Pipeline for narrative extraction

Departamento de Ciência de Computadores

Faculdade de Ciências da Universidade do Porto

Pedro Eduardo Nogueira da Mota

Advised by:

Prof. Alípio Mário Guedes Jorge

May 2021

Contents

1 Introduction

Narrative texts are often characterized by narrative sequences with features such as the chronological succession and causality relations between events, and the presence of one or more protagonists who suffer a process of transformation throughout the story [1]. These features make narratives both appealing and useful, namely, to aid humans to communicate complex concepts, ideas, realities.

Given the huge body of texts containing narratives in cultural heritage and their continuous production, there is a pressing demand for applying and developing Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques for extracting narratives from texts.

Currently, there is an extensible framework to generate visual representations of narratives from annotated texts. A Narrative Annotation Visualization tool, *Brat2Viz* [2], has been developed for supporting the debugging of narrative annotation done with the Brat annotation tool [3]. *Brat2Viz* implements a pipeline that transforms the annotation into a formal representation (DRS) [4] and from this, to Message sequence charts [5] and Knowledge Graphs [6] visualizations.

The purpose of this project was to complement this visualization tool, by implementing some narrative extraction algorithms, in order to remove the manual effort of generating an annotation from the raw text, giving support for both the English and Portuguese language. A broader objective is to lay the grounds for a wider narrative extraction package that can be used by different future research lines and applications.

The final result will be a Narrative Extraction Visualization Pipeline, for the Python language, designated *Text2Story.*

1.1 The implemented pipeline

The pipeline starts with the story and its publication time as input. After this, through some methods of extraction, one can extract the actors and the times entity structures and the objectal link structure defined in our semantic annotation framework (see section 3). These methods allow to specify which tool and their combinations are to be used in the process of extraction. In the case of no specification, all the tools available for that particular extraction are used.

The actor and time entity structures are extracted through named entity recognition and cross-domain temporal tagging, respectively. The objectal link structure is extracted through co-reference resolution. These extractions, besides identifying our structures, also extract information that we can use in their sub-tags. This extra information that results from the extractions is processed in order to keep it accordingly to the annotation scheme used in our project. The sub-tags defined in our structures that can't be filled by the information extracted by the tools are defined by some heuristic, typically filling the sub-tag with its most common value.

In the package, all the structures have their own classes, with their attributes being their sub-tags. Within each narrative, we keep a list for every type of structure. Each identified structure has as an unique identifier (a letter - 'T' for actors and times structures, 'E' for events and 'R' for links - followed by an unique number). In this way, after the extractions, one can generate the annotations, accordingly to the brat format, very easily, which can then be utilized in the *Brat2Viz* module.

Section 4 explains in more detail the implemented pipeline.

2 Background

Natural Language Processing, usually shortened as NLP, refers to the branch of artificial intelligence that deals with the interaction between computers and humans using the natural language, whereas the main objective is to the give the computers the ability to understand text and spoken words in much the same way human beings can [7].

The problem with human language is that it is ambiguous and full of irregularities, such as homonyms and idioms, which makes it extremely difficult to write software that accurately determines the intended meaning of the text.

Typically, the process of NLP is broken into several tasks, in order to deal with all complexity. Some of these tasks, that intersect with the work done in this project, include the following:

- **Part of speech tagging** is the process of determining the part of speech of a particular word based on its use and context. Part of speech identifies 'fly' as a verb in 'I can't fly because I'm afraid of heights' and as a noun in 'A fly is an insect'.
- **Named entity recognition**, also known as NER, mainly identifies words as useful entities, also classifying them into pre-defined categories such as person names, organizations and locations, for instance. As an example, NER identifies 'Portugal' as a location and 'FIFA' as an organization.
- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or the object to which a certain pronoun refers.

There are currently many tools, implemented in various programming languages, but specially in Python, that solve these tasks with a good accuracy. These tasks are first steps in a pipeline for a narrative extraction from text.

Narrative extraction from text consists of identifying the events and protagonist that appear throughout the story and the relations between them, such as causality relations between the events or even the chronological succession of events. The main goal is to extract, from raw text, narratives that can then be presented in various forms. Narratives are both appealing and useful, namely, to aid humans to communicate complex concepts, ideas and realities.

Also, related to this project is Labeling and Discourse Representation Scheme, also known as DRT, created by Hans Kamp in the early 1980s. In formal linguistics, discourse representation theory is a framework for exploring meaning under a formal semantics approach. DRT includes a level of abstract mental representations (discourse representation structures, DRS) within its formalism, which gives it an intrinsic ability to handle meaning across sentence boundaries [4].

A discourse representation structure (DRS) is a mental representation built up by the hearer as the discourse unfolds. A DRS consists of two parts: a universe of so-called "discourse referents", which represent the objects under discussion, and a set of DRS-conditions which encode the information that has accumulated on these discourse referents.

To illustrate, the following DRS represents the information that there are two individuals, one of which is a teacher, the other a student, and that the former congratulated the latter:

[x, y: teacher(x), student(y), congratulated(x,y)]

The universe of this DRS contains two discourse referents, x and y, and its condition set is teacher(x), student(y), congratulated(x,y).

The generated annotations will be used to construct a DRS for the story, which is then used as input to the *DRS2Viz* module, currently under development by the rest of team, that produces the visualizations in the web browser.

Our focus will reside on the automating the process of annotations and then link this process to the *Brat2DRS* module, creating the Narrative Extraction Visualization Pipeline, *Text2Story*.

3 Semantic annotation framework

Semantic annotation or tagging is the process of attaching to a text document or other unstructured content, metadata about concepts (e.g., people, places, organizations, products or topics) relevant to it. Semantically tagged documents are easier to find, interpret, combine and reuse and they also can be used by machines [8]. In our case, we are interested in capture the information relevant to the construction of a narrative.

With this in mind, the annotation scheme used in our project follows the semantic framework from ISO 24617-1/9 [9, 10], for the referential and temporal level, and from Linguistic InfRastructure for Interoperable ResourCes and Systems (LIRICS) for the semantic role labeling level [11, 12]. Some adaptations regarding, for instance, the number of tags and types of attributes were made due to the multilayer annotation and to some properties of the language (European Portuguese) and of the genre of the corpus (news) [13]. As such, our current annotation scheme has three types of tags:

- Actors, to annotate characters in the story (e.g., 'um homem' – 'a man');
- Events, for events (e.g., 'assaltou'– 'robbed');
- Times, for temporal expressions.

In our multilayer semantic framework there are also tags for link structures. These link structures capture the relations between the mentioned entities and there are five types of them:

- Objectal, to state how two discourse entities are referentially related to one another;
- Semantic Role, to detect the arguments associated with the predicate or verb of a sentence and how they are classified into their specific roles;
- Temporal, to represent different temporal relationships (simultaneous, after, before) holding between two events, two times, or between an event and a time;
- Subordination, to represent modal, factive, counter-factive, evidencial conditional relations between two events;
- Aspectual, to represent relations between aspectual events and their event arguments.

A scheme of the framework is presented in figure 1.

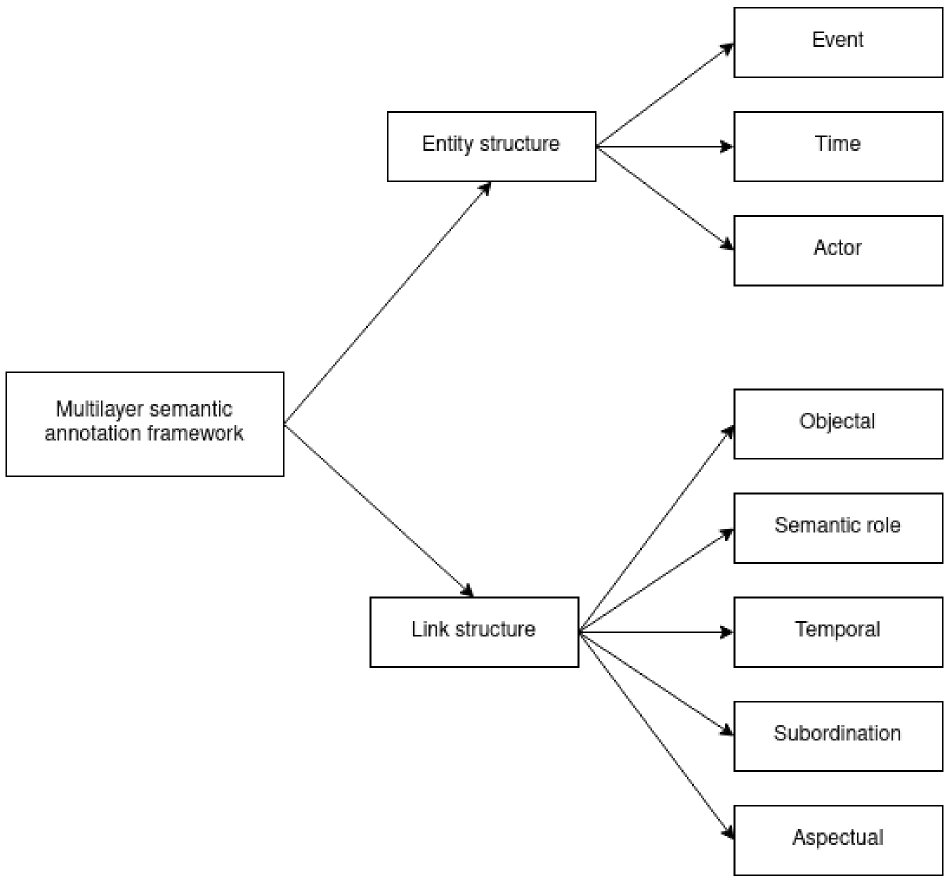


Figure 1: Semantic annotation framework scheme

As stated in section 5.1, the Event entity structure and the links of Semantic Role, Temporal, Subordination and Aspectual were left as future developments.

Each one of the tags in our framework has attributes (sub-tags) so that we have complete meanings for every component annotated. The sub-tags for the entity and link structures are presented and explained in sections 3.2 and 3.1.

3.1 Entity structures

As stated, the entity structures Actor and Time represent the participants and the temporal expressions presented in the narrative, respectively.

3.1.1 Actor entity structure

Relatively to the Actor entity structure, its lexical head is annotated within the tag 'lexical head', which assumes the values 'Noun' and 'Pronoun', which corresponds naturally to the noun or pronoun lexical heads. There are also tags relatively to the 'domain' of the actor, which then subdivides into two tags named 'individuation' and 'type'. The tag 'individuation' is a stipulation of whether it is a set, a single individual, or a mass quantity (such as a bit of fresh air, some coffee, or some music), thus take values 'Set', 'Individual' and 'Mass', corresponding to these descriptions. The 'type' tag is used to classify the actor into the following categories:

- 'Per', for person names;
- 'Org', for organizations;
- 'Loc', for locations;
- 'Obj', for objects;
- 'Nat', if it's nature related;
- 'Other', if it doesn't fit in any of the above categories.

They are also annotated with the tag 'involvement', which is the specification of how many entities or how much of the domain are/is participating in an event. The values that these tags take are represented in the figure 2, as well the structure of the actor entity.

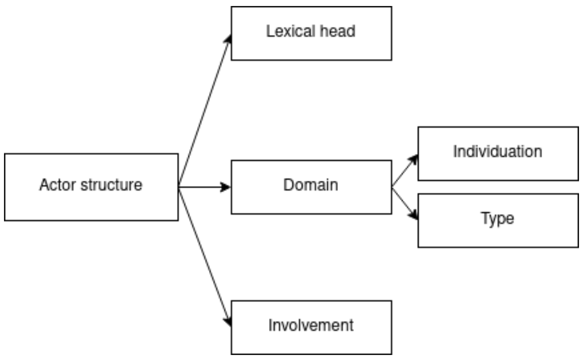


Figure 2: Actor entity scheme

3.1.2 Time entity structure

Regarding the Time entity structure they are annotated with three tags, as shown in figure 3.

The attribute 'type', naturally, captures the type of the temporal expression, which can be a 'Date', 'Time', 'Duration' or a 'Set', with all being self-explanatory. The attribute 'value' has the specific value of the temporal expression and, finally, the attribute 'temporal_function' has a boolean value, which is true, if the temporal expression is relative to the publication time, otherwise, it's false.

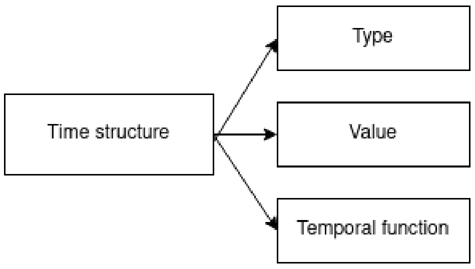


Figure 3: Time entity scheme

3.2 Link structures

The link structures represent the relations between the mentioned entities, such as causality relations between the events.

3.2.1 Objectal link structure

Relatively to the Objectal link structure, this link states how two discourse entities are referentially related to one another and it is further divided into a set of types, as illustrated in figure 4.

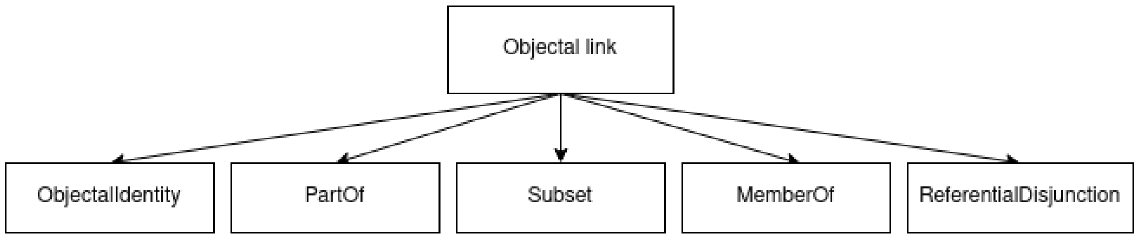


Figure 4: Objectal link scheme

The type 'ObjectalIdentity' captures the relation between referring expressions that have identical referents, the type 'PartOf' captures the relation between two referring expressions, where one referent is considered as a part of the other, the type 'Subset' represents the relation between two referring expressions where one referent is considered as a set and the other referent as a subset of the other, type 'MemberOf' captures the relation between two referring expressions where one referent is considered as a set and the other referent as a member of the other and, finally, the type 'ReferentialDisjunction' captures the relation between two referring expressions where one referent is explicitly disjoint from the other.

4 Developed software

As already stated in section 1, the main focus of the project, besides automating the process of annotating text, was to produce a extensible pipeline in Python, where all the different modules being currently worked by the team would culminate, namely the *Brat2Viz* module.

In order to keep the code base maintainable and extensible, a very modular architecture was followed, as demonstrated in figure 5. All the related source code can be found here.

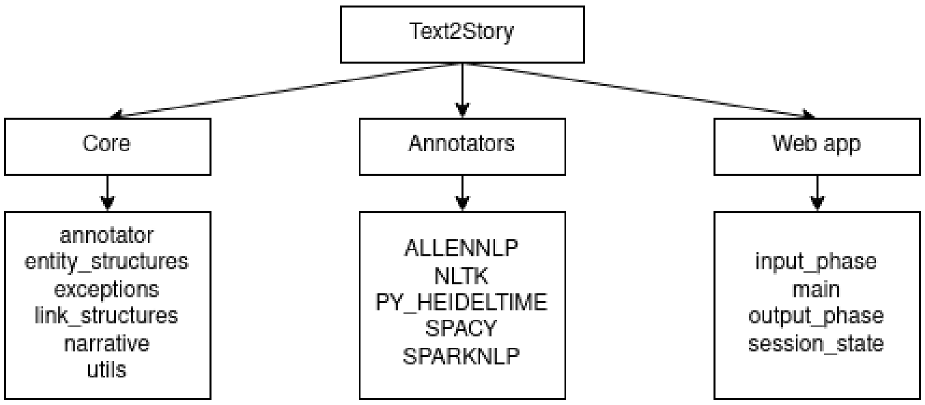
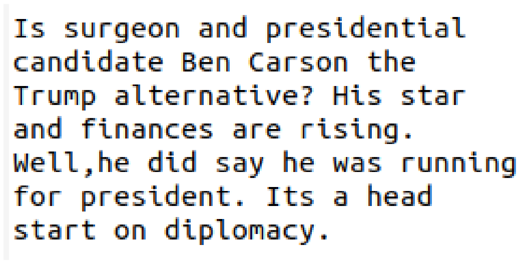


Figure 5: Text2Story package architecture

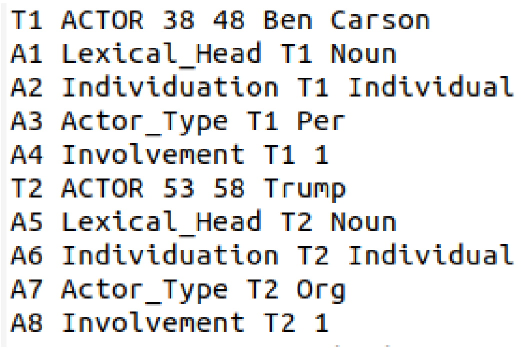
The construction of the pipeline started from representing and creating a narrative and then we focused on the extractions. The actor extraction was done first, followed by the times entities and finally, the objectal links between them.

The class Narrative is the main class of the package and implements modules of input and output. The user interacts only with this class: either for creating the narrative or to do the extractions. A narrative is created by given the story that we want to extract the narrative from, its language, and its publication date. After the extractions, that will be explained later, this class implements a method to generate the annotation, accordingly to the brat standoff format. An example of a small story and its corresponding generated annotation is illustrated in figure 6. As already stated, this generated annotations are then used as input in the *Brat2Viz* module.



```
Is surgeon and presidential
candidate Ben Carson the
Trump alternative? His star
and finances are rising.
Well,he did say he was running
for president. Its a head
start on diplomacy.
```

(a) Original story



```
T1 ACTOR 38 48 Ben Carson
A1 Lexical_Head T1 Noun
A2 Individuation T1 Individual
A3 Actor_Type T1 Per
A4 Involvement T1 1
T2 ACTOR 53 58 Trump
A5 Lexical_Head T2 Noun
A6 Individuation T2 Individual
A7 Actor_Type T2 Org
A8 Involvement T2 1
```

(b) Generated annotations

Figure 6

The modules 'entity_structures' and 'link_structures' define the entities and link structures, previously defined in our semantic framework.

The modules 'exceptions' and 'utils' exist mainly for good programming practices, since they define the exceptions that can be generated by our entire package, such as an invalid language, when the user choose a language that is currently not supported, for instance, and some utilities functions, respectively.

The module 'annotator' defines our META-Annotator which will be explained later, following the main reasons why it exists in the first place.

The sub-package 'Annotators' is where all the current tools to do the extractions co-exist. It was mandatory that the package should had support for different tools. This presented a problem, since each tool, despite conceptually doing the same, work in completely different ways. So, in order to keep the package configurable and extensible, a rigid structure was adopted. In this way, it should be easy to add news tools or even change the models of the current tools employed.

The solution came from defining an uniform interface to interact with the different tools. In order to achieve this, each supported tool simply implement one function for loading the models either for the English and Portuguese language (with the only purpose being to load the models when the package starts, so when we create new narratives, one doesn't need to reload their pipelines again, we can simply reuse the already loaded pipeline) and functions to do the extractions.

Throughout the development, namely after the implementation of the actor extraction, some tools and their respective models were giving unsatisfactory results compared to a manual annotation. Nevertheless, they have different strong and weak points, so the idea of combining their outputs in a final output, gathering each tool strong cases, seemed reasonable. This lead to the idea of a META-Annotator, presented in the 'annotator' module. In a nutshell, the META-annotator interacts with the already referred interface of the annotators and implements the same interface, while allowing the result of one or more annotators to be combined, if specified so. In the end, all the interactions from the core of the package within the annotations should be through this META-Annotator.

With this design, we accomplished the proposed objectives: the META-Annotator is easily configurable and extensible, while providing a simple interface.

In sections 4.1, 4.2 and 4.3, we explain in more detail how each structure is being extracted. These are shown in ascending chronologically order as they were implemented.

4.1 Actor entity extraction

For the extraction of the entities, we did it mainly by named entity recognition.

Relative to the named entity recognition, we used the tools: the Natural Language Toolkit (NLTK) [14], spaCy [15] and the Spark NLP [16] annotator. We used the default models for the English and Portuguese language in each tool.

The interface function *extract_actors*, implemented in the modules corresponding to these annotators, under the 'Annotators' sub-package, receives the text and it's corresponding language as input and returns a list of three dimensional tuples. Each tuple corresponds to a named entity identified and consists of the character span, the part-of-speech tag and the named entity itself. These entities will correspond to our actors: the character span is used to identify the actor in the text, the part-of-speech tag will become the lexical head of our actor and the named entity will become the actor's type.

Since the different tools and models work with different labels, we also normalize their labels to the labels defined by our semantic framework. For the part-of-speech tags, we converted the various labels related to pronouns and nouns to just the labels 'pronoun' and 'noun', respectively. Note that the entity will be discarded if it has a POS tag different than a pronoun or a noun. For the named entity, the relation between the identified labels and the labels defined by our semantic framework is typically one-to-one, that is, the predefined categories implemented by the tools that we are using to do the named entity recognition are similar to our actor types, so we simply map this categories. When the category has no direct relation, we label it with the general label 'Other'. Also, this process typically identifies dates and times entities, similar to our Time entity structure, however the information given through this process is rather incomplete, in the sense it mainly identifies the temporal expression, giving no information about it, thus we discard these entities and do the extraction through another process, as stated in section 4.2.

The META-Annotator, which interacts with this interface, also implements the same interface.

To combine the outputs from these tools, we used the character span. We start by the entity with the lowest character offset identified, then we find intersections in the character span from the entities identified and if we find one, then we know it's the same entity. Thus we can accumulate the new information about the POS tag and the named entity label and also the character span. We do this process until the character span of the entity stabilizes. After the character span stabilizes, we know we have the complete information about this entity, given from all the tools selected to the extraction.

This process of combining tools got us better results. Besides the immediate increase in the number of entities identified, we also got more precise results. For instance, one tool could classify 'UNESCO' with the general label 'Other', but if another tool made a more specific classification, like 'Org', for organization, then we could improve the final result, simply by favoring more specific labels.

Apart from the effort put in to improve the results of the annotations, compared to a human-made annotation, the results can still be improved a lot. As reported in section 5.1, this can be done either by adding new tools or by training new, more specific, models.

4.2 Time entity extraction

For the timex extraction, the tool py_heideltime [17] was used, which is a Python wrapper for the multilingual temporal tagger HeidelTime.

The interface function *extract_times*, implemented in the module of the these annotators, receives the text, it's corresponding language and the publication date of the story as input and returns a list of three dimensional tuples. Each tuple corresponds to a temporal expression and consists of the character span, it's value and it's type respectively. These temporal expressions will correspond to our time entity structures: the character span is used to identify the time in the text, and the value and type attributes correspond to the tags 'value' and 'type' in the time entity structure, respectively.

The results from the tool py_heideltime consist of the text, with the identified temporal expressions in tags. The attributes of the tags are the 'tid', an identifier, the 'type' and the 'value', which correspond directly to our tags 'value' and 'type' in the time entity structure.

For instance, the output for the text 'Thurs August 31st - News today that they are beginning to evacuate the London children tomorrow. Percy is a billeting officer. I can't see that they will be much safer here.' is 'Thurs <TIMEX3 tid="t2" type="DATE" value="XXXX-08-31">August 31st</TIMEX3> - News <TIMEX3 tid="t3" type="DATE" value="PRESENT_REF">today</TIMEX3> that they are beginning to evacuate the London children <TIMEX3 tid="t4" type="DATE" value="XXXX-XX-XX">tomorrow</TIMEX3>. Percy is a billeting officer. I can't see that they will be much safer here.'

To extract the identified tags and then the information within them, we parsed the text, with the help of regular expressions. The attribute 'tid' of the tags were discarded, since we have our own identifiers. The attribute 'type' and 'value' became the same attributes in our time entity structure.

4.3 Objectal link extraction

The objectal link extraction was done through co-reference resolution.

The annotator AllenNLP [18] was used to this extraction, with the default model, for the

English language. As stated in section 5.1, there's currently no support for the Portuguese language, since neither of the tools that we are currently using offer support for the Portuguese language.

The interface function *extract_objectal_links*, implemented in the module of the corresponding annotator and in the META-Annotator, receives the text and it's corresponding language and returns a list of clusters, where each cluster is a list itself, consisting of the character spans, which are represented by two-dimensional tuples. These clusters are the parts in the text that are co-references. For instance, consider the following text, in English: 'John is my friend. I love him.'. In this text, 'John' and 'him' are co-references, thus they are part of a cluster, in this case, the cluster is just formed by this two co-references, so it's represented as [(0,4), (25,28)].

Also important to note, is that this process of co-reference resolution will add new actors to our narrative. In our framework, we defined the lexical head of actors as being either a pronoun or a noun. Since our approach to actor extraction is through named entity extraction, which simply classifies named entities mentioned in the text into predefined categories such as organizations and persons names, this process will fail to identify actors with pronoun as a lexical head, since these don't correspond to named entities. On the other side, the process of co-reference resolution identifies these actors, whose lexical head is a pronoun, so, after the objectal link extraction, new actors may appear.

4.4 Graphical interface

Alongside the development of the package, a graphical interface was implemented, inspired by the current tool *Brat2Viz*, developed by the team. The graphical interface, aside from being a clear final use of the package, can be quite useful during the development by allowing us to visualize some of the implemented work, while executing some tests.

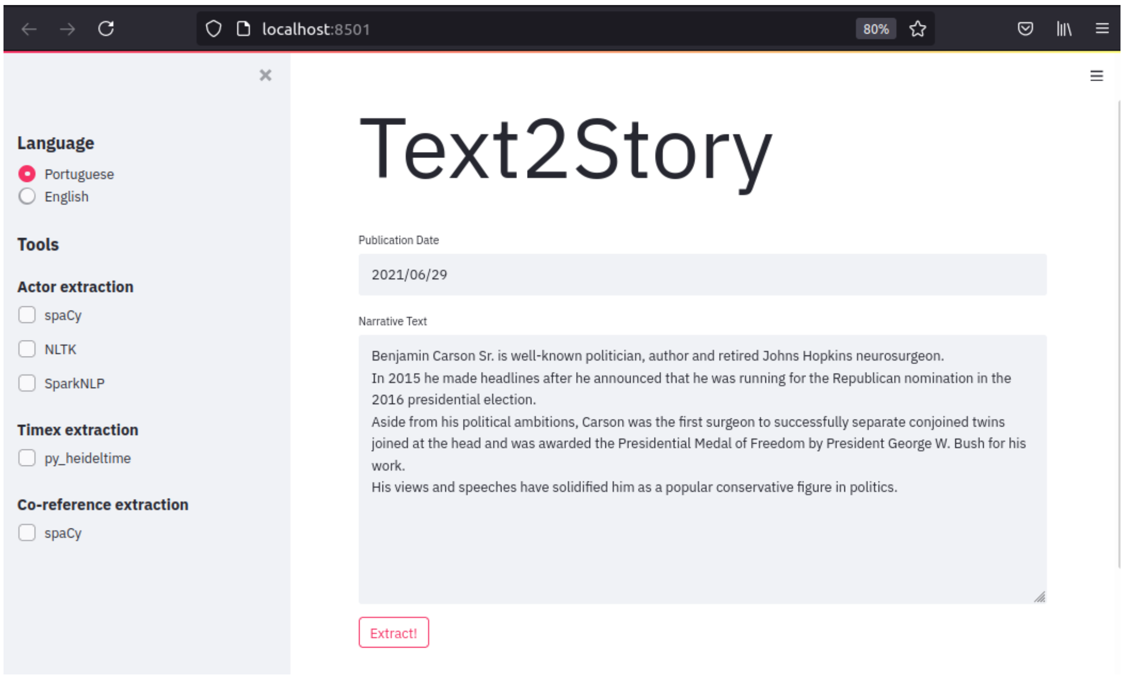
The graphical interface was implemented uniquely with the tool Streamlit.

The interface should be intuitive to use and, simultaneously, aesthetically pleasing. To achieve this, a minimalist design was followed. With regards to the interaction with the package, the graphical interface should provide a way to input the text and to specify the tools to do the extraction. After the extraction, the user should be able to navigate through the different visualizations, namely the generated annotation, the formal representation (DRS) and the final MSC and Knowledge Graphs visualizations.

The interface has two phases: an input phase, where the user adds the story, its publication time and selects the tools to do the extractions and an output phase, that occurs after the input phase, where the user can navigate through the different modes of visualization.

The design of the input and output interface are illustrated in figures 7 and 8, respectively.

An online demonstration can be found here.



[✓ Portuguese] English

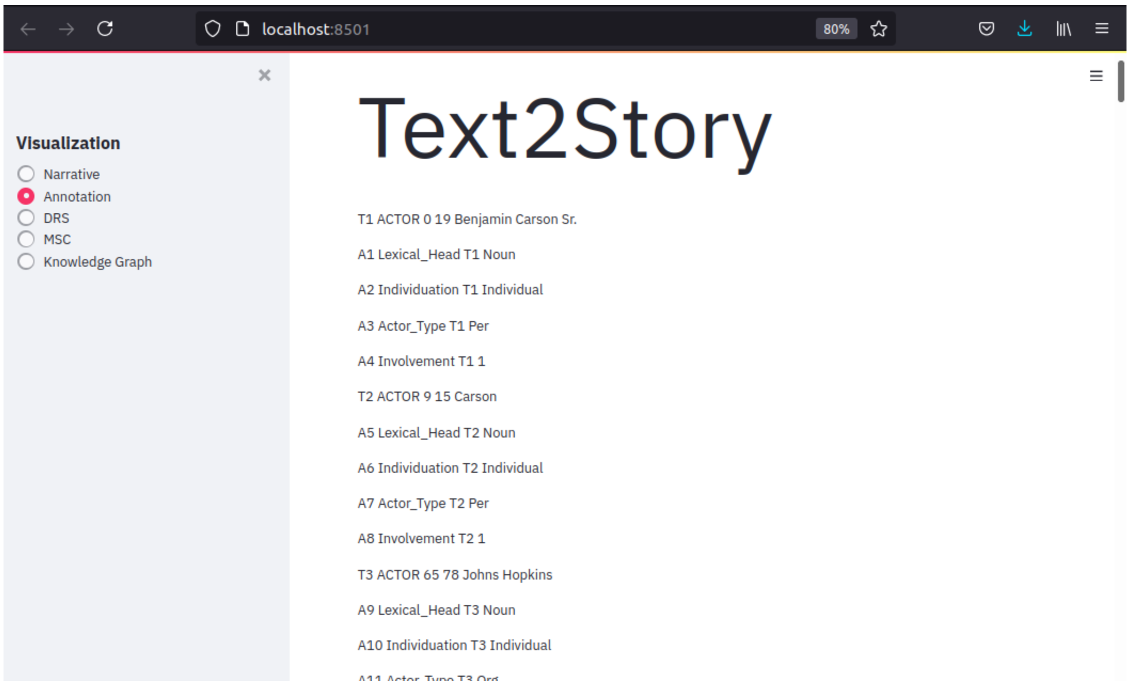
Publication Date

2021/06/29

Narrative Text

Benjamin Carson Sr. is well-known politician, author and retired Johns Hopkins neurosurgeon.
In 2015 he made headlines after he announced that he was running for the Republican nomination in the 2016 presidential election.
Aside from his political ambitions, Carson was the first surgeon to successfully separate conjoined twins joined at the head and was awarded the Presidential Medal of Freedom by President George W. Bush for his work.
His views and speeches have solidified him as a popular conservative figure in politics.

Figure 7: Input interface



Narrative [✓ Annotation] DRS MSC Knowledge Graph

```
T1 ACTOR 0 19 Benjamin Carson Sr.
A1 Lexical_Head T1 Noun
A2 Individuation T1 Individual
A3 Actor_Type T1 Per
A4 Involvement T1 1
T2 ACTOR 9 15 Carson
A5 Lexical_Head T2 Noun
A6 Individuation T2 Individual
A7 Actor_Type T2 Per
A8 Involvement T2 1
T3 ACTOR 65 78 Johns Hopkins
A9 Lexical_Head T3 Noun
A10 Individuation T3 Individual
A11 Actor_Type T3 Org
```

Figure 8: Output interface

4.5 Features of the interface

4.5.1 Input interface

- **Inputting the text**

  Inputting the text is done by writing the text in the text area.

- **Selecting the publication time**

  Selecting the publication time is done by clicking on the field of the data publication and then by navigating through a calendar.

- **Selecting the language of the text**

  To select the language of the text, one simply choose the desired language from the left sidebar.

- **Selecting the tools to do the extraction**

  Like in the language selecting, to select the tools to do the extraction, one simply choose from the left sidebar.

- **Start the extraction**

  To start the extraction, one can click on the 'Extract!' button, which will lead to the output page. This transaction is only done if the input configuration is valid. For instance, if no tool is specified to the actor extraction, then, when clicking the button, the output page won't appear, but an error message will.

4.5.2 Output interface

- **Navigation through the different visualization options** On the output page, to navigate through the different modes of visualization, one can simply select from the left sidebar the desired one.

5 Conclusions

Having acknowledged the difficulty inherit to the natural language, this project tried to automate the process of annotating raw text, generating annotations that then can be utilized to produce all sorts of narrative visualizations, such as MSC and Knowledge Graphs.

During this project, we implemented some of this process, by identifying the elements of our semantic framework in the story through some NLP common tasks, such as named entity recognition and co-reference resolution.

Alongside, we also developed a Graphical User Interface, which is a by-product of the pipeline and where different visualizations can be developed in the future.

Despite not having reached a fully functional pipeline, this project resulted in a solid ground-base pipeline, paving the way for future work towards the narrative extraction visualization pipeline.

5.1 Future work

The event entity structure and the links, except from the objectal one, have been left for the future due to lack of time. So, in a future interaction, the immediate work would be to complement the missing parts relatively to the semantic framework.

In addition to these completions, the extractions, especially the actor extraction, could gain a lot from specific trained models. One can still gather the extraction through named entity recognition, but the current tools lack some support for some of the types of actors that our semantic framework defines, namely the 'Obj' and 'Nat' types. So a model that could identify these types more accurately, would improve the generated annotations a lot, just by reducing the number of general labels.

Another point that concerns a deeper analysis were the attributes where we used some heuristic approach, in order to label some attributes of the entities, namely the involvement and the individuation in the actor entity. If required so, one can train a model to this categorization.

Also, there's no support for co-reference resolution for the Portuguese language.

After this completions, the tool *Brat2Viz* can be fully integrated in the package and the set of languages supported could be expanded.

5.2 Self-evaluation

With regards to my performance, I believe more work could have been done. Nevertheless, with the odd situation we are currently living and the increased difficulty due to the incomplete, lack or even wrong documentation that I had to face, I find that my contributions to the project were still valuable, either quantitatively or qualitatively, so I am happy with the work that was done and with the broad knowledge that I gained in the field of natural language processing.

References

[1] Daniela Alderuccio and Luciana Bordoni. An ontology-based approach in the literary research: twocase-studies. InLREC, 2002.

[2] Evelin Amorim, Alexandre Ribeiro, Brenda Salenave Santana, Inês Cantante, A. Jorge, Sérgio Nunes, P. Silvano, Antonio Leal, and Ricardo Campos. Brat2viz: a tool and pipeline for visualizing narratives from annotated texts. In *Text2Story@ECIR*, 2021.

[3] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April 2012. Association for Computational Linguistics.

[4] David I. Beaver Bart Geurts and Emar Maier. Discourse representation theory. In Edward N.Zalta, editor, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, spring 2020 edition, 2020.

[5] David Harel and P. S. Thiagarajan. Message sequence charts, page 77–105. Kluwer AcademicPublishers, USA, 2003.

[6] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. InSEMANTiCS(Posters, Demos, SuCCESS), 2016.

[7] Medium. A simple introduction to natural language processing, 2018. `https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32`, Acessed on 30-06-2021.

[8] Ontotext. What is semantic annotation?, 2018. `https://www.ontotext.com/knowledgehub/fundamentals/semantic-annotation/`, Acessed on 30-06-2021.

[9] International Organization for Standardization. ISO/WD 24617-1, Language resource management-semantic annotation framework (semaf)—part 1: Time and events, 2007.

[10] International Organization for Standardization. ISO 24617-9, Language resource management—semantic annotation framework —part 9: Reference annotation framework (RAF), 2019.

[11] Volha Petukhova and Harry Bunt. LIRICS Semantic Role Annotation: Design and Evaluation of a Set of Data Categories. InLREC. Citeseer, 2008.

[12] V. Petukhova A. Schiffrin, H. Bunt and Susanne Salmon-Al. LIRICS Deliverable D4. 3. Documented compilation of semantic data categories, 2007.

[13] Marıa Teresa Pisa Cañete. La construction discursive de l' événement rapport e dans les textes desgenres informatifs de la presse francaise. Çedille. Revista de Estudios Franceses, (7):272–305, 201.

[14] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python.* O'Reilly Media Inc., 2009.

[15] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[16] Veysel Kocaman and David Talby. Spark nlp: Natural language understanding at scale. *Software Impacts*, page 100058, 2021.

[17] Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013.

[18] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.