Building a corpus of the Croatian parliamentary debates using UDPipe open source NLP tools and Neo4j graph database for creation of social ontology model, text classification and extraction of semantic information

Benedikt Perak,* Filip Rodik,†

* Department of Cultural Studies, Faculty of Humanities and Social Sciences, University of Rijeka
Sveučilišna avenija 4, 51000 Rijeka, Croatia
bperak@uniri.hr

† Tune informacijske tehnologije d.o.o.,
Levanjska 5, 10040 Zagreb, Croatia
filip.rodik@gmail.com

Abstract

This paper describes a process of creating morphosyntactically tagged corpus of the Croatian parliamentary debates using NLP tool UDapi for tokenization, morpho-syntactic parsing and processing Universal Dependencies data to process over 300 thousand transcribed parliamentary speech utterances produced over the period from 2003-2017 and store the data in a Neo4j graph database.

Introduction

This paper[1] describes a pipeline for creating morphosyntactically tagged corpus of the Croatian parliamentary debates using open source NLP tool UDapi (https://github.com/udapi) for tokenization, morpho-syntactic parsing and processing Universal Dependencies data. The pipeline was used to process over 300 thousand transcribed parliamentary speech utterances produced over the period from 2003-2017 and store the data in a Neo4j graph database. The aim of using the graph database is to create a complex representation of the social ontology of the political behaviour involving various social entities, communication processes as well as to apply basic statistic summarization, text classification, community and centrality graph analytics for the research of social, linguistic and conceptual networks.

This computational linguistic and data science research is valuable for the humanities because these parliamentary texts represent one of the biggest available transcribed corpus of public speech in Croatian, while the graph analytics and social model of communication can be valuable for the research in different social sciences because The Croatian Parliament (Croatian: Hrvatski sabor) is one of the most important representative and legislative body of the citizens of the Republic of Croatia. The Parliament is composed of 151 members elected to a four-year term that convene regularly twice a year, the first session runs between 15 January and 15 July, while the second session runs from 15 September to 15 December. The Croatian Parliament can also hold extraordinary sessions (http://www.sabor.hr/Default.aspx?sec=713).

The parliament debates are transcribed and published on the http://www.sabor.hr/ web site. The site comprises of current debates in the 9th term of the Parliament, along with the material from the previous 5th, 6th, 7th and 8th terms of the Parliament, covering sessions from the year 2003-2017. However, this type of repository is not suitable for extensive analysis of the communicative or linguistic features of the delivered speeches. Therefore, we developed a pipeline for tokenization, lemmatization, syntactic parsing of dependencies and meta data integration for creation of complex queries and exploration of linguistic features related to speakers, topics, and sessions.

Goal of the paper

The goal of the paper is to present tools, methods and resources used for the a) data harvesting and extraction of the Croatian Parliament speeches, b) tokenization, lemmatization and syntactic parsing of the files, and c) data storing, modelling and integration. The structure of the paper follows these steps.

Data gathering

The texts of the Croatian parliamentary debates corpus are gathered using a RSelenium scraper (https://github.com/ropensci/RSelenium) on the Parliament web-repository (http://edoc.sabor.hr/). The data gathering process is published as a github

[1] This research is part of the EmoCNet project http://emocnet.uniri.hr, supported by the University of Rijeka's initial grant for the researchers 2017-18.

project (https://github.com/rodik/Sabor). The debates of the 5th to 9th Parliamentary Assembly are downloaded as datasets in a CSV format (Table 1-2). The structure of the data representation on the web-repository yielded two datasets – a session dataset and a transcripts dataset, each with unique metadata features.

The sessions dataset (table 1.) collected the information about the Parliamentary sessions with features: 1) unique ID, 2) number of the parliamentary assembly, 3) number of the parliamentary session, 4) identifying number, 5) title of the session, 6) url of the session, 7) logical value on the existence of the recording (illustration 2).

The transcripts dataset (table 2.) harvested the transcripts data with the following features: 1) person, 2) transcript, 3) number of the utterance, 4) unique ID of the session, 5) date, 6) announcement, 7) parliamentary club.

```
> Rasprave
# A tibble: 1,720 x 7
      ID Saziv Sjednica RedniBroj Naziv                    URL               ImaSnimku
   <int> <chr>    <int>     <int> <chr>                    <chr>                 <dbl>
 1 15668 VII         20        50 Prijedlog odluke o rasp~ http://edoc.sabo~         1
 2 15667 VII         20        49 Konačni prijedlog zakon~ http://edoc.sabo~         1
 3 15666 VII         20        48 Konačni prijedlog zakon~ http://edoc.sabo~         1
 4 15665 VII         20        47 Prijedlog odluke o razr~ http://edoc.sabo~         1
 5 15664 VII         20        46 Prijedlog odluke o dopu~ http://edoc.sabo~         1
 6 15663 VII         20        45 Konačni prijedlog zakon~ http://edoc.sabo~         1
 7 15662 VII         20        44 Prijedlog odluke o osni~ http://edoc.sabo~         1
 8 15661 VII         20        43 Konačni prijedlog zakon~ http://edoc.sabo~         1
 9 15660 VII         20        42 Prijedlog odluke o upuć~ http://edoc.sabo~         1
10 15659 VII         20        41 Izvješće o sudjelovanju~ http://edoc.sabo~         1
# ... with 1,710 more rows
```

Table 1: Example of sessions dataset CSV file.

```
> Transkripti
# A tibble: 12,368 x 7
   `Osoba ___` Transkript                  RB Rasprava_ID Datum      Najava Klub
   <chr>       <chr>                    <int>      <int> <date>     <lgl>  <chr>
 1 67          Prijedlog odluke o izmjen~     1    2012625 2016-06-20 TRUE   NA
 2 "Reiner, Želj~ Sada bih vas zamolio da r~     2    2012625 2016-06-20 FALSE  HDZ
 3 "Ivić, Tomisl~ Hvala lijepo gospodine pr~     3    2012625 2016-06-20 FALSE  HDZ
 4 "Reiner, Želj~ Hvala lijepa. Žele li izv~     4    2012625 2016-06-20 FALSE  HDZ
 5 "Strenja-Lini~ Poštovani predsjedniče, p~     5    2012625 2016-06-20 FALSE  MOST
 6 "Reiner, Želj~ Hvala vam lijepa. S time ~     6    2012625 2016-06-20 FALSE  HDZ
 7 66          Prijedlog odluke o raspuš~     1    2012624 2016-06-20 TRUE   NA
 8 "Reiner, Želj~ Sada bismo naravno prešli~     2    2012624 2016-06-20 FALSE  HDZ
 9 "Zgrebec, Dra~ Predsjedniče, točka koje ~     3    2012624 2016-06-20 FALSE  SDP
10 "Reiner, Želj~ Dobro, čini mi se da post~     4    2012624 2016-06-20 FALSE  HDZ
# ... with 12,358 more rows
```

Table 2: Example of transcript dataset CSV file.

Initial data integration and modelling

The aim of the project is to integrate existing data with an ontology that can intuitively represent the entities and their relations in the process of the Parliamentary debates and possible future data enrichments with some other informational structures and corpora. Two csv datasets were integrated with a Python script using Py2Neo, a client library and toolkit for working with Neo4j from within Python applications (https://py2neo.org/v4/). Neo4j is is one of the most used open-source, fully transactional database, a persistent Java engine where it is possible to store structures in the form of graphs instead of tables (Webber, 2012). It has its own programmatic Cypher language, created by the Neo4j company for developing unique approach to graph query methods. Through the combination of Python code and Cypher queries this language that we can store and get the data from the graph database (Panzarino, 2014). The ontology of the data is represented in the illustration 1. It has 6 structurally different nodes stored with different properties and labels. These structures are connected using the partonomic type of description: 1) Parliament Assembly HAS Session with unique ID, 2) Session HAS Number of the parliamentary session, 3) Number of the parliamentary session HAS Utterance, 4) Person IS_MEMBER_OF Parliamentary club, and a process type description: 5) Person DELIVERED Utterance.

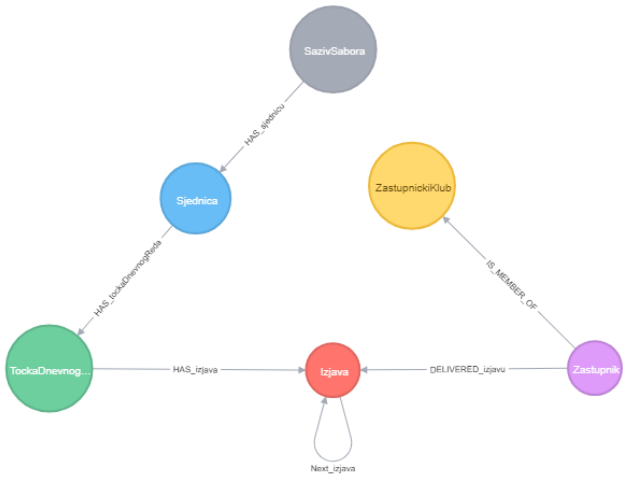

Figure 1. Ontological model of the sessions and trancrips datasets: Parliamentary Assembly – HAS –> Session – HAS –> Discussion point –> HAS –> Utterance <– DELIVERED – Representative –> IS_MEMEBER_OF –> Parliamentary club

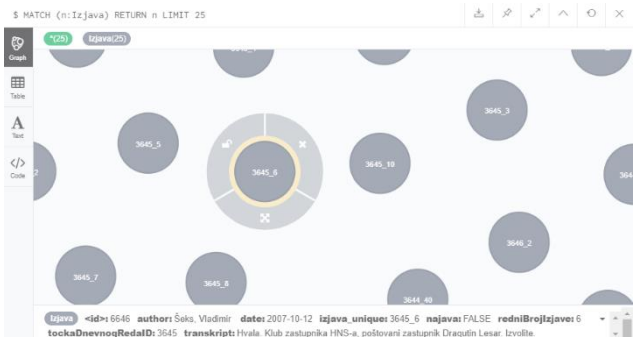

Figure 2. Screenshot of the Neo4j graph data base application browser presenting random 25 nodes labelled *Izjava* 'Utterances' with respective properties.

Corpus creation

The data from the transcripts have been extracted and stored as a batch of separate files with unique identifier of the session and number of the utterance, for example: 2012726_35.txt. These files have been sent to local installation of the UDPipe

(http://ufal.mff.cuni.cz/udpipe), and specifically R package for Tokenization, Parts of Speech Tagging, Lemmatization and Dependency Parsing Based on the UDPipe Natural Language Processing Toolkit (https://bnosac.github.io/udpipe). The model used for parsing was croatian-ud-2.0-170801.udpipe from Universal Dependencies 2.0 Models for UDPipe repository at (https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2364). The parsed files were stored with _conllu.txt endings, for example: 2012726_35_conllu.txt. The output of each transcribed utterance uses a revised version of the CoNLL-X format called CoNLL-U. Annotations are encoded in plain UTF-8 encoded text files with three types of lines: word lines containing the annotation of a word/token in 10 fields separated by single tab characters. Blank lines marking sentence boundaries. Comment lines starting with hash (#). The 10 fields are respectively: 1) ID: Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes. 2) FORM: Word form or punctuation symbol. 3) LEMMA: Lemma or stem of word form. 4) UPOS: Universal part-of-speech tag. 5) XPOS: Language-specific part-of-speech tag; underscore if not available. 6) FEATS: List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available. 7) HEAD: Head of the current word, which is either a value of ID or zero (0). 8) DEPREL: Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one. 9) DEPS: Enhanced dependency graph in the form of a list of head-deprel pairs. 10) MISC: Any other annotation. (http://universaldependencies.org/format.html). The example of the structure is the following:

newdoc id = doc1
newpar
sent_id = 1
text = Poštovani predsjedniče, poštovani premijeru članovi Vlade uz čestitke za pozitivnom aviju.
1 Poštovani poštovan ADJ _ Case=Nom|Definite=Def|Degree=Pos|Gender=Masc|Number=Plur 2 amod _ _
2 predsjedniče predsjednik NOUN _ Case=Voc|Gender=Masc|Number=Sing 0 root _ SpaceAfter=No
3 , , PUNCT _ _ 4 punct _ _
4 poštovani poštovati ADJ _ Case=Nom|Definite=Def|Degree=Pos|Gender=Masc|Number=Plur|VerbForm=Part 2 acl _ _
5 premijeru premijer NOUN _ Case=Dat|Gender=Masc|Number=Sing 4 iobj _ _
6 članovi član NOUN _ Case=Nom|Gender=Masc|Number=Plur 4 nsubj _ _
7 Vlade Vlada NOUN _ Case=Gen|Gender=Fem|Number=Sing 6 nmod _ _
8 uz uz ADP _ Case=Acc 9 case _ _
9 čestitke čestitka NOUN _ Case=Acc|Gender=Masc|Number=Plur 6 nmod _ _
10 za za ADP _ Case=Acc 12 case _ _
11 pozitivnom pozitivan ADJ _ Case=Ins|Definite=Def|Degree=Pos|Gender=Fem|Number=Sing 12 amod _ _
12 aviju avija NOUN _ Case=Acc|Gender=Fem|Number=Sing 9 nmod _ SpaceAfter=No
13 . . PUNCT _ _ 2 punct _ _

From these files additional two structures: a) Sentences and b) Tokens, were created in the graph database. The Sentences nodes were connected to the utterances nodes using the HAS_Sentence relation, and every sentence in a utterance was connected with the NEXT_sentence relation (illustration 3)

The words of a sentences have been stored as

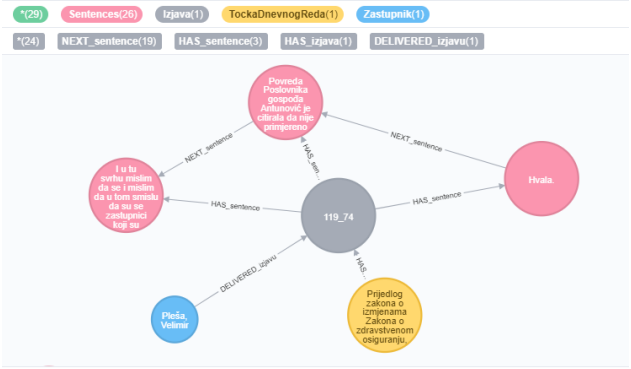

Figure 3 Screenshot of the Neo4j graph data base application browser representing relation of the nodes labelled Izjava 'Utterances' to Sentence.

Token nodes with all ten data fields from the UDPipe parser as properties. Each token relates to a Sentence node with HAS_token relation and stores mutual dependency information with other Tokens in a sentence using HAS_dependency relation. The graph representation is depicted in the figure 4.

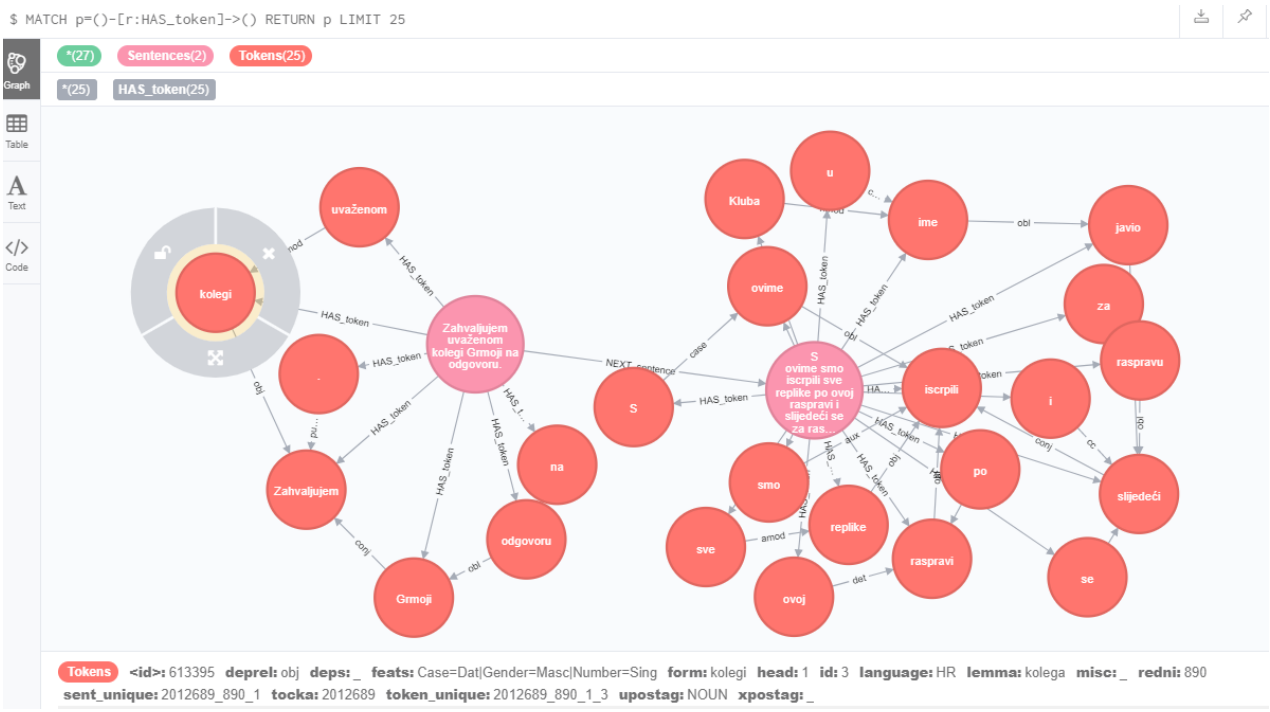

Figure 5 Screenshot of the Neo4j graph data base application browser representing random token nodes of two Sentences with respective properties and dependency relations.

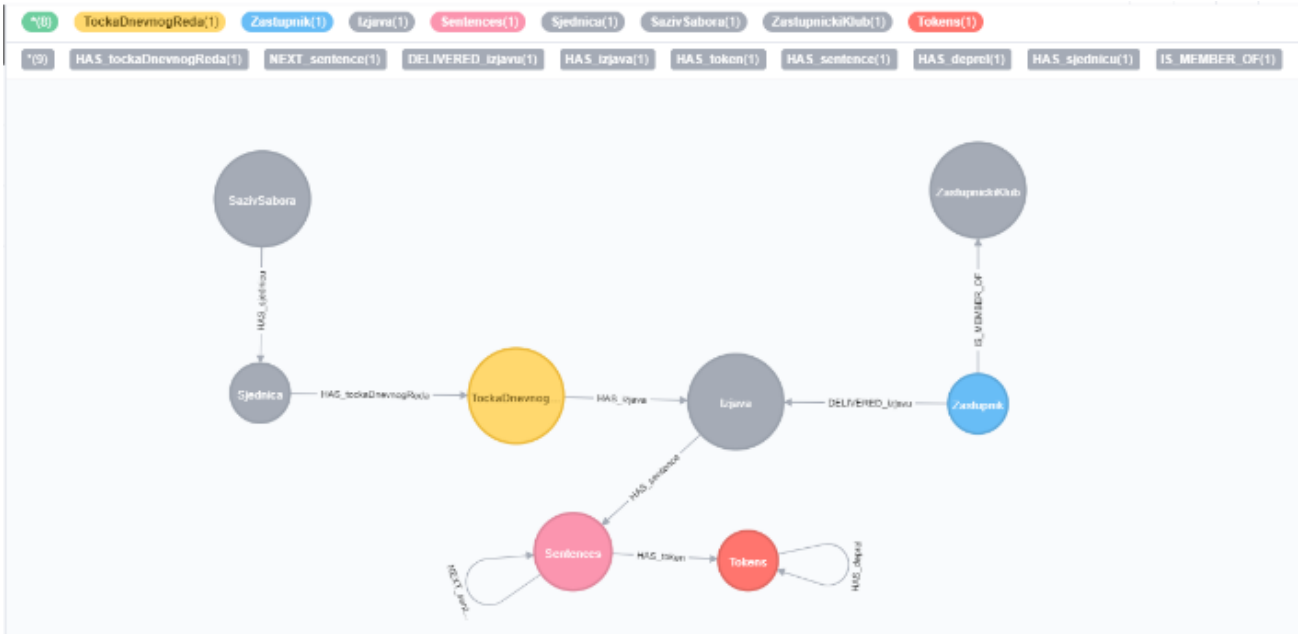

Figure 4. Ontological model stored in the Neo4j database: Parliamentary Assembly – HAS –> Session – HAS –> Discussion point –> HAS –> Utterance (-HAS->Sentence– HAS->Token)<– DELIVERED – Representative –> IS_MEMEBER_OF –> Parliamentary club

The final ontology in the database has the structure as depicted in the figure 5.

Further research of the application

The graph database enables highly connected storage of the disparate datasets thus allowing for an intuitive and yet complex structural development of the data according to the custom created ontology. Besides creating the statistical summarization of the entities, the graph data structure allows creation of complex queries about relations between the interconnected levels within a single text or for multiple texts. In this manner, a local corpus with universally described features can be created allowing for the analysis of the various informational features within the patterns that form the linguistic corpus and its metadata. A special feature of the Neo4j graph database is related to the native graph algorithms library (https://neo4j.com/developer/graph-

algorithms/) that extends the basic summarization procedures to allow the community detection and centrality tagging for any given set of patterns. Furthermore, the graph storage of the parsed text enables the data enrichment for each level of the entities and relations. This means that the level of texts can be enriched with connections the new structures (mentioned Persons, Institution, and Organization) that can be used for further ontological description and contextualization of the text (Perak forthcomming).

References

Onofrio Panzarino2014. *Learning Cypher*. Packt Publishing Ltd.

Benedikt Perak (forthcoming) "Ontological and constructional approach to the discourse analysis of the commemorative speeches in Croatia". In: *Framing the Nation*. Pavlaković, Vjeran, Pauković, Davor (eds.) Routledge.

Jim Webber 2012. A programmatic introduction to neo4j. In: *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity* (pp. 217-218). ACM.

https://neo4j.com/developer/graph-algorithms

https://bnosac.github.io/udpipe

http://ufal.mff.cuni.cz/udpipe

https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2364

https://github.com/ropensci/RSelenium

https://github.com/rodik/Sabor

https://github.com/udapi

http://www.sabor.hr/Default.aspx?sec=713