

6. Based on matching, conclude which type of conceptual metonymy and metaphor is detected: if the word has a metaphorical symbol and at the same time does not have a core verb then the metonymy is detected.

7. If the sentence does not have a core domain and has a metaphorical symbol, then the metaphor is detected.

For such algorithm to work properly it is necessary to assign proper tags to all words. Together with these automatically extracted relations, MWE lexicon holds multiword that were created by the human. That enables an algorithm to check into local metaphor repository if the given expression is a metaphor, before going into any deeper analysis. One such repository is the Croatian Metaphor Repository made within the Croatian Science Foundation (CSF) project, led by Kristina Štrkalj Despot [155].

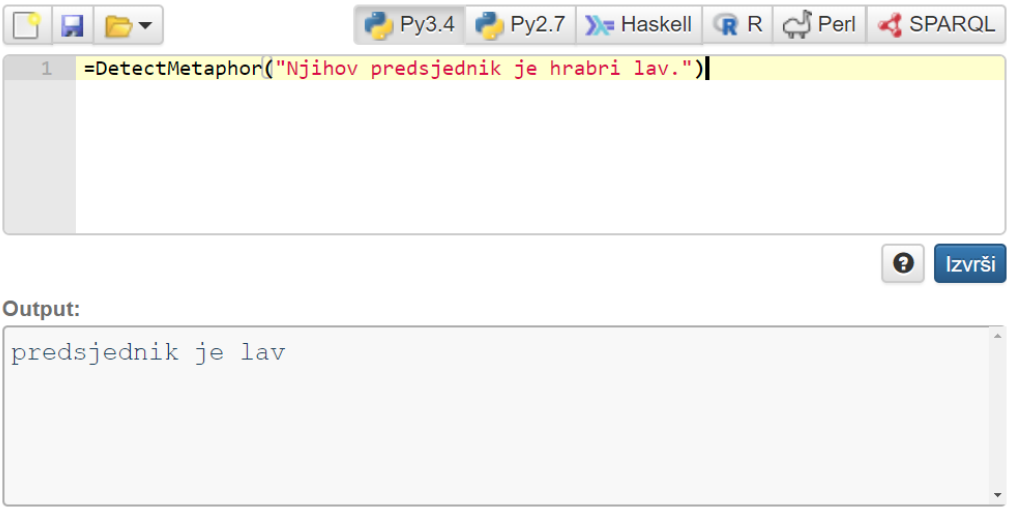


Figure 64: Metaphor detection in the SSF

Figure 64 shows function `DetectMetaphor()` which analyses the given sentence for metaphors. In the first step, the function extracts SPO roles from the sentence. For the sentence cro. '*Njihov predsjednik je hrabri lav*' (eng. Their president is brave lion) the subject is cro. '*predsjednik*' (eng. president), the predicate is cro. '*je*' (eng. is) the object is cro. '*lav*' (eng. lion). If the word in the SPO is tagged as a metaphorical symbol , as in the above example the word cro. '*lav*' is, then the function checks words which form associated core

domains for the given word. For example, the word cro. '*lav*' is associated with several domains which contain words such as 'animal', 'jungle', 'dangerous', 'mane', etc. but none of the domains contains the word 'president'. If the observed word is not in the domains (and in above example, the word 'president' is not), the metaphor is detected. In order for this functions to work properly, it is necessary to have a lexicon which is properly tagged with metaphorical symbols and core domains.

7.4. Extraction of relations using O-structures

O-structures are extension to a WOS/SOW tagging concept (therefore their name is a letter which is obtained by the intersection of abbreviations WOS and SOW). They represent a specific WOS/SOW pattern which can be either a part or a whole sentence. Figure 65 shows one sentence cro. '*Iva će sutra ići u vrtić*' (eng. Iva will go to the kindergarten tomorrow) which can be observed as a linked list of words. Since each word has WOS/SOW tags associated to it, instead of looking at the words interrelations, tags and their interrelations can be observed. Parts of such patterns which represent a specific figure or has a meaning can be stored in a form of a O-structures.

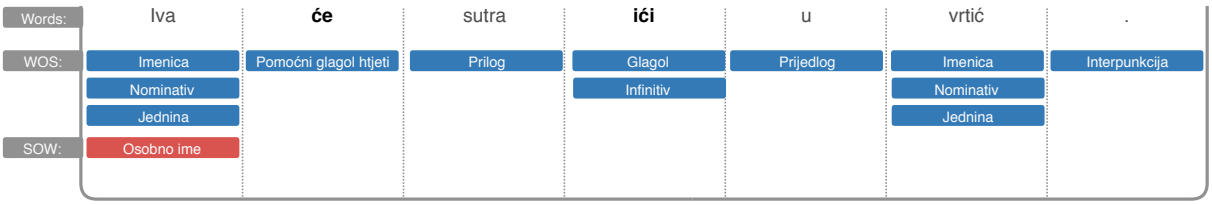


Figure 65: Example of sentence tagging patterns

For example, one of O-structures used for extraction of the Future tense in Croatian language (called *futur*) in the SSF looks as shown in Figure 67. When matched against the sentence, as shown in Figure 66, highlighted words point out.

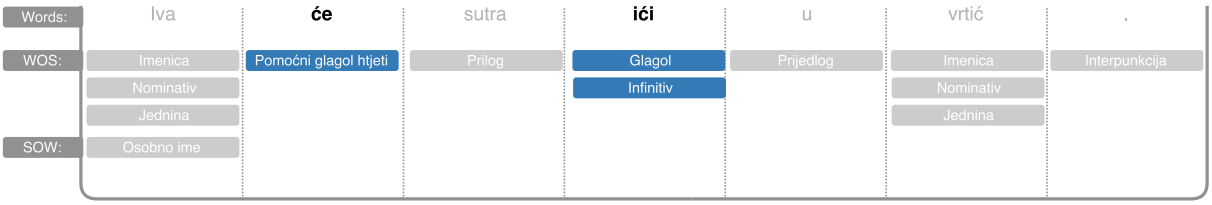


Figure 66: Future tense O-structure pattern

Figure 67 shows only one variant of the O-structure called '*futur*', which matches sentences that matches specific regular expression pattern. As the first pattern element is `.*?` which

is basically any word, followed by the word tagged as a '*Pomoćni glagol htjeti*' (WOS ID 153), after which any other word may or may not appear (also `.*?`), and then the word tagged as a verb in infinitive, followed by any other word.

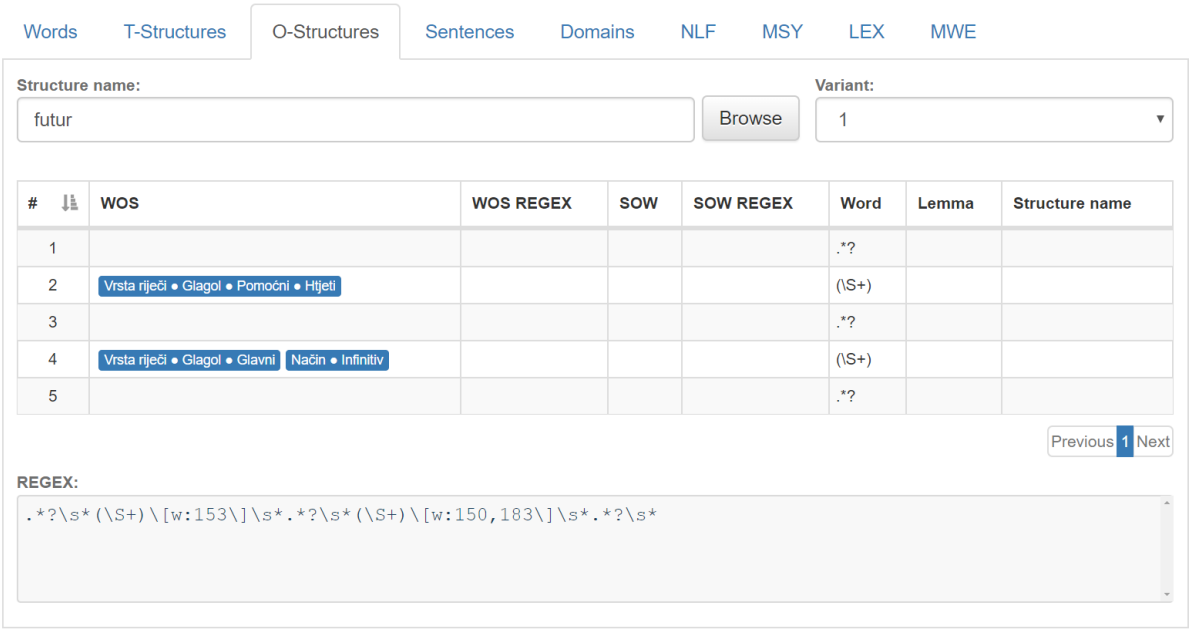


Figure 67: Screenshot of O-structures editor

Similarly to the matching algorithm described in Section 5.6, the sentence is enriched with WOS/SOW features and then matched against regular expression which in an example above will look like:

```
.*?\s*(\S+)\[w:153\]\s*.*?\s*(\S+)\[w:150,183\]\s*.*?\s*
```

And then when matched against the enriched version of the sentence which looks like:

```
Iva␣će[w:153]␣sutra␣ići[w:150,183]␣u␣vrtić
```

will result with matching words 'Iva **će** sutra **ići** u vrtić' as shown in Figure 66. In a similar way extraction of relations can be done in the Natural language functions subsystem. For example, three functions called `DetectS()`, `DetectP()`, and `DetectO()` will be used. These functions use O-structure in order to extract subject, predicate and object. For example, to detect predicate in the simple sentence cro. '*Ivan čita knjigu*' over 10 different pattern variants can be defined for predicate detection. For the mentioned example, pattern variant no. 5 of predicate pattern, as shown in Table 22 is used. It defines such sentence pattern where at the position #1 can (but not necessary) be any word. At position #2 must be a word which is tagged with WOS tag 2 (noun), followed by any other word at position #3. At the position #4 is word tagged with WOS tag 5 (verb) which is captured into a resulting group. At the position #5 is another word tagged with WOS tag 2 (noun)

which is also captured in a group, and at the last position #6 any other word.

Table 22: O-structures for predicate extraction

#	pattern_tag	pattern_variant	wos	sow	lemma	word
1.	predicate	5				.*?
2.	predicate	5	2			.*?
3.	predicate	5				.*?
4.	predicate	5	5			(\S+)
5.	predicate	5	2			(\S+)
6.	predicate	5				.*?

Such table structure is row by row transformed (serialized) into a regular expression:

```
.*?\s*.*?\[w:2\]\s*.*?\s*(\S+)\[w:5\]\s*(\S+)\[w:2\]\s*.*?\s*
```

and when mached over the enriched version of the sentence:

```
Ivan[w:2]␣čita[w:5]␣knjigu[w:2]
```

The resulting group is: `čita knjigu`. In a similar way, all other O-structures matchings are performed. The result of `DetectS()`, `DetectP()` and `DetectO()` functions is shown in Figure 68. List of all other NLF functions that deal with sentence syntax can be found in Appendix D

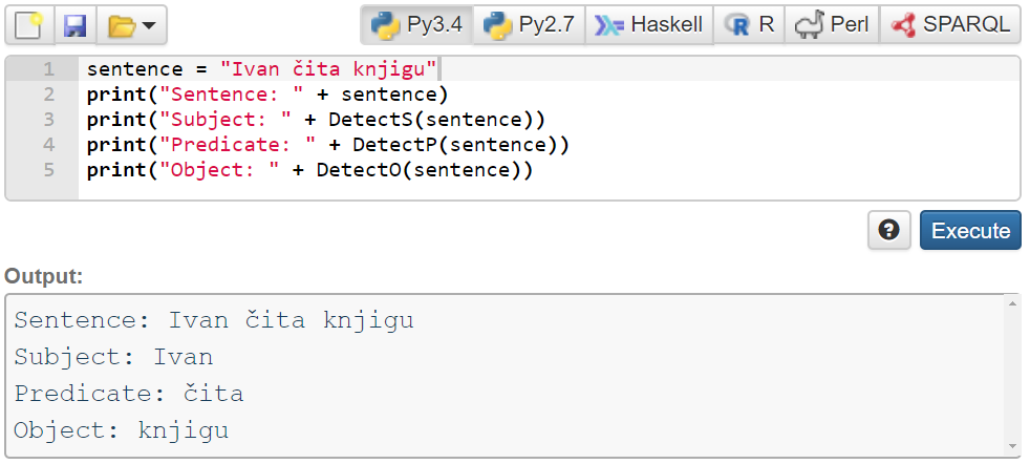


Figure 68: Division of a simple sentence into SPO

Lexical relations from corpora can also be extracted using web interface (as shown in Figure 69) which is performed in three steps: query forming, sentence enrichment and matching. In the first step a user, using a web form of the SSF creates a query by entering a word (or its part, by using valid regular expressions). Along with the morphological

form of the word, a user can also select WOS/SOW tags that every matched word needs to have. In the example shown in Figure 69, the first word needs to start with the letter O, and must be tagged with a WOS tag cro. '*zamjenica*' (eng. pronoun). As well as with the word features, it is necessary to enter the number of occurrences of searched word. The number of occurrences is an integer with values from 0 to N. If the value of 0 is entered, then the current word can be repeated from 0 to N times. After the pronoun, as the next search element shown in Figure 69, the user defined a wildcard that matches any word with a number of occurrences from 0 to N, followed by a word which has SOW tag on it. The selected SOW tag is named cro. '*sinonim*' (eng. synonym) with a value cro. '*mama*' (eng. mom). This means that any word from the same synonymy set (e.g. cro. '*majka*', (eng. mother)) will be matched. As the last element of a search query, the user again entered a wildcard that matches any word with a 0 to N number of occurrences.

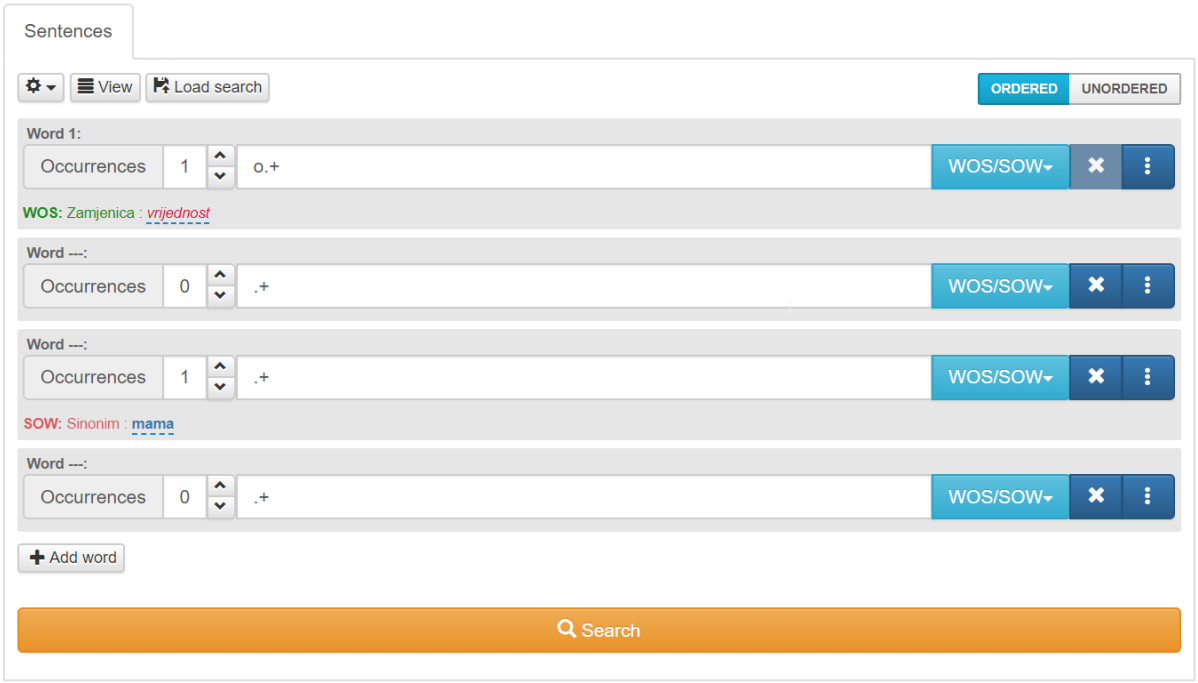


Figure 69: Screenshot of the web form for pattern search

After a form is submitted, in a process of query pattern generation, the following regular expression (for a given example) is formed:

```
/^^(o\S+?)[^\s]*\[w:3\].+␣(\S+?)[^\s]*\[s:104.*:mama:.*\].+$$/iUsx
```

Before matching procedure can be executed, plain sentences from the selected corpus must be enriched with proper WOS/SOW tags. Only those tags that are used in search query are relevant and included into the enrichment process. In the above example, one WOS tag (pronoun cro. *zamjenica* with ID 3) and one SOW tag (synonym cro. *sinonim* with ID number 104) is relevant. Therefore, every sentence from the selected corpus is extended

only with these tags. After enrichment process is done sentences have additional marks near every word. One sentence that meets regular expression criteria in its enriched form looks like this:

```
Ona[w:3]␣mu[w:3]␣je␣majka[s:104:mama:mater:majka:mati]␣.
```

The sentence is significantly expanded, and every word is now followed with relevant WOS/SOW marks. Enrichment procedure is then iterated over selected corpus and those sentences that have a match with a regular expression from the first step, are displayed as a result (as shown in Figure 70).

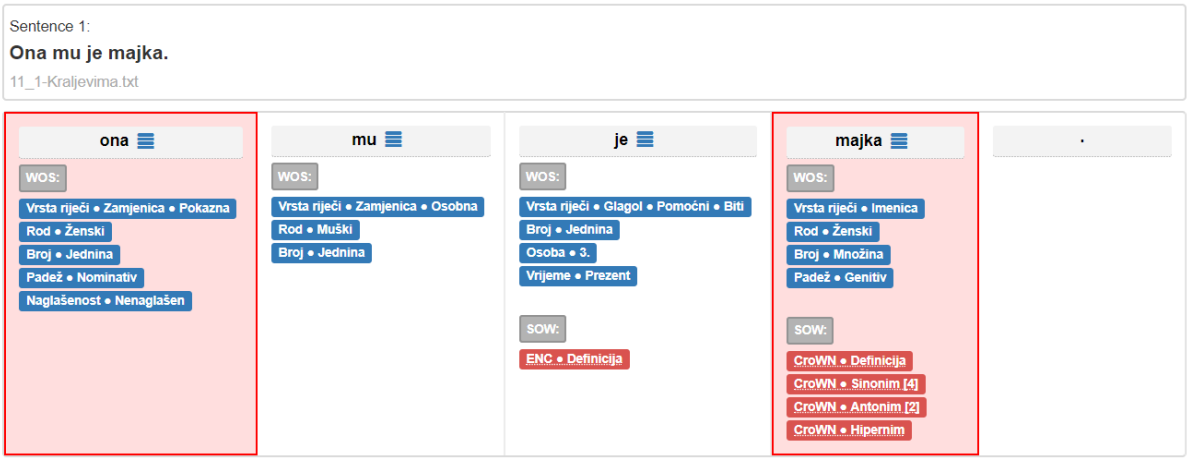


Figure 70: Screenshot of the results of pattern searching

There are two types of sentences search: ordered and unordered. Ordered search, takes into account the word order that is searched within the sentence, while the unordered search minds only that the words that meet the search criteria appear in the sentence, inrespective of their position.

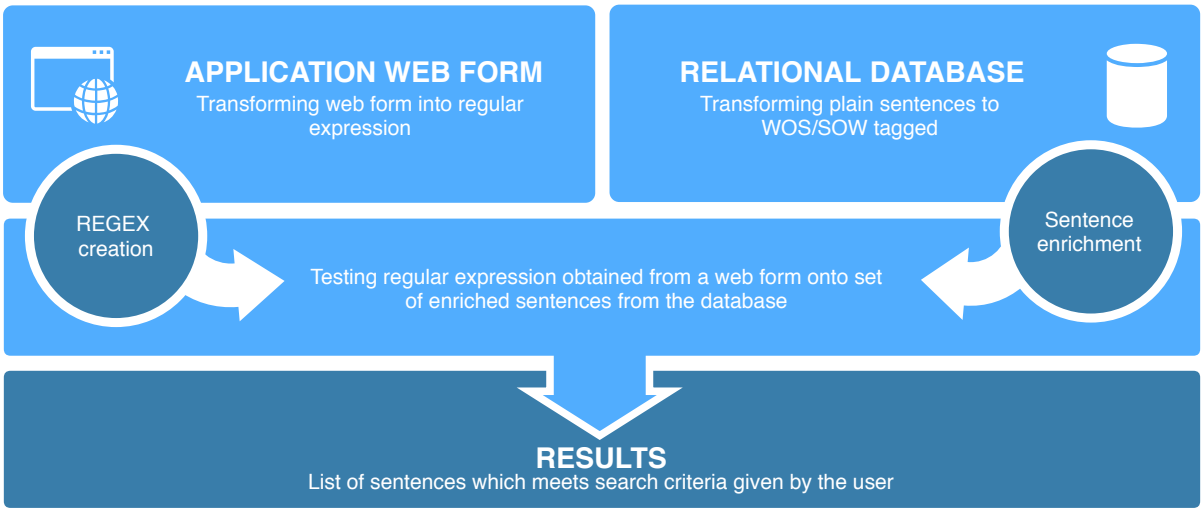


Figure 71: Process of extraction of sentences from the corpus

Figure 71 shows the whole process of extraction of sentences from the corpus, as described in the example above. WOS and SOW integration forms the new O-structures, which follows the sentence syntax. By storing and later using of such patterns it is possible to extract lexical information from the previously unknown corpus, which confirms hypothesis **H2** the regarding extraction of syntactic and semantic relation, and also gives an answer to the research question **Q2**, that it is possible to develop such network framework.

7.5. Artifact API functions

Every system that tends to be interoperable with a wider software ecosystem must provide an API interface. The SSF exposes all of its Natural Language Functions over the representational state transfer (REST) API. The REST was defined by Roy Fielding in his PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures" [61] where he proposed standard API structure which would enable interaction between computer programs and allow them to exchange information. Modern web applications commonly apply the design of APIs in the REST style and for that reason are called 'RESTful' [114]. The SSF's API endpoint is available on the (`http://www.ss-framework.com/api/`). The user (application that uses the API) has to provide at least three parameters: the key obtained from the SSF system, the one of NLF programs (Pyhton, Haskell, Perl, SPARQL or R) and the code to be executed (written in one of these languages). The parameters are sent in the HTTP POST request. After executing such HTTP request, the SSF returns the message in JSON format, which is commonly used format nowadays. Listing 1 shows an example of one such call. The function `ChangeTense()` which is standard NLF function is called over the API. The function accepts two parameters: the sentence and the tense to which the sentence should be transformed. After the HTTP request with all these parameters is sent to the SSF, the NLF module executes the provided program code and returns a result in a JSON form. Although the example is written in Python, the same logic applies to any other programming language which is capable of calling HTTP requests. The complete list of NLF functions by each programming language is described in Appendix D.

Listing 1: API call example

```
# -*- coding: utf-8 -*-
import requests

# API key
key = "429ae4bfe8088f071abef86ac021653b"

# Execute in Python, Haskel, SPARQL, R
program = "Python"

# Code to be executed
code = "=ChangeTense('vidim plavu kuću', 'aorist')"
url = "http://www.ss-framework.com/api/fpj"

req = requests.post(url, data = {'apiKey':key, 'program':program,
    'code':code})

# Set the output encoding to UTF-8
req.encoding = 'UTF-8'

# Print the output
print req.text
```

After execution of above example, two variables are returned: the status of the request and the result. List of possible statues is inspired by the HTTP protocol (defined in RFC 2616 [62]) where `1xx` statuses are of informal type, `2xx` are success statuses and `5xx` statuses denotes that error occurred. The output looks like this:

```
{"status":200, "result":"vidjeh plavu kuću"}
```

The `status` 200 indicates success, and the variable `result` contains the output of the `ChangeTense()` function. The same result would be if the function was called through the NLF module within the SSF GUI.

APIs are powerful tools for integration with various third party application. One such example, is a simple game which is oriented towards children who are learning how to write. The game is developed in PHP programming language by Juraj Benić and is a very simple example on how SSF is used in other external system. When the user enters the game he gets simple sentences where some of words are replaced with images or sounds. Below such words there is a text box where the child needs to enter the missing word.

When the word is correctly written it indicates that the answer is correct and offers a step to the next level.



Figure 72: API call diagram

This simple game uses SSF's API interface for obtaining images and sounds for words. The administrator of the game enters a list of sentences, and for every word which needs to be replaced with either image or a sound puts an SOW id near it. When the user approaches the start screen of the game, the system in the backend connects to the SSF's API endpoint and sends HTTP request for every word, containing call for the function `GetSOW(word, sowid)` (as shown in Listing 1). The SSF looks for the asked word in its lexicon and if the word is tagged with defined SOW tags, it returns a string value, containing an URL of asked resource (image or sound). Figure 72 shows a diagram of these steps. This rather simple application shows how a rich lexicon and complex linguistic functions which are a part of the SSF can easily be integrated in any external application. In a similar way the SSF's NLFs can be used in other systems dealing with the lexicographic content (e.g. name entity recognition, detection of tropes, machine translation, etc.).

8. Semantic Web integration

Traditional forms of data representation like web pages were primarily oriented toward easier human readability. Although, such representation is suitable for humans it is almost unusable in terms of automated data processing and interoperability. As an upgrade to a World Wide Web (WWW) concept, its author Tim Berner Lee together with James Hendler and Ora Lassila described the new web of data (i.e. Semantic Web) [11]. It uses the already existing infrastructure of the WWW to define specific terms from the real world and enables the same resource to be described in multiple instances which together forms a global network of information. The Semantic Web goal was not to develop intelligent agents that could make decisions and act but rather to develop a technical platform for development of such systems that could use organized knowledge and support humans. Since Semantic Web focuses only on the technical infrastructure, many researches worldwide made effort to standardize ontology languages for publishing on the Semantic Web. In that sense the Web Ontology Language (OWL) was designed to represent complex knowledge about things, groups of things, and relations between things. OWL is currently an official recommendation by the World Wide Web Consortium (W3C) [116]. On top of it, researchers have developed a model for data representation in a form of triples (*subject, predicate, object*) which is referenceable and de-referenceable using standard web protocols and is built on the top of already existing web standards like Resource Description Framework (RDF) [108]. The RDF is a data model for describing resources on the web in a structured way. Each resource has a Uniform Resource Identifier (URI) which identifies it. For example, the URI `http://www.ss-framework.com/owl/word/majka` identifies the word cro. *majka* (eng. mother) in the SSF and if accessed through the web browser will show all the information about it. Since RDF is essentially a graph of triples, each node can be three kinds of nodes: a URI, literal (for textual and numeric values) and *blank node* which can only be used for a subject or an object in cases where the URI or literal is not given. The knowledge that is used to classify the terms, their relationships and constrains which are used in Semantic Web is usually defined in ontologies. Ontologies play a key role in the knowledge interpretation. Each ontology is modelled based on its purpose. How deep ontology will be in the end depends on the application that utilizes that ontology. One of the main features and ideas behind the linked data is a possibility of extending them over time, not only in base ontology but globally. With that in mind during ontology modelling it's possible to define only these hierarchies and individuals that are relevant for the application that is being developed. Other applications or users could then take these individuals and connect them to other same individuals using relation *owl:sameAs*, and then extend them with additional information which is relevant for them. The publication

of linguistic data in a form of Linked Data does not solve the problem of interoperability *per se*, so the links toward other resources (e.g. DBpedia, LexInfo and WordNet) are made within the SSF. Such newly generated data networks are especially valuable since they contain linguistic knowledge for a plethora of languages. This section describes the development of linguistic ontology (as a part of the SSF) that tends to become a part of global Linguistic Linked Open Data cloud (described in Section 8.4).

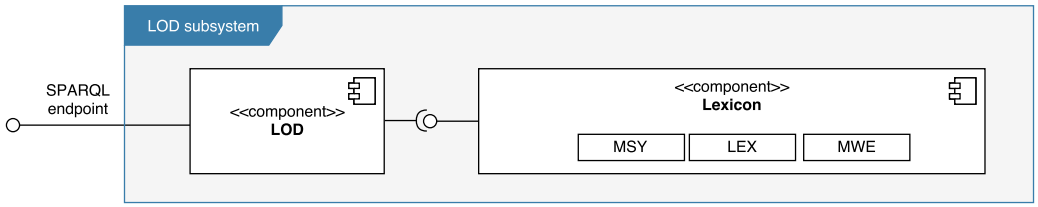


Figure 73: Conceptual model of the LOD subsystem

When considering methods of transforming relational data into RDF triples there are two most commonly used approaches. The first is to use triple store application (e.g. Virtuoso Server) and then periodically synchronize relational database with it. The second approach is to simply use a LOD wrapper (e.g. D2RQ) and represent a relational data as a RDF in a real time without any unnecessary redundancy. This is an answer to research question **Q5**. In cases where there is no need to update the data this approach is certainly a better choice since it avoids synchronization and the data available through a RDF endpoint are the same as the relational database in any time. In this thesis both ways will be demonstrated. In Section 8.1, it is demonstrated how the SSF's lexicon can be easily served as a RDF triples in a read-only mode, using D2RQ platform, and Section 8.2 deals with another viable option of synchronizing data to a Virtuoso triplestore. Regardless the chosen approach the first step is to model an ontology. From technical point of view, ontology can be built using only text editor tools (e.g. Notepad or vi), but it requires special knowledge on how this ontology file should be structured in the end. An easier way for modelling an ontology is by using of special tools like *Protégé* which has graphical user interface for an easier and more user friendly ontology modelling. *Protégé* was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine and is offered publicly in a form of free open source tool. It is fully compatible with OWL2 ontology and offers modelling and visualization of ontologies and databases. Figure 74 shows user interface of the *Protégé* platform with one lexical entry (`http://www.ss-framework.com/owl/word/barcelona`). The same triple is shown in Figure 75 in a form of an ontology graph. The complete SSF's ontology currently contains over 65.000 triples (which is 8% of SSF's lexicon), and is growing daily.

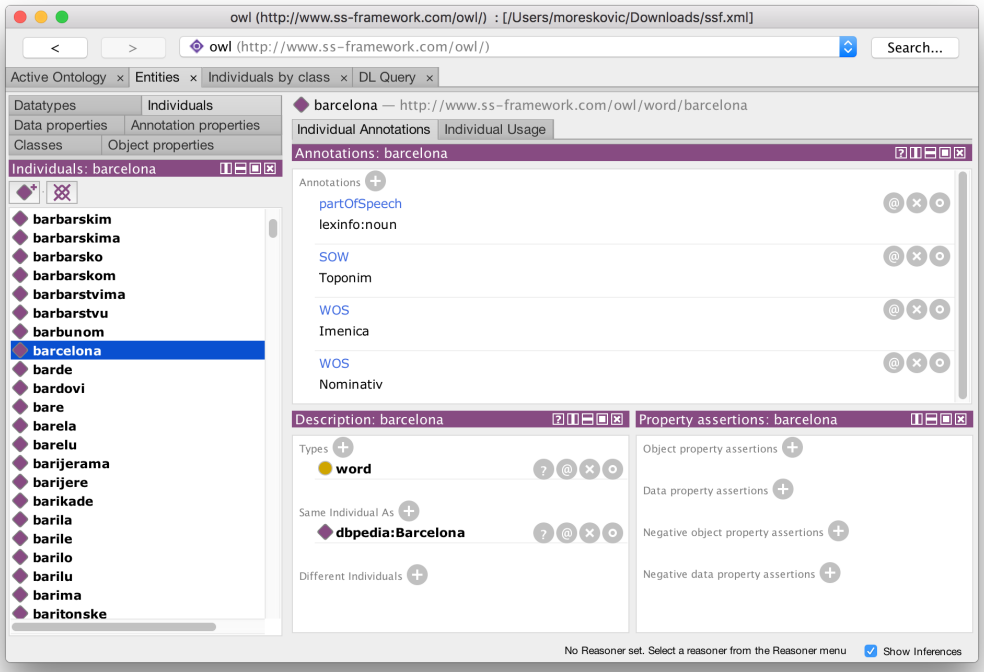


Figure 74: Building Linguistic Ontology in Protégé

Class `word` holds information about lexicon words. There are two data properties, `SOW` and `WOS`. Every individual in the ontology has at least one `WOS` or `SOW` property assigned, and many of them are also linked to LexInfo and DBpedia. Figure 75 shows ontology graph of one triple with links to external sources.

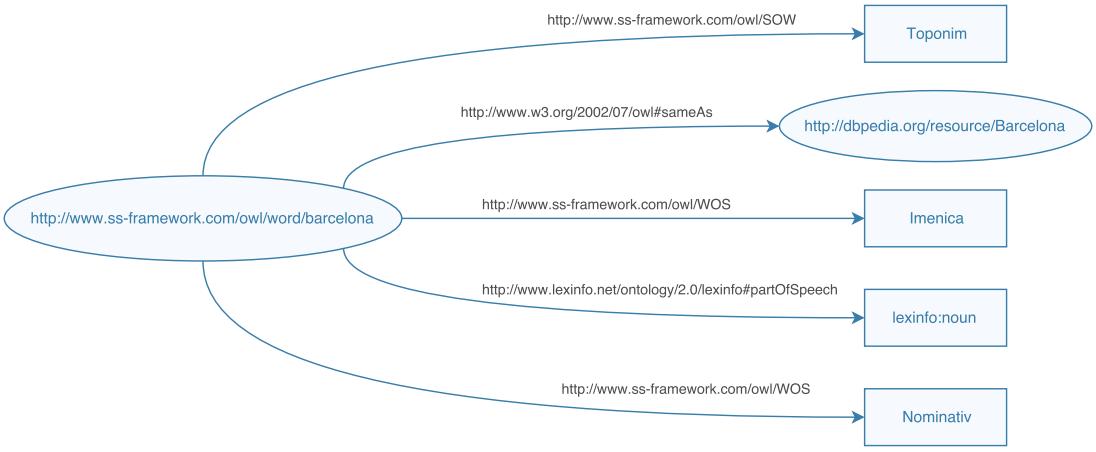


Figure 75: Graph representation of one lexical entry in RDF

8.1. LOD wrapper

The first step in implementing a D2RQ wrapper was to prepare the relational data to be described in a RDF form of triples (subject-predicate-object). The SSF's lexicon could be transformed in a way that each word can be observed as a subject, and its tags as an object. The predicate which connects them can be either `WOS` or `SOW`. The easiest way of transforming the Lexical data (the segment of the ER diagram related to this issue is shown in Figure 15) to the RDF form is to create a database `VIEW` which output corresponds to a subject-predicate-object form. The view `hasWOS` in a SQL notation looks like this:

```
CREATE VIEW `hasWOS` AS
    SELECT
        `w`.`word` AS `word`, `wos`.`wos_name` AS `wos_name`
    FROM
        ((`word_has_wos` `whw`
        LEFT JOIN `words` `w` ON ((`whw`.`wordid` = `w`.`wordid`)))
        LEFT JOIN `wos` ON ((`whw`.`wosid` = `wos`.`wosid`)))
    ORDER BY `w`.`word`
```

and the view for `hasSOW` looks like this:

```
CREATE VIEW `hasSOW` AS
    SELECT
        `w`.`word` AS `word`, `sow`.`sow_name` AS `sow_name`
    FROM
        ((`word_has_sow` `whs`
        LEFT JOIN `words` `w` ON ((`whs`.`wordid` = `w`.`wordid`)))
        LEFT JOIN `sow` ON ((`whs`.`sowid` = `sow`.`sowid`)))
    ORDER BY `w`.`word`
```

When executed these queries result in two columns. The first is the word itself and the second is WOS or SOW attribute the word has been tagged with. Such output is a good foundation for mapping to a subject-predicate-object format.

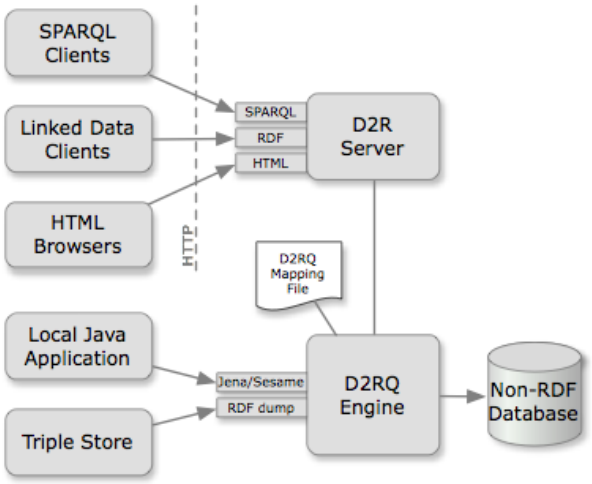


Figure 76: The architecture of D2RQ platform [17]

As shown in Figure 76, the D2RQ uses special mapping file which defines which attribute/table/view of the relational database relates to a RDF field. The example of such mapping file which is used for SFF's lexicon transformation is shown in Listing 2.

Listing 2: D2RQ mapping file

```
@prefix map: <#> .
@prefix db: <> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix ssf: <http://www.ss-framework.com/owl/> .
@prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#> .

map:Word a d2rq:ClassMap;
    d2rq:dataStorage map:database;
    d2rq:class ssf:Word;
    d2rq:uriPattern "ssf:word/@@hasWOS.word@@";
   d2rq:classDefinitionLabel "Word"
    .
map:WOS a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Word;
    d2rq:property ssf:WOS;
    d2rq:column "hasWOS.wos_name";
    .

map:SOW a d2rq:PropertyBridge;
```

```
    d2rq:belongsToClassMap map:Word;
    d2rq:property ssf:SOW;
   #d2rq:refersToClassMap map:Word;
   d2rq:column "hasSOW.sow_name";
    d2rq:join "hasSOW.word => hasWOS.word";
    .

map:ExtendedBridge a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Word;
    d2rq:property lexinfo:partOfSpeech;
    d2rq:column "hasWOS.wos_name";
    d2rq:translateWith map:ExtendedTable;
    .
map:ExtendedTable a d2rq:TranslationTable;
    d2rq:translation [ d2rq:databaseValue "Imenica"; d2rq:rdfValue
        lexinfo:Noun; ];
   d2rq:translation [ d2rq:databaseValue "Glagol"; d2rq:rdfValue
      lexinfo:Verb; ];
   d2rq:translation [ d2rq:databaseValue "Nominativ"; d2rq:rdfValue
      lexinfo:nominativeCase; ];
   d2rq:translation [ d2rq:databaseValue "Genitiv"; d2rq:rdfValue
      lexinfo:genitiveCase; ];
   d2rq:translation [ d2rq:databaseValue "Dativ"; d2rq:rdfValue
      lexinfo:dativeCase; ];
   d2rq:translation [ d2rq:databaseValue "Akuzativ"; d2rq:rdfValue
      lexinfo:accusativeCase; ];
   d2rq:translation [ d2rq:databaseValue "Vokativ"; d2rq:rdfValue
      lexinfo:vocativeCase; ];
   d2rq:translation [ d2rq:databaseValue "Lokativ"; d2rq:rdfValue
      lexinfo:locativeCase; ];
   d2rq:translation [ d2rq:databaseValue "Instrumental"; d2rq:rdfValue
      lexinfo:instrumentalCase; ];
   d2rq:translation [ d2rq:databaseValue "Pridjev"; d2rq:rdfValue
      lexinfo:Adjective; ];
   d2rq:translation [ d2rq:databaseValue "Zamjenica"; d2rq:rdfValue
      lexinfo:Pronoun; ];
   d2rq:translation [ d2rq:databaseValue "Veznik"; d2rq:rdfValue
      lexinfo:Conjunction; ];
    .
```

At the beginning of the mapping file are definitions of prefixes. The SSF's prefix is defined as `http://www.ss-framework.com/owl/`. After prefixes, the set of rules for defining classes and relational database mappings are defined. Since one of the Linked Data requirements is that the entities should be linked to other resources, the SSF's is linked to the LexInfo ontology which was defined to provide data categories for the Lemon model.

8.2. Virtuoso triplestore

Another approach to RDF representation of relational database is by synchronization to a triple store server. For this purpose, the Virtuoso triplestore is chosen. After the ontology model is built triple data is exported from MySQL database and converted to RDF/XML and N-Triples format which are both suitable for loading into Virtuoso triplestore. N-Triples is a line-based, plain text serialization format for RDF (Resource Description Framework) graphs, and a subset of the Turtle (Terse RDF Triple Language) format. It is the simplest way of storing RDF triples in textual files. Each line holds one triple with subject, relation and predicate delimited with space. Similar to N-triples is a Turtle serialization type which allows multiple lines to be used for storing triples' information. RDF/XML format is defined by the W3C for serialization of RDF graphs in an XML syntax. Since it is less human readable than N-Triples or Turtle it is not used very often.

To achieve data transformation from the relation database to RDF triples a transforming PHP script has been developed (Appendix B). The script connects to MySQL database and iterates the whole dictionary. The observed word becomes a subject of a RDF triple, WOS/SOW relation is a predicate, and the value of WOS/SOW tag is an object. All links to external resources (e.g. DBpedia, LexInfo, WordNet or BabelNet) are also stored in WOS/SOW tags and therefore automatically a part of resulting ontology. The end results are N-triple and RDF/XML files which can then be imported into Virtuoso. In a case of RDF/XML, the content of output file for one sample triple is shown below:

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:lexinfo="http://www.lexinfo.net/ontology/2.0/lexinfo#"
xmlns:ssf="http://www.ss-framework.com/owl/"
xml:base="http://www.ss-framework.com/owl/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<owl:Ontology rdf:about="http://www.ss-framework.com/owl/"/>
```

```
<owl:DatatypeProperty rdf:about="http://www.ss-framework.com/owl/sow"/>
<owl:DatatypeProperty rdf:about="http://www.ss-framework.com/owl/wos"/>
<owl:Class rdf:about="http://www.ss-framework.com/owl/word"/>
   <rdf:Description
       rdf:about="http://www.ss-framework.com/owl/word/barcelona">
   <rdf:type rdf:resource="http://www.ss-framework.com/owl/word"/>
      <ssf:SOW>Toponim</ssf:SOW>
      <owl:sameAs rdf:resource="http://dbpedia.org/resource/Barcelona"/>
      <ssf:WOS>Imenica</ssf:WOS>
      <lexinfo:partOfSpeech>lexinfo:noun</lexinfo:partOfSpeech>
      <ssf:WOS>Nominativ</ssf:WOS>
      <ssf:WOS>Ženski</ssf:WOS>
   </rdf:Description>
</rdf:RDF>
```

The W3C consortium offers an RDF Validator web application in the `https://www.w3.org/RDF/Validator`. Figure 77 shows the results of RDF validation for the above example.

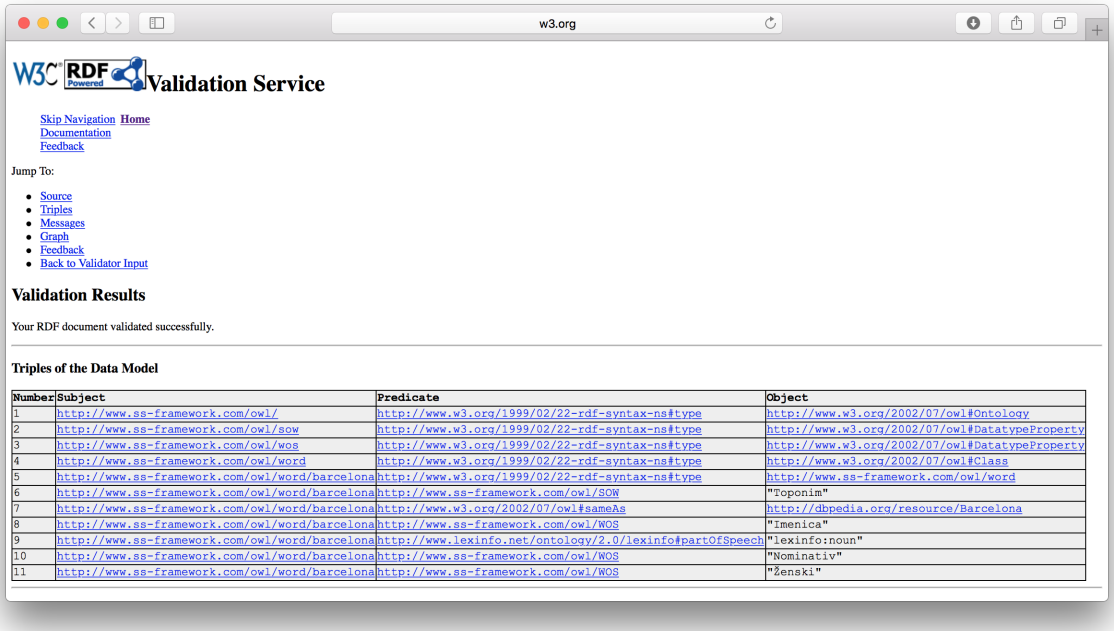


Figure 77: W3C Validator output for SSF ontology

As the validation of RDF triples generated from the SSF is successful, it means that the framework is interoperable with global linguistic linked data cloud and that hypotheses **H3** is confirmed.

8.3. SPARQL queries

Standard language for the linked data searching is SPARQL. What SQL is to relation databases SPARQL is to RDF data model. Like SQL, SPARQL also uses `SELECT` operator to define a subset of the data that is retrieved. The first version of SPARQL 1.0 became standard back in 2008 and has been approved by W3C. After that the entire latest version SPARQL 1.1 was developed in 2013, and basically became a new standard for RDF searching. SPARQL enables searching of the data based on overall or partial RDF triple matching while offering a possibility to bound multiple queries in one. It also implements filtering, grouping and sorting algorithms. Like SQL, SPARQL uses a `WHERE` clause filter out results. Parameters of `WHERE` clause consists of the subject, predicate and object triple to find a match in the RDF data. The latest version (SPARQL 1.1) also enables data manipulation (`INSERT`, `DELETE` or `UPDATE`). One of the main advantages of SPARQL language is an ability of searching public and open databases on remote servers as well as local ones. In that way it is possible to link local RDF triples with remote ones.
SPARQL queries syntax is based on Turtle syntax. General structure of every SPARQL query is defined in the following way:

```
PREFIX
SELECT
WHERE { }
ORDER BY
LIMIT
OFFSET
```

At the beginning of the query is a place reserved for prefix definitions which are later used in queries. After prefixes, one of commands which defines query type (`SELECT`, `ASK`, `CONSTRUCT` or `DESCRIBE` are used for data retrieval or `INSERT`, `DELETE`, `CLEAR` and `DROP` which are used for data manipulation). After that an optional clause `WHERE` can be used. Between the curled brackets it is possible to define triple that will be matched against RDF database. At the very end of SPARQL query it is possible to define the result set ordering by `ORDER BY` clause or limiting output by using `LIMIT` and/or `OFFSET` clause.

To offer an easier inclusion into a global linked data cloud, the SSF's SPARQL endpoint (as described in Section 8.1) is publicly exposed. The following example shows simple a SPARQL query which retrieves all words that are tagged as a noun (cro. *imenica*) in alphabetical order and limits the output to only 10 results.

```
PREFIX ssf: <http://www.ss-framework.com/owl/>
SELECT DISTINCT *
WHERE {?word ssf:WOS "Imenica"}
ORDER BY ?word
LIMIT 10
```

The result is a list of first 10 words that are tagged with a noun WOS tag:

word
http://www.ss-framework.com/owl/word/abdikacije
http://www.ss-framework.com/owl/word/abecedi
http://www.ss-framework.com/owl/word/abecedu
http://www.ss-framework.com/owl/word/abelu
http://www.ss-framework.com/owl/word/abolicijama
http://www.ss-framework.com/owl/word/abolicije
http://www.ss-framework.com/owl/word/abonenti
http://www.ss-framework.com/owl/word/abonentima
http://www.ss-framework.com/owl/word/abonentu
http://www.ss-framework.com/owl/word/abrama
http://www.ss-framework.com/owl/word/abrame

The next example shows a SPARQL query that retrieves all WOS marks for a certain word, for example cro. *majka* (eng. mother):

```
PREFIX ssf: <http://www.ss-framework.com/owl/>
PREFIX ssfword: <http://www.ss-framework.com/owl/word/>
SELECT DISTINCT *
WHERE {ssfword:majka ssf:WOS ?wos}
```

The output of such query is a list of WOS tags that are assigned to the word cro. majka and looks like this:

wos
Imenica
Nominativ
Ženski
Jednina

In the same way it is possible to ask for a word tagged with specific SOW tags, for example all words that are tagged as Toponyms:

```
PREFIX ssf: <http://www.ss-framework.com/owl/>
SELECT DISTINCT *
WHERE {?word ssf:SOW "Toponim"}
ORDER BY ?word
LIMIT 5
```

And the result is the following:

word
http://www.ss-framework.com/owl/word/barcelona
http://www.ss-framework.com/owl/word/berlin
http://www.ss-framework.com/owl/word/ljubljana
http://www.ss-framework.com/owl/word/london
http://www.ss-framework.com/owl/word/zagreb

Furthermore, the real power of semantic web is the so called federated queries across diverse data sources. Previous query can, for example, be extended in a way to integrate the data from DBpedia. For each word which is tagged with the SOW tag Toponym, the data about its population (which is not stored in the SSF's database but is automatically acquired during the SPARQL query execution) is acquired.

```
PREFIX dbo:<http://dbpedia.org/ontology/>
PREFIX ssf:<http://www.ss-framework.com/owl/>
SELECT ?word ?population {
    ?word ssf:SOW "Toponim" .
    ?word owl:sameAs ?city_dbpedia.
    SERVICE <http://dbpedia.org/sparql> {
        ?city_dbpedia rdfs:label ?city_name .
        FILTER (lang($city_name) = 'en') .
        ?city_dbpedia dbo:populationTotal ?population .
    }
}
ORDER BY DESC(?population)
```

After the above query is executed, a semantic reasoner gathers the data from stated sources and produces the following output:

word	population
http://www.ss-framework.com/owl/word/london	8673713
http://www.ss-framework.com/owl/word/berlin	3610156
http://www.ss-framework.com/owl/word/barcelona	1604555
http://www.ss-framework.com/owl/word/zagreb	792875
http://www.ss-framework.com/owl/word/ljubljana	279756

All SPARQL queries to the SSF can be executed either directly via SPARQL endpoint which is publicly available at the `http://www.ss-framework.com/sparql` or through the SSF's NLF tab like these shown in Figure 78. The first type of access is more suitable for external applications or other systems performing federated queries, whereas the second approach is more suitable for usage in the custom made NLF functions for SSF users.

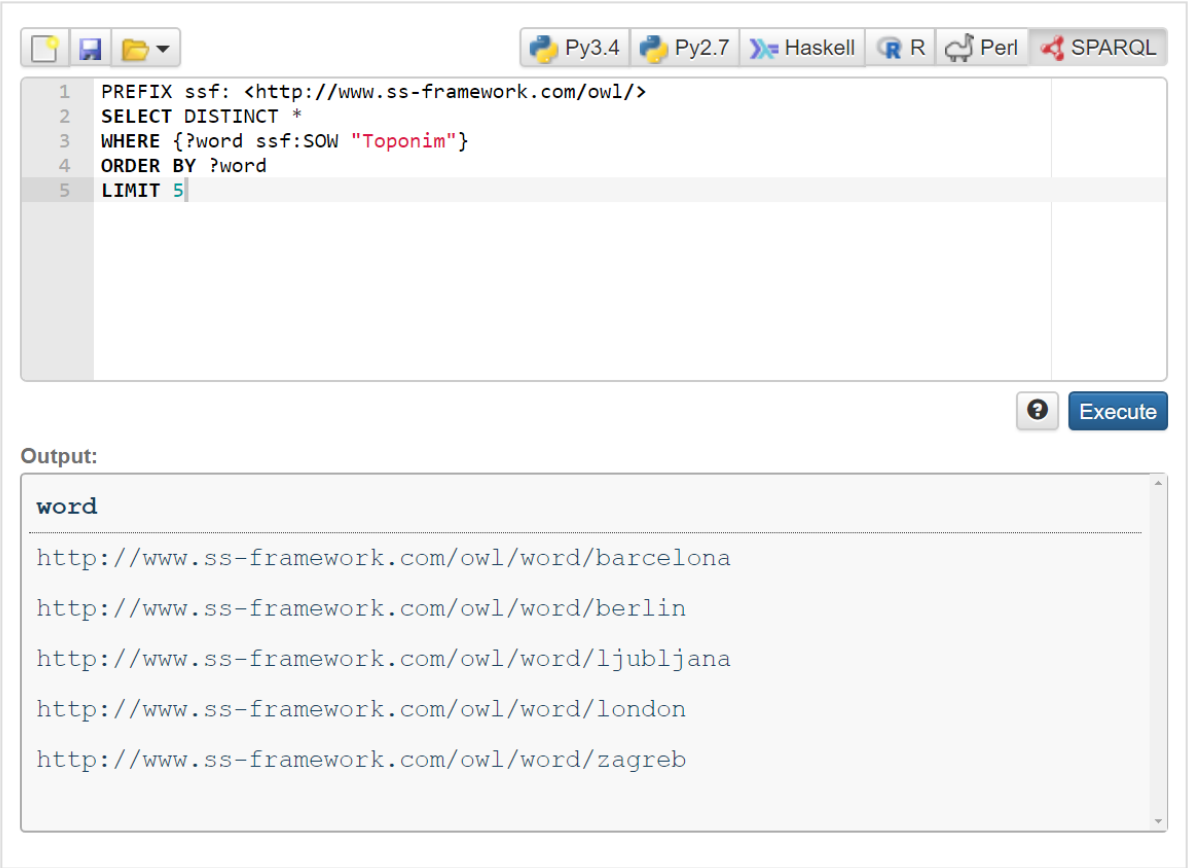


Figure 78: Running SPARQL queries in the SSF

8.4. Croatian word in the Linguistic Linked Open Data Cloud

The Linguistic Linked Open Data [101] is a movement about publishing data for linguistics and natural language processing using the following principles:

- Data should be openly licensed using licenses such as the Creative Commons licenses;
- The elements in a dataset should be uniquely identified by means of a URI;
- The URI should resolve, so users can access more information using web browsers;
- Resolving an LLOD resource should return results using web standards such as HTML, RDF or JSON-LD; and
- Links to other resources should be included to help users discover additional resources and provide semantics.

Finally, to be visible by others the data should be registered in a public repository. Figure 79 shows the screenshot of SSF's RDF data in the Datahub repository.

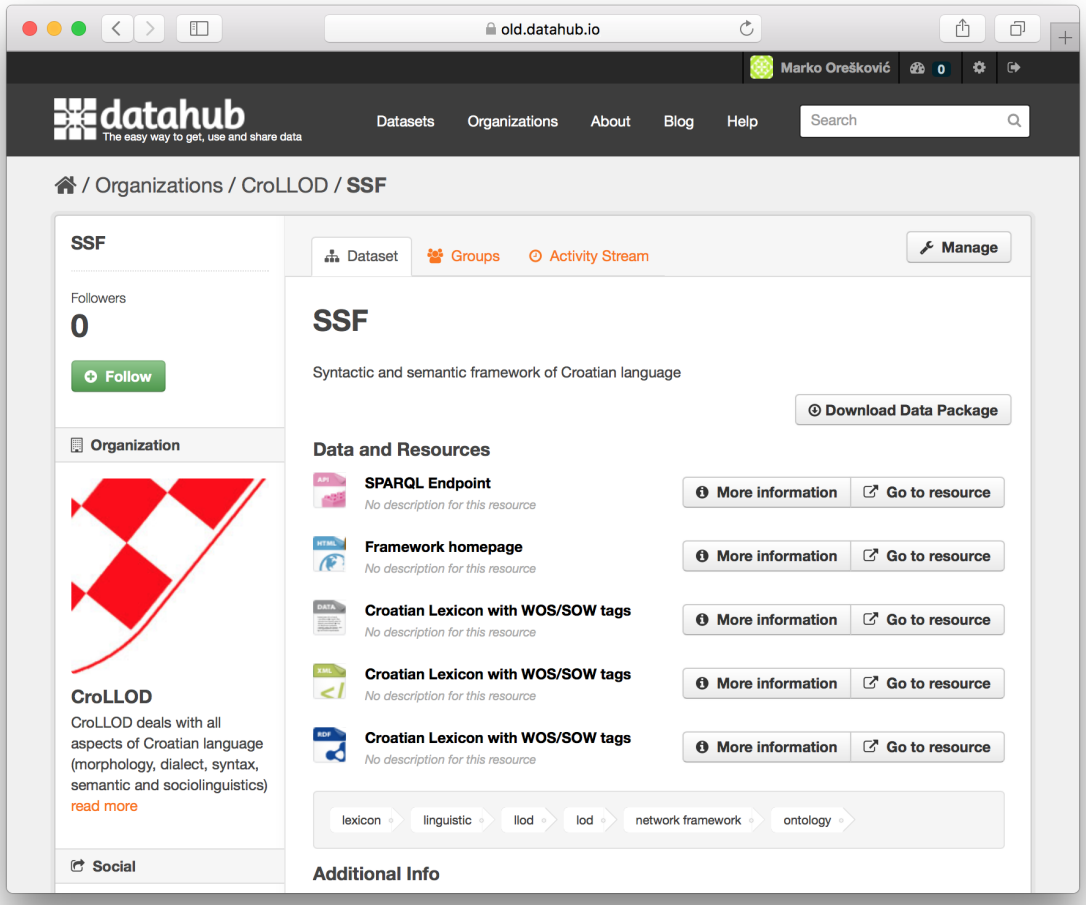


Figure 79: SSF's Lexicon as a dataset in the Datahub

The primary benefits of LLOD have been identified as:

- Representation - linked graphs are a more flexible representation format for linguistic data;
- Interoperability - common RDF models can easily be integrated;
- Federation - data from multiple sources can trivially be combined;
- Ecosystem - tools for RDF and linked data are widely available under open source licenses;
- Expressivity - existing vocabularies such as OWL, Lemon and NIF help express linguistic resources;
- Semantics - common links express what you mean; and
- Dynamism - web data can be continuously improved.

Since April 2018, the SSF became a part of the global linguistic linked data network with over 20,000 links to the BabelNet ontology and over 67,000 links to the LexInfo (Figure 80). The inclusion of the SSF in the LOD cloud one of this research objectives.

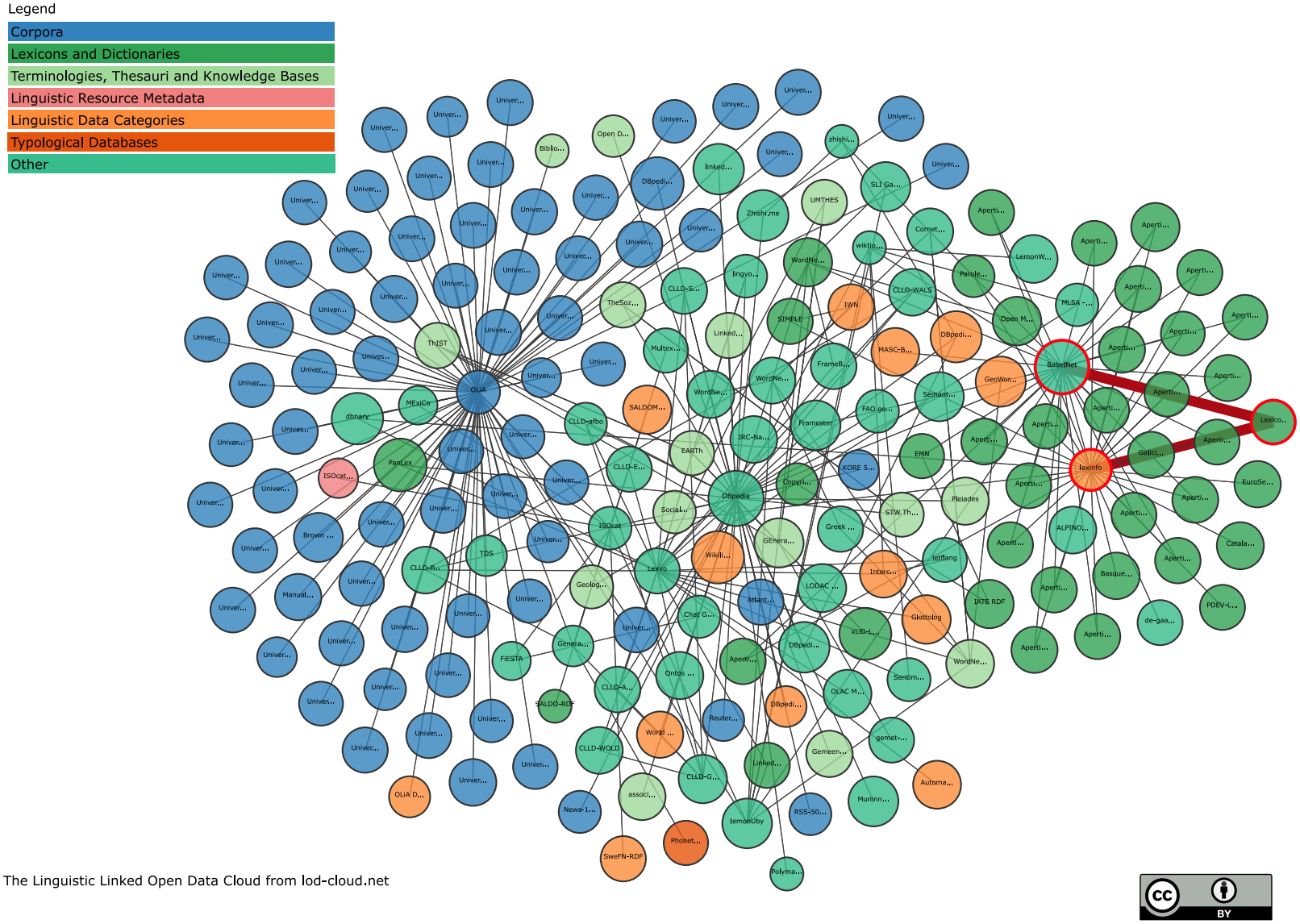


Figure 80: The SSF in the LOD Cloud

9. Conclusion

This thesis proposes a new deterministic language model with the associated artifact, the Syntactic and Semantic Framework (SSF), which extracts lexical relationships from unstructured text. Although the artifact is based on the Croatian language, it is also suitable for any Indo-European language. The model is based on deterministic lexical, morphological, and syntactical settings, and it is realised as an online application written in the PHP programming language with a parser component made in Python, while the database management system that powers it is the relational database MariaDB, which contains 40 tables, 250 attributes, and over 200 indices. The frontend of the SSF was developed using the Bootstrap framework combined with the jQuery JavaScript library and served over the nginx web server. The theoretical language model represents an extension of Mel'čuk's Meaning Text Theory [117] with a new type of syntactic and semantic mark-up. By utilizing WOS/SOW tags, MWE lexicon and syntactic pattern, the SSF enables the realization of any Mel'čuk's function for a given language (which cannot be translated between different languages). On the other hand, the SSF expands these functions due to the usage of syntactic patterns in O-structures which are defined with general characteristics (POS marks, categories, and semantic characteristics) and this renders a more general character to such functions. They are no longer related solely to words, but to syntax, too (e.g. colligations).

The extraction of semantic relationships from unstructured text represents the most complex task in the computational processing of language due to its connectivity to all aspects of language: lexicon, syntax (grammatical relationships between words and sentences), and semantic features of words and multiword expressions. The building of the fundamental language resource, the lexicon, required the development of three demanding lexicons: a subatomic lexicon of syllables, morphs, and syllable morphs; a lexicon of words (with grammatical and semantic features), and a lexicon of multiword expressions (e.g. collocations, phrases, etc.). A unique aspect of such an approach was the introduction of a new tree-like structure (WOS/SOW) for each word in the lexicon instead of the commonly used MULTEXT-East tags. The same structure is also used in multiword expression tagging and multilevel searching. Syntactic structure, due to the deterministic approach, is role-based, which means that it is possible to build and store sets of grammatical patterns (e.g. congruity of noun groups, correct patterns of multiword conjugation classes, etc.) from well-tagged words.

The thesis also shows the possibility of processing the syntactic structure through the stochastic classes in the NLF module using the statistics tool R or the Python programming language. Finally, thanks to the SOW tags, it is possible to conduct semantic analysis,

such as the extraction of semantic relationships and the recognition of semantic patterns (e.g. metonymy, metaphors, etc.). For this complex task, static and semantic structures (domains) are created from existing knowledge taken from local network resources, such as the Miroslav Krleža Institute of Lexicography (LZMK), the Croatian Language Portal (HJP), and Croatian WordNet (CroWN), as well as foreign ones, such as BabelNet and WordNet. The processing of such encyclopaedic articles is carried out in two directions: extraction and storage of the subject-predicate-object (S-P-O) information and creation and storage of parts of speech (POS) information for semantically valuable word types (e.g. verbs, nouns, adjectives, etc.). This semantic information, combined with SOW tags, makes it possible to extract all semantic relationships (e.g. synonyms, antonyms, hyponyms, etc.) as well as any conceptual structures that may occur. All these possibilities are shown in the thesis with examples.

The deterministic language model and its associated artifact have two applications: as a standalone network framework and as an ontological lexical structure in the global Linguistic Linked Open Data (LLOD) cloud. Both applications are verified by linguistic experts and are tested in suitable environments: the Department of Mathematics at J. J. Strossmayer University of Osijek (`https://www.mathos.unios.hr/`) and the global LLOD cloud (`https://lod-cloud.net/`).

The future development of the artifact is possible in many ways:

- As an innovative subatomic lexicon type (three types of lexicons) that enables the permanent storage of Croatian words in local and global repositories and data clouds;
- As an educational tool for students and researchers in the field of complex linguistics, especially since the API functionality allows for easy integration with various external systems; and
- As the foundation for machine translation when the different language lexicons are loaded into the database.

The main scientific contributions of this thesis are:

- A new computer model of natural language that integrates all fields of linguistics;
- A new network artifact that results from a language model, such as the SSF, that enables extraction of lexical information from digitised textual documents in a new, deterministic way;
- A new network lexicon with a new WOS/SOW tagging system and links to other external network resources;

– The integration of different programming languages (e.g. PHP, Python, Haskell, R, etc.), which enables the execution of programs developed by third parties within the SSF; and

– The integration of linguistic information (e.g. lexicons, patterns, semantic relationships, etc.) into the global LLOD.

The thesis fulfils all of the hypotheses and research questions:

– Hypothesis **H1** is confirmed by the implementation of the language model in three segments: lexicographical (as described in Section 4.2), syntactic (as described in Section 5.3) and semantic (as described in Section 6.2).

– Hypothesis **H2** is confirmed by the development of lexical patterns (O-structures) as described in Section 7.4, which enables the storage and extraction of information from corpora, which is a fundamental scientific contribution of this thesis.

– Hypothesis **H3** is confirmed by the successful validation of RDF data as shown in Section 8.2, which is included in the global LLOD cloud.

– An answer to research question **Q1** is given in Section 3.1, which describes how all annotation models used today (e.g. MULTEXT-East, Penn Treebank POS tags, SWETWOL, UD POS tags, etc.) can be implemented in the network framework using a new hierarchical structure of tags (T-structure).

– An answer to research question **Q2** is given in Section 2.3, which confirms that the development of a network framework that integrates morphological, syntactic, and semantic features is feasible, because the network framework is publicly available

– An answer to research question **Q3** is given in Section 4.2, which shows different possibilities of external resources (e.g. Croatian WordNet, Miroslav Krleža Institute of Lexicography Encyclopedia, Croatian Language Portal, BabelNet, etc.) to be included in the network framework.

– An answer to research question **Q4** is given in Section 6.5, which shows how the network framework can be used to conduct sentiment analysis.

– An answer to research question **Q5** is given in Chapter 8, which shows different approaches to the transformation of relational data to triples that are suitable for publishing in a semantic cloud.

I hope my work will continue to develop and will be useful to the Croatian linguistic community in particular and the worldwide linguistic community as a whole.

References

[1] Aggarwal, Charu C. and Zhai, ChengXiang. 'A Survey of Text Clustering Algorithms'. In: *Mining Text Data*. Boston, MA: Springer US, 2012. Chap. 4, pp. 77–128. ISBN: 978-1-4614-3223-4. DOI: 10.1007/978-1-4614-3223-4_4.

[2] Eugene Agichtein and Luis Gravano. 'Snowball: Extracting Relations from Large Plain-text Collections'. In: *Proceedings of the Fifth ACM Conference on Digital Libraries*. DL '00. ACM. San Antonio, Texas, USA: ACM, 2000, pp. 85–94. ISBN: 1-58113-231-X. DOI: 10.1145/336597.336644.

[3] Alan Agresti. *Categorical data analysis*. 3rd ed. Wiley Series in Probability and Statistics, 2012. ISBN: 978-0-470-46363-5.

[4] Alan Akbik, Oresti Konomi and Michail Melnikov. 'Propminer: A Workflow for Interactive Information Extraction and Exploration using Dependency Trees'. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (4th–9th Aug. 2013). Sofia, Bulgaria: The Association for Computational Linguistics, 2013, pp. 157–162.

[5] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. 1st ed. Elsevier, 2011.

[6] Juri D Apresjan. 'Principles of systematic lexicography'. In: *Practical Lexicography. A reader* (2008), pp. 51–60.

[7] Mark Aronoff. *Morphology by Itself: Stems and Inflectional Classes*. Vol. 22. Linguistic Inquiry Monograph. Cambridge, Massachusetts: Massachusetts Institute of Technology Press, 1994. ISBN: 9780262011365.

[8] Stjepan Babić. *Tvorba riječi u hrvatskome književnome jeziku*. Hrvatska akademija znanosti i umjetnosti Zagreb, 2002. ISBN: 86-407-0025-7.

[9] Stefano Baccianella, Andrea Esuli and Fabrizio Sebastiani. 'SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining'. In: *Lrec*. Vol. 10. 2010. 2010, pp. 2200–2204.

[10] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead and Oren Etzioni. 'Open Information Extraction from the Web'. In: *IJCAI*. Vol. 7. 2007, pp. 2670–2676.

[11] Tim Berners-Lee, James Hendler and Ora Lassila. 'The Semantic Web'. In: *Scientific American* 284.5 (2001), pp. 34–43. ISSN: 00368733, 19467087.

[12] Steven Bethard, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou and Dan Jurafsky. 'Automatic extraction of opinion propositions and their holders'. In: *2004 AAAI spring symposium on exploring attitude and affect in text*. Vol. 2224. 2004.

[13] Douglas Biber. *Variation across speech and writing*. Cambridge University Press, 1991.

[14] Ante Bičanić, Anđela Frančić, Lana Hudček and Milica Mihaljević. *Pregled Povijesti,Gramatike i Pravopisa Hrvatskoga Jezika*. Zagreb: Croatica, 2013.

[15] Juhani Birn. *Morphological Tags - SWECG: A Short Presentation*. 1998. URL: `http://www2.lingsoft.fi/doc/swecg/intro/mtags.html` (visited on 30/06/2018).

[16] Matea Birtić. *Baza hrvatskih glagolskih valencija: e-Glava*. 2018. URL: `http://valencije.ihjj.hr` (visited on 16/08/2018).

[17] Chris Bizer. *The D2RQ Platform – Accessing Relational Databases as Virtual RDF Graphs*. 2018. URL: `http://d2rq.org/` (visited on 28/01/2018).

[18] Anders Björkelund, Love Hafdell and Pierre Nugues. 'Multilingual Semantic Role Labeling'. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task* (4th June 2009). Association for Computational Linguistics. 2009, pp. 43–48. ISBN: 978-1-932432-29-9.

[19] Geert Booij. 'Morphology: An International Handbook on Inflection and Word-Formation'. In: ed. by Geert Booij, Christian Lehmann, Joachim Mudgan and Stavros Skopeteas. Vol. 1. Walter de Gruyter, 2002. Chap. Inflection and derivation, pp. 360–369. ISBN: 3110111284.

[20] Thorsten Brants. 'TnT: A Statistical Part-of-speech Tagger'. In: *Proceedings of the Sixth Conference on Applied Natural Language Processing*. ANLC '00. Seattle, Washington: Association for Computational Linguistics, 2000, pp. 224–231. DOI: `10.3115/974147.974178`.

[21] Mario Brdar and Rita Brdar-Szabó. 'On constructional blocking of metonymies'. In: *Review of Cognitive Linguistics. Published under the auspices of the Spanish Cognitive Linguistics Association* 15.1 (2017), pp. 183–223.

[22] Bresnan, Joan and Asudeh, Ash and Toivonen, Ida and Wechsler, Stephen. *Lexical Functional Syntax*. 2nd ed. John Wiley & Sons, 2015. ISBN: 9781405187817.

[23] Eric Brill. 'Transformation-based Error-driven Learning and Natural Language Processing: A Case Study in Part-of-speech Tagging'. In: *Computational Linguistics* 21.4 (Dec. 1995), pp. 543–565. ISSN: 0891-2017.

[24] Robbins Burling. *Patterns of language: Structure, variation, change*. Academic Press, 1992.

[25] Erik Cambria, Amir Hussain, Catherine Havasi and Chris Eckl. 'Sentic computing: Exploitation of common sense for the development of emotion-sensitive systems'. In: *Development of Multimodal Interfaces: Active Listening and Synchrony*. Springer, 2010, pp. 148–156.

[26] Erik Cambria, Daniel Olsher and Dheeraj Rajagopal. 'SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis'. In: *Twenty-eighth AAAI conference on artificial intelligence*. 2014.

[27] Erik Cambria and Bebo White. 'Jumping NLP curves: A review of natural language processing research'. In: *IEEE Computational intelligence magazine* 9.2 (2014), pp. 48–57.

[28] Andrew Carstairs-McCarthy. 'Morphology: An International Handbook on Inflection and Word-Formation'. In: ed. by Geert Booij, Christian Lehmann, Joachim Mudgan and Stavros Skopeteas. Vol. 1. Walter de Gruyter, 2000. Chap. Lexeme, word-form, pp. 595–607. ISBN: 3110111284.

[29] Christian Chiarcos, Sebastian Nordhoff and Sebastian Hellmann, eds. *Linked Data in Linguistics*. Springer, 2012. ISBN: 978-3-642-28248-5. DOI: 10.1007/978-3-642-28249-2.

[30] Chomsky, Noam. *Aspects of the Theory of Syntax*. Vol. 11. MIT press, 2014.

[31] Noam Chomsky. *Syntactic structures*. Martino Fine Books, 2002.

[32] Christensen, Janara and Soderland, Stephen and Etzioni, Oren. 'An Analysis of Open Information Extraction based on Semantic Role Labeling'. In: *Proceedings of the sixth international conference on Knowledge capture* (26th–29th June 2011). ACM. 2011, pp. 113–120. ISBN: 978-1-4503-0396-5.

[33] Christensen, Janara and Soderland, Stephen and Etzioni, Oren. 'Semantic Role Labeling for Open Information Extraction'. In: *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading* (6th July 2010). Association for Computational Linguistics. Los Angeles, California, 2010, pp. 52–60.

[34] Kenneth Ward Church. 'A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text'. In: *Proceedings of the Second Conference on Applied Natural Language Processing*. ANLC '88. Austin, Texas: Association for Computational Linguistics, 1988, pp. 136–143. DOI: 10.3115/974235.974260.

[35] Guglielmo Cinque. *Typological studies: Word order and relative clauses*. Taylor & Francis, Routledge, 2014.

[36] Montserrat Civit Torruella. *Guía para la anotación morfológica del corpus CLiC–TALP (versión 3)*. Tech. rep. Centre de Llenguatge i Computació (CLiC), Barcelona, Catalunya, 2002.

[37] Jan Cloeren. 'Toward a cross-linguistic tagset'. In: *Workshop On Very Large Corpora: Academic And Industrial Perspectives* (1993).

[38] Alan Cruse. *Meaning in Language: An Introduction to Semantics and Pragmatics*. Oxford University Press, 2011. ISBN: 978-0199559466.

[39] Walter Daelemans, Jakub Zavrel, Peter Berck and Steven Gillis. 'MBT: A Memory-based Part of Speech Tagger-Generator'. In: *Proceedings of the Fourth Workshop on Very Large Corpora*. 1996, pp. 14–27.

[40] Walter Daelemans, Jakub Zavrel, Ko Sloot and Antal Van den Bosch. *TiMBL: Tilburg Memory-Based Learner - version 4.0 Reference Guide*. Sept. 2001.

[41] Scott Deerwester. 'Improving Information Retrieval with Latent Semantic Indexing'. In: ed. by Christine L. Borgman and Edward Y. H. Pai. Vol. 25. American Society for Information Science. Atlanta, Georgia, 1988. ISBN: 9780938734291.

[42] Ivan Derzhanski and Natalia Kotsyba. 'Towards a consistent morphological tagset for Slavic languages: Extending MULTEXT-East for Polish, Ukrainian and Belarusian'. In: *Mondilex Third Open Workshop* (15th–16th Apr. 2009). Ed. by Radovan Garabík. Ľ. Štúr Institute of Linguistics, Slovak Academy of Sciences, Bratislava, 2009, pp. 9–26. ISBN: 978-80-7399-745-8.

[43] Guillaume Desagulier. 'Introduction'. In: *Corpus Linguistics and Statistics with R: Introduction to Quantitative Methods in Linguistics*. Cham: Springer International Publishing, 2017, pp. 1–12. ISBN: 9783319645728. DOI: 10.1007/978-3-319-64572-8_1.

[44] Guillaume Desagulier. 'Visualizing distances in a set of near-synonyms'. In: *Corpus Methods for Semantics: Quantitative studies in polysemy and synonymy* 43 (2014), p. 145.

[45] Institut für Deutsche Sprache. *Das elektronische Valenzwörterbuch deutscher Verben*. 2018. URL: http://hypermedia.ids-mannheim.de/evalbu/ (visited on 16/08/2018).

[46] Ann Devitt and Khurshid Ahmad. 'Sentiment polarity identification in financial news: A cohesion-based approach'. In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 984–991.

[47] Wolfgang U Dressler. 'Prototypical differences between inflection and derivation'. In: *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung* 42.1 (1989), pp. 3–10. DOI: https://doi.org/10.1515/stuf-1989-0102.

[48] Wolfgang U Dressler. 'The cognitive perspective of "naturalist" linguistic models'. In: *Cognitive Linguistics* 1.1 (1990), pp. 75–98. DOI: https://doi.org/10.1515/stuf-1989-0102.

[49] Tomaž Erjavec. 'MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora'. In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC'04, ELRA*. Paris, France, 2004, pp. 1535–1538.

[50] Tomaž Erjavec, Cvetana Krstev, Vladimr Petkevič, Kiril Simov, Marko Tadić and Duško Vitas. 'The MULTEXT-East morphosyntactic specifications for Slavic languages'. In: *Proceedings of the 2003 EACL Workshop on Morphological Processing of Slavic Languages*. Association for Computational Linguistics. 2003, pp. 25–32.

[51] Mario Essert, Ivana Kurtović Budja and Marko Orešković. 'Pozivnica Oxford dictionaryja hrvatskomu jeziku'. In: *Izazovi nastave hrvatskoga jezika* (10th–17th Nov. 2017). Ed. by Srećko Listeš and Linda Grubišić Belina. 8. Simpozij učitelja i nastavnika Hrvatskoga jezika. Školska knjiga d.d., Zagreb, Croatia, 2017, pp. 10–25. ISBN: 978-953-0-51739-4.

[52] Oren Etzioni, Michele Banko and Michael J. Cafarella. 'Machine Reading'. In: *Proceedings of the 21st National Conference on Artificial Intelligence* (16th–20th July 2006). Vol. 2. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, pp. 1517–1519. ISBN: 978-1-57735-281-5.

[53] Oren Etzioni, Michele Banko, Stephen Soderland and Daniel S. Weld. 'Open Information Extraction from the Web'. In: *Communications of the ACM* 51.12 (2008), pp. 68–74. DOI: 10.1145/1409360.1409378.

[54] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland and Mausam Mausam. 'Open Information Extraction: the Second Generation'. In: *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. Vol. 11. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 3–10. ISBN: 978-1-57735-513-7. DOI: 10.5591/978-1-57735-516-8/IJCAI11-012.

[55] Stefan Evert. 'The Statistics of Word Cooccurrences: Bigrams and Collocations'. Doctoral dissertation. University of Stuttgart, 2004.

[56] Anthony Fader, Stephen Soderland and Oren Etzioni. 'Identifying Relations for Open Information Extraction'. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Association for Computational Linguistics. Edinburgh, United Kingdom: Association for Computational Linguistics, 2011, pp. 1535–1545. ISBN: 978-1-937284-11-4.

[57] Gilles Fauconnier and Mark Turner. 'Conceptual integration networks'. In: *Cognitive science* 22.2 (1998), pp. 133–187.

[58] Ronen Feldman. 'Techniques and applications for sentiment analysis'. In: *Communications of the ACM* 56.4 (2013), pp. 82–89.

[59] Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

[60] Andy Field, Jeremy Miles and Zoë Field. *Discovering statistics using R*. Sage publications, 2012.

[61] Roy T Fielding. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Doctoral dissertation, 2000.

[62] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach and Tim Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. RFC Editor, 1999. URL: `http://www.rfc-editor.org/rfc/rfc2616.txt`.

[63] John Firth. 'A Synopsis of Linguistic Theory 1930-1955'. In: *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957.

[64] Vladimir Fomichov. *Semantics-oriented Natural Language Processing Mathematical Models and Algorithms*. Springer, 2010. DOI: `10.1007/978-0-387-72926-8`.

[65] Thierry Fontenelle. 'WordNet, FrameNet and Other Semantic Networks in the International Journal of Lexicography – The Net Result?' In: *International Journal of Lexicography* 25.4 (2012), pp. 437–449. ISSN: 0950-3846. DOI: `10.1093/ijl/ecs027`.

[66] Nelson Francis and Henry Kucera. 'Brown Corpus'. In: *Department of Linguistics, Brown University, Providence, Rhode Island* 1 (1964).

[67] Gil Francopoulo, ed. *LMF Lexical Markup Framework*. John Wiley & Sons, 2013. ISBN: 978-1-84821-430-9.

[68] Roger Garside. 'The CLAWS word-tagging system'. In: (1987). Ed. by Roger Garside, Geoffrey Leech and Geoffrey Sampson, pp. 30–41.

[69] Alexander Gelbukh and Olga Kolesnikova. *Semantic Analysis of Verbal Collocations with Lexical Functions*. Vol. 414. Studies in Computational Intelligence. Springer, 2012. ISBN: 978-3-642-28770-1. DOI: 10.1007/978-3-642-28771-8.

[70] Namrata Godbole, Manja Srinivasaiah and Steven Skiena. 'Large-Scale Sentiment Analysis for News and Blogs'. In: *Icwsm* 7.21 (2007), pp. 219–222.

[71] Cliff Goddard. *Semantic analysis: A practical introduction*. Oxford University Press, 2011.

[72] Joseph H Greenberg. 'Some universals of grammar with particular reference to the order of meaningful elements'. In: *Universals of language* 2 (1963), pp. 73–113.

[73] Alexander Grosu. 'Towards a more articulated typology of internally headed relative constructions: The semantics connection'. In: *Language and Linguistics Compass* 6.7 (2012), pp. 447–476.

[74] Jan Hajič. *Disambiguation of Rich Inflection-Computational Morphology of Czech*. Prague, Czech Republic: Charles University Press, 2004.

[75] Michael Alexander Kirkwood Halliday and Christian M.I.M. Matthiessen. *Halliday's Introduction to Functional Grammar*. Routledge, 2013.

[76] Douglas Harper. *Online Etymology Dictionary*. 2010. URL: http://www.dictionary.com/browse/parse (visited on 20/07/2018).

[77] Roland Hausser. *Computational Linguistics and Talking Robots: Processing Content in Database Semantics*. Springer-Verlag Berlin Heidelberg, 2011. ISBN: 978-3-642-22431-7. DOI: 10.1007/978-3-642-22432-4.

[78] Roland Hausser. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. 3rd ed. Springer, 2014. DOI: 10.1007/978-3-642-41431-2.

[79] Marti A. Hearst. 'Automatic Acquisition of Hyponyms from Large Text Corpora'. In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*. COLING '92. Nantes, France: Association for Computational Linguistics, 1992, pp. 539–545. DOI: 10.3115/992133.992154.

[80] Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram. 'Design Science in Information Systems Research'. In: *MIS Quarterly* 28.1 (2004), pp. 75–105. ISSN: 0276-7783.

[81] *Hrvatski jezični portal*. 2018. URL: http://hjp.znanje.hr (visited on 18/07/2018).

[82] Ray Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press, 2003. ISBN: 9780199264377.

[83] Ray Jackendoff. *Semantics and cognition*. Vol. 8. MIT press, 1983.

[84] Paul Johannesson and Erik Perjons, eds. *An Introduction to Design Science*. Springer International Publishing, 2014. ISBN: 978-3-319-10631-1. DOI: 10.1007/978-3-319-10632-8.

[85] Stig Johansson, Geoffrey Leech and Helen Goodluck. *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computer*. Department of English, University of Oslo, 1978.

[86] Jurafsky, Daniel and Martin, James H. 'Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition'. In: MIT Press, 2017. Chap. 22, pp. 378–379.

[87] Ronald Kaplan and Joan Bresnan. 'Lexical-functional grammar: A formal system for grammatical representation'. In: *Formal Issues in Lexical-Functional Grammar* 47 (1982), pp. 29–130.

[88] Soo-Min Kim and Eduard Hovy. 'Determining the sentiment of opinions'. In: *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics. 2004, p. 1367.

[89] Nikolaos Konstantinou and Dimitrios-Emmanuel Spanos. *Materializing the Web of Linked Data*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-16074-0.

[90] Zoltan Kovecses. *Language, mind, and culture: A practical introduction*. Oxford University Press, 2006.

[91] Madhav Krishna. 'Retaining semantics in relational databases by mapping them to RDF'. In: *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference*. IEEE. 2006, pp. 303–306.

[92] George Lakoff. 'The death of dead metaphor'. In: *Metaphor and symbol* 2.2 (1987), pp. 143–147.

[93] George Lakoff and Mark Johnson. *Metaphors we live by*. University of Chicago press, 2008.

[94] Ronald Langacker. *Cognitive grammar: A basic introduction*. Oxford University Press, 2008.

[95] Ronald W Langacker. *Concept, image, and symbol: The cognitive basis of grammar*. Vol. 1. Walter de Gruyter, 2002.

[96] Ronald W Langacker. *Foundations of cognitive grammar: Theoretical prerequisites*. Vol. 1. Stanford university press, 1987.

[97] Leksikografski zavod Miroslav Krleža. *Hrvatska Enciklopedija*. 2018. URL: http://enciklopedija.hr/ (visited on 09/07/2018).

[98] Natalia Levshina. *How to do linguistics with R: Data exploration and statistical analysis*. John Benjamins Publishing Company, 2015.

[99] Hai-yun Ling and Shu-feng Zhou. 'Translating relational databases into RDF'. In: *Environmental Science and Information Application Technology (ESIAT), 2010 International Conference*. Vol. 3. IEEE. 2010, pp. 464–467.

[100] Linguistic Data Consortium. *Penn Treebank P.O.S. Tags*. 2003. URL: https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html (visited on 30/06/2018).

[101] *Linguistic Linked Open Data*. 2018. URL: http://linguistic-lod.org/ (visited on 28/01/2018).

[102] Bing Liu. 'Sentiment Analysis and Opinion Mining'. In: *Synthesis Lectures on Human Language Technologies* 5.1 (2012). Ed. by Graeme Hirst, pp. 1–167. ISSN: 1947-4040.

[103] Hugo Liu and Push Singh. 'ConceptNet—a practical commonsense reasoning toolkit'. In: *BT technology journal* 22.4 (2004), pp. 211–226. DOI: 10.1023/B:BTTJ.0000047600.45421.6d.

[104] Nikola Ljubešić. *MULTEXT-East Morphosyntactic Specifications, revised Version 4*. 2013. URL: http://nlp.ffzg.hr/data/tagging/msd-hr.html (visited on 05/04/2018).

[105] Félix López and Víctor Romero. *Mastering Python Regular Expressions*. Packt Publishing Ltd, 2014.

[106] Christopher D Manning, Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[107] Christopher D Manning, Prabhakar Raghavan and Hinrich Schütze. 'Text classification and Naïve Bayes'. In: *Introduction to Information Retrieval*. Cambridge University Press, 2008. Chap. 4, pp. 234–265. DOI: 10.1017/CBO9780511809071.014.

[108] Manola, Frank and Miller, Eric and McBride, Brian. 'RDF primer'. In: *W3C recommendation* (2004). URL: https://www.w3.org/TR/rdf-primer/.

[109] Stela Manova. *Understanding morphological rules: with special emphasis on conversion and subtraction in Bulgarian, Russian and Serbo-Croatian*. Vol. 1. Springer, 2011.

[110] J. A. Zárate Marceleño, R. A. Pazos R. and A. Gelbukh. 'Customization of Natural Language Interfaces to Databases: Beyond Domain Portability'. In: *2009 Mexican International Conference on Computer Science*. 2009, pp. 373–378. DOI: 10.1109/ENC.2009.52.

[111] Marcus, Mitchell P and Marcinkiewicz, Mary Ann and Santorini, Beatrice. 'Building a large annotated corpus of English: The Penn Treebank'. In: *Computational linguistics* 19.2 (1993), pp. 313–330.

[112] Ivan Marković. *Uvod u jezičnu morfologiju*. Disput, 2012. ISBN: 978-953-260-154-1.

[113] Joško Markučič and Klemen Govedić. *Mrežni Morfološki Program za Hrvatski jezik*. the paper is applied for Rectors award of University of Zagreb. 2014.

[114] Mark Masse. *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc.", 2011.

[115] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland and Oren Etzioni. 'Open Language Learning for Information Extraction'. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 523–534.

[116] Deborah L McGuinness and Frank Van Harmelen. 'OWL web ontology language overview'. In: *W3C recommendation* (2004). URL: https://www.w3.org/TR/owl-features/.

[117] Igor Mel'čuk. *Semantics: From meaning to text*. Ed. by David Beck and Alain Polguère. Vol. 3. John Benjamins Publishing Company, 2015. DOI: 10.1075/slcs.129.

[118] Dunja Mladenić. 'Learning word normalization using word suffix and context from unlabeled data'. In: *Proceedings of the Nineteenth International Conference on Machine Learning* (8th–12th July 2002). Ed. by Claude Sammut and Achim G. Hoffmann. Morgan Kaufmann Publishers Inc. 2002, pp. 427–434. ISBN: 1558608737.

[119] Richard Montague. 'Universal grammar'. In: *Theoria* 36.3 (1970), pp. 373–398.

[120] Roberto Navigli and Simone Paolo Ponzetto. 'BabelNet: Building a very large multilingual semantic network'. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 216–225.

[121] Roberto Navigli and Simone Paolo Ponzetto. 'BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network'. In: *Artificial Intelligence* 193 (2012), pp. 217–250. ISSN: 0004-3702. DOI: 10.1016/j.artint.2012.07.001.

[122] Bruno Ohana. 'Opinion mining with the SentWordNet lexical resource'. In: (2009).

[123] Marko Orešković, Juraj Benić and Mario Essert. 'The Network Integrator of Croatian Lexicographical Resources'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 267–272. ISBN: 978-9941-13-542-2.

[124] Marko Orešković, Marta Brajnović and Mario Essert. 'A step towards machine recognition of tropes'. In: *Third International Symposium on Figurative Thought and Language* (26th–28th Apr. 2017). Faculty of Humanities and Social Sciences University of Osijek, Croatia. 2017, p. 71.

[125] Marko Orešković, Mirko Čubrilo and Mario Essert. 'The Development of a Network Thesaurus with Morpho-semantic Word Markups'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 273–279. ISBN: 978-9941-13-542-2.

[126] Marko Orešković, Ivana Kurtović Budja and Mario Essert. 'Encyclopedic knowledge as a semantic resource'. In: *The Future of Information Sciences, INFuture2017 : Integrating ICT in Society* (8th–10th Nov. 2017). Ed. by Iana Atanassova, Wajdi Zaghouani, Bruno Kragić, Aas Kuldar, Hrvoje Stančić and Sanja Seljan. Department of Information Sciences, Faculty of Humanities and Social Sciences, University of Zagreb, Croatia, 2017, pp. 151–160.

[127] Marko Orešković, Jakov Topić and Mario Essert. 'Croatian Linguistic System Modules Overview'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 280–283. ISBN: 978-9941-13-542-2.

[128] Krešimir Pinjatela. 'Hrvatska riječ'. Lexical database. 2001.

[129] Soujanya Poria, Alexander Gelbukh, Amir Hussain, Newton Howard, Dipankar Das and Sivaji Bandyopadhyay. 'Enhanced SenticNet with affective labels for concept-based opinion mining'. In: *IEEE Intelligent Systems* 28.2 (2013), pp. 31–38.

[130] Nicolas Prat, Isabelle Comyn-Wattiau and Jacky Akoka. 'A Taxonomy of Evaluation Methods for Information Systems Artifacts'. In: *Journal of Management Information Systems* 32.3 (2015), pp. 229–267. DOI: 10.1080/07421222.2015.1099390.

[131] Nives Mikelic Preradović. *The Croatian Valency Lexicon of Verbs, Version 2.0008 (CROVALLEX 2.0008)*. 2018. URL: http://theta.ffzg.hr/crovallex/ (visited on 16/08/2018).

[132] Princeton University. *WordNet: An Electronic Lexical Database*. 2018. URL: https://wordnet.princeton.edu/ (visited on 09/07/2018).

[133] Adam Przepiórkowski and Marcin Woliński. 'A flexemic tagset for Polish'. In: *Proceedings of the 2003 EACL Workshop on Morphological Processing of Slavic Languages*. Association for Computational Linguistics. 2003, pp. 33–40.

[134] Vasin Punyakanok, Dan Roth and Wentau Yih. 'The Importance of Syntactic Parsing and Inference in Semantic Role Labeling'. In: *Computational Linguistics* 34.2 (2008), pp. 257–287. ISSN: 0891-2017. DOI: 10.1162/coli.2008.34.2.257.

[135] James Pustejovsky. 'Generativity and Explanation in Semantics: A Reply to Fodor and Lepore'. In: *Linguistic Inquiry* 29.2 (1998), pp. 289–311. DOI: 10.1162/002438998553752.

[136] James Pustejovsky. *The generative lexicon*. The MIT Press, 1995. ISBN: 9780262161589.

[137] James Pustejovsky, Pierrette Bouillon, Hitoshi Isahara, Kyoko Kanzaki and Chungmin Lee, eds. *Advances in Generative Lexicon Theory*. Vol. 46. Springer Netherlands, 2013. DOI: 10.1007/978-94-007-5189-7.

[138] Ida Raffaelli and Daniela Katunar. 'Lexical-Semantic Structures in Croatian WordNet'. In: *Filologija* 59 (2013), pp. 69–101.

[139] Ida Raffaelli, Marko Tadić, Božo Bekavac and Željko Agić. 'Building croatian wordnet'. In: *Fourth Global WordNet Conference (GWC 2008)*. 2008.

[140] Gillian Ramchand. 'Against a generative lexicon'. In: *University of Geneva* (2006).

[141] Adwait Ratnaparkhi. 'A Maximum Entropy Part-of-speech Tagger'. In: *Proceedings of the Empirical Methods in Natual Language Processing (EMNLP) Conference*. Philadelphia, USA: University of Pennsylvania, 1996, pp. 133–142.

[142] Stuart Russell and Peter Norvig. 'Artificial intelligence: a modern approach'. In: *Prentice Hall Press, Upper Saddle River, NJ, USA* (2009).

[143] Patrick Saint-Dizier and Evelyn Viegas. *Computational Lexical Semantics*. Studies in Natural Language Processing. Cambridge University Press, 1995. DOI: 10.1017/CBO9780511527227.

[144] Gerard Salton, Anita Wong and Chung-Shu Yang. 'A Vector Space Model for Automatic Indexing'. In: *Communications of the ACM* 18.11 (1975), pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220.

[145] Ljiljana Šarić and Wiebke Wittschen. *Rječnik sinonima hrvatskoga jezika*. Zagreb: Naklada Jesenski i Turk, 2008. ISBN: 978-953-222-362-0.

[146] Helmut Schmid. 'Part-of-speech Tagging with Neural Networks'. In: *Proceedings of the 15th Conference on Computational Linguistics*. COLING '94. Kyoto, Japan: Association for Computational Linguistics, 1994, pp. 172–176. DOI: 10.3115/991886.991915.

[147] Helmut Schmid. 'Probabilistic Part-of-Speech Tagging Using Decision Trees'. In: *International Conference on New Methods in Language Processing*. Stuttgart, Germany, 1994, pp. 44–49.

[148] Fabrizio Sebastiani and Andrea Esuli. 'Determining term subjectivity and term orientation for opinion mining andrea esuli'. In: *In Proceedings of the 11th conference of the european chapter of the association for computational linguistics (EACL'06)*. Citeseer. 2006.

[149] Stephen Soderland, John Gilmer, Robert Bart, Oren Etzioni and Daniel S Weld. 'Open Information Extraction to KBP Relations in 3 Hours'. In: *TAC*. 2013.

[150] Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu and Oren Etzioni. 'Adapting Open Information Extraction to Domain-Specific Relations'. In: *AI magazine* 31.3 (2010), pp. 93–102. DOI: 10.1609/aimag.v31i3.2305.

[151] Krešimir Šojat. 'Morfosintaktički razredi dopuna u Hrvatskom WordNetu'. In: *Suvremena lingvistika* 35.68 (2009), pp. 305–339.

[152] Robert Speer and Catherine Havasi. 'ConceptNet 5: A large semantic network for relational knowledge'. In: *The People's Web Meets NLP*. Springer, 2013, pp. 161–176.

[153] Mateusz-Milan Stanojević and Jelena Parizoska. 'Conventional conceptual metaphors and idiomaticity'. In: *Semantika prirodnog jezika i metajezik semantike*. 2005.

[154] Anatol Stefanowitsch and Stefan Th Gries. 'Collostructions: Investigating the interaction of words and constructions'. In: *International journal of corpus linguistics* 8.2 (2003), pp. 209–243. DOI: 10.1075/ijcl.8.2.03ste.

[155] Kristina Štrkalj Despot, Mario Brdar, Mario Essert, Mirjana Tonković, Benedikt Perak, Ana Ostroški Anić, Bruno Nahod and Ivan Pandžić. *MetaNet.HR – Croatian Metaphor Repository*. 2015. URL: http://ihjj.hr/metafore/ (visited on 19/07/2018).

[156] Håkan Sundblad. 'Automatic Acquisition of Hyponyms and Meronyms from Question Corpora'. In: *OLT2002, France* (2002).

[157] Mihai Surdeanu. 'Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling'. In: *The Proceedings of the TAC-KBP 2013 Workshop*. 2013.

[158] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll and Manfred Stede. 'Lexicon-based methods for sentiment analysis'. In: *Computational linguistics* 37.2 (2011), pp. 267–307.

[159] Lucien Tesnière. *Elements of structural syntax*. John Benjamins Publishing Company, 2015, p. 571. ISBN: 9789027212122.

[160] Stjepko Težak and Stjepan Babić. *Gramatika hrvatskoga jezika; Priručnik za osnovno jezično obrazovanje*. Zagreb: Školska knjiga, 2009.

[161] Kristina Toutanova, Dan Klein, Christopher D. Manning and Yoram Singer. 'Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network'. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. NAACL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 173–180. DOI: 10.3115/1073445.1073478.

[162] David Tuggy. 'Cognitive Approach to Word-Formation'. In: *Handbook of Word-Formation*. Ed. by Pavol Štekauer and Rochelle Lieber. Springer Netherlands, 2005, pp. 233–265. ISBN: 978-1-4020-3596-8. DOI: 10.1007/1-4020-3596-9_10.

[163] Peter D Turney. 'Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews'. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 417–424.

[164] Friedrich Ungerer and Hans-Jeorg Schmid. *An Introduction to Cognitive Linguistics*. Longman, 1996. ISBN: 0582239664.

[165] *Universal Dependencies*. 2017. URL: http://universaldependencies.org/ (visited on 02/07/2018).

[166] Josip Užarević. 'Bilježenje naglasaka u hrvatskome i dvoznakovni sustav'. In: *Jezik: Časopis za kulturu hrvatskoga književnog jezika* 59.4 (2012), pp. 126–143.

[167] Robert D Van Valin Jr. *An Introduction to Syntax*. Cambridge University Press, 2001.

[168] Robert D Van Valin Jr. *Exploring the Syntax-Semantics Interface*. Cambridge University Press, 2005.

[169] Robert D Van Valin Jr. 'Role and Reference Grammar as a framework for linguistic analysis'. In: *The Oxford handbook of linguistic analysis*. Ed. by Bernd Heine and Heiko Narrog. 2010. Chap. 28, pp. 703–738. DOI: 10.1093/oxfordhb/9780199544004.013.0028.

[170] John Venable, Jan Pries-Heje and Richard Baskerville. 'FEDS: a Framework for Evaluation in Design Science Research'. In: *European Journal of Information Systems* 25.1 (2016), pp. 77–89. ISSN: 1476-9344. DOI: 10.1057/ejis.2014.36.

[171] Atro Voutilainen. 'Orientation'. In: *Syntactic Wordclass Tagging*. Ed. by Hans van Halteren. Dordrecht: Springer Netherlands, 1999, pp. 3–7. ISBN: 978-94-015-9273-4. DOI: 10.1007/978-94-015-9273-4_1.

[172] Shaohua Wang. 'Conceptual blending and the on-line meaning construction of metaphors [J]'. In: *Contemporary Linguistics* 2 (2002), p. 004.

[173] Leo Wanner. *Lexical functions in lexicography and natural language processing*. Vol. 31. John Benjamins Publishing, 1996.

[174] Ralph Weischedel, Richard Schwartz, Jeff Palmucci, Marie Meteer and Lance Ramshaw. 'Coping with Ambiguity and Unknown Words Through Probabilistic Models'. In: *Computational Linguistics* 19.2 (1993), pp. 361–382.

[175] Fei Wu and Daniel S. Weld. 'Open Information Extraction Using Wikipedia'. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (11th–16th July 2010). ACL '10. Association for Computational Linguistics. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 118–127.

[176] Jeonghee Yi, Tetsuya Nasukawa, Razvan Bunescu and Wayne Niblack. 'Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques'. In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE. 2003, pp. 427–434.

[177] Shu-feng Zhou. 'Relational Databases Access based on RDF View'. In: *E-Business and E-Government (ICEE), 2010 International Conference*. IEEE. 2010, pp. 5486–5489.

[178] Shufeng Zhou. 'Exposing relational database as RDF'. In: *Industrial and Information Systems (IIS), 2010 2nd International Conference*. Vol. 2. IEEE. 2010, pp. 237–240.

[179] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang and Ji-Rong Wen. 'StatSnowball: A Statistical Approach to Extracting Entity Relationships'. In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 101–110. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526724.

APPENDIXES

A. Creation of static domains from SOW definitions

```
#!/usr/bin/php
<?PHP
   require_once("../class.db.php");
   $db = new DB();

   Class DomainCreator {
      protected $_connection;

      public function __construct($conn)
      {
         $this->_connection = $conn;
      }

      public function getWosId($wosName) {
         # Get ID of WOS based on its name
         $res = $this->_connection->query("SELECT wosid FROM wos WHERE
            wos_name='$wosName'");
         $row = $res->fetch_object();
         return $row->wosid;
      }

      public function getSowId($sowName) {
         # Get ID of SOW based on its name
         $res = $this->_connection->query("SELECT sowid FROM sow WHERE
            sow_name='$sowName'");
         $row = $res->fetch_object();
         return $row->sowid;

      }

      public function getLemmaId($wordId) {
         # Get ID of Lemma for a given wordid
         $res = $this->_connection->query("SELECT lemmaid FROM words WHERE
            wordid=$wordId");
         $row = $res->fetch_object();
         return $row->lemmaid;
      }
```

```
35
36      public function getLemmaIdFromWord($word) {
37         # Get ID of Lemma for a given word
38         $res = $this->_connection->query("SELECT lemmaid FROM lemmas WHERE
               lemma='$word'");
39         $row = $res->fetch_object();
40         return $row->lemmaid;
41      }
42
43      public function getDefinitions($wordId) {
44         # For a given word ID return list of definitions if exist
45         $tmp = array();
46         $res = $this->_connection->query("SELECT DISTINCT sow_value FROM
               word_has_sow WHERE wordid=$wordId AND sowid IN (SELECT sowid
               FROM sow WHERE sow_name = 'Definicija')");
47         while ($row = $res->fetch_object()) {
48            array_push($tmp, $row->sow_value);
49         }
50         return $tmp;
51      }
52
53      public function str2words($str) {
54         # For a given string extracts only words and returns as array
55         $tmp = array();
56         $re = '/\b([a-zA-Zčćšžđ]+)\b/u';
57         preg_match_all($re, mb_strtolower($str), $matches, PREG_SET_ORDER,
               0);
58         foreach ($matches as $val) {
59            array_push($tmp,$val[0]);
60         }
61         return $tmp;
62      }
63
64      public function words2lemmas($words) {
65         # For a given list of words returns list of lemmas
66         $tmp = array();
67         foreach ($words as $word) {
68            $res = $this->_connection->query("SELECT l.lemma FROM words w
                  LEFT JOIN lemmas l ON (w.lemmaid=l.lemmaid) WHERE
                  w.word='$word'");
```

```
69            $row = $res->fetch_object();
70            array_push($tmp, $row->lemma);
71         }
72         return $tmp;
73      }
74
75
76      public function filterIncludeWOS($words, $woses) {
77         # For a given list of words include these that have some WOS values
78         $tmp = array();
79         $wosarr = array();
80         # Variable $wosarr contains ID of WOSes we're looking for
81         foreach ($woses as $wos) {
82            array_push($wosarr, $this->getWosId($wos));
83         }
84         foreach ($words as $word) {
85            $res = $this->_connection->query("SELECT COUNT(*) AS br FROM
                  word_has_wos WHERE wordid IN (SELECT wordid FROM words WHERE
                  word='$word') AND wosid IN (".implode(",",$wosarr).")");
86            $row = $res->fetch_object();
87            if ($row->br > 0) array_push($tmp, $word);
88         }
89         return $tmp;
90      }
91      public function filterExcludeWOS($words, $woses) {
92         # For a given list of words exclude these that have some WOS values
93         $tmp = array();
94         $wosarr = array();
95         # Variable $wosarr contains ID of WOSes we're looking for
96         foreach ($woses as $wos) {
97            array_push($wosarr, $this->getWosId($wos));
98         }
99         foreach ($words as $word) {
100           $res = $this->_connection->query("SELECT COUNT(*) AS br FROM
                  word_has_wos WHERE wordid IN (SELECT wordid FROM words WHERE
                  word='$word') AND wosid IN (".implode(",",$wosarr).")");
101           $row = $res->fetch_object();
102           if ($row->br == 0) array_push($tmp, $word);
103        }
104        return $tmp;
```

```
105     }
106     public function cleanup($words) {
107        # Final cleanup of words
108        $tmp = array();
109        $words = array_unique($words);
110        foreach ($words as $word) {
111           if (mb_strlen($word) > 2) array_push($tmp, $word);
112        }
113        return $tmp;
114     }
115     public function createDomain($domain_name, $words) {
116        $res = $this->_connection->query("INSERT INTO domains
              (domain_type, domain_name) VALUES ('static', '$domain_name')");
117        $id = $this->_connection->insert_id();
118        $orderno = 1;
119        foreach ($words as $word) {
120           $lid = $this->getLemmaIdFromWord($word);
121           $res = $this->_connection->query("INSERT INTO domain_parts
                 (domainid, element_type, element_id, orderno) VALUES ($id,
                 'lid', $lid, $orderno)");
122           $orderno++;
123        }
124        return $id;
125     }
126
127  }
128
129
130  # Create new instance of DomainCreator class
131  $dc = new DomainCreator($db);
132  $domain_counter = 1;
133  # For each WORD in the Lexicon find its definitions
134  $res = $db->query("SELECT wordid FROM words");
135  while ($row=$res->fetch_object()) {
136     foreach($dc->getDefinitions($row->wordid) as $def) {
137        if (mb_strlen($def) == 0) continue;
138        # Get words from definition
139        $def_words = $dc->str2words($def);
140        # Extract only Nouns, Adjectives, Verbs and Adverbs
141        $def_na = $dc->filterIncludeWOS($def_words, array("Imenica",
```

```
            "Pridjev", "Glagol", "Prilog"));
142         # Exclude eventualy ambigous words Conjunctions and Pronouns
143         $def_exConjunct = $dc->filterExcludeWOS($def_na, array("Veznik",
                "Zamjenica"));
144         # Convert words to lemmas
145         $def_lemmas = $dc->words2lemmas($def_exConjunct);
146         # Remove eventual duplicates
147         $def_cleanup = $dc->cleanup($def_lemmas);
148         # Create domain
149         $domain_name = "Domain #". $domain_counter;
150         $did = $dc->createDomain($domain_name, $def_cleanup);
151         $domain_counter++;
152         # Assign a domain to a word
153         $db->query("INSERT INTO domains_has_words (domainid, wordid)
                VALUES (".$did.",".$row->wordid . ")");
154     }
155 }
156 ?>
```

B. Creation of RDF triples

```
#!/usr/bin/php
<?PHP

   require_once("../class.db.php");
   $db = new DB();
   $total = 0;

   Class LodCreator {
      protected $_connection;

      public function __construct($conn)
      {
         $this->_connection = $conn;
         file_put_contents("../owl/ssf.n3", "# SSF Ontology - " .
             date("Y-m-d H:i:s") . " \n");
         file_put_contents("../owl/ssf.xml", "<?xml version=\"1.0\"?>\n");
      }

      public function getWOS($wordid) {
         $tmp = array();
         $res = $this->_connection->query("select w.wos_name,
             whw.wos_value_html from word_has_wos whw LEFT JOIN wos w ON
             (whw.wosid=w.wosid) WHERE whw.wordid=". $wordid;
         while ($row = $res->fetch_object()) {
            array_push($tmp, array($row->wos_name, $row->wos_value_html));
         }
         return $tmp;

      }

      public function getSOW($wordid) {
         $tmp = array();
         global $dbpedia, $wordnet, $babel;
         $res = $this->_connection->query("select s.sow_name,
             whs.sow_value_html from word_has_sow whs LEFT JOIN sow s ON
             (whs.sowid=s.sowid) WHERE whs.wordid=". $wordid;
         while ($row = $res->fetch_object()) {
```

```
33            if (strpos($row->sow_value_html,"dbpedia") > 0) $dbpedia++;
34            if (strpos($row->sow_value_html,"babelnet") > 0) $babel++;
35            if (strpos($row->sow_value_html,"wordnet") > 0) $wordnet++;
36            array_push($tmp, array($row->sow_name, $row->sow_value_html));
37         }
38         return $tmp;
39      }
40
41      public function getSOWIndividuals() {
42         $tmp = "";
43         $res = $this->_connection->query("SELECT distinct sow_name FROM
               sow WHERE languageid=1");
44         while($row = $res->fetch_object()) {
45               $tmp .= "<http://www.ss-framework.com/owl/sow/" .
                     str_replace(",","",str_replace(" ", "_",$row->sow_name))
                     . "> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                     <http://www.ss-framework.com/owl/sow> .\n";
46               $tmp .= "<http://www.ss-framework.com/owl/sow/" .
                     str_replace(",","",str_replace(" ", "_",$row->sow_name))
                     . "> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                     <http://www.w3.org/2002/07/owl#NamedIndividual> .\n";
47         }
48         return $tmp;
49      }
50
51      public function getWOSIndividuals() {
52         $tmp = "";
53         $res = $this->_connection->query("SELECT distinct wos_name FROM
               wos WHERE languageid=1");
54         while($row = $res->fetch_object()) {
55               $tmp .= "<http://www.ss-framework.com/owl/wos/" .
                     str_replace(",","",str_replace(" ", "_",$row->wos_name))
                     . "> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                     <http://www.ss-framework.com/owl/wos> .\n";
56               $tmp .= "<http://www.ss-framework.com/owl/wos/" .
                     str_replace(",","",str_replace(" ", "_",$row->wos_name))
                     . "> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                     <http://www.w3.org/2002/07/owl#NamedIndividual> .\n";
57         }
58         return $tmp;
```

```
59      }
60
61      public function push2n3($str) {
62          #echo $str . "\n";
63          file_put_contents("../owl/ssf.n3", $str . "\n", FILE_APPEND);
64      }
65
66      public function push2xml($str) {
67          #echo $str . "\n";
68          file_put_contents("../owl/ssf.xml", $str . "\n", FILE_APPEND);
69      }
70
71  }
72
73
74  # Create new instance of LodCreator class
75  $lc = new LodCreator($db);
76
77  # For each WORD in the Lexicon
78  $res = $db->query("select * from words;");
79   $lc->push2xml("<rdf:RDF
        \nxmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"\n
        xmlns:lexinfo=\"http://www.lexinfo.net/ontology/2.0/lexinfo#\"
        \nxmlns:ssf=\"http://www.ss-framework.com/owl/\"
        \nxml:base=\"http://www.ss-framework.com/owl/\"\n
        xmlns:owl=\"http://www.w3.org/2002/07/owl#\"\n
        xmlns:rdfs=\"http://www.w3.org/2000/01/rdf-schema#\">\n");
80   $lc->push2xml("<owl:Ontology
        rdf:about=\"http://www.ss-framework.com/owl/\"/>");
81   $lc->push2n3("@prefix : <http://www.ss-framework.com/owl/> .");
82   $lc->push2n3("@prefix owl: <http://www.w3.org/2002/07/owl#> .");
83   $lc->push2n3("@prefix rdf:
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .");
84   $lc->push2n3("@prefix xml: <http://www.w3.org/XML/1998/namespace> .");
85   $lc->push2n3("@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .");
86   $lc->push2n3("@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .");
87   $lc->push2n3("@prefix lexinfo:
        <http://www.lexinfo.net/ontology/2.0/lexinfo#> .");
88   $lc->push2n3("@prefix lemon: <http://www.lemon-model.net/lemon#> .");
89   $lc->push2n3("@base <http://www.ss-framework.com/owl> .");
```

```
90      $lc->push2n3("<http://www.ss-framework.com/owl> rdf:type owl:Ontology
            .");
91      $lc->push2n3(":wordid rdf:type owl:DatatypeProperty ;");
92      $lc->push2xml("<owl:DatatypeProperty
            rdf:about=\"http://www.ss-framework.com/owl/sow\"/>");
93      $lc->push2xml("<owl:DatatypeProperty
            rdf:about=\"http://www.ss-framework.com/owl/wos\"/>");
94      $lc->push2n3("      rdfs:domain :Word .");
95      # Classes
96      $lc->push2n3(":SOW rdf:type owl:Class .");
97      $lc->push2n3(":WOS rdf:type owl:Class .");
98      $lc->push2n3(":Word rdf:type owl:Class .");
99      $lc->push2xml("<owl:Class
            rdf:about=\"http://www.ss-framework.com/owl/word\"/>");
100     # Individuals
101     $lc->push2n3($lc->getWOSIndividuals());
102     $lc->push2n3($lc->getSOWIndividuals());
103    while ($row=$res->fetch_object()) {
104        $total++;
105        $woses = $lc->getWOS($row->wordid);
106        $sows = $lc->getSOW($row->wordid);
107        $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
               str_replace(",","",str_replace(" ", "_",$row->word)) . ">
               <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
               <http://www.ss-framework.com/owl/word> .");
108        $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
               str_replace(",","",str_replace(" ", "_",$row->word)) . ">
               <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
               <http://www.w3.org/2002/07/owl#NamedIndividual> .");
109        $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
               str_replace(",","",str_replace(" ", "_",$row->word)) . ">
               <http://www.ss-framework.com/owl/wordid> \"".$row->wordid."\"
               .");
110        $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
               str_replace(",","",str_replace(" ", "_",$row->word)) . ">
               rdf:type lemon:LexicalEntry .");
111        $lc->push2xml("\t<rdf:Description
               rdf:about=\"http://www.ss-framework.com/owl/word/" . $row->word
               . "\">");
112        $lc->push2xml("\t<rdf:type
```

```
          rdf:resource=\"http://www.ss-framework.com/owl/word\"/>");
113       foreach ($sows as $sow) {
114       if (empty($sow[1])) {
115          $lc->push2xml("\t\t<ssf:SOW>".$sow[0]."</ssf:SOW>");
116          $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
              str_replace(",","",str_replace(" ", "_",$row->word)) . ">
              <http://www.ss-framework.com/owl/hasSOW>
              <http://www.ss-framework.com/owl/sow/" .
              str_replace(",","",str_replace(" ", "_", $sow[0])) . "> .");
117       } else {
118          if ($sow[0] == "owl:sameAs") {
119             $lc->push2xml("\t\t<owl:sameAs
                 rdf:resource=\"".$sow[1]."\"/>");
120          } else {
121             $lc->push2xml("\t\t<".$sow[1].">".$sow[0]."</".$sow[1].">");
122          }
123          $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
              str_replace(",","",str_replace(" ", "_",$row->word)).">
              <".$sow[0]."> <" . str_replace(",","",str_replace(" ", "_",
              $sow[1]))."> .");
124       }
125      }
126    foreach ($woses as $wos) {
127   if (empty($wos[1])) {
128      $lc->push2xml("\t\t<ssf:WOS>".$wos[0]."</ssf:WOS>");
129      $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
          str_replace(",","",str_replace(" ", "_",$row->word)).">
          <http://www.ss-framework.com/owl/hasWOS>
          <http://www.ss-framework.com/owl/wos/" .
          str_replace(",","",str_replace(" ", "_", $wos[0]))."> .");
130   } else {
131      $lc->push2xml("\t\t<".$wos[1].">".$wos[0]."</".$wos[1].">");
132      $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
          str_replace(",","",str_replace(" ", "_",$row->word)).">
          <".$wos[0]."> <" . str_replace(",","",str_replace(" ", "_",
          $wos[1]))."> .");
133   }
134   if ($wos[0] == "Imenica") {
135      $lexinfo++;
136      $lc->push2xml("\t\t
```

```
        <lexinfo:partOfSpeech>lexinfo:noun</lexinfo:partOfSpeech>");
137     $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
            str_replace(",","",str_replace(" ", "_",$row->word)) . ">
            lexinfo:partOfSpeech lexinfo:noun .");
138   }
139   if ($wos[0] == "Glagol") {
140     $lc->push2xml("\t\t
            <lexinfo:partOfSpeech>lexinfo:verb</lexinfo:partOfSpeech>");
141     $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
            str_replace(",","",str_replace(" ", "_",$row->word)) . ">
            lexinfo:partOfSpeech lexinfo:verb .");
142     $lexinfo++;
143   }
144   if ($wos[0] == "Pridjev") {
145     $lc->push2xml("\t\t
            <lexinfo:partOfSpeech>lexinfo:adjective</lexinfo:partOfSpeech>");
146     $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
            str_replace(",","",str_replace(" ", "_",$row->word)) . ">
            lexinfo:partOfSpeech lexinfo:adjective .");
147     $lexinfo++;
148   }
149   if ($wos[0] == "Zamjenica") {
150     $lc->push2xml("\t\t
            <lexinfo:partOfSpeech>lexinfo:pronoun</lexinfo:partOfSpeech>");
151     $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
            str_replace(",","",str_replace(" ", "_",$row->word)) . ">
            lexinfo:partOfSpeech lexinfo:pronoun .");
152     $lexinfo++;
153   }
154   if ($wos[0] == "Veznik") {
155   $lc->push2xml("\t\t
156   <lexinfo:partOfSpeech>lexinfo:conjunction</lexinfo:partOfSpeech>");
157   $lc->push2n3("<http://www.ss-framework.com/owl/word/" .
        str_replace(",","",str_replace(" ", "_",$row->word)) . ">
        lexinfo:partOfSpeech lexinfo:conjunction .");
158   $lexinfo++;
159   }
160   }
161     $lc->push2xml("\t</rdf:Description>");
162   }
```

```
163     $lc->push2xml("</rdf:RDF>");
164     echo "Total triples: " . $total . "\n";
165 ?>
```

C. Vocal changes in Croatian

Sound Assimilation is a vocal change which occurs when a voiced consonant is preceding a devoiced one. It changes into its devoiced pair, and vice versa. If a devoiced consonant is preceding a voiced one, it changes into its voiced pair. This assimilation may influence the voiced consonant before a devoiced one (nominative cro. *lažac* – genitive cro. *lašca* (eng. liar)) or a devoiced consonant before a voiced one (verb cro. *svjedočiti* – noun cro. *svjedodžba* (eng. testimonial)).

Articulation assimilation is a change which occurs when the phoneme *s* is replaced with *š* when preceding *č, ć, nj, lj* (cro. *misao - mišlju* (eng. thought), nominative and instrumental case, cro. *tijesan – tješnji* (eng. tight), positive and comparative) and *z* is replaced with *ž* before *đ, dž, lj, nj* (cro. *voziti – vožnja* (eng. drive), verb and noun). This assimilation is sometimes conditioned by the changes which have already taken place (iotation, palatalization, insertion of a, sound assimilation, cro. *pisac* > *pisče* > *pišče* (eng. writer), nominative and vocative singular, cro. *zvijezda* (eng. star) > cro. *zvijezđe* > *zviježđe* (eng. constellation)). It is important to note that the assimilation does not take place if *s*, or *z* respectively, is the last consonant of the prefix in compound words (cro. *izljubiti* (eng. kiss many times), cro. *raspustiti* (eng. 1. let go; 2. let down; 3. dismiss)), or if it is before sonants *lj* or *nj* which are results of the yat alterations (cro. *slijediti* (eng. follow) – cro. *sljedba* (eng. the act of following), cro. *snijeg – snjegovi* (eng. snow) singular and plural). Furthermore, articulation assimilation states that *n* is replaced with *m* before labial consonants *b, p* (cro. *obranben* > *obramben* (eng. defensive), cro. *stanben* > *stamben* (eng. related to an apartment)). This is regulated in writing by orthography as a convention of writing *n* (cro. *izvanparnični, stranputica*), but is pronounced *m*. The assimilation also takes place when *h* is replaced by *š* if it is preceding *č, ć*, including diminutives with suffixes *-čić, -če* (cro. *kruh* eng. bread – cro. *kruščić* (eng. little bread), cro. *orah* (eng. nut) – *oraščić* (eng. little nut)).

Consonant elimination occurs when phonemes *t* and *d* are eliminated from the clusters *stk, stl, stn, štn, štnj, ždn* (cro. *naprstak - naprstka* > *naprska* (eng. thimble); in nominative and genitive case, cro. *rastao - rastla* > *rasla* (eng. grow) pres part *m* and *f* singular, cro. *mjesto* (eng. place) - *mjestni* > *mjesni* (eng. local)). The same phonemes are eliminated before the cluster *št* (cro. *gospodština* > *gospoština* (eng. attributed to gentlemen), cro. *hrvatština* > *hrvaština* (eng. attributed to Croatia)), and before *c, č* and *ć* (cro. *sudac – sudca* > *suca* (eng. judge) in nominative and genitive singular, cro. *otac – otca* > *oca* (eng. father) in nominative and genitive singular). Orthographic rules determine the way these clusters must be written, and according to the newest orthography, the phonemes *t* and *d* are to be written in all clusters except in cro. *otca / oca* (eng.

father) in genitive singular. It is important to note that sometimes the orthographic norm is different from the spoken language in use. For example, the orthography says that the adjective is cro. *hrvatski* (eng. Croatian), while it is pronounced cro. *hrvaski*. If the same two consonants are found in the immediate vicinity of each other, one of them is eliminated (cro. *preddvorje* > *predvorje* (eng. lobby)), except for the *jj* cluster in the superlative forms of adjectives and adverbs (cro. *najjači* (eng. the strongest), cro. *najjasnije* (eng. the clearest)). This phenomenon is often caused by other changes, such as articulation assimilation (cro. *bezžični* > *bežžični* > *bežični* (eng. wireless)). In the Croatian language, an elimination of sound groups is also possible. This occurs in the following ways: *-in* is eliminated in the plural of masculine nouns ending in *-(j)aninin*, *-čanin*, *-đanin*, *-ljanin*, *-njanin* (cro. *Splićanin – Splićani* (eng. man from Split) in nominative singular and plural, cro. *državljanin – državljani* (eng. citizen) in nominative singular and plural); *-ij-* and *-in-* are eliminated from the derivatives of geographic names ending in *-ija, -ina* (cro. *Etiopija* (eng. Ethiopia) – cro. *Etiopljanin* (eng. man from Ethiopia), cro. *Slovenija* (eng. Slovenia) – cro. *Slovenac* (eng. man from Slovenia); and the adverbial ending *-ak, -ek, -ok* in the comparative from (cro. *kratak* (eng. short) – *kraći* (eng. shorter), cro. *visok* (eng. tall) – *viši* (eng. taller)).

Insertion of the sonant '*j*' is a change that occurs always, but it is not always noted in writing. The written rule is to note the *j* in all cases except in *io* (cro. *radio* 'radio'), on the boundary between the prefix and the root, and in the groups *ai*, *ei* and *ui*, which are primarily derived from internationalisms adapted to Croatian.

Insertion of the Vowel '*a*' is a change when the vowel *a* is inserted in nominative singular and genitive plural of masculine nouns ending in *-(a)c*, *-l(a)c* (cro. *konac – konca – konaca* 'thread'), *-(a)k* (cro. *mačak – mačka – mačaka* (eng. male cat)) *-(a)l/-(a)o* (cro. *kotao – kotla – kotala* (eng. cauldron)), *-(a)lj* (cro. *pedalj – pedalja* (eng. span)); *-(a)m* (cro. *jaram – jarama* (eng. yoke)); *-(a)n* (cro. *ovan – ovnova* (eng. ram)); *-(a)nj* (cro. *režanj – režanja* (eng. slice)). Furthermore, it is in the genitive plural of polysyllabic feminine e-declination nouns with a consonant cluster at the root ending (cro. *mačka - mačaka* (eng. cat), cro. *banka - banaka* (eng. bank), cro. *bukva - bukava* (eng. beech tree)), then in the nominative singular of i-declination nouns ending with: *-(a)n* (cro. *plijesan – plijesni* (eng. mold), cro. *sablazan – sablazni* (eng. terror)); *-(a)o* (cro. *misao – misli* (eng. thought), including the words with prefixes, for example cro. *zamisao* (eng. idea)), in genitive plural of neuter nouns whose root ends with a consonant cluster (cro. *deblo – debala* (eng. bole, trunk), cro. *veslo – vesala* (eng. oar)), in nominative singular of masculine indefinite adjectives with the endings: *-(a)k* (cro. *gorak, kratak*), *-(a)lj* (cro. *šupalj*), *-(a)n* (cro. *bučan*), *-ao* (> *(a)l*) (cro. *nagao – nagli*, cro. *topao – topli*), *-(a)r* (cro. *bistar – bistri*), *-(a)v* (cro. *mrtav – mrtvi*); and finally in the interrogative and relative pronouns ending in *-(a)v* (cro.

kakav (eng. what kind), cro. *takav* (eng. such), cro. *kojekakav* (eng. any)). Croatian has three groups of dialects (Čakavian (Chakavian), Kajkavian and Štokavian (Stokavian)). Along with the onomastic data, the dialects are reflected in standard Croatian, through the phenomenon of insertion of the vowel *e*, which is found in Kajkavian names, surnames, nicknames and toponyms (cro. *Čakovec, Črnomerec, Vugrovec; Gubec, Ozimec, Tkalec*), and derivatives from them (cro. *tuheljski, čakovečki*). In some surnames, the vowel *e* is found consistently throughout the paradigm (*Vrabec – Vrabeca, Zebec – Zebeca*). The vowel *e* is not consistent in the genitive of names, nicknames and toponyms (cro. *Čakovca, Vugrovca, Gupca, Ozimca, Tkalca*), and in the adjectives derived by the suffix *-ov/-ev* (cro. *Brezovčev, Gupčev*).

Replacement of Final '*l*' with '*o*' or vocalization is a change that occurs when *l* is changed to *o* at the ending of a syllable or a word: cro. *mil > mio* (eng. nice), cro. *vesel > veseo* (eng. happy). This change occurred early in the development of Štokavian (Stokavian) and its results are visible in the standard since it is based on a Štokavian dialect. Čakavian (Chakavian) and Kajkavian dialects preserve the *-l*. The change is found in the present participle of masculine singular forms: cro. *čitao (< čital)* (eng. read), cro. *čuo (<čul)* (eng. hear), in nominative and accusative singular of indefinite adjectives (cro. *cio* (eng. whole), cro. *nagao* eng. abrupt), cro. *vreo* (eng. hot), cro. *topao* (eng. warm)), in nominative and accusative singular of certain masculine and feminine nouns (cro. *anđeo* (eng. angel), cro. *dio* (eng. part), cro. *kabao* (eng. bucket), cro. *pomisao* (eng. thought), cro. *idea*, cro. *pogibao* (eng. jeopardy)), and in the declination of nouns ending with *-lac*, except in genitive plural, where it remains *-lac*.

Ablaut is a change when the vowel *o* is replaced with the vowel *e* in instrumental singular of masculine nouns (cro. *mačem* (eng. sword), cro. *konjem* (eng. horse)), and in the expanded plural of the masculine nouns (cro. *mačevi, krajevi* (eng. area)), while the examples without expansion are cro. **mači*, cro. **kraji*. There are exceptions, such as monosyllabic, and occasionally bi-syllabic words which have the vowel *e* in the syllable preceding the suffix (cro. *Bečom, kaležom*). The nouns formed with the suffix *-ar* can have different case suffixes in instrumental singular (cro. *ribarom/ribarem* (eng. fisherman), *rudarom/rudarem* (eng. miner)). The exceptions are nouns in which *e* is not a part of the suffix (cro. *carem* (eng. emperor), cro. *darom* (eng. gift), cro. *parom* (eng. pair), cro. *žarom* (eng. spark)). The change is often caused by consonant alternations (iotation: cro. *smetje > smeće* (eng. trash), cro. *predgradje > predgrađe* (eng. suburb)), and the final *e* is a variant of the grammatical suffix. Thus, a morphological change is also noted as a replacement of the suffix *-om > -em* (cro. *kraljom > kraljem*).

Replacement of yat is a change when the short syllable of the long root *yat* is replaced with the short *e / je* if the root morpheme preceding it consists of a consonant cluster

ending with *r* (cro. *brijeg – brjegovi / bregovi*). Orthographic manuals determine the notation *je* or *e*, except in certain examples in which *e* always prevails (cro. *vrijeme – vremena* (eng. time)).

Palatalization occurs in the vocative singular of masculine nouns (cro. *težak – težače* (eng. farmer), cro. *stric – striče* (eng. uncle)), in the nouns cro. *oko – oči* (eng. eye), cro. *uho – uši* (eng. ear), in the present tense of certain verbs ending with *-ći* (cro. *vući – vučem – vuku* (eng. pull), cro. *strići – strižem – strigu* (eng. shave [sheep])), in the aorist tense of some verbs on *-ći* (cro. *rekoh - reče* (eng. say), cro. *digoh - diže* (eng. lift)), in the word formation of nouns with the suffixes *-ica, -ina, -ić* (cro. *djevojka* (eng. girl) - *djevojčica* (eng. little girl), cro. *Mjesec* (eng. Moon) - *mjesečina* (eng. moonlight), cro. *zec* (eng. rabbit) - *zečić* (eng. little rabbit)), and in the word formation of adjectives ending with *-ev* derived from the nouns ending with c (cro. *bijelčev* (eng. white man's), cro. *stričev* (eng. uncle's), cro. *zečev* (eng. rabbit's)).

Sibilarization occurs in the dative and locative singular of feminine nouns (cro. *ruka – ruci* (eng. arm), cro. *snaha – snasi* (eng. daughter in law), cro. *sloga – slozi* (eng. harmony)), in nominative, vocative, dative, locative and instrumental plural of masculine nouns (cro. *seljak – seljaci* (eng. peasant), cro. *jastog – jastozi* (eng. lobster)), exceptionally in dative, locative and instrumental plural of the noun cro. *klupko – klupcima* (eng. hank), and in the imperfect tense of verbs (cro. *pecijah* (eng. bake)) and imperative (cro. *peci, pecite* (eng. bake) singular and plural, cro. *lezite* (eng. lay down) plural). It is important to say that the e-declination has a lot of exceptions.

Iotation occurs with the formulas: $c + j > č$ (cro. *micati – mičem* (eng. move) infinitive and present 1st person singular), $d + j > đ$ (cro. *glad – glađu* (eng. hunger), in nominative and instrumental, cro. *mlad – mlađi* (eng. young) positive and comparative), $g + j > ž$ (cro. *drag – draži* (eng. dear), positive and comparative), $h + j > š$ (cro. *tih – tiši* (eng. quiet) positive and comparative), $k + j > č$ (cro. *jak – jači* (eng. strong), positive and comparative), $l + j > lj$ (cro. *sol – solju* (eng. salt), Nominative and Instrumental), $n + j > nj$ (cro. *tanak – tanji* (eng. thin), positive and comparative), $s + j > š$ (cro. *visok – viši* (eng. tall), positive and comparative), $t + j > ć$ (cro. *cvijet – cvijeće* (eng. flower), singular and collective plural), $z + j > ž$ (cro. *brz – brži* (eng. quick), positive and comparative).

D. Python Natural Language Functions

Def2domain Semantic function

Function definition:

```
json Def2domain (word [, wos=[], source=true|false])
```

Parameters:

```
string word - searched word
int wos [optional] - list of WOS marks in the output
boolean source [optional] - show definition source
```

Return values:

```
Returns JSON object (domain) with words that are related to the input
word based on the word's definition.
```

Example:

```
=Def2domain('mozak')

# OUTPUT:
[['mozak', 'visok', 'dio', 'središnji', 'u', 'životinja', 'i',
    'središte', 'primati', 'informacija', 'iz', 'obrađivati', 'ih',
    'slati', 'uputa', 'izvršan', 'organ', 'on', 'sjedište',
    'inteligencija', 'nizak', 'živčan', 'sustav', 'čin', 'mreža',
    'međusoban', 'dug', 'funkcija', 'im', 'primanje', 'podražaj',
    'predavanje' ... ]]
```

Function `Def2domain()` uses the definition of the word from the related SOW tags to create a domain of words related to the source word. Every word from such definition is lemmatized and returned as a JSON object. The function accepts two additional optional parameters (`wos` and `source`). If `wos` parameter is given then the output will be filtered only to these words that are tagged with listed values. It is extremely useful if some word classes (e.g. conjunctions) for which it is usually not important that they are a part of resulting domain are excluded from the output. Finally, the `source` parameter defines whether the source of the used definition will be shown or not.

MarkovChain Statistical function

Function definition:

```
string MarkovChain (docs, n [, startword])
```

Parameters:

```
int docs - documents that are used as a source for Markov chain
           algorithm
int n - number of words in resulting sentence
string startword [optional] - starting word of resulting sentence.
                              If left empty, random word is used.
```

Return values:

```
Returns sentence generated using Markov chain algorithm and based on
given input parameters.
```

Example:

```
=MarkovChain([1,2,3,4,5], 5, 'i')

# OUTPUT:
'i vjetar ljulja ogradu polako'
```

Generation of new sentences based on the already loaded corpora in the SSF can be done using the function `MarkovChain()`. The function accepts two mandatory and one optional parameter. The first parameter `docs` is a list of ID's of documents from the corpora that will be used as a base for calculation of word occurrence probabilities. The second parameter `n` is a number of words that will appear in the resulting sentence. The third optional parameter `startword` is a string that will be used as a starting point of Markov chain algorithm. If this parameter is left out, the random word from corpora is used.

ChangeTense Morphological function

Function definition:

```
string ChangeTense (sentence, time)
```

Parameters:

```
string sentence - source sentence
string time - verb time to which source sentence will be converted
```

Return values:

```
Returns sentence in verb time defined as the second parameter
```

Example:

```
=ChangeTense('vidim plavu kuću', 'aorist')

# OUTPUT:
'vidjeh plavu kuću'
```

Function `ChangeTense()` is used when there is a need to change the verb's tense in sentence(s). The first parameter sentence accepts sentence(s) in which verb's tense will be changed. It is not necessary to provide whole sentence; the function will also accept only one word that needs to be changed. The second parameter is target `time`. The function first iterates over words inside a sentence. If the currently observed word is a verb (based on WOS tags the word is associated to) then the function steps into a procedure of verb tense change. If the observed word is not a verb it is simply copied to an output. The verb tense change procedure firstly finds a lemma of the word and after that it goes through the lexicon and searches for a word that is tagged as a verb, in the same number and person as the source verb but with targeted verb's tense. When such a word is found it is pushed to the output stack. When the iteration process is ended, the stacked output is returned as a string from the function.

The NLF function `Gets()` converts the source sentence to a list of tags under the WOS branch which is passed as a second parameter. Source sentence is tokenized and then iterated. In each iteration step, the algorithm checks which WOS tags from the corresponding branch (from the second parameter) the observed word has assigned to it. If a child tag is found it is pushed to an output stack, and if there is no appropriate tag that is assigned to an observed word - the `ULO` (Unidentified Linguistic Object) mark is returned.

Gets WOS/SOW function

Function definition:

```
string Gets (sentence, tags)
```

Parameters:

```
string sentence - source sentence
string tags - parent WOS/SOW tag
```

Return values:

```
Converts a sentence to a tag pattern based on WOS parent parameter
```

Example:

```
=Gets('vidim plavu kuću', 'Vrsta riječi')

# OUTPUT:
Glagol Pridjev Imenica
```

Function `Plural()` as an input parameter takes a string (a source sentence) which is converted into plural. The function iterates over words in the sentence and for every word that is in the singular first it finds its lemma which is used to find a word in the lexicon with the same lemma but is tagged with WOS mark of plural. Such word is pushed to an output stack. Once the iteration process is finished the stack output is returned as a string value.

Plural Morphological function

Function definition:

```
string Plural (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Converts a sentence in singular to plural
```

Example:

```
=Plural('vidim plavu kuću')

# OUTPUT:
vidimo plave kuće
```

Functions `wos()` and `sow()` are used when there is a need to get an ID of a specific WOS/SOW tag for later usage inside other functions. Both return JSON objects and are called without any parameters.

wos WOS/SOW function

Function definition:

```
json wos
```

Parameters:

```
NONE
```

Return values:

```
Returns JSON object with all WOS marks.  Each element has ID of
the WOS mark and its name.
```

Example:

```
=wos()

# OUTPUT:
[(1, 'Vrsta riječi'), (2, 'Imenica'), (3, 'Zamjenica'), (4,
   'Pridjev'), (5, 'Glagol'), (6, 'Broj'), (7, 'Prilog'), ... ]
```

sow WOS/SOW function

Function definition:

```
json sow
```

Parameters:

```
NONE
```

Return values:

```
Returns JSON object with all SOW marks.  Each element has ID of
the SOW mark and its name
```

Example:

```
=sow()

# OUTPUT:
[(1, 'Opće'), (2, 'Živo'), (3, 'Pojam'), (9, 'Ime'), (16, 'Osoba'),
(17, 'Tijelo'), (18, 'Doživljaj'), (19, 'Spoznaja') ... ]
```

The following two functions (`CountWOS()` and `CountSOW()`) are used for counting words in the lexicon which are tagged with some specific tag. The functions accept ID of the WOS/SOW tag as the input parameter, and returns the number of occurrences as an integer.

CountWOS — WOS/SOW function

Function definition:

```
int CountWOS (id)
```

Parameters:

```
int id - ID of the WOS tag
```

Return values:

```
Returns the number of words tagged with the target tag
```

Example:

```
=CountWOS(2)

# OUTPUT:
178968
```

CountSOW — WOS/SOW function

Function definition:

```
int CountSOW (id)
```

Parameters:

```
int id - ID of the SOW tag
```

Return values:

```
Returns the number of words tagged with the target tag
```

Example:

```
=CountSOW(97)

# OUTPUT:
35699
```

Above examples shows the output of function `CountWOS()` for WOS tag with the ID 2 (Nouns), and the result shows that in Croatian language (according to the SSF's lexicon), there are 178,968 words which are tagged as nouns. The function `CountSOW()` when called with the parameter ID 97 (Croatian WordNet definition), show that there are 35,699 words in the SSF's lexicon which have definition.

The following set of functions are intended for administrative purposes (creating new words, multiwords, lemmas and assigning or removing specific WOS/SOW tags to them). For usage of all assignment and removing functions, the administrative rights are required.

AssignWOS — WOS/SOW function

Function definition:

```
boolean AssignWOS (wordid, wosid, [, wosvalue])
```

Parameters:

```
int wordid - ID of the target word
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignWOS(2301, 5)
# OUTPUT:
True
```

AssignSOW — WOS/SOW function

Function definition:

```
boolean AssignSOW (wordid, sowid, [, sowvalue])
```

Parameters:

```
int wordid - ID of the target word
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignSOW(2301, 12)
# OUTPUT:
True
```

Both functions, `AssignWOS()` and `AssignSOW()` are used for assigning WOS/SOW tags to words. These functions are extremely useful in the process of automated enrichment of lexicon (either by the inclusion of external resources, or for redacting purposes).

Similarly to functions `AssignWOS()` and `AssignSOW()`, function `AssignWOSLemma()` and `AssignSOWLemma()` are used for assigning WOS/SOW tags to lemmas.

AssignWOSLemma — WOS/SOW function

Function definition:

```
boolean AssignWOSLemma (lemmaid, wosid, [, wosvalue])
```

Parameters:

```
int lemmaid - ID of the target lemma
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignWOSLemma(1200, 5)

# OUTPUT:
True
```

AssignSOWLemma — WOS/SOW function

Function definition:

```
boolean AssignSOWLemma (lemmaid, sowid, [, sowvalue])
```

Parameters:

```
int lemmaid - ID of the target lemma
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignSOWLemma(1200, 12)

# OUTPUT:
True
```

Finally, the last lexicon entities, multiword expressions (MWEs), can also be tagged with special functions `AssignWOSMWE()` and `AssignSOWMWE()`.

AssignWOSMWE — WOS/SOW function

Function definition:

```
boolean AssignWOSMWE (mweid, wosid, [, wosvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignWOSMWE(3103, 5)

# OUTPUT:
True
```

AssignSOWMWE — WOS/SOW function

Function definition:

```
boolean AssignSOWMWE (mweid, sowid, [, sowvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignSOWMWE(3103, 12)

# OUTPUT:
True
```

Removing tags from words, lemmas and MWE's can be done with functions: `RemoveWOS()` and `RemoveSOW()` for word's lexicon, `RemoveWOSLemma()` and `RemoveSOWLemma()` for lexicon of lemmas and `RemoveWOSMWE()` and `RemoveSOWMWE()` for the MWE lexicon. The remove functions have the same parameters layout as assignment functions.

RemoveWOS — WOS/SOW function

Function definition:

```
boolean RemoveWOS (wordid, wosid, [, wosvalue])
```

Parameters:

```
int wordid - ID of the target word
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveWOS(2301, 5)

# OUTPUT:
True
```

RemoveSOW — WOS/SOW function

Function definition:

```
boolean RemoveSOW (wordid, sowid, [, sowvalue])
```

Parameters:

```
int wordid - ID of the target word
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveSOW(2301, 12)

# OUTPUT:
True
```

RemoveWOSLemma — WOS/SOW function

Function definition:

```
boolean RemoveWOSLemma (lemmaid, wosid, [, wosvalue])
```

Parameters:

```
int lemmaid - ID of the target lemma
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveWOSLemma(1200, 5)

# OUTPUT:
True
```

RemoveSOWLemma — WOS/SOW function

Function definition:

```
boolean RemoveSOWLemma (lemmaid, sowid, [, sowvalue])
```

Parameters:

```
int lemmaid - ID of the target lemma
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveSOWLemma(1200, 12)

# OUTPUT:
True
```

RemoveWOSMWE — WOS/SOW function

Function definition:

```
boolean RemoveWOSMWE (mweid, wosid, [, wosvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int wosid - ID of the WOS tag
string wosvalue [optional] - if applicable, the value of the WOS tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveWOSMWE(3103, 5)

# OUTPUT:
True
```

RemoveSOWMWE — WOS/SOW function

Function definition:

```
boolean RemoveSOWMWE (mweid, sowid, [, sowvalue])
```

Parameters:

```
int mweid - ID of the target MWE
int sowid - ID of the SOW tag
string sowvalue [optional] - if applicable, the value of the SOW tag
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=RemoveSOWMWE(3103, 12)

# OUTPUT:
True
```

The creation of new words in the SSF's lexicon is usually done with the integrated morphology generator (which can be accessed from the administrative module of the GUI). In cases when the process of words creation is done from external applications (over the API), or as a part of the NLF script, the usage of functions `InsertWord()`, `InsertMWE()` and `InsertLemma()` is necessary.

InsertWord Syntactic function

Function definition:

```
int InsertWord (word)
```

Parameters:

```
string word - the word that is inserted
```

Return values:

```
Returns the inserted word ID
```

Example:

```
=InsertWord("informatika")

# OUTPUT:
288588
```

InsertMWE Syntactic function

Function definition:

```
int InsertMWE (mwe)
```

Parameters:

```
string mwe - the multiword expression that is inserted
```

Return values:

```
Returns the inserted MWE ID
```

Example:

```
=InsertMWE("morski pas")

# OUTPUT:
43317
```

Each multiword expression is first tokenized, and then for every token (word), checked over the words lexicon. If the word exists in the lexicon, the ID of the word is used in the MWE building procedure. When the word does not exist within the word's lexicon, it is automatically inserted, and assigned to the multiword expression as its part.

InsertLemma Syntactic function

Function definition:

```
int InsertLemma (lemma)
```

Parameters:

```
string lemma - the lemma that is inserted
```

Return values:

```
Returns the inserted lemma ID
```

Example:

```
=InsertLemma("raditi")

# OUTPUT:
3252
```

The inserted lemma is automatically visible in the lemmas lexicon, but to assign it to the specific word in the word's lexicon, the function `AssignLemma2Word()` is used. The function accepts two parameters, the ID of the word, and the ID of the lemma and return True if the operation is successfully completed, or False if the error in the process of assignment occurred (e.g. invalid ID of word/lemma was used).

AssignLemma2Word Syntactic function

Function definition:

```
boolean AssignLemma2Word (wordid, lemmaid)
```

Parameters:

```
int wordid - ID of the word
int lemmaid - ID of the lemma
```

Return values:

```
Returns True upon success, or False upon failure
```

Example:

```
=AssignLemma2Word(3223, 2452)

# OUTPUT:
True
```

Function `Antonym()` as the first parameter takes a word and generates its antonym based on a antonymity criteria which is given as the second parameter. This function relies on semantic domains which are described in Section 6.3. For the given example in the `Antonym()` function the word cro. *mladić* (eng. young man) is given as the first parameter and the second parameter (criteria of antonymity) is the given 'gender'. The output is cro. *djevojka* (eng. young girl). If for instance, instead of 'gender' an 'age' is set as a criteria of antonymity the word cro. *starac* (eng. old man) would be returned as an output. In order that this function might work, the semantic domain with properly ordered words must exist. For the given example at least two such domains are used. First, the domain `gender` which is made of two elements (cro. *muško* (eng. male) and cro. *žensko* (eng. female), and second - the domain `age` which is made of elements cro. *rođenje* (eng. birth), cro. *mladost* (eng. youth), cro. *zrelost* (eng. maturity), cro. *starost* (eng. senility) and cro. *smrt* (eng. death). Since all words are tagged with proper SOW features (e.g. the word cro. *mladić* (eng. young man) is tagged with *male* and *youth* tag and the word cro. *djevojka* (eng. young girl) is tagged with *youth* and *female* tag), it is easy to use semantic domains and find a corresponding word with opposite meaning.

Antonym Semantic function

Function definition:

```
json Antonym (word, criteria)
```

Parameters:

```
string word - source word
string criteria - criteria of antonimity
```

Return values:

```
Returns an antonym of a given word based on specific criteria
```

Example #1:

```
=Antonym('mladić','spol')

# OUTPUT:
['djevojka']
```

Example #2:

```
=Antonym('mladić','dob')

# OUTPUT:
['starac']
```

Synonym Semantic function

Function definition:

```
json Synonym (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Returns a list of synonyms for a given word
```

Example:

```
=Synonym("raditi")

# OUTPUT:
['izrađivati', 'uraditi', 'proizvoditi', 'stvoriti', 'proizvesti',
    'tvoriti', 'praviti', 'činiti', 'učiniti', 'stvarati', 'izraditi',
    'načiniti', 'napraviti', 'izgraditi', 'izgrađivati',
    'konstruirati', 'graditi', 'sagraditi', 'specifičan', 'prosječan',
    'obraditi', 'obrađivati', 'fabricirati', 'upravljati', 'dvoriti',
    'služiti', 'poslužiti', 'posluživati', 'umjeren', 'funkcionirati',
    'slab', 'vršiti', 'obavljati', 'izvršavati', 'provoditi',
    'producirati', 'robijati', 'dirinčiti', 'nespecifičan', 'crnčiti',
    'kulučiti', 'rintati']
```

As described in Section 3.3, the function `Synonym()` uses the information from WOS or SOW tags to create a list of synonyms for a given word and return it in the form of the JSON object. These lexical functions can also be chained, for example, if the function `Anyonym()` is called with parameters cro. *mladić* (eng. young man) as a word, and cro. *dob* (eng. age) as a criteria of anonymity, the output is word cro. *djevojka* (eng. young girl). This output can be directly the input for the function `Synonym()`. The function call `=Synonym(Antonym("mladić", "spol")[0])` will result with the following JSON object: `['dekla', 'gospođica', 'gđica', 'cura', 'mlada dama']`. In the same way any other function output can be input in another function. The function `Antonym("mladić", "spol")` is executed first, resulting with the list of antonyms for the word cro. *mladić* (eng. young man). The first element of the list (i.e. word cro. *djevojka*) is then passed as the first argument in the function `Synonym()`.

Collocation Semantic function

Function definition:

```
json Collocation (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Returns JSON object with all collocations for a given word.
```

Example:

```
=Collocation('labav')

# OUTPUT:
['labava carinska unija', 'labava federacija', 'labava granica',
    'labava kompozicija', 'labava konfederacija', 'labava unija',
    'labave cijene', 'labave uzde', 'labave veze', 'labav režim',
    'labava politička suradnja', 'labava veza među koalicijskim
    strankama']
```

Collocations are multiword expressions (MWE) which are related to a specific word. The function `Collocation()` outputs all these MWEs as a JSON object.

Freq Statistical function

Function definition:

```
int Freq (docs, tags)
```

Parameters:

```
int docs - list of document ID's
string tags - tags that are counted
```

Return values:

```
Returns number of occurances of a words which are tagged with
WOS/SOW tags from the second parameter.
```

Example:

```
=Freq([1,2], ['Imenica'])

# OUTPUT:
7283
```

MatchPattern Syntactic function

Function definition:

```
string MatchPattern (sentence, patternname)
```

Parameters:

```
string sentence - sentence which is tested against a pattern
string patternname - pattern name
```

Return values:

```
Returns part of sentence which matches certain pattern.
```

Example:

```
=MatchPattern('Sutra ću više raditi', 'futur')

# OUTPUT:
ću raditi
```

The function `MatchPattern()` tests a source sentence against an already stored pattern from the database. The process of pattern testing is similar to the one described in Section 5.6. In the first step the sentence is enriched with relevant WOS/SOW marks and then tested against the pre-prepared regular expressions which are stored in the database. Some of possible values (pattern names) are: `s` (subject), `p` (predicate), `o` (object), `aorist`, `apozicija`, `futur`, `imperfekt`, `infinitiv`, `perfekt`, `prezent`, etc.). Example pattern `MatchPattern()` function (future tense), which is used to test if a sentence has auxiliary verb will (WOS ID 153) followed by a main verb (WOS ID 150) in infinitive (WOS ID 183), may look like:

```
.*?\s*(\S+)\[w:153\]\s*.*?\s*(\S+)\[w:150,183\]\s*.*?\s*
```

and when tested against enriched version of the sentence:

```
Sutra␣ću[w:153]␣više␣raditi[w:150,183]
```

the matched result is:

```
ću␣raditi
```

This is a very simple example involving only WOS tags, but in some complex scenarios it is possible to combine it also with SOW tags, or even define specific word forms independently to WOS/SOW marks, just by using standard regular expression rules.

DetectPattern Syntactic function

Function definition:

```
string DetectPattern (sentence, patternname)
```

Parameters:

```
string sentence - sentence which is tested against a pattern
boolean extend - shows matched part of the sentence
```

Return values:

```
Detects a pattern within the sentence.
```

Example:

```
=DetectPattern('Prestao sam pisati', extend=True)

# OUTPUT:
prezent (Prestao sam)
```

Like the `MatchPattern()`, the function `DetectPattern()` tests the sentence in the same way but against all stored patterns in the database and outputs those that have a match.

FreqDist Statistical function

Function definition:

```
int FreqDist (docs, wos, pattern, n, len)
```

Parameters:

```
int docs - list of documents
int wos - list of WOS marks in the output
string pattern - pattern for filtering words
int n - number of letters, morphs or syllables
string len - sylab, morph or char
```

Return values:

```
Returns a number of occurances of words that have n letters, morphs
or syllables and are tagged with specific WOS tags.
```

Example:

```
=FreqDist([1,2], wos=['Imenica'], pattern='%', n=4, len='sylab')

# OUTPUT:
258
```

Function `FreqDist()` returns the number of occurrences of a specific word defined by parameters within selected documents. The example above shows that there are 258 nouns that have 4 syllables in documents 1 and 2.

Ngrams Syntactic function

Function definition:

```
string[] Ngrams (sentence, n, d)
```

Parameters:

```
string sentence - source sentence
int n - size of ngram
int d - number of neighbouring words to combine
```

Return values:

```
Returns a list of n-grams from the source sentence based on
the size of a sliding window
```

Example:

```
=Ngrams("Čitala je njegove pjesme kao da prvi put otkriva snagu
    pjesničke riječi", 2, 2)

# OUTPUT:
[['Čitala', 'je'], ['Čitala', 'njegove'], ['je', 'njegove'], ['je',
    'pjesme'], ['njegove', 'pjesme'], ['njegove', 'kao'], ['pjesme',
    'kao'], ['pjesme', 'da'], ['kao', 'da'], ['kao', 'prvi'], ['da',
    'prvi'], ['da', 'put'], ['prvi', 'put'], ['prvi', 'otkriva'],
    ['put', 'otkriva'], ['put', 'snagu'], ['otkriva', 'snagu'],
    ['otkriva', 'pjesničke'], ['snagu', 'pjesničke'], ['snagu',
    'riječi'], ['pjesničke', 'riječi']]
```

Function `Ngrams()`, as name says, extracts n-grams from sentences. It accepts three parameters, the first parameter is sentence, the second parameter `n` defines the size of n-gram, and parameter `d` defines how many neighbouring words next to the observed word will be in the output.

GwMSY WOS/SOW function

Function definition:

```
json GwMSY (ms, like, orderno)
```

Parameters:

```
string ms - morphs or syllables
string like - pattern to match a syllable or a morph
int orderno - position of a morph or a syllable within the word
              (-1 means last)
```

Return values:

```
Returns JSON object with words split into morphs or syllables
matching a specific pattern.
```

Example:

```
=GwMSY('syllable','ja',-1)

# OUTPUT:
{'ja': ['bo-ja', 'im-ple-men-ta-ci-ja', 'i-de-ja',
    'ko-mu-ni-ka-ci-ja', 'ge-sti-ku-la-ci-ja', 'op-ci-ja',
    'se-lek-ci-ja', 'kog-ni-ci-ja', 'ap-strak-ci-ja', 'po-zi-ci-ja',
    'fun-kci-ja', 're-gi-ja', 'kre-a-ci-ja', 'or-ga-ni-za-ci-ja',
    'si-tu-a-ci-ja', 'for-ma-ci-ja', 'spo-zna-ja', 'pro-fe-si-ja',
    'ka-te-go-ri-ja', 'in-for-ma-ci-ja', 'e-du-ka-ci-ja',
    're-la-ci-ja', 'pu-bli-ka-ci-ja', 'lo-ka-ci-ja', 'e-mo-ci-ja',
    'kon-struk-ci-ja', 'e-sen-ci-ja', 'di-stri-bu-ci-ja', ... ]}
```

Function `GwMSY()` returns JSON object with words split into syllables or morphs based on the criteria provided in parameters. The first is `ms` parameter which decides whether the morphs or syllables are displayed. The `like` parameter is a pattern or an exact string of syllable/morph which is queried, and `orderno` parameter defines the position of the searched syllable/morph within the word. If `orderno` value is set to -1, it means that only syllables/morphs which are at the end of the word are returned. The example above queries lexicon for all words with the last syllable '*ja*'.

SplitSentences | Syntactic function

Function definition:

```
string[] SplitSentences (sentence [, allmatches])
```

Parameters:

```
string sentence - source sentence
boolean allmatches [optional] - Show all possible matches
```

Return values:

```
Returns a list of sentences parts along with corresponding types.
```

Example:

```
=SplitSentences("Koji ne može sebi zapovijedati, ne može ni drugom.")

# OUTPUT:
[['subjektna', 0, '(^\S+\[w:136\].+\[w:5\].+),(.+\[w:5\].+)',
   'Koji[w:136] ne može[w:5] sebi zapovijedati[w:5] , ne može[w:5] ni
   drugom . ', ['Koji ne može sebi zapovijedati', 'ne može ni drugom
   .']]]
```

The `SplitSentences()` function uses a predefined set of syntactical patterns (known as O-structures) to split the enriched version of the complex sentence in its parts (see function `EnrichSentence()`) and detect the type of the sentence. As an output it returns a list of elements. The first element is a type of sentence the algorithm has found. The second element indicates if the matched sentence is in its regular or inverted version (value `0` means the sentence is in regular form, whereas `1` means that it is an inverted sentence). The third element of the output list is the regular expression which was tested against the enriched version of the sentence. The fourth element is the enriched sentence which was tested with the regular expression (the third element), and finally, the last element of the output list is another list which contains all of the sentence elements. The SSF currently contains 14 different sentence types (cro. *apozicijska* (eng. appositional), cro. *atributna* (eng. attribute), cro. *dopusna* (eng. permissible), cro. *mjesna* (eng. place), cro. *namjerna* (eng. intentional), cro. *načinska* (eng. modal), cro. *objektna* (eng. object), cro. *poredbena* (eng. comparable), cro. *posljedična* (eng. subsequent), cro. *predikatna* (eng. predicate), cro. *subjektna* (eng. subject), cro. *uvjetna* (eng. conditional), cro. *uzročna* (eng. causative), cro. *vremenska* (eng. time)) which in all their variants make over 40 different syntactic patterns used by the splitting algorithm. These patterns can be edited in the O-structure tab as described in Section 7.4, with the possibility of creating completely new patterns.

The following three functions: `DetectS()`, `DetectP()` and `DetectO()` are used to detect subject, predicate and object within the sentences. These functions basically implement `MatchPattern()` function.

DetectS — Syntactic function

Function definition:

```
string DetectS (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns the subject from the sentence.
```

Example:

```
=DetectS('Željka kuha ručak')

# OUTPUT:
Željka
```

DetectP — Syntactic function

Function definition:

```
string DetectP (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns the predicate from the sentence.
```

Example:

```
=DetectS('Željka kuha ručak')

# OUTPUT:
kuha ručak
```

DetectO Syntactic function

Function definition:

```
string DetectO (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns the object from the sentence.
```

Example:

```
=DetectO('Željka kuha ručak')

# OUTPUT:
ručak
```

For easier usage all three function are aggregated into one function `DetectSPO()`.

DetectSPO Syntactic function

Function definition:

```
json DetectSPO (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns the subject, predicate and object from the sentence as a
JSON object.
```

Example:

```
=DetectSPO('Željka kuha ručak')

# OUTPUT:
{'subject': 'Željka', 'predicate': 'kuha ručak', 'object': 'ručak'}
```

Detection of metonymies and metaphors is described in Section 7.3 and the function `DetectMetonymy()` implements these principles to extract metonymy from the sentence. In the first pass sentence elements (words) are lemmatized, and checked if the word is tagged as a symbol. If such word is found, in the next step the algorithm looks for core verbs that are tagged to a symbol word. If the verb in the sentence is not in the list of the core verbs it is recognized as a metonymy.

DetectMetonymy Semantic function

Function definition:

```
json DetectMetonymy (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a metonymy if detected.
```

Example:

```
=DetectMetonymy('Kruna je rekla da su se odnosi poboljšali')

# OUTPUT:
Metonimija: Kruna rekla
```

Similarly to the `DetectMetonymy()` function, the function `DetectMetaphor()` uses SOW symbol tag and word's domains to detect metaphors from sentences. In the first step the function extracts SPO roles, and for any SPO role which is tagged as a symbol, checks core domains. If the word is not in core domains, the metaphor is detected.

DetectMetaphor Semantic function

Function definition:

```
json DetectMetaphor (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a metaphor if detected.
```

Example:

```
=DetectMetaphor("Njihov predsjednik je hrabri lav.")

# OUTPUT:
Metafora: predsjednik je lav
```

The function `DetectColoc()` extracts collocations from the sentence. Collocations are by definition - sequences of words or terms that co-occur more often than would be expected by chance. The function has only one parameter (sentence) which is parsed and matched against the MWE dictionary.

DetectColoc Semantic function

Function definition:

```
json DetectColoc (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns collocations if detected.
```

Example:

```
=DetectColoc('Bila je to labava carinska unija')

# OUTPUT:
labava carinska unija
```

The function `DetectFigure()` parses a sentence and tries to find a stylistic figure using O-structures patterns (described in Section 7.4)

DetectFigure Semantic function

Function definition:

```
json DetectFigure (sentence)
```

Parameters:

```
string sentence - source sentence
```

Return values:

```
Returns a stylistic figure if detected.
```

Example:

```
=DetectFigure('Bilo je dugo bablje ljeto')

# OUTPUT:
Antonomasia => bablje ljeto
```

The function `EnrichSentence()` is commonly used in situations where sentence needs to be expanded with WOS/SOW marks in order to be easily matched against O-structure patterns.

EnrichSentence Syntactic function

Function definition:

```
json EnrichSentence (sentence, wos, sow)
```

Parameters:

```
string sentence - source sentence
int wos - list of WOS tags which are relevant for enrichment
int sow - list of SOW tags which are relevant for enrichment
```

Return values:

```
Returns the enriched version of the sentence.
```

Example:

```
=EnrichSentence('Mario voli čitati knjige','1,2,3','1,2,3')

# OUTPUT:
Mario[w:2] voli čitati knjige[w:2]
```

The function `Imperative()` takes as an argument the word and returns its imperative version. This function is a part of Mel'čuks MTT.

Imperative Syntactic function

Function definition:

```
string Imperative (word)
```

Parameters:

```
string word - source word
```

Return values:

```
Imperative of the source word.
```

Example:

```
=Imperative("stajati")

# OUTPUT:
stoj
```

Following two functions `Word2MULTEXT()` and `Wid2MULTEXT()` use the WOS/SOW tags of words for creating the MULTEXT-East tags. Both functions work on the same principle; the only difference is that the function `Word2MULTEXT()` as a parameter expects a string representation of a word, whereas function `Wid2MULTEXT()` expects an integer value of word's ID. For the given word function check WOS/SOW tags which are assigned to words and generate the MULTEXT-East string using the rules from Appendix F.

Word2MULTEXT WOS/SOW function

Function definition:

```
string[] Word2MULTEXT (word)
```

Parameters:

```
string word - target word
```

Return values:

```
Returns a list of MULTEXT-East tags for a given word.
```

Example:

```
=Word2MULTEXT("stol")

# OUTPUT:
['Nmsa-', 'Nmsn-']
```

Wid2MULTEXT WOS/SOW function

Function definition:

```
string Wid2MULTEXT (wid)
```

Parameters:

```
int wid - target word's ID
```

Return values:

```
Returns MULTEXT-East tags for a given word.
```

Example:

```
=Wid2MULTEXT(2301)

# OUTPUT:
Nfpg-
```

E. WOS/SOW marks

Table 23: List of WOS and SOW marks

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
1	Vrsta riječi		1	Opće	
2	└Imenica	1	2	└Živo	1
145	└a	2	3	└Pojam	1
146	└e	2	111	└Tvar	1
147	└i	2	112	└Tvorevina	1
149	└0	2	113	└Relacija	1
3	└Zamjenica	1	114	└Stanje	1
131	└Osobna	3	115	└Proces	1
132	└Povratna	3	116	└Prostor	1
133	└Posvojna	3	117	└Vrijeme	1
134	└Upitna-odnosna	3	118	└Terminološko	1
135	└Pokazna	3	294	└Obilježje	1
136	└Neodređena	3	9	Ime	
137	└Povratno-posvojna	3	119	└Antroponim	9
4	└Pridjev	1	125	└Ime	119
5	└Glagol	1	126	└Prezime	119
150	└Glavni	5	127	└Nadimak	119
151	└Pomoćni	5	120	└Toponim	9
152	└Biti	151	128	└Hidronim	120
153	└Htjeti	151	129	└Oronim	120
154	└Modalni	5	130	└Etnonim	120
6	└Broj_vr	1	131	└Oikonim	120
138	└Glavni	6	121	└Ustanova	9
139	└Redni	6	122	└Tvrtka	9
140	└Zapis	6	123	└Zanimanje	9
141	└arapski	140	124	└Mjerne jedinice	9
142	└rimski	140	16	Osoba	
155	└strojni	140	17	└Tijelo	16
7	└Prilog	1	18	└Doživljaj	16
8	└Prijedlog	1	19	└Spoznaja	16
156	└Uz G	8	20	└Duhovnost	16
157	└Uz D	8	21	Skup	
158	└Uz A	8	22	└Brojivo	21
159	└Uz L	8	23	└Nebrojivo	21
160	└Uz I	8	24	└Dio	21
9	└Veznik	1	25	└Poredano	21
10	└Uzvik	1	234	└Relacija	21
11	└Čestica	1	235	└Sastavni	234

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
12	└Kratica	1	236	└Rastavni	234
255	└Interpunkcija	1	237	└Suprotni	234
256	└Navodnik	255	238	└Isključni	234
257	└’	256	239	└Zaključni	234
258	└„	256	241	└Kontinuirano	21
259	└“	256	242	└Diskretno	21
333	└"	256	26	Djelovanje	
260	└Navezak	255	27	└Gibanje	26
261	└a	260	28	└Označivanje	26
262	└-crtica	260	29	└Trajanje	26
263	└_crtica	260	30	└Ponavljanje	26
264	└Međa	255	31	└Provjera	26
265	└,	264	32	└Prijenos	26
266	└;	264	33	└Zamjena	26
267	└:	264	34	└Stanje	26
268	└|	264	35	Valentnost	
269	└/	264	39	└1	35
270	└\	264	40	└2	35
271	└Zagrada	255	41	└3	35
272	└(	271	42	└4	35
273	└)	271	43	└0_neosobno	35
274	└[	271	44	Mjera	
275	└]	271	45	└Veličina	44
276	└{	271	46	└Količina	44
277	└}	271	47	└Kvaliteta	44
278	└<	271	48	└Oblik	44
279	└>	271	49	Oštrina	
280	└Simbol	255	52	└Objektivno	49
281	└@	280	53	└Boje	52
282	└&	280	54	└Zvuk	52
283	└#	280	132	└Relativno	49
284	└$	280	133	└Apsolutno	49
285	└%	280	134	└Ustrojbeno	49
286	└*	280	137	└Koherentno	49
287	└Ostalo	280	138	└Omeđeno	49
288	└Svršetak	255	162	└Opisno	49
289	└.	288	163	└Gradivno	49
290	└!	288	164	└Posvojno	49
291	└?	288	240	└Subjektivno	49

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
14	Padež		277	└─Negativno	49
15	└─Nominativ	14	278	└─Pozitivno	49
16	└─Genitiv	14	279	└─Minimalno	49
17	└─Dativ	14	280	└─Maksimalno	49
18	└─Akuzativ	14	281	└─Umanjenica	49
19	└─Vokativ	14	282	└─Uvećanica	49
20	└─Lokativ	14	63	Okvir	
21	└─Instrumental	14	64	└─Prostor	63
22	Rod		65	└─Dimenzija	64
23	└─Muški	22	66	└─Lik	64
24	└─Ženski	22	67	└─Površina	64
25	└─Srednji	22	303	└─Slika	64
26	Broj		304	└─Zvuk	64
27	└─Jednina	26	68	└─Vrijeme	63
173	└─Singularia tantum	27	69	└─Interval	68
28	└─Množina	26	70	└─Trenutak	68
30	└─Pluralia tantum	28	71	Način	
161	└─Dvojina	26	78	└─Izrično	71
162	└─Malina	26	79	└─Namjerno	71
31	Komparacija		80	└─Poredbeno	71
32	└─Pozitiv	31	81	└─Pogodbeno	71
33	└─Komparativ	31	82	└─Dopusno	71
34	└─Superlativ	31	83	└─Uzročno	71
35	Određenost		84	└─Posljedično	71
36	└─Određen	35	85	└─Zaključno	71
37	└─Neodređen	35	91	Uzvik	
38	Naglašenost		92	└─Osjećaji	91
39	└─Naglašen	38	93	└─Poticanje	91
40	└─Nenaglašen	38	94	└─Oponašanje	91
51	Vrijeme		95	└─Pokazivanje	91
52	└─Aorist	51	96	CroWN	
53	└─Imperfekt	51	97	└─Definicija	96
54	└─Prezent	51	102	└─Hipernim	96
55	Vid		103	└─Homonim	96
56	└─Svršen	55	104	└─Sinonim	96
57	└─Nesvršen	55	110	└─Antonim	96
185	└─Trajni	57	98	HJP	
186	└─Učestali	57	99	└─Definicija	98
187	└─Dvovidan	55	100	└─Etimologija	98

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
163	Kombinabilne vrste		101	└Frazeologija	98
164	└Imenica	163	105	HOL	
165	└Pridjev	163	106	└Definicija	105
167	└Prilog	163	107	ENC	
168	Osoba		108	└Definicija	107
169	└Neosobno	168	109	└Profesija	107
170	└1.	168	165	Narječja	
171	└2.	168	166	└Čakavsko	165
172	└3.	168	167	└Buzetski dijalekt	166
174	└1.	168	168	└Sjevernočakavski	166
175	└2.	168	169	└Srednjočakavski	166
176	└3.	168	170	└Jugozapadni istarski	166
177	Particip		171	└Južnočakavski	166
178	└Prošli	177	172	└Lastovski dijalekt	166
179	└Sadašnji	177	173	└Kajkavsko	165
180	└Radni	177	174	└Zagorsko-međimurski	173
181	└Trpni	177	175	└Križevačko-podravski	173
182	Način		176	└Turopoljsko-posavski	173
183	└Infinitiv	182	177	└Donjosutlanski dijal	173
184	└Imperativ	182	178	└Prigorski dijalekt	173
292	└Supin	182	179	└Goranski dijalekt	173
293	└Negacija	182	180	└Štokavsko	165
188	Prijelaznost		181	└Slavonski dijalekt	180
189	└Prijelazan	188	182	└Istočnobosanski	180
190	└Neprijelazan	188	183	└Zapadni dijalekt	180
191	└Povratan	188	184	└Novoštokavski jekavs	180
343	└Nepovratan	188	185	Specifičnosti	
294	Akcentuacija		186	└Arhaizam	185
295	└Riječ	294	187	└Nekrotizam	185
296	└Slogovi	294	188	└Novotvorenica	185
297	└Broj slogova	294	189	└Pejorativ	185
299	└Naglašen	294	190	└Vulgarizam	185
300	└Kratkosilazni	299	191	└Žargonizam	185
301	└Dugosilazni	299	192	Posuđenice	
302	└Kratkouzlazni	299	193	└Anglizam	192
303	└Dugouzlazni	299	194	└Germanizam	192
304	└Akut	299	195	└Grecizam	192
306	└Kratki iktus	299	196	└Hungarizam	192
307	└Dugi iktus	299	197	└Latinizam	192

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
334	└─Broj	299	198	└─Rusizam	192
315	└─Nenaglašen	294	199	└─Srbizam	192
316	└─Broj	315	200	└─Talijanizam	192
318	└─Klitike	294	201	└─Turcizam	192
319	└─Proklitike	318	251	└─Preporuka	192
320	└─Enklitike	318	202	CroVallex	
321	└─Paradigma	294	203	└─Gloss example	202
322	└─A	321	204	└─Functor	202
323	└─B	321	205	└─Class	202
324	└─C	321	206	└─Idiom	202
325	└─Broj morfema	294	207	└─Idiom example	202
326	└─Morfemi	294	210	└─Gloss	202
328	└─Raščlamba	294	212	Termin	
330	└─Nenaglašen	294	213	└─Područje	212
332	└─Broj nng. sloga	330	214	└─Polje	212
335	└─Dužina1	294	215	└─Grana	212
336	└─Broj	335	216	Teorije	
339	└─Dužina2	294	217	└─Lieber R.	216
340	└─Broj	339	218	└─material	217
341	└─Kratak	339	219	└─-material	217
342	└─Dug	339	220	└─dynamic	217
			221	└─-dynamic	217
			222	└─IEPS	217
			223	└─-IEPS	217
			224	└─Pustejovsky J	216
			225	└─Constitutive	224
			226	└─Formal	224
			227	└─Telic	224
			228	└─Agentive	224
			229	└─Qualia	224
			230	└─Event	224
			231	└─Proccess	230
			232	└─State	230
			233	└─Arguments	224
			252	└─Melčuk	216
			253	└─Funkcija	252
			254	└─Domena X	252
			255	└─Kodomena Y	252
			256	└─X	252

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
			257	└─Y	252
			258	└─Uvjeti	252
			259	└─MIP(VU)	216
			260	└─Basic meaning	259
			261	└─MRW	259
			262	└─Direct	261
			263	└─Indirect	261
			264	└─Cross-domain	261
			265	└─Exception	259
			266	└─Polyword	265
			267	└─Phrasal	265
			268	└─Compaunds	265
			269	└─MFlag	259
			270	└─Figure	216
			271	└─Amfibolija	270
			272	└─Metalepsa	270
			273	└─Silepsa	270
			274	└─Katahreza	270
			275	└─Sinegdoha	270
			276	└─Metonimija	270
			287	└─Šarić	216
			288	└─Sinonimski skup	287
			243	Stav	
			244	└─Poticajno	243
			245	└─Dvojbeno	243
			246	└─Ravnodušno	243
			248	└─Upitno	243
			249	└─Niječno	243
			250	└─Potvrdno	243
			283	└─Polaritet	243
			284	└─Smjer	243
			285	└─Pozitivno	243
			286	└─Negativno	243
			289	Kolokacija	
			290	└─Čvrsta sveza	289
			291	└─Frazem	289
			292	└─Poslovica	289
			293	└─Frazem u kontekstu	289

WOS			SOW		
ID	Name	ParentID	ID	Name	ParentID
			305	OWL	
			306	└─owl:sameAs	305
			307	Simbol	
			311	BabelNet	
			312	└─ID	311
			313	└─Definicija	311
			314	└─Kategorija	311
			315	Jezgreni	
			308	└─Glagol	315
			309	└─Pridjev	315
			310	└─Imenica	315
			316	└─Domena	315

F. List of tags in different tagging systems

Penn Treebank Project

Table 24: Alphabetical list of Penn Treebank tags [100]

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential *there*
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

MULTEX-East Project

As described in Chapter 3 the MULTEXT-East Project distincts between 12 different word categories (as shown in Table 25), and each category has different specific tags which are relevant for it [104].

Table 25: MULTEXT-East Croatian categories

Category	Code	Attributes
Noun	N	5
Verb	V	6
Adjective	A	7
Pronoun	P	11
Adverb	R	2
Adposition	S	1
Conjunction	C	2
Numeral	M	6
Particle	Q	1
Interjection	I	0
Abbreviation	Y	0
Residual	X	1

The tag is a string of characters in which each character position and value means as noted in following tables.

Table 26: MULTEXT-East Croatian Specification for Nouns

Position	Attribute	Code	Meaning
1	Type	c	common
		p	proper
2	Gender	m	masculine
		f	feminine
		n	neuter
3	Number	s	singular
		p	plural
4	Case	n	nominative
		g	genitive
		d	dative
		a	accusative
		v	vocative
		l	locative
		i	instrumental
5	Animate	n	no
		y	yes

Table 27: MULTEXT-East Croatian Specification for Verbs

Position	Attribute	Code	Meaning
1	Type	m	main
		a	auxiliary
		c	copula
2	VForm	n	infinitive
		p	participle
		r	present
		f	future
		m	imperative
		a	aorist
		e	imperfect
3	Person	1	first
		2	second
		3	third
4	Number	s	singular
		p	plural
5	Gender	m	masculine
		f	feminine
		n	neuter
6	Negative	n	no
		y	yes

Table 28: MULTEXT-East Croatian Specification for Adverbs

Position	Attribute	Code	Meaning
1	Type	g	general
		r	participle
2	Degree	p	positive
		c	comparative
		s	superlative

Table 29: MULTEXT-East Croatian Specification for Adpositions

Position	Attribute	Code	Meaning
1	Case	g	genitive
		d	dative
		a	accusative
		l	locative
		i	instrumental

Table 30: MULTEXT-East Croatian Specification for Adjectives

Position	Attribute	Code	Meaning
1	Type	g	general
		s	possessive
		p	participle
2	Degree	p	positive
		c	comparative
		s	superlative
3	Gender	m	masculine
		f	feminine
		n	neuter
4	Number	s	singular
		p	plural
5	Case	n	nominative
		g	genitive
		d	dative
		a	accusative
		v	vocative
		l	locative
		i	instrumental
6	Definiteness	n	no
		y	yes
7	Animate	n	no
		y	yes

Table 31: MULTEXT-East Croatian Specification for Conjunctions

Position	Attribute	Code	Meaning
1	Type	s	coordinating
		s	subordinating
2	Formation	s	simple
		c	compound

Table 32: MULTEXT-East Croatian Specification for Particles

Position	Attribute	Code	Meaning
1	Type	z	negative
		q	interrogative
		o	modal
		r	affirmative

Table 33: MULTEXT-East Croatian Specification for Pronouns

Position	Attribute	Code	Meaning
1	Type	p	personal
		d	demonstrative
		i	indefinite
		s	possessive
		q	interrogative
		r	relative
		x	reflexive
2	Person	1	first
		2	second
		3	third
3	Gender	m	masculine
		f	feminine
		n	neuter
4	Number	s	singular
		p	plural
5	Case	n	nominative
		g	genitive
		d	dative
		a	accusative
		v	vocative
		l	locative
		i	instrumental
6	Owner_Number	s	singular
		p	plural
7	Owner_Gender	m	masculine
		f	feminine
		n	neuter
8	Clitic	n	no
		y	yes
9	Referent_Type	p	personal
		s	possessive
10	Syntactic_Type	n	nominal
		a	adjectival
11	Animate	n	no
		y	yes

Table 34: MULTEXT-East Croatian Specification for Numerals

Position	Attribute	Code	Meaning
1	Form	d	digit
		r	roman
		l	letter
2	Type	c	cardinal
		o	ordinal
		m	multiple
		s	special
3	Gender	m	masculine
		f	feminine
		n	neuter
4	Number	s	singular
		p	plural
5	Case	n	nominative
		g	genitive
		d	dative
		a	accusative
		v	vocative
		l	locative
		i	instrumental
6	Animate	n	no
		y	yes

Table 35: MULTEXT-East Croatian Specification for Residuals

Position	Attribute	Code	Meaning
1	Type	f	foreign
		t	typo
		p	program

SWETWOL Tags

The structure of SWETWOL tags, as defined on SWECG website [15] is:

```
< >* Part-of-speech Inflection
   |    |             |
   |    Major         Nominal inflection categories:
   |    syntactic     gender, definiteness, number, case.
   |    category      Verbal inflection categories:
   |                  voice, finiteness (tense or mood or nonfinite form).
   |
   Additional features (zero or more)
```

Table 36: SWETWOL Part of Speech tags

Code	Meaning
N	Noun
A	Adjective
V	Verb
PRON	Pronoun
DET	Determiner
ADV	Adverb
PREP	Preposition
CC	Coordinating Conjunction
SC	Subordinating Conjunction
INFMARK	Infinitive Marker
INTERJ	Interjection
ABBR	Abbreviation

Table 37: SWETWOL Verbal inflection tags

Code	Meaning
ACT	Voice: active
PASS	Voice: passive
DEP	Voice: deponential
PRES	Tense: present
PAST	Tense: past
IMP	Mood: imperative
CNJV	Mood: conjunctive
INF	Nonfinite form: infinitive
SUPINE	Nonfinite form: supine

Table 38: SWETWOL Nominal inflection tags

Code	Meaning
UTR	Gender: common, Swe. utrum
NEU	Gender: neutre, Swe. neutrum
UTR/NEU	Gender: common/neutre
UTR-MASC	Gender: common-masculine
DEF	Definiteness: definite
INDEF	Definiteness: indefinite
DEF/INDEF	Definiteness: definite/indefinite
SG	Number: singular
PL	Number: plural
SG/PL	Number: singular/plural
NOM	Case: nominative
ACC	Case: accusative
GEN	Case: genitive
NOM/ACC	Case: nominative/accusative
NOM/GEN	Case: nominative/genitive
?	Undetermined
	("*kennedy" <*> N ? SG NOM, "133" <DIGIT> <NUM> ?)

Table 39: SWETWOL Derivational tags

Code	Meaning
DER-are	Derived noun in -are
DER-arinna	Derived noun in -arinna
DER-else	Derived noun in -else
DER-erska	Derived noun in -erska
DER-het	Derived noun in -het
DER/-nde	Derived noun in -nde
DER/-ning	Derived noun in -ing
DER-bar	Derived adjective in -bar
DER-ig	Derived adjective in -ig
DER-isk	Derived adjective in -isk
DER-lig	Derived adjective in -lig
<V/DER>	Deverbal
<PCP2>	Past Participle
<PCP1>	Present Participle

Table 40: SWETWOL Governmental definiteness tags for determiners

Code	Meaning
<ID>	Indefinite: governs A INDEF and N INDEF
<DF>	Definite: governs A DEF and N DEF
<MD>	Mixed definite: governs A DEF and N INDEF
<DF/ID>	Definitie/indefinite: <DF> or <ID>
<DF/MD>	Definite/mixed definite: <DF> or <MD>
<ID/MD>	Indefinite/mixed definite: <ID> or <MD>

Table 41: SWETWOL Other tags specifically for pronouns and determiners

Code	Meaning
<DEM>	Demonstrative
<DEM/ART>	Demonstrative/Articular
<NUM>	Numeral
<NUM/ART>	Numeral/Articular
<ORD>	Ordinal
<PERS-SG1>	Personal, 1st person singular
<PERS-SG2>	Personal, 2nd person singular
<PERS-SG3>	Personal, 3rd person singular
<PERS-PL1>	Personal, 1st person plural
<PERS-PL2>	Personal, 2nd person plural
<PRE>	Potential predeterminer
<POSS-SG1>	Possessive, 1st person singular
<POSS-SG2>	Possessive, 2nd person singular
<POSS-PL1>	Possessive, 1st person plural
<POSS-PL2>	Possessive, 2nd person plural
<POSS>	Possessive
<REFL>	Reflexive
<WH>	WH-word, i.e. interrogative or relative pronoun, determiner, or adverb

Table 42: SWETWOL Other additional tags

Code	Meaning
<ARCH>	Archaic
<AUX>	Auxiliary verb
<CLB>	Clause boundary
<CLLQ>	Colloquial
<CMP>	Comparative
<COLLOCATION>	Idiomatic construction or collocation
<COP>	Copular verb
<DIGIT>	Digit
<N>	Compound: noun as first part.
	Similar tags for other parts of speech, e.g. <A>.
<NEG>	Negative
<PARAGRAPH>	Paragraph boundary
<PROP>	Proper noun.
	In the present implemenation, <PROP> is most of the time replaced by some more specific tag, e.g. <FIRST_NAME>, <FAMILY>, <COUNTRY>, <CITY>.
<PUNCT>	Punctuation mark (including . ? ,)
<ROMAN>	Roman number (XII)
<SUP>	Superlative
<TrunCo>	Truncated compound

Table 43: SWETWOL Miscellaneous tags

Code	Meaning
<COERCE!>	Marks a reading to be eliminated by local disambiguation
<RETAIN!>	Marks a reading not to be eliminated by local disambiguation
<RARE!>	Rare reading
<NON-SWETWOL>	Marks a reading assigned by morphological heuristics
<?>	Stands for inflectional features in <NON-SWETWOL> readings
*	Upper case (*den = Den)
<*>	Upper case represented in the lexicon
<**>	Upper case not represented in the lexicon

UD POS Tags

UD POS Tags have core part-of-speech categories and universal features [165]. The core categories are:

Table 44: UD POS Tags for open class words

Code	Meaning
ADJ	Adjective
ADV	Adverb
INTJ	Interjection
NOUN	Noun
PROPN	Proper noun
VERB	Verb

Table 45: UD POS Tags for closed class words

Code	Meaning
ADP	Adposition
AUX	Auxiliary
CCONJ	Coordinating conjunction
DET	Determiner
NUM	Numeral
PART	Particle
PRON	Pronoun
SCONJ	Subordinating conjunction

Table 46: UD POS Tags for other words

Code	Meaning
PUNCT	Punctuation
SYM	Symbol
X	Other

Universal features are: `Abbr` (abbreviation), `AbsErgDatNumber` (number agreement with absolutive/ergative/dative argument), `AbsErgDatPerson` (person agreement with absolutive/ergative/dative argument), `AbsErgDatPolite` (politeness agreement with absolutive/ergative/dative argument), `AdpType` (adposition type), `AdvType` (adverb type), `Animacy`, `Aspect`, `Case`, `Clusivity`, `ConjType` (conjunction type), `Definite` (definiteness or state), `Degree` (degree of comparison), `Echo`, `ErgDatGender` (gender agreement with ergative/dative argument), `Evident` (evidentiality), `Foreign`, `Gender`, `Hyph` (hyphenated compound or part of it), `Mood`, `NameType` (type of named entity), `NounType`

(noun type), `NumForm` (numeral form), `NumType` (numeral type), `NumValue` (numeric value), `Number`, `PartType` (particle type), `Person`, `Polarity`, `Polite` (politeness), `Poss` (possessive), `PossGender` (possessor's gender), `PossNumber` (possessor's number), `PossPerson` (possessor's person), `PossedNumber` (possessed object's number), `Prefix` (Word functions as a prefix in a compund construction), `PrepCase` (case form sensitive to prepositions), `PronType` (pronominal type), `PunctSide`, `PunctType` (punctuation type), `Reflex` (reflexive), `Style` (style or sublanguage to which this word form belongs), `Subcat` (subcategorization), `Tense`, `VerbForm` (form of verb or deverbative), `VerbType` (verb type) and `Voice`.

CV

Marko Orešković was born on 23rd of January 1984 in Požega, Croatia where he finished elementary school and Mathematical Gymnasium. He graduated in 2009 at the University of Zagreb, Faculty of Organization and Informatics with the thesis cro. "*OpenCV biblioteka kao osnova za implementaciju računalnog vida*" (eng. "Implementation of computer vision using OpenCV library") and mentor Professor Neven Vrček, PhD. During the study he received many honours and awards for professional work and innovations, of which the Dean's award and the Rector's Award 2006 is particularly emphasized, placing in the finals of the *Microsoft Imagine Cup 2006* final competition in India and ranking among the six most innovative teams in the world. In the late 2012 he enrolled in the postgraduate doctoral study of *Information Science*, at the same faculty. He currently works as a Head of Information Technology department at the National and University Library in Zagreb. His fields of interest are software engineering, data analysis and database modeling. He is married and has two children.

List of scientific papers

[1] Marko Orešković, Sandra Lovrenčić and Mario Essert. 'Croatian Network Lexicon within the Syntactic and Semantic Framework and LLOD Cloud'. In: *International Journal of Lexicography* (2018). ISSN: 1477-4577. DOI: 10.1093/ijl/ecy024

[2] Marko Orešković, Juraj Benić and Mario Essert. 'A Step toward Machine Recognition of Complex Sentences'. In: *TEM Journal* 7.4 (Nov. 2018), pp. 823–828. ISSN: 2217-8333. DOI: 10.18421/TEM74-20

[3] Marko Orešković, Ivana Kurtović Budja and Mario Essert. 'Encyclopedic knowledge as a semantic resource'. In: *The Future of Information Sciences, INFuture2017 : Integrating ICT in Society* (8th–10th Nov. 2017). Ed. by Iana Atanassova et al. Department of Information Sciences, Faculty of Humanities and Social Sciences, University of Zagreb, Croatia, 2017, pp. 151–160

[4] Marko Orešković, Marta Brajnović and Mario Essert. 'A step towards machine recognition of tropes'. In: *Third International Symposium on Figurative Thought and Language* (26th–28th Apr. 2017). Faculty of Humanities and Social Sciences University of Osijek, Croatia. 2017, p. 71

[5] Marko Orešković, Juraj Benić and Mario Essert. 'The Network Integrator of Croatian Lexicographical Resources'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 267–272. ISBN: 978-9941-13-542-2

[6] Marko Orešković, Mirko Čubrilo and Mario Essert. 'The Development of a Network Thesaurus with Morpho-semantic Word Markups'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 273–279. ISBN: 978-9941-13-542-2

[7] Marko Orešković, Jakov Topić and Mario Essert. 'Croatian Linguistic System Modules Overview'. In: *Proceedings of the 17th EURALEX International Congress* (6th–10th Sept. 2016). Ed. by Tinatin Margalitadze and George Meladze. Tbilisi, Georgia: Ivane Javakhishvili Tbilisi University Press, 2016, pp. 280–283. ISBN: 978-9941-13-542-2

[8] Igor Baj, Vesna Golubović and Marko Orešković. 'Istraživanje korisnika Nacionalne i sveučilišne knjižnice u Zagrebu o novom obliku usluge: tematsko pretraživanje'. In: *Vjesnik bibliotekara Hrvatske* 56.4 (2014), pp. 107–128

[9] Dijana Machala and Marko Orešković. 'Measuring Information and Digital Literacy Activities through Learning Record Store Repository of the National Training Centre for Continuing Education for Librarians in Croatia'. In: *Information Literacy. Lifelong Learning and Digital Citizenship in the 21st Century. ECIL 2014*. Ed. by Serap Kurbanoğlu et al. Cham: Springer International Publishing, 2014, pp. 580–588. ISBN: 978-3-319-14136-7. DOI: 10.1007/978-3-319-14136-7_61

[10] Neven Vrček, Miroslav Novak and Marko Orešković. 'Konvergencija mobilnih tehnologija na primjeru mobilne elektrokardiografije'. In: *Metode i alati za razvoj poslovnih i informatičkih sustava: CASE 19* (18th–20th June 2007). Ed. by Mislav Polonijo. CASE d.o.o., Rijeka, 2007, pp. 233–236

[11] Marko Velić et al. 'Smart ECG, solution for mobile heart work analysis and medical intervention in case of heart work problems'. In: *Abstracts of The 56th Annual Scientific Session of the American College of Cardiology: Special Topics* (25th Mar. 2007). Elsevier, 2007. DOI: 10.1016/j.jacc.2007.01.041

List of professional papers

[1] Mario Essert, Ivana Kurtović Budja and Marko Orešković. 'Pozivnica Oxford dictionaryja hrvatskomu jeziku'. In: *Izazovi nastave hrvatskoga jezika* (10th–17th Nov. 2017). Ed. by Srećko Listeš and Linda Grubišić Belina. 8. Simpozij učitelja i nastavnika Hrvatskoga jezika. Masarykova 20, Zagreb, Croatia: Školska knjiga d.d., Zagreb, Croatia, 2017, pp. 10–25. ISBN: 978-953-0-51739-4

[2] Mario Essert and Marko Orešković. 'Označiteljska pomagala i povezani podatci u predmetnoj obradi'. In: *Znanstveno-stručni skup "Predmetna obrada : pogled unaprijed", Knjižnice grada Zagreba, Gradska knjižnica, 20. svibnja 2016.* Ed. by Branka Purgarić-Kužić and Sonja Špiranec. Hrvatsko knjižničarsko društvo, 2016, p. 217. ISBN: 978-953-8176-02-9

[3] Dijana Machala and Marko Orešković. 'Skupni katalog Nacionalne i sveučilišne knjižnice u Zagrebu te knjižnica iz sustava znanosti i visokog obrazovanja Republike Hrvatske'. In: *Stručni skup knjižnični podaci: interoperabilnost, povezivanje i razmjena* (28th–29th Nov. 2017). Nacionalna i sveučilišna knjižnica u Zagrebu, 2017, pp. 20–22. ISBN: 978-953-500-166-9

[4] Marko Orešković, Tamara Krajna and Jelena Bolkovac. 'Aplikacije otvorenog koda za korištenje u knjižnicama'. In: *Vjesnik bibliotekara Hrvatske* 58.1-2 (2015), pp. 81–92

[5] Antica Bračanov, Vesna Golubović and Marko Orešković. 'Mrežna aplikacija - Novo u čitaonicama: bilten prinova otvorenog pristupa građi'. In: *Vjesnik bibliotekara Hrvatske* 56.1-2 (2013)

Bazična dokumentacijska kartica na hrvatskom jeziku

DD (FOI - Sveučilište u Zagrebu) UDK 004.4'412/'414(043.3)

Doktorska disertacija

Mrežni sintaksno-semantički okvir za izvlačenje leksičkih relacija deterministračkim modelom prirodnoga jezika

M. Orešković
Sveučilište u Zagrebu
Fakultet organizacije i informatike
Varaždin

Pojavom velikoga broja digitalnih dokumenata u okružju virtualnih mreža (interneta i dr.), postali su zanimljivi, a nedugo zatim i nužni, načini identifikacije i strojnoga izvlačenja semantičkih relacija iz (digitalnih) dokumenata (tekstova). U ovome radu predlaže se novi, deterministički jezični model s pripadnim artefaktom (Syntactic and Semantic Framework - SSF), koji će služiti kao mrežni okvir za izvlačenje morfosintaktičkih i semantičkih relacija iz digitalnog teksta, ali i pružati mnoge druge jezikoslovne funkcije. Model pokriva sva temeljna područja jezikoslovlja: morfologiju (tvorbu, sastav i paradigme riječi) s leksikografijom (spremanjem riječi i njihovih značenja u mrežne leksikone), sintaksu (tj. skladnju riječi u cjeline: sintagme, rečenice i pragmatiku) i semantiku (određivanje značenja sintagmi). Da bi se to ostvarilo, bilo je nužno označiti riječ složenijom strukturom, umjesto do sada korištenih vektoriziranih gramatičkih obilježja predložene su nove T-strukture s hijerarhijskim, gramatičkim (Word of Speech - WOS) i semantičkim (Semantic of Word - SOW) tagovima. Da bi se relacije mogle pronalaziti bilo je potrebno osmisliti sintaktički (pod)model jezika, na kojem će se u konačnici graditi i semantička analiza. To je postignuto uvođenjem nove, tzv. O-strukture, koja predstavlja uniju WOS/SOW obilježja iz T-struktura pojedinih riječi i omogućuje stvaranje sintagmatskih uzoraka. Takvi uzorci predstavljaju snažan mehanizam za izvlačenje konceptualnih struktura (npr. metonimija, simila ili metafora), razbijanje zavisnih rečenica ili prepoznavanje rečeničnih dijelova (subjekta, predikata i objekta). S obzirom da su svi programski moduli mrežnog okvira razvijeni kao opći i generativni entiteti, ne postoji nikakav problem korištenje SSF-a za bilo koji od indoeuropskih jezika, premda su provjera njegovog rada i mrežni leksikoni izvedeni za sada samo za hrvatski jezik. Mrežni okvir ima tri vrste leksikona (morphovi/slogovi, riječi i višeriječnice), a glavni leksikon riječi već je uključen u globalni lingvistički oblak povezanih podataka, što znači da je interoperabilnost s drugim jezicima već postignuta. S ovako osmišljenim i realiziranim načinom, SSF model i njegov realizirani artefakt, predstavljaju potpuni model prirodnoga jezika s kojim se mogu izvlačiti leksičke relacije iz pojedinačne rečenice, odlomka, ali i velikog korpusa (eng. *big data*) podataka.

Voditelji rada: **Prof. dr. sc. Mirko Čubrilo**
Prof. dr. sc. Mario Essert

Povjerenstvo za obranu: **Prof. dr. sc. Jasminka Dobša**
Prof. dr. sc. Sanja Seljan
Prof. dr. sc. Markus Schatten

Obrana: 15.03.2019. **Promocija:**

Rad je pohranjen u knjižnici Fakulteta organizacije i informatike u Varaždinu.

(237 stranica, 80 slika, 46 tablica, 6 priloga, 179 bibliografska podatka, original na engleskom jeziku)

M. Orešković

DD-2

UDK 004.4’412/’414(043.3)

1. Mrežni sintaksno-semantički okvir za izvlačenje leksičkih relacija deterministričkim modelom prirodnoga jezika

I. Orešković, M.

II. Fakultet organizacije i informatike, Varaždin, Hrvatska

Sintaksna analiza
Semantička analiza
Izvlačenje leksičkih relacija
Novi tip leksikona
Hierarhijska struktura označivanja
Otvoreni povezani podaci

Bazična dokumentacijska kartica na engleskom jeziku

DD (FOI - University of Zagreb) UDC 004.4'412/'414(043.3)

Doctoral Thesis

An Online Syntactic and Semantic Framework for Lexical Relations Extraction Using Natural Language Deterministic Model

M. Orešković
University of Zagreb
Faculty of Organization and Informatics
Varaždin, Croatia

Given the extraordinary growth in online documents, methods for automated extraction of semantic relations became popular, and shortly after, became necessary. This thesis proposes a new deterministic language model, with the associated artifact, which acts as an online Syntactic and Semantic Framework (SSF) for the extraction of morphosyntactic and semantic relations. The model covers all fundamental linguistic fields: Morphology (formation, composition, and word paradigms), Lexicography (storing words and their features in network lexicons), Syntax (the composition of words in meaningful parts: phrases, sentences, and pragmatics), and Semantics (determining the meaning of phrases). To achieve this, a new tagging system with more complex structures was developed. Instead of the commonly used vectored systems, this new tagging system uses tree-like T-structures with hierarchical, grammatical Word of Speech (WOS), and Semantic of Word (SOW) tags. For relations extraction, it was necessary to develop a syntactic (sub)model of language, which ultimately is the foundation for performing semantic analysis. This was achieved by introducing a new 'O-structure', which represents the union of WOS/SOW features from T-structures of words and enables the creation of syntagmatic patterns. Such patterns are a powerful mechanism for the extraction of conceptual structures (e.g., metonymies, similes, or metaphors), breaking sentences into main and subordinate clauses, or detection of a sentence's main construction parts (subject, predicate, and object). Since all program modules are developed as general and generative entities, SSF can be used for any of the Indo-European languages, although validation and network lexicons have been developed for the Croatian language only. The SSF has three types of lexicons (morphs/syllables, words, and multi-word expressions), and the main words lexicon is included in the Global Linguistic Linked Open Data (LLOD) Cloud, allowing interoperability with all other world languages. The SSF model and its artifact represent a complete natural language model which can be used to extract the lexical relations from single sentences, paragraphs, and also from large collections of documents.

Supervisors: **Full Prof. Mirko Čubrilo, PhD**
Full Prof. Mario Essert, PhD

Examiners: **Assoc. Prof. Jasminka Dobša, PhD**
Full Prof. Sanja Seljan, PhD
Assoc. Prof. Markus Schatten, PhD

Oral examination: 2019-03-15 **Promotion:**

The thesis deposited at the Library of the Faculty of Organization and Informatics, Varaždin, Croatia.

(237 pages, 80 figures, 46 tables, 6 appendices, 179 references, original in English)

M. Orešković

DD-2

UDC 004.4’412/’414(043.3)

1. An Online Syntactic and Semantic Framework for Lexical Relations Extraction Using Natural Language Deterministic Model

Syntactic analysis
Semantic analysis
Lexical relations extraction
New lexicon types
Hierarchical tagset structure
Linked open data

I. Orešković, M.

II. Faculty of Organization and informatics, Varaždin, Croatia