Agreement number: INEA/CEF/ICT/A2017/1565710

Action No: 2017-EU-IA-0136

Deliverable 4
Sustainability

Version No. 1

30/04/2020

Document Information

Activity:	Activity 4: Sustainability
Deliverable number:	D4
Deliverable title:	Dissemination plan
Indicative submission date:	30/04/2020
Actual submission date of deliverable:	1/05/2020
Main Author(s):	Dan Tufiș
Participants:	Vasile Florian Păiș, Maria Carp, Radu Ion
Version:	V1

History of Versions

Version	Date	Status	Name of the Author (Partner)	Contributions	Description/ Approval Level
V1	30/04/2020	Completed	RACAI		

EXECUTIVE SUMMARY

The report presents the sustainability plan and the proposed solution for its implementation. The objective of this Activity is to build a single access point service that will integrate 7 language specific processing pipelines. The proposed platform for seven language processing pipe-lines is based on the concept of containerization in order to ensure time-persistence of the language specific implementations against the OS updates and other changes between hosts and environments. The structure of a processed document is the same irrespective of its language and an example of a processed sentence is exemplified for one of the consortium languages.

Access platform for sustainable data processing for Marcell project

Contents

1. Goals

Sustainability of the project has two aspects: continuous feeding of the repository with new incoming data and ensuring time-persistence and low maintenance times of the processing pipelines against the OS updates and other changes between hosts and environments.

For the data collection sustainability, we opted to leave the individual crawlers out of the language processing chains. Their complexity and implementation depend on the data structuring at each source provider, the access rights granted to the project partner, the format of the published documents, the possible necessary conversions into raw texts, the rate of data updates, among the others. The new data may be sent to a partner by the owners based on a contractual agreement, or may be periodically (e.g. monthly) downloaded by partners from some open-access sites. Irrespective of the data acquisition procedure, the new text data shall be archived and sent (by each partner) to the single-access point language processing platform. On the platform, automatically there will be activated the corresponding language dependent processing flow.

The second aspect of sustainability refers to containerisation of the language specific processing flows. The Marcell primary text processing and annotation platform must ensure the processing of new documents that will become available in the future from each of the consortium members. For this purpose, it must be resistant to operating system updates as well as unforeseen configuration changes at operating system level. Furthermore, it must accept raw texts in the specific languages of the project and produce output in the common CoNLL-U Plus format agreed between consortium members.

Each document begins with a newdoc marker holding the file id (# newdoc id = ro.legal). Each sentence in a document is labelled by a unique ID (# sent_id = ro_legal.4), is followed by the text of the respective sentence (# text = …) and then the vertical analysis, UD-like with 14 columns, of the tokens occurring in the sentence.

The structure of a line is the following, the line fields being tab-separated:

<ID><FORM><LEMMA><UDPOS><XPOS><FEATS><HEAD><DEPREL><_><_><NER><NP>
<IATE><EUROVOCID><EUROVOCMT>

1. ID: Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes (decimal numbers can be lower than 1 but must be greater than 0).
2. FORM: Word form or punctuation symbol.
3. LEMMA: Lemma or stem of word form.
4. UPOS: Universal part-of-speech tag.
5. XPOS: Multext East morpho-syntactic tag (MSD)
6. FEATS: List of morphological features MSD
7. HEAD: Head of the current word, which is either a value of ID or zero (0).
8. DEPREL: Universal dependency relation to the HEAD (root iff HEAD = 0)
9. Not used
10. Not used
11. NER: the BIO format annotation of the current token if it is part of a name entity, (‘_’ otherwise)
12. NP: the BIO format annotation of the current token if it is part of a NP, (‘_’ otherwise)

13. IATE: the annotation of a IATE term by the language-independent code if it is (part of) a IATE term, ('_' otherwise)
14. EUROVOCID: the EUROVOC label if it is a term, ('_' otherwise)
15. EUROVOCMT: the MT label in the EUROVOC thesaurus if it is a term, ('_' otherwise)

Underscore (_) is used to denote unspecified values in all fields. Trailing underscores are not shown.

An example of the information generated for a sentence (in Romanian) is shown in the next figure:

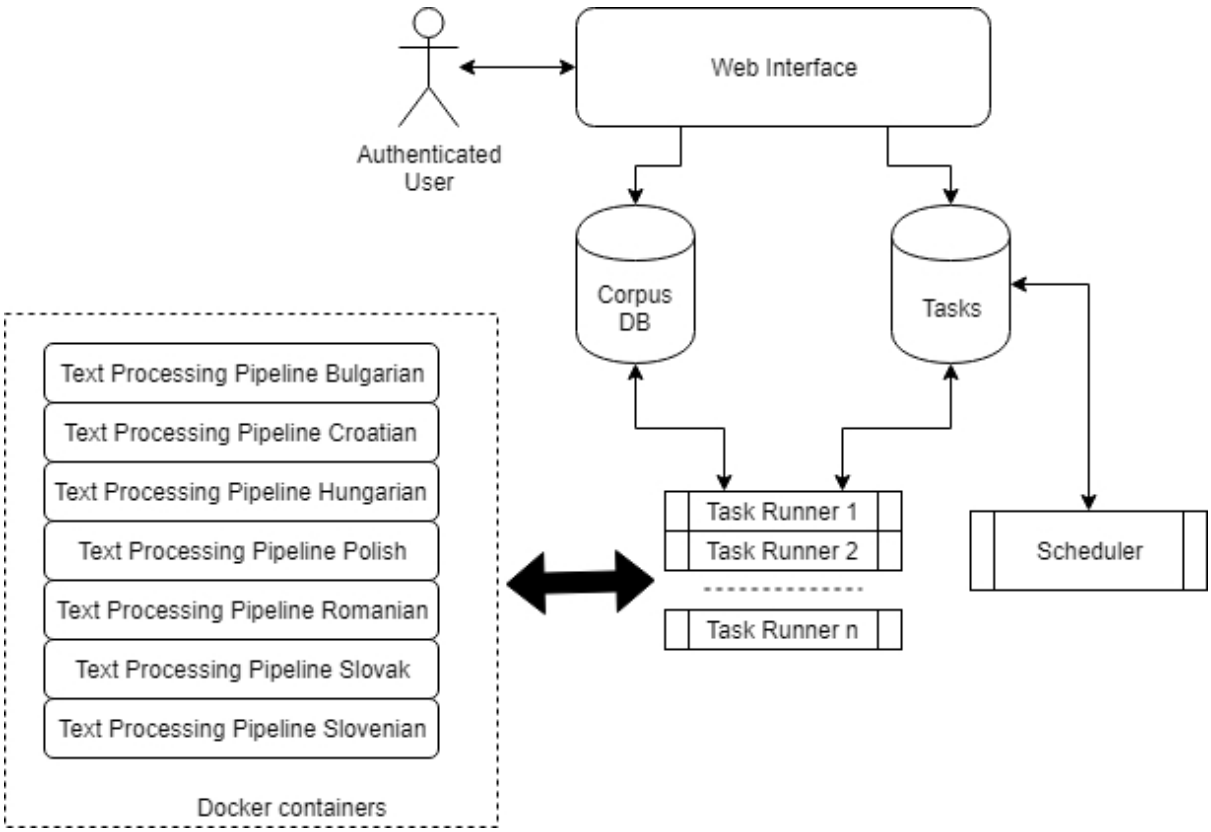
```
# sent_id = ro_legal.4
# text = MONITORUL OFICIAL nr. 799 din 18 septembrie 2018
1 MONITORUL monitor NOUN Ncmsry Case=Nom|Definite=Def|Gender=Masc|Number=Sing 0 root _ _ B-ORG B-NP 10:1394636;11:3522817 10:3206;11:3221,7206,7231
2 OFICIAL oficial ADJ Afpms-n Definite=Ind|Degree=Pos|Gender=Masc|Number=Sing 1 amod _ _ I-ORG I-NP 11:3522817 11:3221,7206,7231
3 nr. nr. NOUN Yn Abbr=Yes 1 nmod _ _ O I-NP _ _
4 799 799 NUM Mc _ 3 nummod _ _ O I-NP _ _
5 din din ADP Spsa AdpType=Prep|Case=Acc 7 case _ _ O O _ _
6 18 18 NUM Mc _ 7 nummod _ _ B-TIME B-NP _ _
7 septembrie septembrie NOUN Ncms-n Definite=Ind|Gender=Masc|Number=Sing 1 nmod _ _ I-TIME I-NP _ _
8 2018 2018 NUM Mc _ 7 nummod _ _ I-TIME I-NP _ _
```

The IATE and EUROVOC labels are prefixed with a number counting the terms in the current document. For multi-word terms this counter allows correct term identification. In the exemple above, MONITORUL (en.: the instructor) is the 10th term in the current document, identified by the IATE code 1394636 and EUROVOC category 3206 (Education and Communication). However, MONITORUL OFICIAL (en.: THE OFFICIAL MONITOR) is a different term (the 11th) with IATE code 3522817 for which three EUROVOC categories applies: 3221, 7206, 7231.

After the language specific processing the documents are archived and sent to the next processing hub, the multilingual clustering and comparable documents semantic alignment phase.

The output of these processing services, together with the raw data will be sent to the ELRC repository of language resources.

2. General platform diagram



3. Platform requirements

3.1. Containerized language processing pipelines

In order to ensure uninterrupted processing of future materials related to the Marcell project, the platform must ensure processing pipelines are isolated from external interferences such as: operating system updates, operating system reconfigurations, changes in other processing pipelines, updates in used libraries. Keeping this in mind, each text processing pipeline will be provided in the form of a Docker container, allowing the transformation of a text file into a common format annotated document.

By using Docker[1] containers and embedding all the necessary runtime libraries, independence of any uncontrolled external updates at OS level is achieved. The single access point will receive an archive with text documents and the language ID of the documents. The contents of the archives will be transferred to the specific “dockerized” language processing chain. Each of the language processing flows has the same input-output behaviour: receives a collection of text documents in the specific language and outputs a collection of the processed documents, which can be downloaded either manually or automatically, at automatically formatted URLs. In case of improvements to processing pipelines for certain languages, consortium members have the ability to provide an updated container which will replace the previous one, without interfering with other processing flows. Furthermore, the use of containerization enables scalability of processing resources with the

[1] https://www.docker.com/

number of new documents, by instantiating as many containers as needed to allow for efficient parallel processing.
The sustainability workflow will have a web-based entry interface where the partners will be able to:

- Verify the progress of the text data annotation
- Download the results of the annotation if desired
- Upload archives of text files if also desired

When faced with new raw text documents, the platform will start the corresponding docker instance(s) and invoke the processing flow.

Containers must offer a web-service like interface, in REST format, callable from the platform.

The format for such a call should be:

Endpoint: *http://<CONTAINER_ADDRESS>:<PORT>/annotate*

CONTAINER_ADDRESS and *PORT* are automatically associated by the platform using the docker interface

Parameter: *file*, will be the actual text file to be processed, sent via HTTP POST method

The result of the call should be the resulting CoNLL-Plus format corresponding to the received text file. In case of error, the response should be either empty or an error message (plain text message, provided without a starting '#' sign).

This allows for the platform to start any number of containers for each individual language pipeline in order to allow for efficient parallel processing of text documents.

Containers should be completely self-contained. No external resources should be used. This includes any language models or additional data files, that should be stored inside the container image.

3.2. Secure data upload interface

The platform will provide an interface for uploading new data files. This will allow for ZIP archives containing raw texts to be uploaded in the platform in a secure way.

Security will be ensured by using specific user accounts and HTTPS encryption during authentication.

Following the upload, the platform will automatically begin extracting the files from the archive and will allow the user to explore the extracted files.

3.3. Task scheduling and monitoring

Each operation within the platform is expected to take a long time (directly proportional with the number and size of uploaded texts). In order to prevent issues arising from possible power outages or other users performing additional annotations, each operation will be scheduled and executed by a background task processing engine. This will allow for each operation to be parallelized and will ensure resuming in case of unexpected system crashes or restarts.

The user interface will allow the user to monitor the task status. It is envisaged to allow at least for the following states:

- NEW : the task was just created
- SCHEDULING: the task was picked up by the scheduler
- SCHEDULED: the task is waiting for runner processes to become available
- RUNNING: the task is currently running on one or more runner processes
- DONE: the task has finished
- ERROR: unexpected errors during task execution

Additionally, the task engine will provide information regarding the user who started the task, start time and end time, with the possibility of adding a custom human readable description to the task.

Actual processing will happen inside the docker containers specific to each language as orchestrated by the runner processes. When all the files are processed, the task status will be changed to “DONE”.

3.4. Secure data export interface

Following the completion of an annotation task and obtaining the CoNLL-U Plus documents, data must be made available for export. This is achieved via a ZIP creation task that will add all the generated documents into a ZIP file. Afterwards, the generated ZIP archive can be downloaded by authorized users.

The archive creation process will make use of the same task engine described above. This will ensure background creation of the ZIP file.

The resulting archive will be stored in the platform and available for future downloads if needed.

3.5. Data exploration

The platform will allow simple exploration and visualization of data (both raw texts and annotated documents). This will allow users to have a quick look on the resulting documents even during the task processing time. Visualized documents can be saved individually, in order to allow checking of interoperability with other processing chains if needed.

This implementation will be based on simple data grids.

3.6. Platform technologies

In order to ensure durability of the platform’s access point as well as independence of the operating system, it will be implemented using a server-side scripting language in the form of PHP and complemented with modern browser technologies in the form of HTML5, JavaScript, CSS.

The platform’s access point will not maintain any connection with a specific underlying operating system, thus enabling it to be containerized if needed, similar to the text processing pipelines. Nevertheless, since it uses PHP it may not be required to containerize it as well.

3.7. Data storage

In order to reduce dependencies on other potentially complex systems, such as database management systems, the platform will store all relevant data inside text-like files on the file system. These files include raw texts (for the input files), CoNLL-U Plus files (for annotated files), JSON files (for internal data structures). This behavior will contribute to the sustainability goal, by keeping dependencies to a minimum and not requiring additional configuration or maintenance of other systems.