Information Extraction from the greek parliament records : An experimentation project.

Fotis Paraschiakos : 3873986

1. Introduction

This report is concerned with a first attempt to extract semantic information from the text records of the greek parliament proceedings. Some of the extraction was done by a Java application created, and the rest was done with GATE (General Architecture of Text Enginnering). GATE is a ready made Natural Language Processing framework that has a plug-in based functionality and uses the JAPE language in order to create grammar rules. It has its own GUI but also a Gate Embedded part which allows for functionality within Java code. The project was conceived as writeen purely in Java. However during the development, it became apparent that it was easier for some tasks (but not all of them) to be done in GATE. The result is a Java program using its own extraction algorithms for some functions and implementing GATE for some others.

The result of the process is a series of objects of various types connected to each other, each containing data of mostly String type. Also, the GATE pipeline produces an XML file of annotated text for each conference file, together with individuals from a sequence of ontologies created to represent entities related to the parliament procedure.

The final goal of this project will be the representation of the annotated text in a user-friendly, web based interface, along with the possibility of complex search and querying of the semantic information. Since the dataset is fairly large, this opens up the possibility of a series of interesting data analysis tasks. For a first simple approach to this problem, the data extracted is being put into a relational database.

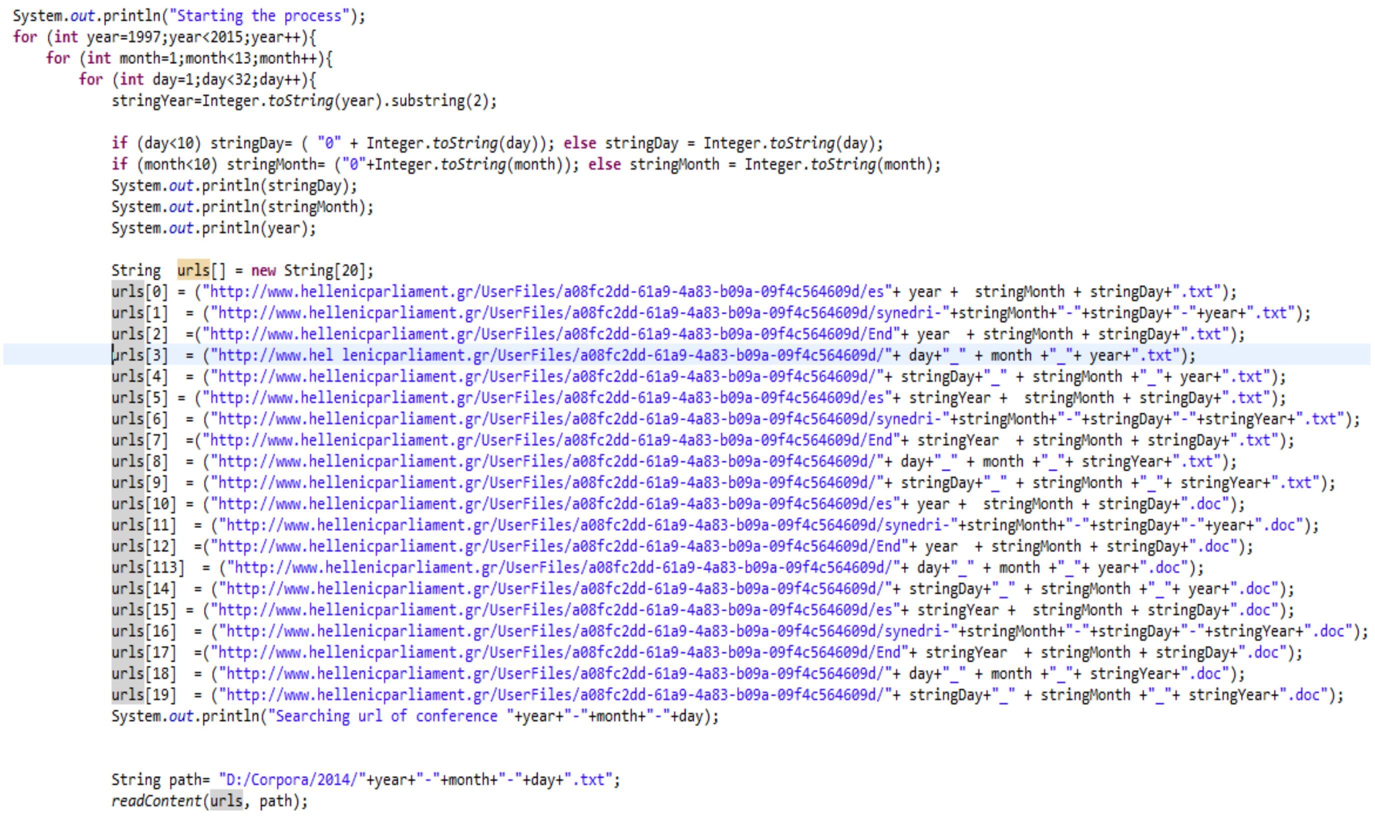
However, the dataset is large and a new document should be added every time a new conference is taken place, so other alternatives are being explored for processing the full dataset, with the most possible choice being to migrate the dataset into a NoSQL Hadoop-based system.

Another goal that will easily be accomplished when the annotation tool is ran in the whole corpus, will be the addition of most of the data extracted to open data repositories such as dbpedia.

2. The Document Corpus

The corpus currently consists of a 4280 text files, which represent greek parliament conferences between the years of 1997 and 2014. Some of them represent full days and some others represent a morning or evening conference. They are not representative of the whole parliamentary activity since some years do not have any conferences digitized and some others are scanned paper copies of the conferences. However, most of them are either in txt, pdf or doc format. They were extracted from the website of the greek parliament (http://www.hellenicparliament.gr/Praktika/Synedriaseis-Olomeleias). Unfortunately, as for almost any site ran by the greek government, it is not well maintained and does not conform to uniform standards, so getting them from the site and storing them properly is a non-trivial task. Looking at the URLs, it was quickly discovered that the date of the conference is contained in the URL in some format. Therefore, a brute force approach would work.

Since they are not in the same format, to extract the date from the filenames and make sure nothing is missed, all the possible formulations of filenames had to be found and checked for every possible date between 1997 and the present date. Also, this way the file names would become standardized or possibly also indexed as they are obtained. The website is still active so the corpus should be constantly updated with the new ones, and with this approach it is easy to automatically (or even manually) check for new updates by implementing a scheduler.



```
System.out.println("Starting the process");
for (int year=1997;year<2015;year++){
    for (int month=1;month<13;month++){
        for (int day=1;day<32;day++){
            stringYear=Integer.toString(year).substring(2);

            if (day<10) stringDay= ( "0" + Integer.toString(day)); else stringDay = Integer.toString(day);
            if (month<10) stringMonth= ("0"+Integer.toString(month)); else stringMonth = Integer.toString(month);
            System.out.println(stringDay);
            System.out.println(stringMonth);
            System.out.println(year);

            String  urls[] = new String[20];
            urls[0] = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/es"+ year +  stringMonth + stringDay+".txt");
            urls[1]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/synedri-"+stringMonth+"-"+stringDay+"-"+year+".txt");
            urls[2]  =("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/End"+ year  + stringMonth + stringDay+".txt");
            urls[3]  = ("http://www.hel lenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ day+"_" + month +"_"+ year+".txt");
            urls[4]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ stringDay+"_" + stringMonth +"_"+ year+".txt");
            urls[5] = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/es"+ stringYear +  stringMonth + stringDay+".txt");
            urls[6]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/synedri-"+stringMonth+"-"+stringDay+"-"+stringYear+".txt");
            urls[7]  =("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/End"+ stringYear  + stringMonth + stringDay+".txt");
            urls[8]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ day+"_" + month +"_"+ stringYear+".txt");
            urls[9]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ stringDay+"_" + stringMonth +"_"+ stringYear+".txt");
            urls[10] = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/es"+ year +  stringMonth + stringDay+".doc");
            urls[11]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/synedri-"+stringMonth+"-"+stringDay+"-"+year+".doc");
            urls[12]  =("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/End"+ year  + stringMonth + stringDay+".doc");
            urls[113]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ day+"_" + month +"_"+ year+".doc");
            urls[14]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ stringDay+"_" + stringMonth +"_"+ year+".doc");
            urls[15] = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/es"+ stringYear +  stringMonth + stringDay+".doc");
            urls[16]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/synedri-"+stringMonth+"-"+stringDay+"-"+stringYear+".doc");
            urls[17]  =("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/End"+ stringYear  + stringMonth + stringDay+".doc");
            urls[18]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ day+"_" + month +"_"+ stringYear+".doc");
            urls[19]  = ("http://www.hellenicparliament.gr/UserFiles/a08fc2dd-61a9-4a83-b09a-09f4c564609d/"+ stringDay+"_" + stringMonth +"_"+ stringYear+".doc");
            System.out.println("Searching url of conference "+year+"-"+month+"-"+day);


            String path= "D:/Corpora/2014/"+year+"-"+month+"-"+day+".txt";
            readContent(urls, path);
```

Figure 1: Part of the file downloading script.

After downloading, they were converted to a YYYY-M(M)-D(D) format. Figure 1 shows part of the process.

3. The parliament documents

Each document is presented in a raw text format. Its layout is standardized, although most of these standards are not consistent throughout the years. Generally, a conference document consists of the following information in this specific order :

- Introductory document information such as the number of the governmental period, the number of the conference (starting from the first conference of the new government) and the date. As shown above, the date is already derived from the file name. The other numbers are written in a greek numeral system and therefore they have to be parsed accordingly so that they can be stored as numeric properties.

- A table of contents consisting of the topics scheduled for discussion that day. The types of topics shown will be discussed below. All the information is repeated afterwards in the discussion itself, so at the present time there is no need to extract any information. It could be used to reduce the running time of some search algorithms used by skipping the examination of large regions of text. However, this only applies to the data extraction algorithms, which will only be ran once for each conference file. During the queries, the data will be already indexed, so taking information from the table of contents is not a serious priority at the moment.

- In the transcripts of the later years, there is also usually a list of the conference chairman, his substitutes and the scheduled speakers for each topic. This is useful and can be used to quickly build an dictionary of speakers for the current conference. However, a chronic problem of greek politics is that speeches are often interrupted and there are constantly people talking out of order or even out of schedule. Thus, there is a constant chance of people talking that will not be on the list and as a consequence it cannot be reliable.

The rest of the document consists of direct quotations of the each person speaking. The procedures are described by the parliament chairman and his substitutes and are also written in a formal manner, very much like the transcripts of a courtroom. In most cased, the chairman announces the topic and the additional information concerning it, gives and takes speaking rights to each speaker, and coordinates any other activity, such as voting or general announcements.

- **Special topics** (Ειδικά θέματα) : These are usually irregular topics which vary from conference to conference. Examples of them could be regulatory topics about the parliament workers themselves, announcement of important visitors in

the viewing stands, or applications for leave of absence of Members of Parliament.
Most special topics are not written formally enough or important enough to be able to extract any useful information.

- **Parliamentary control** (Κοινοβουλευτικος ελεγχος): This chapter includes parliamentary reports and questions to the ministers.

  Parliamentary reports are information that one or many MPs state to the parliament for the record. They are usually either information that the MP wants to make public to the parliament, or small time scale requests from the government. These are presented on a list together with the names of the MPs reporting so it can be easily parsed and indexed.

  Questions to ministers are more formal questions posed from a MP or a group of them to a minister of the government, together with the minister's answer. These are also presented on a list having the names of the MPs and the minister in specific places, so they can be also easily parsed and indexed. There are also the ''current questions'' (Επίκαιρες ερωτήσεις) that are accompanied by a discussion in the parliament. They can also be stored but the answers will have to be the whole discussion.

- **Law making work** (Νομοθετικη εργασια) consists of work relevant to the laws and law proposals. Specifically, reports of work done by law making committees, presentations of law proposals, discussion about them and internal elections for passing a law or not.

4. Creating the application

The java application created consists of the following parts :

a. The script for downloading the documents, discussed above, together with helper methods like ReadContent() or exists() for an URL on the web. Since the file formats vary, a .pdf and .doc to txt converter also had to be made.
b. Classes named :
   - ParlSpeech
   - ParlMember
   - ParlReport
   - ParlQuestion

   These classes represent the data intended to be extracted from the conference. They contain all the text content associated with them.
c. The information is processed in the Conference class. There, the text of the conference is kept, together with the text extracting methods and the data structures containing objects of the type mentioned above. When a

conference class is created (by passing the content as an argument), a number of scripts are being run.
The content is tokenized, being converted into an array of Strings. After that, a trash-removing method is run, deleting any unwanted values from the array. Examples are page counts, headers and other kinds of metadata put into the text by the transcript writer and not corresponding to the real content of the transcript (mostly text not corresponding to speeches).
Following, the data-extracting methods are being run. If they run successfully, all the information specified in (b) is found. So, a Conference object contains all the parliament members speaking linked with their speeches, their reports and their questions to the ministers, also objects of ParlMember. The parliament chairman (also a ParlMember) is also found.
The algorithms use either string manipulation or regular expression algorithms. Fortunately, the names of the parliament members are always in capital letters, so that makes the job somewhat easier. However, the content of the Reports, speeches and questions for each ParlMember still needs to be found, so the context of the ParlMember name appearance needs to be searched (by traversing through the string array), and the output to the database needs to be sanitized.

d. Some dictionaries also have to be created. Dictionaries are text files that contain a list of string elements. They are mainly combined with the GATE Gazetteer in order to create annotations of type Lookup. The dictionaries are :
    - PoliticalParties.lst
    - ParlMembers.lst
    - Provinces.lst

   The PoliticalParties list is manually created, and contains all known and common occurencies of a political party name.

   The list of the ParlMembers is created by scanning each text in the updateDictionary() method in Conference and taking all pairs (or triples in the case of a name with 3 parts) of words in capital letters. Then the temporary dictionary is manually edited to keep only the entries which are actual parliament member names.

   Provinces.lst is updated by the method that extracts the report. A presentation of a report always included the reporter and their legislative province, so when a report is found, together with sending the reporter, the province and the report body in the database, the Provinces and ParlMember dictionary are updated or checked for possible updates.

   Besides their use in the GATE gazeteer, the main reason that these data is kept both in the database and in a dictionary is the coreference problem. It was found that there are a lot of ways that the name of each object (Parliament Member, province, party) is writtenin the transcripts, some times omitting punctuation and other times using a different way of writing that name. This has to be manually managed by modeling in the database the associations between the name of the object and the different ways that this object can be expressed, for example with a list containing alternative spellings for each word that has one. However, the different parliament members from 1997 until 2014 are around 2.000 and there are probably 3-4 ways that each one is being expressed, Therefore, the full task of manually setting up these connections is too labor intensive and beyond the scope of this report.

A dictionary handler class is also created in order to host the methods for easily updating the dictionary. It allows for either looking up a particular string in the dictionary, or also inserting it If it does not exist.

e. A database handler with methods for conflict-free insertion of each class's element to the database. To ensure conflict-free insertion and for the sake of simplicity, it was preferable for the project to have its own jdbc-based handler instead of using an object-relational mapping system.

5. The Database

In order to manage simple queries, a database was created in H2. It has the following tables with the corresponding fields:

CONFERENCE : ID(Primary key Integer) , **ChairmanID** (Integer, references PARLMEMBER(ID)), **Date**(Date), **Content**(Clob).

PARLMEMBER : ID(Primary key Integer) , **Name** (Varchar), **Province**(Varchar), **Politicalparty**(Varchar).

QUESTION : ID(Primary key Integer) , **Question_content** (Clob), **Answer_content** (Clob) **SpeakerID**(Integer, references PARLMEMBER(ID),**MinisterID**(Integer, references PARLMEMBER(id)), **ConferenceID**(Integer, references CONFERENCE(ID))

REPORT: ID(Primary key Integer) , **Content** (Clob), **ParlmemberID**(Integer, references PARLMEMBER(id)), **ConferenceID**(Integer, references CONFERENCE(ID)).

SPEECH: ID(Primary key Integer) , **Content** (Clob), **ParlmemberID**(Integer, references PARLMEMBER(id)), **ConferenceID**(Integer, references CONFERENCE(ID)).

The fields contained here are similar to the ones discussed and describe in more detail in the next section that deals with the ontology. The database will be used manage queries to the dataset, either within the program (for example using a scheduler to repeat the data extraction process when a new conference Is uploaded and pushing the data to the database), as a part of a website or web service that helps the visualization or for executing data analysis tasks.

6. Creating the Ontology

In order to store all this data and the connection between them, an ontology model had to be created. The ontology presented is not the final version and there are a lot of possibilities for further enrichment.

- **Conference :** This represents one text file of parliament transcripts. It has the following properties :
    - **Activities of Conference (Object property, Range : Parliamentary Activity):** All parliamentary activities linked in this conference. These include anything that has a body of text and is linked to a MP or a group of MPs. It will be discussed below.
    - **Conference Chairman (Object property, Range : Member_of_Parliament) :** The conference chairman, or chairmen, usually more than one.
    - **Conference_date (Data property) :** The date that this conference happened, extracted from the conference file name.
    - **Parliament_period (Data property):** The period that the conference belongs to. A period is defined as the time between two parliamentary elections.
    - **Conf_number (Data property) :** The number of the conference for this parliamentary period.

- **Parliament_member :** This represents a Member of parliament that is a part of some parliamentary activity.
    - **Activity_ of_member (Object property, Range : Parliamentary_Activity) :** The parliamentary activities of each parliament member.
    - **Member_of_Party (Object Property, Range : Political_party)** : The political party the parliament member is a member of.
    - **Biography (Data property)** : A block of text describing the Member of Parliament. This can be extracted from various open data sources.
    - **Name (Data property) :** Name and surname of the MP
    - **Province (Data property) :** The Province (administrative region) that this MP is elected in.

    Subclasses:

    - ***Minister***: A subclass of the Parliament Member class, a minister is an MP that participates in a Questions_of_mp_to_minister event.

- **Parliamentary_Activity :** This abstract class represents any type of parliamentary activity that is represented by at least one member of parliament and a body of text.

    - **Parliamentary_activity_body (Data property, String)** : This is the text content of the parliamentary activity. What exactly it contains depends on the particular subclass.
    - **Members (Object property, Range : Parliament_member) :** The MP (or MPs) participating in the activity.
    - **Body (Data property, String) :** The text of the parliamentary activity.

    Subclasses :

    - ***Report*** **:** Indicates a parliamentary report that a MP makes, as described above
    - ***Speech :*** Indicates a speech that a MP makes.

- **Questions_of_MP_to_Minister :** This corresponds to the questions asked by members of parliament to the ministers.
    - **Member_of_parliament_asking (Object property, Range : Parliament_member) :** The member(s) of parliament asking the question.
    - **Minister_questioned (Object property, Range : Minister) :** The minister answering the question.
    - **Question_body (Data property, String)** : The text of the question extracted.
    - **Answer_body (Data property, String)** : The text of the answer extracted.

7. Usage of GATE

During the process of writing the data extraction algorithms in Java, it became apparent that string/array search was not the optimal way to write them. Also, since annotating and extracting chunks of text with metadata from a corpus is not a special job by itself, it was decided that most of the busywork could have been avoided by using GATE. 36

GATE (General Architecture for Text Enginnering) is a framework developed by the Computer Science department of the University of Sheffield. It provides a series of libraries related to all aspects of text engineering that can be embedded in Java (Gate Embedded). Also, it contains a GUI (GATE Developer) for an easier way to test GATE projects without losing in functionality.

The basic unit of data in GATE is the Annotation. An annotation is created by a processing resource. It usually has a type, a start and end index, showing the place that the relevant text exists in the text, and some metadata fields called Features.

GATE enables the ability to create a pipeline made of the different modules available. They are called processing resources and they are being ran on the text (language resources). They are the input of the pipeline and they are typically is a text file (GATE supports multiple file formats like .txt , .doc and .pdf) or a corpus made from a set of them. The output is the annotated corpus

For English-language semantic analysis, GATE provides its own ready made pipeline with processing resources already set up (ANNIE : A Nearly-New information Extraction System). However, it is customized for the English language so a Unicode tokenizer had to be used. GATE also supports JAPE (Java Annotation Patterns Engine), a finite state transducer that enables pattern matching through a regular expression-like syntax over annotated text.

Lots of different modules were tested. For simple annotation of the text and extraction of the data needed the pipeline (in :

1. Document Reset PR: A utility that resets the annotations in the active GATE text.
2. The GATE Unicode tokenizer : A tokenizer coverts a chunk of text into a set of smaller chunks called tokens. In the case of this particular tokenizer, a Token annotation is created, with the feature 'kind' having the value of 'number', 'word' or 'punctuation'.
3. Gazeteer : The Gazeteer produces Lookup annotation from dictionaries. Its input is a file that contains the filenames of all dictionaries, and the output is the Lookup annotations. Dependent of which dictionary it comes from, the Lookup annotation can be configured to have different MajorType and MinorType features.
   In our case, with the 3 dictionaries available, Lookup annotations of MajorType of value parlmember, politicalparty or province are produced.
4. Jape+ Transducer : The Jape+ transducer is a finite state JAPE transducer that takes as input a number of regular expression-type scripts written in JAPE and produces a set of annotations based on them. The scripts work on the existing annotations produced by the tokenizer and the gazeteer.

The result can be seen in the GATE viewer (Figure 3). Through the GATE Developer GUI, all the annotations can be seen in the text with different colour coding for each one.