A Framework for Fostering Easier Access to Enriched Textual Information

Gabriel Silva
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Coimbra, Portugal

Mário Rodrigues
IEETA, ESTGA, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Coimbra, Portugal

António Teixeira
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Coimbra, Portugal

Marlene Amorim
GOVCOPP, DEGEIT, University of Aveiro, Portugal

Abstract

Considering the amount of information in unstructured data it is necessary to have suitable methods to extract information from it. Most of these methods have their own output making it difficult and costly to merge and share this information as there currently is no unified way of representing this information. While most of these methods rely on JSON or XML there has been a push to serialize these into RDF compliant formats due to their flexiblity and the existing ecosystem surrounding them.

In this paper we introduce a framework whose goal is to provide a serialization of enriched data into an RDF format, following FAIR principles, making it more interpretable, interoperable and shareable. We process a subset of the WikiNER dataset and showcase two examples of using this framework: One using CoNLL annotations and the other by performing entity-linking on an already existing graph. The results are a graph with every connection starting from the document and finishing on tokens while keeping the original text intact while embedding the enriched data into it, in this case the CoNLL annotations and Entities.

2012 ACM Subject Classification Information systems → Document representation; Information systems → Ontologies

Keywords and phrases Knowledge graphs, Enriched data, Natural language processing, Triplestore

Digital Object Identifier 10.4230/OASIcs.SLATE.2023.2

Supplementary Material
Software (Dev Repository): `https://github.com/gabrielrsilva11/GraphBuilderAPI`

Funding This research is funded by National Funds through the FCT - Foundation for Science and Technology (UI/BD/153571/2022). It is also funded, Research Unit IEETA, by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

1 Introduction

With the the expansion of the internet and IoT, the world saw a dramatic increase in the amount of data that is generated every day [12]. While some of this data comes structured, what we observe, especially in the last decade, is a huge amount of unstructured data that was previously mostly ignored due to the difficulty in processing it. This difficulty in processing comes not only in the form of lack of processing power but also in the fact that the information in this type of data is enconded in natural language. This data is scattered

throughout the web and comes from many sources. Tweets, forums, comments, anything that is written in natural language can be a valuable source of data to be converted into information. For example, twitter, can be used to identify adverse drug reactions [8] or analyse comments to have a better understanding of patient feedback [13].

There are many methods to enrich this data that have been developed over the years, Named Entity Recognition (NER), keyword extraction, topic modeling, among many others to fit the needs of those working with this data. Another of such methods is text annotations [23]. Text annotations are labels that can be added to specific parts of a text to provide additional information about the content. These annotations can include information such as part-of-speech tags, named entities, or even more complex structures such as a syntactic trees. This enriched data comes in different formats, for example, NER identifies which words are recognized as named entities while something like syntactic annotations need something more structured (like a table or a tree) for their representation. As a result of these different processing methods of large amounts of data that is enriched by different frameworks or humans is a complex process which can consume a lot of resources to accommodate.

Traditionally these annotations have been made and shared using, mostly, XML or JSON. Both the Text Encoding Initiative [1] and the ISO TC37 SC4 WG1 [2][27] have XML defined guidelines for publishing/sharing data with text annotations. However, there is a growing trend to integrate this enriched data with knowledge graphs in favour of JSON or XML formats due to the flexibility and the existing tools for the semantic web [7].

In this work we aim to tackle common problems that are found while working with enriched data. In general, these problems are:

- Difficulty in managing the enriched data with possibly many levels/layers and diversity of formats for each.
- Difficulty in sharing the results of the data enrichment.
- Lack of a common, unified way of representing the enriched data.
- Processing and exploration of the enriched data requires costly development of custom computational solutions.

Having identified these problems we set a few objectives for this work: (1) Creating a framework that can integrate different annotations within a single knowledge graph; (2) Create a knowledge graph that represents a unified way of representing documents (keeping every connection possible) along with their annotations; (3) Make this knowledge graph shareable and integratable with other software according to the user needs.

To tackle these problems our goal is to serialize enriched data into an RDF/OWL compliant format. Achieving this serialization will make systems that this format more interpretable, interoperable, and integrable with the already existent semantic web ecosystem.

By integrating this enriched data with knowledge graphs (both new and existing) we can create richer relationships that have the potential to help improve algorithms. It will help build more robust and accurate applications as well as sharing the information obtained.

The framework should make it possible to go directly from documents to a knowledge graph. Building a framework capable of doing this would avoid certain issues like having different data enrichment tools generate different representations. There is also the added benefit of being more convenient to whoever uses the framework.

[1] https://tei-c.org/
[2] https://www.iso.org/committee/48104.html

The work will be developed using CoNLL and Entity Linking as our main use-cases to showcase the framework as well as the resulting knowledge graph. The decision to use CoNLL was because, as mentioned before, it is a widely used and accepted standard despite the existing challenges. Entity linking will be to show the extensibility of the framework and how it can be adapted to different annotations.

Paper structure. Following this introductory section, the paper structure is as follows: Section 2 will focus on the framework and the different modules it comprises. How they were developed and what each module does. Section 3 we present the use-cases of the framework: CoNLL annotations and Entity Linking. Section 4 is the related work chapter, and Section 5 will present a conclusion and future work.

2 Our proposal: A framework for accessing enriched textual information

This section will focus on the framework that was developed as well as the decisions behind each module and how the requirements set at the beginning of the work were accomplished. It starts by describing the system and how its modules work, followed by what was used during the implementation as well as its issues. In the end we present a use-case of the framework which will be entity-linking.

2.1 Requirements

The requirements set for this framework relate to both the general problems identified as well as the concrete objectives defined for this work. As such, the requirements are:

- Standards adoption – Adopting standards, such as RDF/OWL or CoNLL, makes the tool be able to be integrated with an already existing suite of software.
- Extensible and Flexible – There are a lot of different text annotations that users might want to contemplate using and as such it should be possible to extend the usability of the tool to cater to these different annotations.
- Keep all the connections intact – That means linking the text to sentences, sentences to words and annotation to words and make it easy to navigate both ways. This will help keep the structure of the text intact and recreate the original document if needed. Ease of navigation in the knowledge graph is a must to develop different applications of the tool.
- Storage/Software Agnostic – The goal here is for the users to be able to integrate the tool with whatever stack they are using making it as easy to use as possible for everyone. Creating and uploading a knowledge graph should not be dependent on, for example, a single triple storage system.
- Abide by FAIR [28] principles – The four FAIR principals are the following: findable, accessible, interoperable and reusable. These are principals we want to abide by because of how important and connected to the problems we found they are, especially the last three.

These were the major requirements that were identified as well as the rationale behind each one of them.

2.2 Proposal/System

The major goal of this framework is to provide professionals with an extensible, easy-to-use library which can be adapted to their needs and relies on standard practices and tools only.

We propose a framework with an initial implementation as a python library that takes a document and transforms it into a direct RDF serialization of enriched data that follows the FAIR principles [28]. The first serialization, and the default one implemented in the library, is done using CoNLL.

By serializing the enriched data into RDF while keeping all the connections between elements intact we create an easy way to navigate this information as well as build upon it.

2.3 Framework overview

This framework uses standard resources that are not unknown to anyone that has ever worked with NLP before. SpaCy and Stanza, which are the two main drivers of this work, are widely used both in industry as well as research. The serialization is done to RDF which is another well-known standard and one that is widely used, not only for NLP tasks. The main difficulty found while developing the library was making it flexible so that users can, for example, write their own queries.

The Fig. 1 presents a simple overview of the framework and its different components for our use-cases.

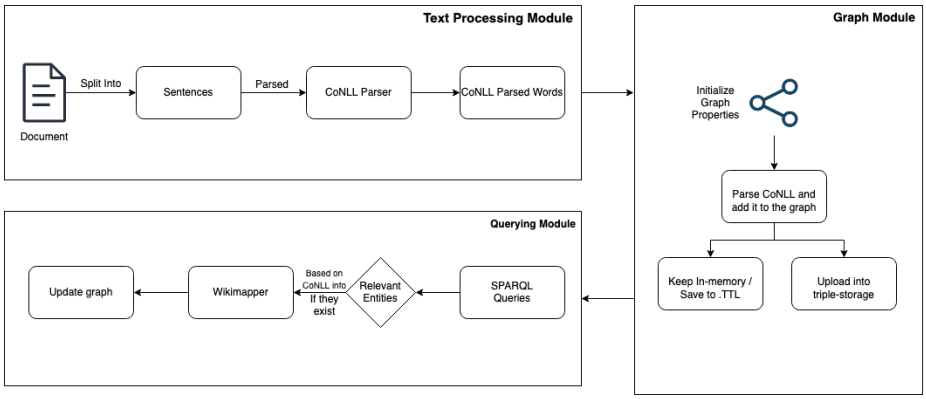

Figure 1 Overview of the current framework architecture and the different modules.

The framework consists of 3 main modules: (1) Text Processing Module; (2) Graph Module; (3) Querying Module. These are the models without which the framework would not function. Each of them handles a different part of the pipeline needed to go from enriched data to the final RDF serialization.

The text processing module objective is to take the raw document and apply the information enrichment methods into it. For example, in our use case, and as seen in Figure 1 we start off by processing the document, running it through a CoNLL parser (our enrichment method) and we then build the knowledge graph based on this.

The Graph module, as the title implies, is responsible for the serialization of the enriched data into an RDF compliant format. This involves parsing the enriched data, creating the necessary connections to maintain the connections in the document intact as well as either generating an RDF file or uploading it into a triple-storage system.

The last module is the querying module. This module is responsible for querying the data and displaying its information or updating an existing graph to add information to it. In our use-cases this is when entity linking is performed to showcase the flexibility of the framework.

The implementation of each module will be presented in the next subsections.

Table 1 Explanation of what each CoNLL data property represents.

Attribute	Description
id	The index of the word in the sentence.
word	The text of the word
lemma	The lemma of the word
pos	Part-of-speech of the word
feats	Morphological features of the word
edge	Universal dependency relation of the word
head	ID of the syntactic head of this word

2.4 First implementation of the Framework

In this section we will discuss how each module was implemented and the functionalities present in each one.

2.4.1 Text processing module

The text processing module is responsible for reading the documents and parsing them into the CoNLL format.

The first step is to split the text into sentences so that we can keep the connection from text into the different sentences. Followed by using SpaCy [18] and Stanza [25] (formerly StanfordNLP) to create the CoNLL representation.

This module allows us to choose the language in which the document comes in and the output is a pandas dataframe in a standard tabular CoNLL format. Table 1 shows each CoNLL attribute and their meaning. From here on out this is the representation we will use to create our graph.

2.4.2 Graph Module

In order to build the graph we first start off by defining the required classes, data properties and object properties. There are three classes defined: Text, sentence and word. This is so that we can preserve every connection possible to be able to navigate the graph whichever way we want and make it so that it is possible recreate the original document, even if just an unformatted version of it.

The data properties are the ones that come from the parsed CoNLL data. These include: edge, feats, lemma, id, pos, poscoarse, word and another data property called "sentence_text" that connects the first word of the sentence with the sentence itself. The head property from CoNLL will be defined as an object property instead of data since it isn't related with the data itself but with the structure of the text and the syntactic tree.

When it comes to object properties most of these were created because of either the structure of the text or with navigation in mind. We create the following object properties: head, depGraph, nextSentence, nextWord, containsSentence, previousWord, fromSentence. Each of these properties are used in different one or various classes/instances. For text the property containsSentence is used to know which sentences belong to a document/text. For sentences we have nextSentence, this is used to know which sentence came next in the text, and depGraph that represents the start of the syntactic tree of that sentence (the

root node of the CoNLL representation). The rest of them are related to words, nextWord and previousWord are related to the order in which the orders appear in the sentence and depGraph are the words that depend on the current word. Figure 2 has a simplification of these connections and showcases how we can navigate in the graph. X is the number of the sentence and Y is the number of the word, so Sentence_1 would be the first sentence and Word_1_3 would be the third word of the first sentence. After adding these properties and

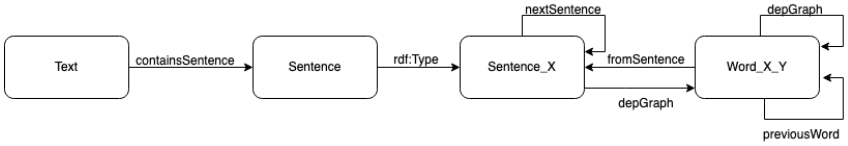

Figure 2 Simplification of the connections existent in the knowledge graph.

classes all that is left is to create the graph. We go through the CoNLL tabular data, parse it, and start by adding the sentences and appending the words to these sentences with the respective tags.

It is also possible to choose how you want your graph. There are two methods available, build it in-memory and export a TTL (Terse RDF Triple Language), a format used to express RDF data, file or upload it into a triple storage system as long as there is an available end-point. Ensuring the need for an endpoint instead of supporting specific triple-storage systems makes the system more agnostic and able to be integrated with different systems to suit different users needs.

The entity linking portion of the work is just a demonstration of a use-case of this framework. We start off by identifying relevant word tags, such as, obj, obl or nsubj and their dependents. From there on we utilize wikimapper which is a project that helps mapping page titles with the corresponding wikipedia articles. If there are any relevant entities and corresponding wikipedia articles we update the sentence to link these to the wikipedia pages and therefor get a more complete knowledge base.

2.4.3 Querying Module

The final module of this system is the Querying Module. This is a module that comes with some pre-built queries and the option to build your own. Currently this module also includes the entity linking module.

The current built-in queries are able to: (1) insert triples into a storage; (2) Find sentences that start by a given string; (3) Find sentences that contain a specific string or list of strings; (4) Find sentence by id;

These queries are built with SPARQL [3], a communication and querying protocol, and work for both triple-storage methods and the in-memory one. There are also functions that use these queries to accomplish other tasks, for example, building sub-graphs with all the dependencies starting from the root, find the syntactic path of a given subset of words to the root node, finding the node of a given word.

This module operates mostly in the same way for every problem a user might have. It starts off with a query to find out the useful nodes, for example, querying by sub-strings or by an attribute. We build a sub-graph of the queried information and then apply our navigation methods to find what the user wants to know.

[3] `https://www.w3.org/TR/sparql11-query/`

3 Use-Cases

To exemplify using this library we make use of two use-cases. The first use-case, which comes built-in with the framework, is the creation of a knowledge graph with CoNLL annotations. This example will show how we keep all the relations in a document and how we add our own annotations to it. The second use-case is entity linking. Using the previously created knowledge graph we perform entity linking and append the extracted entities to the corresponding sentences.

We used the WikiNER [9] data for the Portuguese language and split it into a small subset of data. We extracted 5121 sentences and processed them, built a knowledge graph with CoNLL data (first use-case) and performed entity linking (second use-case).

3.1 CoNLL

Looking at one of the most popular annotation methods and one of our use-cases, CoNLL [2], we can already identify some challenges when working with it. While this is a format that is widely adopted by several systems and used by researchers everywhere it is not the most human-readable format which makes interpreting the results hard, using tools to parse the result table may lead to transformation errors or loss of information and sharing results is not the easiest task.

To build the knowledge graph, we started off by processing the information as shown in Figure 1. The input is a text file with each line being a sentence. The library starts off by reading the file and converting the sentences into CoNLL using a combination of SpaCy [18] and stanza [25]. This is done in chunks to avoid using more memory than what the system would be capable of handling. As the chunks are parsed, they are being added to a new knowledge graph. In this case we are not doing this in-memory but using a triple-storage system called Virtuoso. We parse the text into CoNLL format and then, the resulting tabular data is converted into triples to be uploaded to our system by an endpoint.

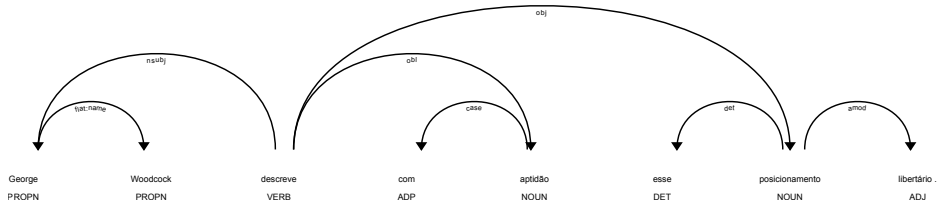

Figure 3 Example of the universal dependencies parsing for the sentence “George Woodcock descreve com aptidão esse posicionamento libertário .”.

To highlight the flexibility of the system we wanted to first create the CoNLL graph based on this subset of data and then add the entity linking part of the work. The processing and upload of the data took 4 hours to complete and the final result was a graph, with every connection intact (document -> sentence -> word -> CoNLL attributes). Figure 3 shows an example of the type of annotations, in this case the universal dependencies, present in our data properties that come from CoNLL.

The result, shown in Protégé using OntoGraf, and for the sentence “George Woodcock descreve com aptidão esse posicionamento libertário .” is shown in Figure 4. The framework is made available as a Python library and to process a document the user has to mention:

the desired relationships uri, an endpoint (if uploading to a triple storage), the language of the document and the document or folder of documents. Then all that is left is to instantiate our *CreateGraph* class and call the *create_graph* method. The output will either be an RDF file or the graph uploaded into a triple storage. A more in-depth explanation of the process is in Section 2.4.

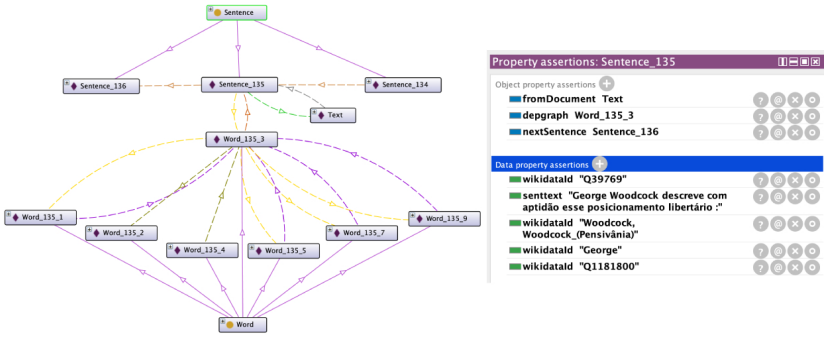

Figure 4 Example of a sentence and its connections seen in Protégé [20] with OntoGraf as well as its properties.

3.2 Entity Linking

The way we perform entity linking here was by using the universal dependency relations found by CoNLL. We look at the 3 nominal core arguments of a sentence, namely, nsubj, obj and iobj. Nsubj is identified as being the syntactic subject of a clause. Obj is the object of a verb. Iobj is an indirect object that is a core argument of the verb but not a subject (nsubj) or a direct object (obj) [10]. Not only do we look at these but also at the relations that are related with these three relations.

For example, in the phrase “George Woodcock descreve com aptidão esse posicionamento libertário”. “descreve” is our verb, “George” our nsubj and “posicionamento” our obj. Instead of looking only at “George” and “posicionamento” to perform the entity linking we also look at ”George Woodcock” and “esse posicionamento libertário”. We do this because, as seen in Figure 3, “Woodcock”, “esse” and “libertário” are associated with our nsubj and obj. This will lead to more robust entities being found in Wikidata as well as a more complete knowledge base.

This relation information is already stored in our knowledge graph. To access it we use a query SPARQL to fetch a sentence and then navigate on it. We start off with query to fetch a single sentence and build its sub-graph. The following query will fetch all the data related to a single sentence and from this data we build a sub-graph that we can navigate. In this query str_id represents the id of a sentence.

```
PREFIX dbp:  <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX defpref: <http://example.pt/framework#>

SELECT ?s ?p ?o
WHERE {
    <""" + str_id + """> defpref:depgraph* ?s .
    ?s ?p ?o .
}
```

From there on it is a matter of navigating between the links and properties of the graph that are shown in Figure 2. We define a root node (the word that is one of the relations previously mentioned) and navigate using the "depGraph" link to fetch all the dependencies of said node and build our partial sentence.

Using Wikimapper [4] we find if these entities exist on the portuguese version of wikipedia or not and fetch their IDs and related names. The final step is to append this ID and names to the sentence using the connection "wikidataId". Figure 5 shows the Virtuoso SPARQL viewer for the following query:

```
PREFIX dbp:  <http://dbpedia.org/resource/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX defpref: <http://example.pt/ontoud#>

SELECT ?s ?p ?o
WHERE {
    defpref:Sentence_148 defpref:depgraph* ?s .
    ?s ?p ?o .
}
```

This query will return the triples related with "Sentence_148" which corresponds to the sentence in Figure 3.

s	p	o
[redacted]/ontoud#Sentence_148	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	[redacted]ontoud#Sentence
[redacted]/ontoud#Sentence_148	[redacted]ontoud#senttext	"George Woodcock descreve com aptidao esse posicionamento libertario :"
[redacted]/ontoud#Sentence_148	[redacted]/ontoud#nextSentence	[redacted]ontoud#Sentence_149
http://ieeta.pt/ontoud#Sentence_148	[redacted]ontoud#depGraph	[redacted]ontoud#word_148_3
[redacted]/ontoud#Sentence_148	[redacted]/ontoud#wikidataId	"Q39769"
[redacted]/ontoud#Sentence_148	[redacted]/ontoud#wikidataId	"Q1181800"
[redacted]/ontoud#Sentence_148	[redacted]/ontoud#wikidataId	"[George]"
[redacted]/ontoud#Sentence_148	[redacted]/ontoud#wikidataId	"[Woodcock, Woodcock_(Pensilvânia)]"

Figure 5 Example of some sentences and the corresponding wikidata IDs found taken from Virtuoso using a SPARQL query.

4 Related Work

In this section we will present the related work. The proposed framework can be used in several areas of work, however, the most directly related ones are annotations (text, semantic), linked data, and since it was our use case, entity linking.

A brief summary of the developments and current state-of-the-art will be presented here in next subsection for each of these topics.

[4] https://github.com/jcklie/wikimapper

4.1 Annotations

Works that follow the Linguistic Annotation Framework which was developed within the ISO TC37 SC4 WG1 are a great starting point for anyone looking into developing a new framework for annotations. This workgroup has put out several documents that can serve as an entry-point, covering most of the language processing field [26]. They have developed standards for morpho-syntactic representation (ISO 24611) [5] and syntactic structures (ISO 24615) [6] and representation of annotated content [27] (ISO 24612) [7] are some of the works published by this work group that are relevant to this work. In general these frameworks provide general requirements that should be followed when creating new annotations framework. For example, the representation of annotated content presents requirements such as expressive adequacy, media independence, semantic adequacy, incrementability, separability, uniformity, openness, extensibility, human readability, processability and consistency [27].

The POWLA (Portable Linguistic Annotation with OWL) [4] is one such framework that follows these guidelines and requirements. It is a framework designed to represent linguistic data structures in a LOD/OWL-compliant way. It is a direct RDF implementation of the PAULA datatypes and annotations. The authors also argue for the benefits of representing annotations in an RDF format over the at-the-time state-of-the-art systems (XML and GrAF).

There are other types of annotations that are not related to semantics but are also relevant to the work developed.

Sentiment annotation consists of looking at a sentence sentiment and assigning it a value of positive, negative or neutral. Assigning these marks is usually done to sentences as a whole but specific words can also convey a negative or positive meaning. As the author of [17] mentions it is not always clear to human annotators the emotion that is being portrayed so there is a need to create more complex annotation schemes that contemplate different parameters especially when considering text such as transcriptions or speeches.

Document Classification is another type of annotation. This one works at a document level instead of sentence or words and the goal is to classify a document into a set of existing categories [14].

4.2 Linked Data

Linked Data and NLP are two highly interconnected fields [7]. The focus of linked data is to create a web of interconnected data which can be queried and analysed in a structured way which pairs well, and even enriches, the goal of NLP which is to better understand and interpret human language [15]. In recent years, the trend has been to use more RDF/OWL based formats due to their interoperability and flexible yet standardized way of representing data [1]. We have seen researchers share more data in this format. For example, CovidPubGraph [24], Protein Ontology [3].

However, NLP is a big field of study and there are many tools which have different and varied outputs making it so that working on large project with many components becomes a complex task due to having to manage all these tools and different standards. There have been attempts to create a standard way to share NLP data by serializing it into RDF, such as the NIF (NLP Interchange Format) [11]. NIF is an RDF/OWL-based format whose goal

[5] https://www.iso.org/standard/51934.html
[6] https://www.iso.org/standard/62508.html
[7] https://www.iso.org/standard/37326.html

was to achieve interoperability and reusability between NLP tools, language resources and annotations. This format specifies several rules to follow as well as defining formal URIs that can be used to maintain consistency between tools and help researchers get over the hurdle of different formats for each tool and even achieve better results due to having the possibility to enrich their data with LOD (Linked Open Data).

There have also been other projects that attempt to convert formats, for example, csv or tsv into ontologies. The W3C keeps a web page of what they call "ConverterToRdf"[8] as well as "RDFImportersAndAdapters"[9] which lists different formats and the tools that can be used for each format, however, this list does not seem to be updated as some of these projects are now offline. This list also does not make any mention of text annotations which is what this work is going for.

The only project that we found who attempted to convert text annotations such as CoNLL into an RDF format was CoNLL-RDF [5, 6]. This is a Java framework where the user provides their own CoNLL file, and the program outputs the RDF representation relying on NIF. This project, however, has some key differences to the framework proposed. The first one being accepting free text / documents instead a CoNLL file. The proposed framework deals with text and serializes it into an ontology with all the CoNLL information in it. This makes it so that the attributes will always be consistent across different corpora. Providing your own file into CoNLL-RDF may break serialization if you use an unknown tagger or there is any error in the generated file. Our work also provides some pre-built SPARQL queries for some useful text processing as well as the option to integrate with different triple-storage systems (Jena, Virtuoso, etc . . .) if there is an available endpoint. With CoNLL-RDF the only output is a RDF or TTL file which you can then upload it into said triple-storage systems, however, if your corpora is too big this isn't an ideal solution.

4.3 Entity Linking

While entity linking is not the main scope of this work it is one of the use cases identified for this work. By using known and established knowledge bases we can improve the quality of our data and of our own knowledge bases. While here we link entities to their corresponding Wikipedia pages, EEL (Entity Extraction and Linking) is a vast field in which a wide variety of approaches are taken.

Representative systme in this area are J-NERD which performs Entity Extraction and Linking with respect to YAGO2/Wordnet and Wikipedia [22], Babelfy which combines Wikipedia WordNet and Babelnet into a semantic network to be used as a multilingual reference knowledge base [19], AIDA-Light focuses on scalability by using a two-step process and uses YAGO2 and Wikipedia [21]. Besides these systems there have also been investments in API/Web Services, for example, IBM has AlchemyAPI[10] which offers a suit of different NLP services (including EEL or Yahoo! Content Analysis API[11] which also includes entity/concept detection. For a more comprehensive study of such systems and a much more in-depth survey of EEL it is recommended to look at [16] and for neural based approaches the survey [29].

[8] https://www.w3.org/wiki/ConverterToRdf
[9] https://www.w3.org/wiki/RDFImportersAndAdapters
[10] https://www.ibm.com/watson/alchemy-api.html
[11] https://developer.yahoo.com/contentanalysis/

5 Conclusion

This paper presents a framework whose goal is to make working with NLP data easier. It offers a common, shareable, unified way of representing data without the need to write immense amounts of custom code. The framework is already capable of creating an RDF-compliant way of working with NLP data, knowledge base completion, text statistics (by navigating the graph) and appending annotations post graph creation. In this version of the framework the requirements that were set at the start were also met. We abide by the FAIR principles [28] by doing the following: it is findable and accessible by being online and available through pypi or github and the dataset being public. It is interoperable by virtue of using a standard format that any triple storage can use. It is reusable due to the data annotation being CoNLL which is a widely known and used format as well not making use of any proprietary frameworks or standards. All the connections from document to the word are intact and navigable both ways. The graphs can be built in-memory or uploaded to a triple storage that accepts SPARQL. We showed that it is extensible and flexible by adding our own annotations after creating the CoNLL graph. We adopt standards and frameworks, such as RDF and CoNLL, that allow integration with already existing software.

While the framework implementation currently only supports CoNLL annotations by default for enriched data, it is already capable of other functions, including (1) Uploading data to a triple-storage as long as there is an available endpoint; (2) Pre-built SPARQL queries to work with the data; (3) Basic entity linking to wikipedia pages.

These functionalities already showcase how powerful the framework is, we also see no issues as to why it could not support other standards, such as POWLA [4] or other types of non-syntactic annotations as long as they are related to either a document, a sentence or words. It is important to make the library adaptable and extensible since enriched data methods are always evolving and the needs of professionals are different depending on their use-cases.

As an example of a use-case we processed a sub-set of the WikiNER dataset for the Portuguese language, we built the knowledge graph with CoNLL annotations and performed a basic form of entity-linking on it. This was a subset with 5121 sentences and it took about 4 hours to process the whole dataset and upload it to Virtuoso[12]. To create the graph in-memory and export as a TTL file it took about 20 minutes. After the graph was created and to show how we can easily extend it with other functionalities we performed entity linking to get a more complete graph. In order to find entities we looked at three main universal dependency relations: obj, obl and nsubj as well as their dependencies to look for entities. When we have these relations and dependencies we use Wikimapper to find, if it exists, the corresponding Wikidata ID and titles, appending them to the sentence it belongs.

These use-cases show the flexibility of the framework and the type of work it can do. We opted to work with a triple-storage system, however it could also be done in-memory but for large documents this is not advised due to memory constraints.

5.1 Future Work

There is still plenty of work that can be done with the current state of the library. The first thing we plan on doing is optimizing the library. We can introduce parallelization when processing large amounts of text. While most of the overhead when working with a triple storage vs in-memory comes from network calls (4 hours vs 20 minutes processing time) for large texts this will be beneficial.

[12] `https://virtuoso.openlinksw.com/`

When it comes to enriched data, adding built-in support to other annotations methods and allow users to choose which ones they want to keep in their serialization is another step that is planned.

Allowing users to export only whichever parts of the graph that they want is also planned if, for example, only part of the graph needs to be shared, or the graph without any custom annotations that might have been done.

References

1 Michael Bergman. Advantages and Myths of RDF. *AI3, April*, 2009.

2 Sabine Buchholz and Erwin Marsi. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June 2006. Association for Computational Linguistics. URL: `https://aclanthology.org/W06-2920`.

3 Chuming Chen, Hongzhan Huang, Karen E. Ross, Julie E. Cowart, Cecilia N. Arighi, Cathy H. Wu, and Darren A. Natale. Protein ontology on the semantic web for knowledge discovery. *Scientific Data*, 7(1):337, October 2020. `doi:10.1038/s41597-020-00679-9`.

4 Christian Chiarcos. POWLA: Modeling Linguistic Corpora in OWL/DL. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, pages 225–239, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

5 Christian Chiarcos and Christian Fäth. CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way. In *International Conference on Language, Data, and Knowledge*, 2017.

6 Christian Chiarcos and Luis Glaser. A Tree Extension for CoNLL-RDF. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7161–7169, Marseille, France, May 2020. European Language Resources Association. URL: `https://aclanthology.org/2020.lrec-1.885`.

7 Philipp Cimiano, Christian Chiarcos, John P. McCrae, and Jorge Gracia. *Modelling Linguistic Annotations*, pages 89–122. Springer International Publishing, Cham, 2020. `doi:10.1007/978-3-030-30225-2_6`.

8 Anne Cocos, Alexander G Fiks, and Aaron J Masino. Deep learning for pharmacovigilance: recurrent neural network architectures for labeling adverse drug reactions in Twitter posts. *Journal of the American Medical Informatics Association*, 24(4):813–821, 2017.

9 Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL: `https://aclanthology.org/D07-1074`.

10 Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL: `http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf`.

11 Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP Using Linked Data. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, pages 98–113, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

12 Martin Hilbert. Big Data for Development: A Review of Promises and Challenges. *Development Policy Review*, 34:135–174, January 2016. `doi:10.1111/dpr.12142`.

13 Mustafa Khanbhai, Patrick Anyadi, Joshua Symons, Kelsey Flott, Ara Darzi, and Erik Mayer. Applying natural language processing and machine learning techniques to patient experience feedback: a systematic review. *BMJ Health Care Inform.*, 28(1):e100262, March 2021.

14 Vandana Korde. Text Classification and Classifiers:A Survey. *International Journal of Artificial Intelligence & Applications*, 3:85–99, March 2012. doi:10.5121/ijaia.2012.3208.

15 Elizabeth D Liddy. Natural language processing, 2001.

16 Jose L Martinez-Rodriguez, Aidan Hogan, and Ivan Lopez-Arevalo. Information extraction meets the semantic web: a survey. *Semantic Web*, 11(2):255–335, 2020.

17 Saif Mohammad. A practical guide to sentiment annotation: Challenges and solutions. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 174–179, 2016.

18 Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. doi:10.5281/ZENODO.1212303.

19 Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014. doi:10.1162/tacl_a_00179.

20 Mark A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015. doi:10.1145/2757001.2757003.

21 Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. AIDA-light: High-Throughput Named-Entity Disambiguation. *LDOW*, 1184, 2014.

22 Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features. *Transactions of the Association for Computational Linguistics*, 4:215–229, 2016. doi:10.1162/tacl_a_00094.

23 Eyal Oren, Knud Möller, Simon Scerri, Siegfried Handschuh, and Michael Sintek. What are semantic annotations. *Relatório técnico. DERI Galway*, 9:62, 2006.

24 Svetlana Pestryakova, Daniel Vollmers, Mohamed Ahmed Sherif, Stefan Heindorf, Muhammad Saleem, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. CovidPubGraph: A FAIR Knowledge Graph of COVID-19 Publications. *Scientific Data*, 9(1):389, July 2022. doi:10.1038/s41597-022-01298-2.

25 Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages, 2020. arXiv:2003.07082.

26 Laurent Romary. Standards for language resources in ISO – Looking back at 13 fruitful years, 2015. arXiv:1510.07851.

27 Laurent Romary and Nancy Ide. International Standard for a Linguistic Annotation Framework, 2007. arXiv:0707.3269.

28 Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, March 2016. doi:10.1038/sdata.2016.18.

29 Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570, April 2022. doi:10.3233/sw-222986.