Starting in Natural Language Processing

Stage de L3 auprès de Serena Villata
WIMMICS, I3S Laboratory, Sophia Antipolis, France

Flora Helmers

Été 2022

Contents

1 Introduction

Academic context of the internship As a bachelor student in the mathematics department at the École Normale Supérieure (ENS) of Lyon, a six weeks internship is mandatory in order to discover research in academic laboratories in France. Some internships are proposed, but it is also possible to find one by oneself, and this is what I chose to do.

My choice of internship in the field of Natural Language Processing (NLP) Natural Language Processing (NLP) is a research field generally considered at the crossroad of linguistics, computer sciences and machine learning. It studies the interaction between the computer and human speech, the goal being that the computer "understands" the subtilities of language which can be given by an unsaid context. Some task in NLP are topic detection, text generation, synthesis ... I wanted to study NLP because I find this topic attractive due to the diversity of techniques being applied in that domain, and I also like the focus on language.

Argument Mining is a subfield in Natural Language Processing. The difficulty even for linguists is to analyze how the arguments are formed and how they convince about an idea or an issue. What is Argument Mining used for? In particular, in the legal field, to detect inconsistancy, but also in medicin, for instance to justify a cure according to the symptoms. These are subjects that are tackled in the WIMMICS team, at I3S, in Sophia Antipolis, France. My goal during this internship was to gain knowledge about NLP, so that I could myself start applying NLP techniques, e.g. on mathematical litterature. Since I know no researcher working on this subject, I thought it would be easier to start by learning in an other domain : politics. The American presidential debates to be more precise. Here there are interesting argumentative structures because the goal of the candidates is to convince the audience that they are the one who should be elected, while the other is attacking the one, and they do so with a mix of prepared and improvised arguments.

Six weeks is a quite short time for discovering a research field, and it seems obvious that e.g. 6 months study would permit a wider discovery of the field with a more thorough focus on important aspects like machine learning for NLP, which I didn't have time to study. But it gave me the opportunity to study literature about NLP in general and scientic article about argument mining specifically, to start to code using some of python's library and to build end-to-end a knowledge graph. So I have done more bibliography and learning than research.

1.1 Coming to a subject of internship

Serena Villata a researcher in artificial intelligence currently working on computational argumentation applied to multiple fields, especially mining arguments from political debates but also in the legal field and in medicin.

A subject we talked about before hands was argument mining on French political debates and statistical analysis of graphs. Finally the subject was reduced due to the lack of time and my real task was to build a knowledgebase in the RDF (Resource Description Framework) format what allowed me to accomplish a complete task.

The report is structured as follow : first I will briefly introduce what tools are used in NLP, especially my dataset. Then I'll explain the concept of knowledge graph and how I almost built one. The final section is devoted to discussion about future directions.

2 Starting point for NLP based analysis

Here, I will briefly introduce NLP methods and the tools that were made available to me during this internship.

2.1 Basic research method in NLP

In this subsection I introduce a schema of research that I read about in the context of Argument Mining.

Even though it can vary, the main task in Argument Mining is detecting arguments and their relationships. To do these detection and classification tasks, the NLP techniques are used. A general method is first building a dataset, then using machine learning methods to detect relationships inside the data, then testing the output with statistical tests and finally seeing how it can be improved. This method follows the global evolution of NLP towards machine learning, which can be clearly seen when comparing the table of contents between the second edition and the third edition's draft of Jurafsky's textbook *Speech and Language Processing* [**jurafsky2**], where the topics get twisted around machine learning techniques.

In this section I will refer to the work of Shoreh Haddada [**shoreh**], Lippi [**Lippi1**], Mestre [**mestre1**] and Visser [**visser1**]. They are all about the detection of the arguments used in American presidential debates or, for Lippi [**Lippi1**], the UK election debates.

The first step is to create a dataset to work with. Necessary in every field of NLP, it is a time and resource consuming task. First raw data has to be gathered. Then annotation guidelines have to be fixed by experts of the field. They leverage definitions and gives examples of what should be identify or not. An example of annotation guideline is the one of the ElecToDebates dataset [**ShorehAnnotation**]. Then annotators have to annotate following the guidelines. They need to be trustworthy, sometimes they need to be trained (e.g. for the dataset US2016 [**visser1**]). Therefore the crowdsourcing solutions, where members of online communities contribute to the annotation, are often impossible to use. Then even if the annotators follows the annotations guidelines, such as the one described in , some unclear cases always appear. Therefore after the annotation, a new conversation starts among the annotators to determine what annotation is gonna be kept and why. This can make the annotation guidelines evolve. If a consensus emerge on some data, they are called gold standard data. In Argument Mining especially, the inter-annotator agreement is important because argument analysis is inherently controversial. Because of all this process, it can take from months to years to annotate a dataset, depending on the size and task. Therefore creating a dataset is a valuable contribution to the Argument Mining research community.

The second step is to do a pretreatment on the data. For spoken language, Mel-frequency cepstral coefficients (MFCC) is first used to represent the spectral components of the audio signal in Lippi's work [**Lippi1**]. For written text, some pretreatment can also be made, for instance by adding Part Of Speech (POS) tags to every words, using a dictionnary. POS tags indicate the type of word, such as noun, verb or even punctuation. This can be done with python's spacy library. Another pretreatment is word embedding, which introduce a distance between words depending on their meaning.

The third step is to define the features of the machine learning model, answering the following questions. What kind of model ? The Latent Dirichlet Allocation (LDA) statistical model is often mentionned for topic detection. Supervised or unsupervised? What tuning for the hyperparameters ? A reproach made by Mark Sandler, an expert in signal analysis, during his keynote at the 2022

Web Audio Conference, was that researchers did not explain enough in their papers how they tuned their models which some use as magic black box. As it is complicated and various, I recommand to an interested reader [**jurafsky2**] for the NLP part and [**goodfellow**] for more details about the maths behind.

The fourth step is to test the experience. For that statistical tests exists such as the F-measure, Cohen's kappa score...

The fifth step is thinking further. Here the work of Lippi and Torrini [**Lippi1**] and Mestre [**mestre1**] highlights the use of mix written and spoken media to enhance the detection of arguments. A greater intertextuality can be achieved by integrating comments from social medias and so on.

So when doing research in NLP, the dataset has a great importance, at least as much as using advanced technology for pretreatment and neuronal training. As resources flourish on the web, researchers get more chances to mix the media to enhance their results. In the following section, I will describe the dataset I had the chance to work on.

2.2 Data overview

The dataset I was able to use is the *USElecDeb60To16 v.01* dataset, whose construction is described in the article "Yes We Can ! Mining Arguments in 50 Years of US Presidential Campaign Debates" by Shoreh Haddadan, Elena Cabrio and Serena Villata [**yeswecan**]. The original data was presented as text in the official website of american political debates : http://debates.org. It is gathered by the Commission on Presidential Debates (CPD). It contains all the debates between the two major candidates from 1960, when the television broadcast the first American presidential debate ever, to 2016. The dataset was then annotated for two distinct purposes : decting argumentative components and their relationships and detecting fallacies. I worked only on the argument components part.

2.2.1 Description and goal of the existing dataset : the annotation guidelines

In this dataset, the argument components are subparts of the text that forms the argumentative structure. They are classified either as claims or as premises. The argumentative model behind that is Toulmin's triad explained in [**toulmin**]. The triad is composed of Claim/Warrant/Data (grounds). The Claim is the main component of the argument. It should carry the standpoint. The Data is provided to support the claim. Here Data will be called premise. The Warrant is the component that justifies the link between the Claim and the Data. As the Warrant is often implicit, it does not appear in the classification.

2.2.2 How to explore a dataset : python library and visualisation tools

Statistical description of my dataset This section provides statistical insight into the data.

In total I have 39 valid speeches from 1960 to 2016. I could have 42 but there were some issues with the structure of the section and/or data types inside the sections for the debates on the October the 11th and the 15th, 1992 and on October the 5th, 2000.

There are 149 participants. Each one is identified by a form : her name in capital letter, eg McGee becomes MCGEE. The problem is for instance that Bill and Hillary Clinton have the same form. So the participant are also associated with the date of the participation into a debates. Participant were then associated with a role. A list was made of the presidential and vice candidates

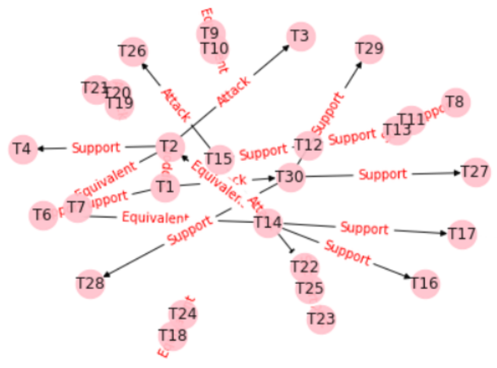

Figure 1: A networkx's representation of the arguments held during the presidential debates on October the 21th, 1960

called participants, containing all the names and surnames of the candidates. I used it to find out who wasn't there who I called the Journalists. Among them, I marked manually as Audience or Unknown the people who hadn't a real name. A finer way to find out who is who, would be to count how many questions where asked by each of the potential journalists.

The debates are split into section for unmentionned reasons, but I guess for homogeneity into the questions and that it is more easy to annotate something short.

There are in total 6517 speeches.

There are 30706 components of argumentation.

There are 24502 relations.

To calculate these statistics, I used the panda library, for working on tabular data. It helps to read the csv file, to put it in a readable form, to check that every column is full and to count the uniques values. These functions where used after every modification to check quickly the success of the operation. It is also useful to reunite all the files of the section in one file. The code of modification can be seen on my git.

Visualisation of the arguments A first way to watch the relationships between the arguments is to use python's networkx library, which enable to create graphics for the graph (section 2.2.2)

2.2.3 Adding new data

I wanted to enrich the data, with more information.

Finding the transcript of the debate A schema appeared for the URI of the debates. https://www.debates.org/voter
education/debate-transcripts/october-9-2016-debate-transcript/ But for 11 of them, it wasn't the case. So I used the following boolean returning function :

```
def url_exists(url):
    response = requests.get(url)
    return (response.status_code == 200)
```

And I used the sommary, to quickly access the rest of the unknown transcript URL. To access the videos of each debate, it wasn't that easy : so I preferred to focus on something else.

Adding metadata about the debate The data I have lack of informations about the place where the debate was held, about the organizers, and about the moderators. On the transcript page, one can access to the location of the event, to the title and full name of the candidates and to the full name of the moderators and the media they work for. The first speech of the debate also gives indications about the debate. For instance, on November the 21^{st}, 1980. Ruth Hinerfeld have the first speech of the debate opposing Ronald Reagan to John Anderson. She explains the debate is organized by the League of Women Voters Education Fund and the purpose of this association. To extract automatically this information, NLP techniques are mandatory.

As I did many manipulations with the files, I failed sometimes and erased them. Here is what I learned about testing the data with panda and saving it properly. The pandas' method DataFrame.info(), and DataFrame.nunique() allow to check regularly if there is no major error while manipulating the dataset. But it is still compulsory to save the data. An excessive method would be to regularly use the "chmod" command to modify the writing access right to the files. Another method is to use git and navigate in the historic when necessary.

3 Building a knowledge graph

3.1 Why among all types of datastructure the knowledge graph is the more useful

3.1.1 Overview of the existing data models

There are many types of databases, and each type organizes the stored data differently. In this project the question is not about the scale of the database (which is small), but about the relationships that we have between the data.

In computer programs data structures are used to organize storage of the data. Data structures are theoretical models such as a vector, a stack, etc. Here we will be interested in graphs which are adapted to analyze connectivity and relationships among entities. As they are represented by a tuple of sets $G = (V, E), E \subseteq V^2$, the relations between the edges have their own representation under the name of vertices. This structure is not standardized in the way it is written on the machine and how you can access it.

The databases are a middleware that store the raw data on the hardware and help the software application to get the data. They have a complex design, which includes considerations about data modeling, data representation, storage, query languages. We won't consider here the security, privacy and distributed computing issues. The database management system (DBMS or database system) is a huge part of the design, since it is the software interacting with end users.

There are different types of databases based on different data structures. The Relational databases (such as the .csv mentionned earlier) became dominant in the 1980s. They model data as rows and columns in a series of table, and the vast majority uses the programming language SQL (for Structured Query Language) for writing and querying data. In the 2000s, non-relational databases, referred to as NoSQL, became popular because they use different query languages. NoSQL databases can have very various forms, as it is only defined by disjunction with the relational databases. An example of NoSQL database is MongoDB. It stores files under a binary JSON format (called BSON). In the JSON format, every element is composed of a set of pairs of keys and their values. The interesting point about MongoDB is that the data is semi-structured and the server is developped to store the data externally. The following example is an individual from the debate collection :

```
{
   "_id" : ObjectId("62d91de33bce166041805bfc"),
   "DebateId" : "https://www.debates.org/voter-education/debate-transcripts/#1992#13Oct",
   "DateISO" : "1992-10-13"
}
```

It has a mandatory unique identifier stored under the key "_id". It has also two other keys : the "DebateId" and the "DateISO". In the collection, individuals can have different keys. Therefore it is semi-structured.

The goal when designing a database is to have a structure reflecting the structure of the real-world data to be stored and to be easily accessed. In our case, as the data is diverse and the goal is to extract knowledge, we will use a knowledge graph to gather the heterogenous data.

3.1.2 Short history of the creation of the knowledge graph

A knowledge graph is a "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" [**KG**]. Formally, the knowledge graph is represented by $G = (V, E, R), E \subset V^2 \times R$, where V and E are the same as defined above, and R is the set of types of relationships.

According to Nurdiati [**25years**], the notion first appeared in 1972. It was created by two dutch researchers : C. Hoede, a discrete mathematician, and F.N. Stokman, a mathematical sociologist. Their goal was to build a structure that could not only represent the content of scientific textbooks but that could also deduce knowledge from it. They used graphs. For the deducing part, it would especially help to automatically find causes and to make automated decisions, as the system would work on the causal relationships between the elements of the graph. Vries, a PhD student at the time, worked on the detection of causal relationships using linguistics marks (such as "because"). In [**structuring_knowledge_in_graph**], F. N. Stokman and Vries explain their motives, such as having a modular exploitation of the database. The latter could be used and improved without being changed totally, only adding elements.

The notion then evolved with different versions and semantics proposed by different researchers, such as Sowa (1984), Zhang (2002). In some versions, the set of types of relations R is limited in order to to facilitate algebras on the graph. In others, the cardinal of R increases indefinitely as new knowledge is added to the graph.

With the expansion of the Web, standards for integrating knowledge have emerged under the name of Semantic Web. They are based on knowledge graphs and their vertices are called resources. Massive knowledge graphs have been created since then on specific topics. Follow some of the most reknown ones with their first date of release. Wordnet (1985) captures semantic (ie related to the meaning) relationships between words and meanings. Geonames (2005) builds relations between geographic names and geopolitical or commercial entities. DBpedia (2007) extracts structured content from Wikipedia. The Google Knowledge Graph (2012) is based on public datasets including DBpedia, and enhances the leverage of the firm's search engine. A more open application of knowledge graph linked to NLP is Zeste, that exploits the Wordnet graph, to predict a topic

[zeste].

Using graphs to represent data helps to integrate a large variety of data coming from different sources. But then how to homologize the data? First step is to define a semantic, in the logical sens, for the knowledge graph. The way to do that is to write an ontology as a schema layer.

3.1.3 A semantic standard to describe such graphs : RDF

As knowledge graphs got widely used, formats emerged to describe then. The most used of them, Resource Description Framework (RDF) is a datamodel for describing and exchanging metadata. It was created by members of the World Wide Web Consortium (W3C), and the first draft was made public in 1997.

The syntax is based on triples of the form: subject - predicate - object. The predicate is the relationship between the subject and the object. For instance, "participant says speech" is a triple.

Some languages to express rdf : Turtle and XML Diverses languages exist to describe triples, they are called serialization format. Among them, Turtle is the most compact and human-friendly, so it is the one that will be used in this project. RDF/XML is the historical serialization format but is not easily readable. There also exist RDF/JSON and others. But whatever the format, the translation from a serialization format to another can easily be made either online or with python's library rdflib.

Ontologies : a way to standardize the data As RDF is a very modular format, it is necessary to define a schema layer to restrict the elements in the graph. These schemas are called ontologies. The name comes from the branch of metahysics concerned with the nature and relations of being. In Information sciences, another description is given by Tom Gruber: "An ontology is a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents. This definition is consistent with the usage of an ontology as a set of concepts' definitions." In the textbook *Knowledge Graphs* [**KG**], it is generalized in : "In the context of computing, an ontology is then a concrete, formal representation of what terms mean within the scope in which they are used."

When describing a graph or writing an ontology with RDF, existing ontology will be used. They are identified with an Universal Resource Identifier (URI) and a prefix. The two most used are

rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
rdfs : <http://www.w3.org/2000/01/rdf-schema#>

They define the base of RDF, by defining what a type is for a resource, etc. RDFS is especially created for writing ontologies. It defines the notions of types, classes, subclasses, properties, sub-properties and helps structure the ontology. Another interesting and essential library is

owl : <http://www.w3.org/2002/07/owl#>

which uses set theory to introduce actions on the classes such as disjoint. It also defines datatypes for litteral such as string or float.

So there are many ways to store data, but for deducing inference rules, RDF that helps building knowledge graphs is the most adapted. But such a modularity implies defining restrictive rules in order to keep a structure. Some point that could be interesting to study, but I didn't got the time, is to study the descriptive logics of such graphs. After defining a semantic for the entities and relationships, inference rules can be defined and new knowledge can be inferred. I did not have enough time to study the descriptive logics of knowledge graph [**potedeChris**]. But I have actually defined an ontology.

3.2 Building the debates ontology

There are two complementary ways to build an ontology : either bottom-up, going from the data and shaping the ontology around this. Or top-down, defining the requests that will be made on the ontology and then describing them. Here the competences expected could be to answer the following questions : What arguments were said during a specific debate? By who ? Who talked ? How many times ? On the arguments, the validity of an argument could also be checked. However as I dispose of an already built and structured dataset, the bottom-up version will also be used. I tried to use W3C standard formats, but here the main objective is to keep it simple, so sometimes I rewrite my own definitions of properties.

As I was writing my ontology, I found out that an ontology on the same topic already exists for the debates of the Dutch parlament under the name of polivoc [**polivoc**]. Its goal is to explore the parlamentary debates according to the topics and the personalities or entities mentionned during the debates. A great importance is given to the chronological order of the debates. I was inspired by this ontology to use the dublin core format, for sources and identifiers.

I called my ontology debates. It is available in the turtle format here. The prefix associated with it is "deb". I defined it's URI as "https://github.com/FloraHelmers/NLP_on_american$_{p}residential\_debates/blob/main$

3.2.1 Global Shape of the ontology

The Debates ontology has a four-points focus, reified in the form of four main classes : the debates , the speeches, the participants and the argument component. The structural schema is visible on figure Figure 2.

In this ontology, the assumption is that the debate happens during an American presidential campaign, but it could also be used for other countries and other political systems. Then the properties such as Country, ElectionType, Language, could be added to the debate property. Furthermore for the European debates the language could be the feature of each speech.

As I have a simple approach of the elements of RDF, my ontology contains only owl:Class and owl:ObjectProperties instances. I followed for that the model of the Argument ontology available at http://rdfs.org/sioc/argument. To justify it, I wrote it "manually" before I discovered the existence of Protégé, so there is less supclasses and supproperties than the good manners would advice. If I was to rewrite it, I would define a temporal property, and an argumentative relationships.

Instances populating the ontology There are several categories of objects in this ontology : classes, relationships and properties.

Every debate has two structural elements : the speeches, which are linked with the predicate deb:speech, and the participants, linked with the predicate deb:participant. The debate's metadata are introduced using dc:identifier and dc:Date. The link to the transcript's URL is made with deb:transcript.

The participants are all people who have a foaf:name. They also have a deb:participation which points to a node deb:Participation with three other properties the deb:debate they participate in, the deb:Role and their text identifier a deb:textIdentifier which is the string used in the transcript to identify them. The deb:Roles are divided in four subclasses. First and second, the deb:PCandidates

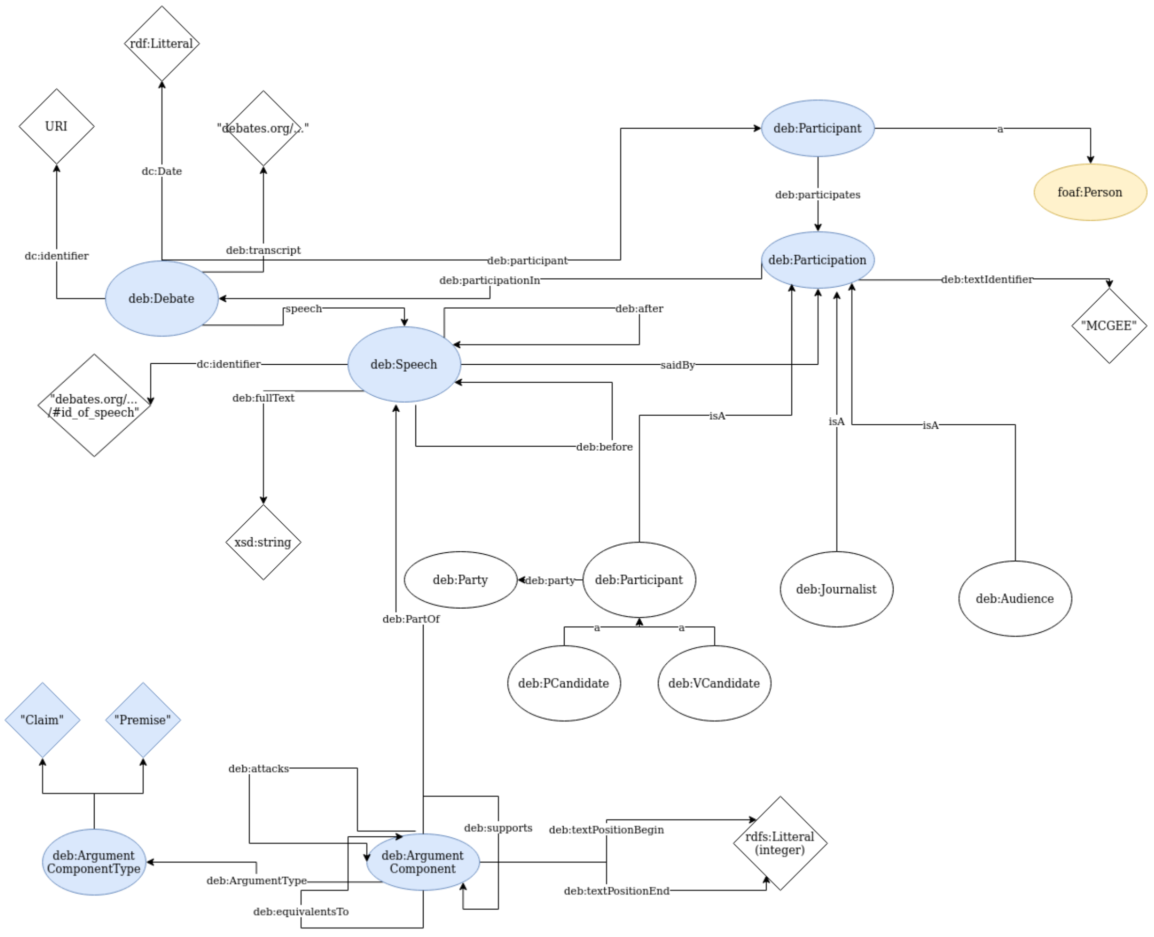

Figure 2: View of my ontology
The rounds represent a class, the arrows a relationship or a property, the diamonds an instance or an example.

and the deb:VCandidates are respectively the presidential and vice candidates and belong to the class deb:Candidates. The property "a" is a shortcut to showing the type. The deb:Candidates have a property pointing to an instance of the deb:PoliticalParty class, which is either democrat, republican or independent. Those party are resources that are linked to the wikipedia page of the party. The third are the deb:Journalist, which are linked to a newspaper with the deb:employer property. The fourth and last is the deb:Audience, which gathers the rest of the others subsection, including the unknown people.

A speech is defined as the text pronounced consecutively by one speaker. Every speech is related to its full transcript and is linked to an unique deb:Participation.

Three interesting relationships which has as range and domain the deb:Speech and deb:ArgumentComponent class are the deb:before, deb:after, deb:during properties, which introduce a chronological order between the subparts of the debate.

The deb:ArgumentComponent have a deb:ArgumentComponentType which is either a deb:Claim or deb:Premise. They are deb:partOf a speech. Those are defined as in [**shoreh**]. A litteral is pointed by deb:TextPositionBegins. It is the integer designing the position of the first character of the text of the argument component. The position is calculated inside of the speech's full text. Because arguments can be cleansed of stop words, the deb:fullText property also apply for deb:ArgumentComponent.

Naming them with an URI In the RDF format, most resources have an identifier. In the semantic web, this identifier is called a URI, an Uniform Resource Identifier. It is a string of characters that uniquely identify a name or a resource on the internet. [**difference-between-uri-url**]. A subtype of these URI are URL which also specifies how to reach the resource on the internet. For instance, "flora.helmers@ens-lyon.fr" is an URI that can be precised into an URL "mailto://flora.helmers@ens-lyon.fr".

To link the project to websemantic, I preferred to use existing URIs instead of creating my own. First, the URI of the ontology is
deb : https://github.com/FloraHelmers/NLP_on_american_presidential_debates/blob/main/my_ontology.ttl/
, which gives direct access to the ontology written in turtle.

Second, for the multiple resources stored in the graph, I built a schema for the four main classes of resource : Participant, Speeches, Debates, ArgumentComponents. Here under is the Python code. So I chose to define hash functions based on the url of debates.org, to make a reference to the provenance of the text. The Python code is written below.

As in the original data, the debates are identified by their dates (two digits for the day and three first letters of the month) and the year of happening (four digits). I kept this representation. As the other elements are related to a debate, their identifier contains the URI of this debate. But such an assuption has got wrong with the notion of Participant (see subsubsection 3.2.2). So it should be changed. For the speeches and the arguments, no natural identifier existed, as the ID in the annotation was arbitrarly defined. So I assigned a new number to them, trying to use the chronological order, and pandas' method sort.

```python
# Python code : obtaining the URI
global_uri = "https://www.debates.org/voter-education/debate-transcripts/"

def URI_debate(year, date, URI=global_uri):
    return f"{URI}#{year}#{date}"

def URI_participation(form, year, date, URI=global_uri):
    return URI_debates(year, date, URI) +"#Participation"+ form

def URI_speech(year, date, speech_id, URI=global_uri):
    return URI_debates(year, date, URI) + f"#Speech{str(speech_id).zfill(3)}"

def URI_argument_component(argument_component_id, year, date, URI=global_uri):
    return URI_debates(year, date, URI) + f"#ArgComp{str(argument_component_id).zfill(4)}"
```

The URI of the Partipants is a string containing their full names.

The URI has no incidence on the structure but it carries indication about it and it's provenance, as every name does.

3.2.2 Defining the classes of Participant and Participation

The Participants of the debates are important because some of them are candidate to the uttermost function in the american democracy. They not only represent the standpoints of political party, but they also want to represent the values of a massive part of the american citizens and their actions tend to have an important impact on the political life. The journalist also are important because even if they were supposed to be neutral, it is her choice to ask one question instead of another, so they are the one shaping the debate. An ontology already exists to deal with the people and their relationships : it is called Friend Of A Friend (foaf). An interesting thing to do would be to link it with a subgraph of foaf (http://xmlns.com/foaf/0.1/).

A tricky point appeared in the definition of the Participant. The role of the candidates is related to a particular debate but what they say is related to a person. My first idea was to build an instance for each participation, as the URI_participant still reminds. But then a participants is a person, that is always the same being as time goes by. So it is compulsory to have both. So several solutions appear. It is not possible to put condition on a predicate. Therefore I introduced the deb:Participation class to add unmixed related properties. It could be a blank node, which cannot be identified but it is not a problem since it directly linked to the participants and to the debate disjointly. But I gave it a URI.

3.2.3 Text : chosing the granularity

The dataset was originally split in subparts of each debate. I assume this choice was made in order to keep a unity of topics, and to focus on smaller part of the text when annotating. But then the dataset was also split into speeches, sentences and even words for POS tag (NLP) purposes.

To keep the graph readable, and not with an extensive number of triples, I chose to use speeches as a text unit. They have a unity since one speech is said by one person during a continous interval of time. They keep the sequence of sentences still readable. The topic is also quite homogenous. And if some part needs to be extracted, a simple offset can be used as explain above for deb:TextPositionBegin.

But at the end, other formats can also added. Some resulting functionality could be interesting to have. For instance, marking the moderators' transitions would together with the temporal links help distinguish unities inside the dialogue. One more time, this needs NLP's classification methods.

Chosing not to used Web Annotation Web Annotation is a W3C standard for annotation : oa : http://www.w3.org/ns/oa# .
It's useful for NLP since it enables to create Annotations, constituted of a body and a target. The strength of Web Annotation is that it has many selectors on many type of resources. At the end, in order to simplify, I didn't kept it. But a project could be to build an intertextual knowledge graph for debates(cf subsection 4.1).

3.2.4 Argument Components

An ontology for organizing arguments already exists at http://rdfs.org/sioc/argument. But as the analyze of the relationships between the arguments doesn't match the classification made in the

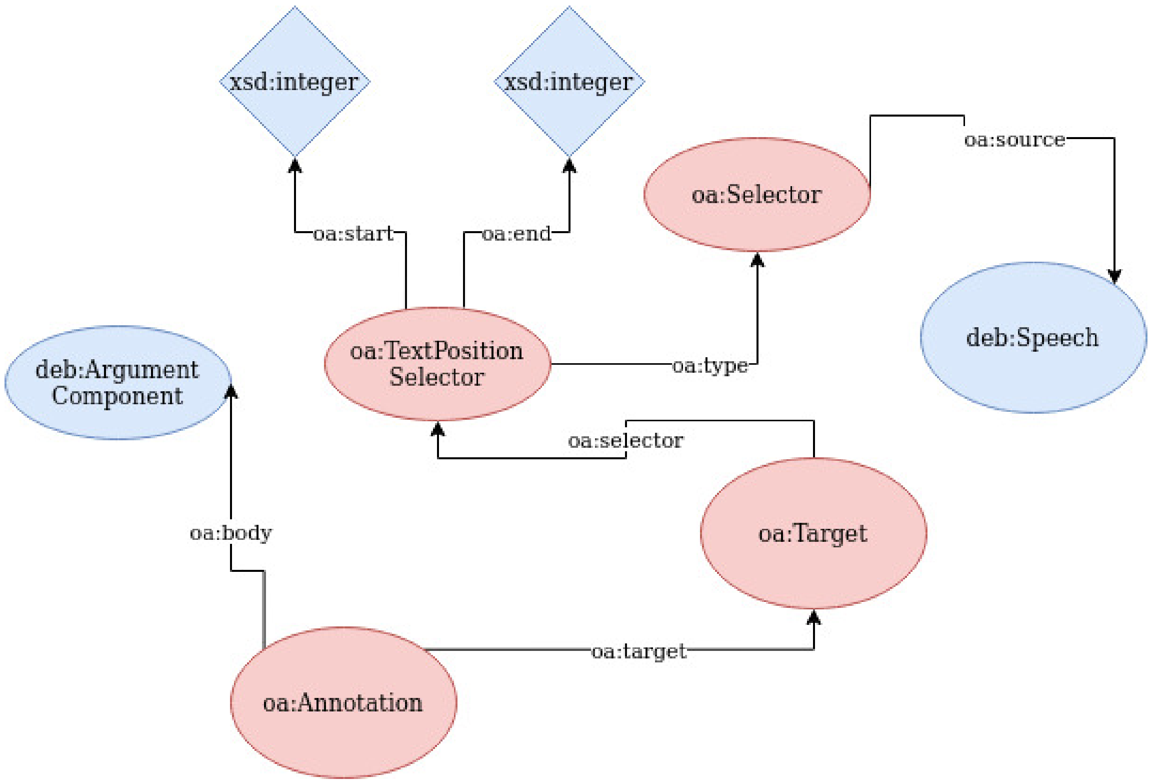

Figure 3: Schema of the Web Annotation Model used with the deb:Argument
The rounds represent a class, the arrows a relationship or a property, the diamonds an instance or an example.

annotation [**ShorehAnnotation**], it could not be used without losing its interesting subtilities.

3.2.5 Writing the rules

Here follows an extract of the ontology written in Turtle. The dc11:description links the class to definition of the class, which I chose to give with the words of the Cambridge dictionnary.

```
deb:Debate
   a owl:Class ;
   dc11:description "a serious discussion between several people, where arguments are
       exchanged" .

deb:transcript
   a owl:ObjectProperty ;
   rdfs:domain deb:Debate ;
   rdfs:range rdf:Resource ;
   dc11:description "Link to the official transcript of the debate." .
```

3.2.6 Checking the validity of my ontology

After having put stops and full stops at the right place, the file can be loaded into Protégé, a software made to write ontologies using a visual help. But a copy should be used since the file is modified. Following a tutorial about Protégé, made me rethink the structure of the ontology.

Moreover if I had more time, I would have check the consistency of the graph. In logics, the consistency is the property of a set of assertions and inference rules from which comes no contradiction. Here the triples would have been considered as being the assumptions.

Conclusion I produced a simple but data-driven ontology, which can be used to explore debates. Now the idea would be to populate it with the existing data.

3.3 Populating the knowledge graph with xR2RML

3.3.1 What is xR2RML?

Created by Franck Michel [**spec_xr2rml**], xR2RML is a RDF mapping language, which is used for translating data stored for instance in JSON files into RDF triples. It is based on RML another RDF Mapping Language. And it is an extension of R2RML which addresses the mapping of relational databases to RDF. Here we will map the data from NoSQL MongoDB database.

To use it, it is necessary to write mapping rules, with one's favorite RDF serialization format. For that, two ontologies are used :

rr : http://www.w3.org/ns/r2rml#
xrr : http://www.i3s.unice.fr /ns/xr2rml#.

The mapping document represent a mapping graph that represents the xR2RML mapping. Indeed to create a subject-predicate-object triple, you have to create a rr:TripleMap. This has at least three properties. The first is the xrr:logicalSource whose query into the database provides the elements to be treated. The second is the rr:subjectMap which specifies which characteristics from the resulting elements are going to be the subject of the triples. And the third is rr:predicateObjectMap that specifies the predicate with rr:predicate, and the object with rr:objectMap. More rr:predicateObjectMap can be added in order to create several types of triples with the same subject. Here is a simple example for mapping the debates.

```
<mappingDebates>
    a rr:TriplesMap;
    xrr:logicalSource [
        xrr:query """db.debates.find()""";
        #xrr:format xrr:JSON ;
    ];

    rr:subjectMap [
        xrr:reference "$.DebateId";
        rr:class deb:Debate;
    ];

    rr:predicateObjectMap [
        rr:predicate dc:date;
        rr:objectMap [
            xrr:reference "$.DateISO" ;
            rr:termType rr:Litteral;
            rr:datatype xsd:dateTime;
        ];
    ].
```

The rest of the mapping rules is available on my git.

But to transform the mapping document into the wanted triples, a xR2RML processor is needed. The processor provides, given an xR2RML mapping and an input database, the access to the output RDF dataset, as explains Franck Michel in [**spec_xr2rml**]. The implementation of this processor is accessible at https://github.com/frmichel/morph-xr2rml.

3.3.2 Transforming the dataset into a MongoDB database

The first step is to convert my .csv files into a MongoDB database. Exporting to Mongodb is easy once mongod is successfully installed and started with 'systemctl'. The names of columns become keys for every individual, each one being created by a row.

However a pretreatment is useful for adapting the form of the mongoDB collections in order to minimize the complexity of the semantic of the queries in the mapping rules. The logical source I want to work with uses xrr:query and rml:iterator like in the following example.

```
<#mapped_Speeches>
   a rr:TripleMap;
   xrr:logicalSource[
       xrr:query "db.SpeechesCollection.find({speeches: { $exists: true} })";
       rml:iterator "$.speeches.*";
   ];
```

I also want easy MongoDB queries such as the one between quote in the latter example. So I rewrote the csv files so that they match the predicate structure of the RDF triples. It means that each row has an URI and that the other columns are propriety or relations with an other URI. To do so, I extensively used python's pandas library, as described in the previous sections.

An interesting feature is however the xrr:template, which helps to avoid defining a specific column for the URI for example.

```
<mapped_Speeches>
   a rr:TripleMap;
   xrr:logicalSource[
       xrr:query "db.SpeechesCollection.find({speeches: { $exists: true} })";
       rml:iterator "$.speeches.*";
   ];

   rr:subjectMap [
       rr:template
           "https://www.debates.org/voter-education/debate-transcripts/#{$.year}#{$.date}#{$.GlobalSpeechID}"
           ;
       rr:class deb:Speech ;
   ];

   rr:subjectMap [
       rr:template
           "https://www.debates.org/voter-education/debate-transcripts/#{$.year}#{$.date}#{$.GlobalSpeechID}"
           ;
       rr:class deb:Speech ;
   ].
```

3.3.3 Result

The process returned only a file with the prefix. I guess the problem comes from an error in the configuration file or in the configuration of the mongo database server. It would still need some time or help to fix it. Then I would visualize the created knowledge graph with tools for ontology visualisation, such as LodLiv. Then I would run some Sparql queries to learn how I can access data.

Conclusion This work made me do an end-to-end construction of a knowledge graph, starting from writing the ontology to populating the graph with elements of the database. After this task, there is always something more to do.

4 Perspective for future works

After exploiting the knowledge graph, there are some improvements I want to do on the structure of the graph. The first is adding more media to the graph. The second is to tag the speeches with topics.

4.1 A better ontology for intertextuality in political debates

As underlined in the work of Mestre [**mestre1**], the mix of transcript and audio recording of a debate enhance the comprehension of the argument. Therefore I think it could be interesting to build an ontology for debates based on the Web Annotation ontology [1]. The interest would be for humanities scholars to be able to explore the debate database and having access to the textual transcript, the audio and the video.

The difficulty in the implementation would be to create automatically a temporal link (like the previously defined deb:after, deb:before, deb:during) between the various types of selector : get to know if the part of the transcript happened before or after a part of a video. This part of annotating the timing was particulary long for the annotators in the experiment of Mestre [**mestre1**]. One idea could be to generate the text from the video's audio using speech recognition, and to match the resembling parts.

Then the comments made on social media could also be linked to the graph. Here also there could be a temporal link.

4.2 Adding topics inside the knowledge graph

Something useful could be to include the topics of each speech. I started applying the Latent Dirichlet Allocation model proposed by python's gensim library on the speeches of my database, to extract important words. But because I used it quickly on just one speech, it didn't make much

[1]https://www.w3.org/TR/annotation-model/

sense. Serena Villata suggested to extract the words only from the journalists speeches. Due to lack of time, I didn't implement it.

To work in a more finer way, a list of topics related to the US politics should be defined. Then associated vocabulary should be defined. And maybe linking it to an already existing knowledge graph such as DBpedia could be interesting. In fact new topic modelling techniques are developped using knowledge graph such as in the Zeste project [2] explained in [**zeste**].

5 Conclusion

To conclude, Knowledge Graphs offers new possibilities to explore and exploit data. As the sources flourish and the resources get numerous, it really helps integrating new data. After having built one, I wish to continue to explore what it has to offer, in particular to learn how to exploit them.

An exploitation of the knowledge graph about American presidential debates could be done, in particular by cooperating with political sociologists, to explore the dynamics around the debates. Another exploitation that could be done for the arguments, would be to use the knowledge graph embedding techniques, to predict the existence of the triples.

I want to thank Serena Villata, for making me discover this field. I also want to warmly thank the WIMMICS team, for welcoming me among them, and showing me diverse perspectives of what argument mining and Web Semantic can be.

[2] https://zeste.tools.eurecom.fr/