Teanga: A Linked Data based platform for Natural Language Processing

Housam Ziad, John P. McCrae, Paul Buitelaar
Insight Centre for Data Analytics, National University of Ireland, Galway
[firstname.lastname]@insight-centre.org

Abstract

In this paper, we describe Teanga, a linked data based platform for natural language processing (NLP). Teanga enables the use of many NLP services from a single interface, whether the need was to use a single service or multiple services in a pipeline. Teanga focuses on the problem of NLP services interoperability by using linked data to define the types of services input and output. Teanga's strengths include being easy to install and run, easy to use, able to run multiple NLP tasks from one interface and helping users to build a pipeline of tasks through a graphical user interface.

Keywords: Natural Language Processing, Linked Data, NLP Architecture, NLP Framework

1. Introduction

Natural language processing (NLP) tasks typically consist of many individual components that need to be used together in order to solve real-world problems. However, it is frequently the case that these components are developed independently and thus far integration of these services is far from trivial. The installation of these services can act as a significant barrier to entry for NLP developers and even once developed these pipelines can be opaque and brittle. These issues are of course endemic to software development and until recently could only be solved by integrating all components within a single development model, for example, such as integrating all NLP tools using the Python language as has been done by NLTK(Bird, 2006). An alternative model has arisen in the form of Web services that provide integration between multiple components through clear and defined protocols such as REST. However, Web services have generally not been adopted by researchers or industry, in part due to the fact that the remote nature of the computation can lead to issues with the availability of services (as external services are often down) and the speed of these services (as sending requests to servers creates significant bottlenecks).

In this paper, we propose a new platform called Teanga[1], which aims to achieve the best of both worlds. We use Web services combined by means of novel linked data standards, in particular, JSON-LD, to provide interoperability between services without the need to have particular programming or framework. We also use containerization technology, in particular, Docker, to ensure that these services and the pipelines that are generated by these tools are highly portable and can easily be used at scale. Furthermore, Teanga has an easy to use UI that allows users to visualise their pipelines as well as the progress (or failure) of each service individually.

2. NLP Services

NLP services cover a broad spectrum of tasks, including term extraction, taxonomy extraction, machine translation, sentiment analysis, suggestion mining, automatic summarisation, entity recognition and text classification, which, along with many others, are used widely in many application domains. For example, using entity recognition to extract the names of diseases or medicines from a healthcare database, or using sentiment analysis to extract emotions from social media websites for the purpose of social studies.

NLP services frequently suffer from one or more of the following problems:

- Services are often very focused on a single task, such as part of speech tagging, which is not clearly of use to the end user.
- Many NLP services are still in an early technological readiness level, so there isn't a full application built for them, and/or they don't provide a Graphical User Interface (GUI).
- Many services can't be installed easily, have required dependencies or programming libraries, do not run on all platforms, or lack sufficient documentation.

As a proposed solution to these problems, we present Teanga, an open-source integrated NLP framework. Teanga can apply many NLP tasks either one by one or multiple tasks at once by chaining user-selected services as a pipeline to reduce the amount of manual work required by researchers, in integrating these tools or manually copying results between services.

Teanga will enable researchers to input text in one interface, create a pipeline of required tasks, click a single button, and get all the required results in JSON-LD[2]. We have chosen JSON-LD as the main output format for Teanga as: a) it is easy to use with all programming languages and environments, especially for Web browsers b) provides deep semantics based on RDF and other Semantic Web technologies, c) introduces few overheads to the encoding of the data.

3. Related Work

In the domain of NLP architecture, multiple frameworks, toolkits, and suites have been created, and each of them

[1] 'teanga' ['tʲaŋgə] means 'language' in Irish

[2] https://json-ld.org/

uses a different approach to creating interoperability among all their services, and, by that, reduce the amount of manual work needed to process data. Among these are the LAPPS Grid (Ide et al., 2015) and its Galaxy front-end (Ide et al., 2016), GernEdiT: A Graphical Tool for GermaNet Development (Henrich and Hinrichs, 2010), Language Grid: An Infrastructure for Intercultural Collaboration (Ishida, 2006), and Unstructured Information Management Architecture (UIMA) (Ferrucci and Lally, 2004).

Some problem with the applications of other platforms is that some of them only run on a desktop machine or rely on a platform-specific program, e.g. Eclipse plugins.
For example, in the case of UIMA, it's only a middleware architecture to be taken into account while developing a new NLP tool. For example, it doesn't provide the user with an interface to process data. UIMA also is like GATE when it comes to the complexity of installing and setting up the environment to be used in the development process. Its intended for an expert who is developing an NLP tool and wants to include it in the UIMA environment.

The other platforms do not sufficiently consider user-experience, as user-experience problems can be seen in two parts. The first is installing and running them for the first time, which, for all of them, requires a high level of expertise in a specific environment or programming language, and for some of them, is a time-consuming process. The other part can be seen in the user interface, as some of them don't include a graphical user interface, and users need to run commands from the terminal. Others like GATE created an interface but, even to its developers, (Cunningham, 2002): "The visual interface is complex and somewhat non-standard.". While in the case of LAPPS, their interface seems to be hard to be used by an unskilled user.
A common issue of all of these platforms is the fact that developers have to follow specific standards or guidelines while developing their services before they can be added to the framework to guarantee the interoperability of the platform. While a recent project, OpenMinTeD[3], is working on the standardisation of tools for NLP, these proposals have yet to bear fruit. With Teanga, developers of already existing services can add their services to the platform only by including a configuration file in the container.

4. Linked Data

4.1. Introduction

While data in human-readable formats such as HTML is fully comprehensive for humans, it's still a problem for machines to understand and analyse that data. In the age of big data, where data is expanding exponentially every day, data wrangling still takes up a significant part of the time in the development of an NLP application.
A solution to this as proposed by (Berners-Lee, 2006) gives four key requirements for data to be considered linked data:

- Use URIs to identify all types of data items, for example, if we have a dataset of papers, we would use a unique URI for each paper.
- Make the URIs accessible globally by using HTTP URIs. This way, people can look up the identifiers over the Internet.
- If someone accessed one of the above mentioned URIs, provide useful structured information in RDF.
- Provide links among different data items by including RDF links that point to other URIs; this would help the discovery of related information.

The W3C has proposed a number of standards to help in the creation of linked data, in particular, the Resource Description Framework (RDF), which provides the standard for linked data resources on the Web. In addition, a number of models have been proposed for detailing the semantics of data described in RDF, in particular, the RDF Schema Model (Brickley and Guha, 2000), which allows for inductive reasoning on properties and the Web Ontology Language (McGuinness et al., 2004, OWL), which allows for more sophisticated reasoning about data using description logic.

4.2. Linked Data in Teanga

The use of Linked Data and Semantic Web technologies in applications delivers structured information, which can be used and queried by a flexible and an extensible way to get a better understanding of the data. In particular, the platform exploits the Semantic Web model of data types, to describe the possible format of input and output to services. Since Teanga is designed to deal with any NLP service, and since we can't predict all possible datatypes that may be used by NLP services, we use Semantic Web technologies to define the data types that pass through the services. By doing this, we can let the machine running Teanga understand what data it is processing and how to handle moving it around all the services in a pipeline. The use of Semantic Web technologies will help Teanga, as a platform, to understand the data input and output for each of the services added to the system, and will contribute to creating data interoperability among services to create clear and straightforward pipelines when the user needs to use them.
In particular, there have been a number of models for the representation of linguistic structures used in natural language processing as linked data. The major type of data handled by Teanga is corpus data, and there are a number of models for stand-off annotation of corpora data that have been developed including the NLP Interchange Format (Hellmann et al., 2013) and the Open Annotation format (Sanderson et al., 2013). In addition, more detailed linguistic models such as POWLA (Chiarcos, 2012) as well as specific models such as for parse trees (Chiarcos and Fäth, 2017). In addition, we rely on common models for linguistic categories such as those proposed by ISOcat (Windhouwer and Wright, 2012), now maintained by the CLARIN Concept Registry (Schuurman et al., 2016), or open repositories such as LexInfo (Cimiano et al., 2011)

[3] http://openminted.eu

and OLiA (Chiarcos and Sukhareva, 2015). Finally, we can also use models for representing lexical information on the Web, in particular, the Lexicon Model for Ontologies (McCrae et al., 2012; Cimiano et al., 2016).
As an example, for a machine translation service, we would find that both the source language and target language share the same type because both are referring to a natural language. In this case, we can use an existing type such as `Language` from the LexVo Ontology (de Melo, 2015) and require that values are given as one of the known inputs to this service. We can use JSON-LD aliases to simplify this creating a mapping between the string, e.g., `"en"`, and the URL, e.g., `http://www.lexvo.org/page/iso639-3/eng`). Moreover, for other datatypes such as strings, we can reuse other standards such as XML Schema to define basic datatypes (such as `xsd:string`) or using custom datatypes that can be defined using the OWL vocabulary.
This can be used to join services, for example, if we have a sentiment analysis service that accepts multilingual input and /or output, this service would have an input to enter the text, and an option to select the text language. In this case, the language in the sentiment analysis is of the same type as the languages in the machine translation. If we want to pass data from the sentiment analysis to the machine translation, we can have something as shown in Figure 1.

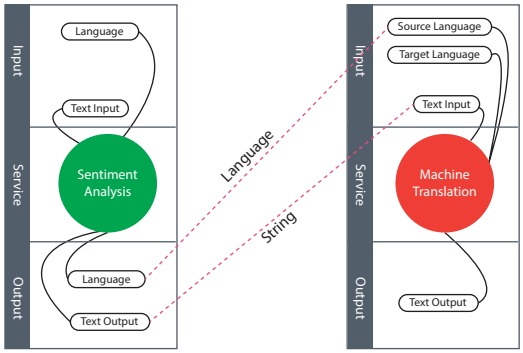

Figure 1: Showing how services share the same types, and how to connect a simple pipeline.

As many of the datatypes used by NLP services are basic or common values such as plain text or language, linked data methods can help the machine understand what type this piece of data is and where to connect that data once we have a pipeline of services. Furthermore, as each of these types is mapped to a URL it is possible to find extra information about it, such as a description, by dereferencing the URL and to provide restrictions, backed by description logic to detect inconsistent combinations of services.

5. Technologies

A key enabling factor for the Teanga platform is the use of the best technologies that exist, in order to enable the service to work efficiently at scale. In particular, we make use of the following open-source tools:

1. Easy-to-use interface by using the Bootstrap[4] library.
2. Stability and maintenance of the Web framework by using the AngularJS[5] library to build the frontend.
3. Using the NodeJS[6] library to run the server and the backend parts.
4. MongoDB[7] is used for data storage, as it uses a JSON-like data structure, which corresponds to our use of JSON-LD files.
5. Using Docker[8] as containerization technology so that the user can download and run Teanga in a simple process of only one step.
6. JSON-LD files for input and output, and to create an interoperable model among the services.

We believe that these technologies will create a highly-performing platform, with an easy-to-use interface, and extremely easy to install and use by third-party users.

6. Design and Implementation

6.1. Introduction

Teanga is designed to host as many NLP services as the users need, for that it should be able to interact with all services and create interoperability among them, especially in the case of running multiple tasks in a pipeline.
Based on that, we had to create a systematic method that helps service developers to add their services to the platform with minimal need to modify their code and architecture to make it compatible with Teanga beforehand. This method must provide three outcomes at once by letting the service provider add a configuration file to the corresponding directory in the platform, with outcomes:

1. Defining the input parameters and types of the service, to make it clear for the interoperability model to understand the service input.
2. Defining the output parameters and types of the service, which makes the service return a compatible output with the platform.
3. By combining the two items above, generating the corresponding user-interface (UI) elements to the user for better interaction with the services.

By using a built-in system to generate the UI, the user or the service provider does not have to write specific code for each service, but rather, by using a format that is widely accepted among Semantic Web applications, the system would create the UI.

6.2. Parameters and Data Types

Any application that interacts with the user would have relevant specific fields in a form or a wizard to collect the data from the user, which can be processed later. These forms and wizards have to interact with the user to collect

[4] `http://getbootstrap.com/`

[5] `https://angularjs.org/`
[6] `https://nodejs.org/`
[7] `https://www.mongodb.com`
[8] `https://www.docker.com`

all corresponding data systematically that the system needs for accurate results.

After analysing many NLP services and considering other cases where we needed to build interfaces for services and applications, we found that we need to build a system that uses semantic structure, JSON-LD files in our case, to generate a user interface automatically and to create interoperability with other services in the platform.

For this, we made our system build an interface depending on the JSON-LD file that describes the service. In this file, we describe the service in details, and then we move to each input and output parameter in it and describe it using linked data technologies.

6.3. Adding a Service to Teanga

Since Teanga is a platform that accepts adding new services to the system, we implemented a simple way to add or remove a service. The method is to add JSON-LD files to the system that represent the service description schema and the context of service components.

The Teanga's file hierarchy contains:

- Ontology: this directory contains the Teanga ontology file, which holds the Teanga ontology that describes all the properties for the services.
- Schema: this directory holds the JSON-LD files that describe the services, one for each service.
- Webapp: This directory has the web applications files and directories, such as the directory named assets, which holds all the static files like images and styles.

To add a new service, we simply need to add a description file for it in the "schema" directory, and to remove the service, we just remove the JSON-LD description file from the directory. As modifying the platform for each service takes some effort, we also scan all loaded Docker images for a scheme directory and load all valid files automatically.

6.4. Containerization Technology using Docker

We're implementing containerization technology in Teanga using Docker to help users overcome the problem of installing many dependencies to run a program. We chose Docker for this task because Docker provides high portability due to a lightweight virtualization with almost no overhead, and that enables running multiple containers on a single machine and adding a layer of abstraction for Teanga. In this case, Teanga would be isolated from other applications on the server, which guarantees performance for the platform and its components inside their containerization environment.
The idea is that we're preparing Teanga inside a downloadable Docker image that holds the Teanga web application and all the libraries, dependencies, and databases elements that it needs to run. This means that the user will only need to download one image that contains everything related to Teanga. By installing a Docker client on their servers, and running the image file, they would have Teanga up and running inside a Docker container by just one step. This is what makes Teanga very easy to install and run. Currently, the Teanga platform is an individual image and the services are in other images; the users can pull them and run them on their servers. It is expected that any third party can easily extend Teanga by simply creating a Docker image and providing an appropriate service description as described above and this image can be published at DockerHub[9] or internal repositories, ensuring that a new service can be added by simply giving the name of its container.

6.5. Error Control

Even perfectly written software can fail due to configuration errors, and most of the software developed by NLP researchers is not developed to industry standards. As such it is a vital goal of the platform to handle failures within services gracefully and show these errors clearly to the user so they may be properly debugged. Teanga can handle the following errors:

- If a service returns an error message, Teanga will display the error message to the user contained inside the results tab.
- If a service fails or has a server error, which usually stops the service and causes it to crash and display default servers messages, Teanga can contain that and return a corresponding message.
- If a service crashed and it returns blank data, Teanga would display an error message that the service is returning an empty message.

We're still working on improving error handling with our continuous experiments with adding services.

6.6. The Interface

We will use an example to describe how the interface works: extracting suggestions from a Spanish text, using a suggestion mining application developed for English.
To run a task in Teanga, we start by entering the text as shown in Figure 2.

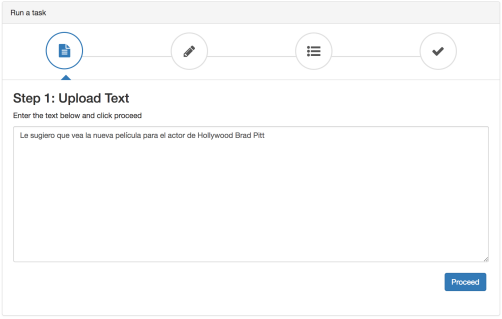

Figure 2: Uploading the text

In Figure 2, we can see that the input text is Spanish, and we need to translate that to English to be ready for the next service, which is suggestion mining.

[9] https://hub.docker.com/

Then, we drag the required services to the service area as in Figure 3. Notice the bottom area, which shows all the available services in the platform represented as a bubble. This means that the system contains three service description files.

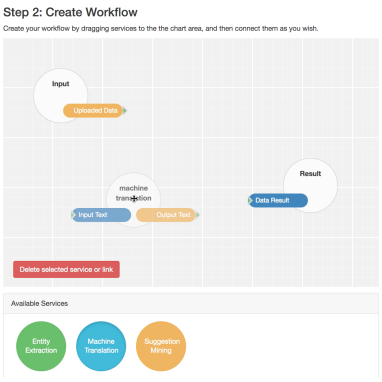

Figure 3: Placing a service in the service area.

Connect the services as shown in in Figure 4. Connecting the service can be done by dragging the points using the mouse.

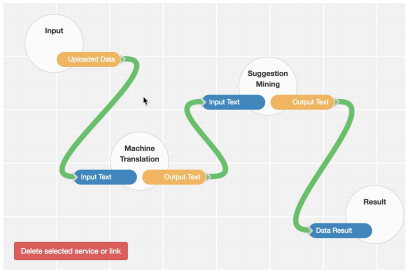

Figure 4: the final workflow graph

Some services do not only have input and output parameters, but they have options as well, e.g. machine translation has options for source and target language. In this step, we select the corresponding options for some of the services as in Figure 5 by selecting the service using the mouse, and then the options sidebar will show up automatically.

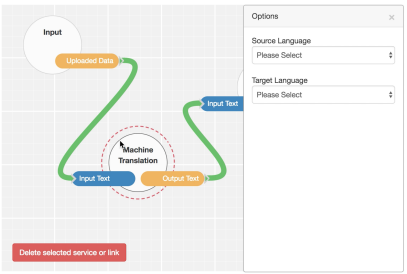

Figure 5: the options panel

And finally proceed to the results page, Figure 6.

Figure 6: The results

6.7. Smart Features in The Interface

As a part of enhancing the user experience, we added two features to Teanga as follows:

1. The predefined tasks, which cover the common experiments that are used in NLP research, we put in a list to choose from, which the system will use to create the whole workflow, saving multiple steps for the user. For example, when the user selects suggestion mining on multilingual text, the system will place the machine translation and the suggestion mining services in the graph, connect them, and then the user only has to choose the languages for the machine translation service.

2. The ability to save and load a saved workflow, in case the user needs to rerun the same experiment in future on a different data set. They can upload the data, and just load the workflow in one click.

7. Testing

Since Teanga is a specialised software, we tested by asking NLP researchers to apply it to their own data. We asked three NLP researchers, two employed at a university and one at a large multinational technology firm, with one of the researchers holding a PhD in NLP.
All the testers agreed that Teanga is well designed and easy to navigate. Yet, they provided 20 comments, of which 13 were easily fixed, while some were kept for future work. The proposed features include:

- Automatic source language detection.
- Downloading the results in JSON-LD files.
- Adding related information to the results page, e.g. how long the experiment took.
- Adding the ability to upload data files to the system instead of direct text entry. These data files can be in different formats.

In addition, some of the comments made us rethink our approach, as in:

- Implementing better ways to choose options for services.
- Changing the view in the last step for the JSON-LD output and adding the ability to dismiss some of the results.
- Enabling the user to choose how to display results.

8. Conclusion

Previous NLP platforms had multiple issues related to their architecture (ability only to apply one task at a time), user experience (no graphical user interface), or interoperability. As a solution for these problems, we developed Teanga, a Linked Data based NLP framework, which aims to enable easily constructed, flexible and high-performance NLP pipelines. We use Linked Data technologies, including JSON-LD files, to define the types of the data that pass through services to deliver structured information that can be used and queried in a flexible and an extensible way and to get a better understanding of the data. We further use a containerization technology (Docker) to ensure that the platform can be easily distributed and installed by end users.

9. Bibliographical References

Berners-Lee, T. (2006). Linked Data - Design Issues. Blog post at www.w3.org.

Bird, S. (2006). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Brickley, D. and Guha, R. V. (2000). Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000. W3C Recommendation, World Wide Web Consortium.

Chiarcos, C. and Fäth, C. (2017). CoNLL-RDF: Linked corpora done in an NLP-friendly way. In *International Conference on Language, Data and Knowledge*, pages 74–88. Springer.

Chiarcos, C. and Sukhareva, M. (2015). OLiA–ontologies of linguistic annotation. *Semantic Web*, 6(4):379–386.

Chiarcos, C. (2012). POWLA: Modeling linguistic corpora in OWL/DL. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 225–239.

Cimiano, P., Buitelaar, P., McCrae, J., and Sintek., M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):29–51.

Cimiano, P., McCrae, J. P., and Buitelaar, P. (2016). Lexicon Model for Ontologies: Community Report. Technical report.

Cunningham, H. (2002). GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

de Melo, G. (2015). Lexvo. org: Language-related information for the linguistic linked data cloud. *Semantic Web*, 6(4):393–400.

Ferrucci, D. and Lally, A. (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using linked data. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 98–113.

Henrich, V. and Hinrichs, E. (2010). GernEdiT: A Graphical Tool for GermaNet Development. In *Proceedings of the ACL 2010 System Demonstrations*, pages 19–24. Association for Computational Linguistics.

Ide, N., Pustejovsky, J., Cieri, C., Nyberg, E., DiPersio, D., Shi, C., Suderman, K., Verhagen, M., Wang, D., and Wright, J. (2015). The Language Application Grid. In *International Workshop on Worldwide Language Service Infrastructure*, pages 51–70. Springer.

Ide, N., Suderman, K., Pustejovsky, J., Verhagen, M., Cieri, C., and Nyberg, E. (2016). The Language Application Grid and Galaxy. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 457–462, Paris, France, may. European Language Resources Association (ELRA).

Ishida, T. (2006). Language Grid: An Infrastructure for Intercultural Collaboration. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, pages 1–5. IEEE.

McCrae, J., de Cea, G. A., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., and Wunner, T. (2012). Interchanging lexical resources on the Semantic Web. *Language Resources and Evaluation*, 46(6):701–709.

McGuinness, D. L., Van Harmelen, F., et al. (2004). OWL Web Ontology Language Overview. W3C recommendation, World Wide Web Consortium.

Sanderson, R., Ciccarese, P., Van de Sompel, H., Bradshaw, S., Brickley, D., Castro, L. J. G., Clark, T., Cole, T., Desenne, P., Gerber, A., et al. (2013). Open Annotation data model. W3c community draft.

Schuurman, I., Windhouwer, M., Ohren, O., and Daniel, Z. (2016). CLARIN concept registry: the new semantic registry. In *Selected Papers from the CLARIN Annual Conference 2015*, pages 62–70.

Windhouwer, M. and Wright, S. E. (2012). Linking to linguistic data categories in ISOcat. In *Linked Data in Linguistics*, pages 99–107. Springer.