This project has received funding from the European Union's Horizon 2020 research and innovation programme under **grant agreement No 780355**
Ref. Ares(2020)253033 - 15/01/2020

FANDANGO DELIVERABLE

Deliverable No.:	D3.3
Deliverable Title:	Preprocessing set of tools
Project Acronym:	Fandango
Project Full Title:	FAke News discovery and propagation from big Data and Artificial iNtelliGence Operations
Grant Agreement No.:	780355
Work Package No.:	3
Work Package Name:	Data gathering and preprocessing for Data Lake population
Responsible Author(s):	UPM
Date:	30.12.2019
Status:	V1.0
Deliverable type:	DEMONSTRATOR
Distribution:	PUBLIC

REVISION HISTORY

VERSION	DATE	MODIFIED BY	COMMENTS
V0.1	11.11.2019	Hernández-Peñaloza Gustavo (UPM)	First draft.
V0.2	03.12.2019	Hernández-Peñaloza Gustavo (UPM), Martín-Gutiérrez David (UPM)	Add preprocessing architecture and methods.
V0.3	10.12.2019	Hernández-Peñaloza Gustavo (UPM) Martín-Gutiérrez David (UPM)	Add Figures, tables and table of contents properly linked
V0.4	11.12.2019	Martín-Gutiérrez David (UPM)	First consolidated version
V0.5	12.12.2019	Magaldi Massimo (ENG), Stalidis Panagiotis (CERTH)	Internal Review
V0.6	26.12.2019	Martín-Gutiérrez David (UPM)	Add modifications and improvements based on the internal review
V0.7	27.12.2019	Martín-Gutiérrez David (UPM)	Final review
V1.0	30.12.2019	Martín-Gutiérrez David (UPM)	Submitted version

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ABBREVIATION	DESCRIPTION
H2020	Horizon 2020
EC	European Commission
WP	Work Package
EU	European Union
REST API	RestFul API
ML	Machine Learning
DL	Deep Learning
NER	Named Entity Recognition
CNN	Convolutional Neural Networks
RNN	Recurrent Neural Networks
TLD	Top-Level Domain
SaaS	Software as a Service

EXECUTIVE SUMMARY

This deliverable is an official document of the FANDANGO project funded by the European Union's Horizon 2020 (H2020) research and innovation programme under grant agreement No 780355. It is a public report that describes the technical requirements for FANDANGO platform and services.

This deliverable reports the strategies adopted by FANDANGO to preprocess the information collected (crawled) from diverse sources, as well as the mechanisms to clean, organise and classify the websites ingested into the FANDANGO ecosystem. Along this deliverable, the manner as pages are parsed and its most relevant fields are obtained, estimated and stored according to the data model defined in Task T3.1.

This deliverable is split into four main sections: the first two sections are devoted to briefly introduce the problem addressed in this deliverable as well as the related work that is being doing by the scientific community when preprocessing in any data mining application.

In Section 3, the architecture of the Preprocessing service is described including the two main procedures that have been developed to cover both a batch and a streaming processes. These two methodologies are denoted as Offline and Online procedures. Additionally, this section is focused on the integration of the services via virtualization for being suitable in Big Data ecosystems.

In Section 4, the different methods and techniques employed for preprocessing an article are explained including intuitive examples and diagrams to promote the understanding and the necessity of this module in the rest of the Project of FANDANGO. Moreover, along this section all the libraries, packages and technology involved in the development will be also remarked with more details.

Finally, Section 5 is devoted to summarize the principal outcomes of this deliverable including more improvements for further implementations. This module is under continuous development and improvement, as new cases appear all the time. The code and repository to use FANDANGO preprocessing tool is released under MIT license and can be found in a Github associated to the project. https://github.com/fandangOrg/fandango/tree/upm/code/Preprocessing%20service.

1. INTRODUCTION

FANDANGO's final goal is to provide journalists and press agencies with an integrated information verification system, able to analyse different typologies of data correlated with news such as images, video, text and metadata (source, author, etc). To achieve such a goal, several different tasks and functionalities need to be developed and integrated to provide a final tool where users can navigate and obtain useful results.

As it is described in D3.2, a large collection of crawlers has been developed to solve the first stage in the FANDANGO pipeline: retrieving the information from different websites and data silos. However, it is well-known that there is not a unique schema or pattern in such sources to normalize all the data, so that it makes this process tedious and complicated. In addition, some data may not be available in some cases or may have a wrong format, making it useless for a posterior analysis.

Moreover, during the analysis of both user and technical requirements, the FANDANGO data model was defined according to the most relevant entities involved in the generation and propagation of a certain article or report including text, images, videos, authors and the publisher among others. Consequently, an additional step is crucial to map directly from the format and schemas provided but the crawlers and the data model proposed for FANDANGO. As a result, a preprocessing stage is needed to provide such mapping in an adequate manner. Along this deliverable, all the methods and techniques employed for such purpose will be described.

According to the task, FANDANGO shall provide mechanisms to: *(a) ingest data into a populated database (b) preprocess the information inserted to extract meaningful features to be used by all modules in FANDANGO (c) Perform an analysis on each modality (text, images, video, entities) to make predictions on the reliability of the information ingested by (d) profile the mentioned entities, serving scorings that will be used as relevant indicators for the decision making process*. In this sense, this task is fully devoted to item (b), by taking information crawled, and making cleaning operations to have the information in a legible, normalized manner to be used by the rest of modules in FANDANGO.

2. RELATED WORK & STATE OF THE ART

FANDANGO seeks to use well-known data mining techniques in different modalities such as text mining, image and video mining or graph mining in order to generate knowledge regarding a certain piece of news or article.

A known problem with data gathered from the internet is that many of these data is unstructured or semi-structured. Thus, preprocessing methods plays a key role in data mining techniques and applications to reduce errors during the posterior analysis of the information throughout Machine or Deep Learning models.

In many research publications, authors have attempted to analyse the impact of using preprocessing techniques in final applications. In (Srividhya, 2010), the authors emphasize the preprocessing methods that are used in text categorization to remark its importance in this topic. Later on, in (Uysal, 2014) authors highlighted the impact of the preprocessing on text classification where the experiments were performed using different domains and languages. After applying different techniques, they demonstrated the improvement of the models when classifying text documents.

Moreover, in (Vijayarani), the authors make a full review regarding all the main preprocessing methods that are considered in different data mining fields including information retrieval, text categorization or Natural Language processing.

In (Denny, 2018), authors analysed the implication of preprocessing decision in the political science domain including text-as-data research. They remark that supervised literature is not helpful enough as a source of advice and they introduce a statistical procedure that investigates the sensitivity of findings under alternate preprocessing regimes. Such technique promotes the researcher's substantive understanding of a problem by providing a characterization of the variability changes in preprocessing decisions may induce when exploring a certain dataset.

Finally, in one of the chapters included in (Anandarajan, 2019), the authors described the importance of a preprocessing step before any performance of Machine or Deep learning approaches and a novel text preprocessing approach for information retrieval is also proposed as part of the research.

3. ARCHITECTURE OVERVIEW

Along this section, a full description of the preprocessing tools will be presented. As it has been explained in previous deliverables, the different components of the FANDANGO platform have been developed as services to improve the scalability and the flexibility of the platform by optimizing the different processes that each component provides.

Moreover, the implementation of the preprocessing component has been divided in two main functionalities: offline and online processes. The former is related to the batch scenario, where the data is collected, processed and passed through Kafka. On the other hand, the latter is used in the user interface scenario, where the backend of the interface launch a request to this service when the article has been collected by the crawlers.

As expected, the result of both offline and online scenarios is the same: a pre-processed article with the data mapped properly to the FANDANGO data model described in previous Deliverables.

In FIGURE 1, a block diagram with the different components that are considered in the Preprocessing service is shown. As the figure shows, the two services share the same Preprocessing methods so that, the output will be the same, whereas the technology involved during the process is distinct according to the General Architecture of the FANDANGO-Project.

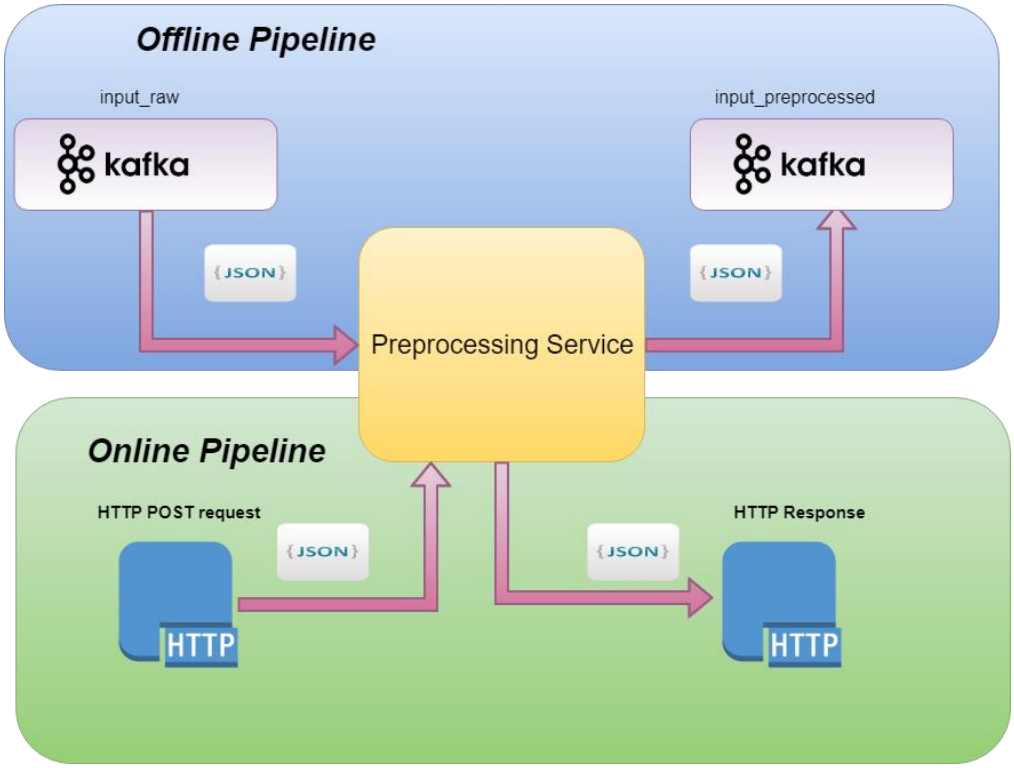

FIGURE 1 GENERAL SCHEMA OF THE PREPROCESSING SERVICE

3.1. SERVICE INTRODUCTION

One of the main objectives of the preprocessing lies in the improvement of the performance of the different analysers that are developed in WP4 including: media-content analyser including image and video, the text analysers and the graph analyser. All of the aforementioned services depend directly from the preprocessing stage, therefore, many of the processes that will be explained along this deliverable will affect directly to WP4 analysers.

As a result, an exhaustive analysis was performed before developing the service in order to understand which features are necessary to be cleaned, pre-processed or restructured and also how this can be achieved automatically using artificial intelligence and big data techniques to speed up the processes. A general schema of the different technologies and features employed in this service is depicted in FIGURE 2.

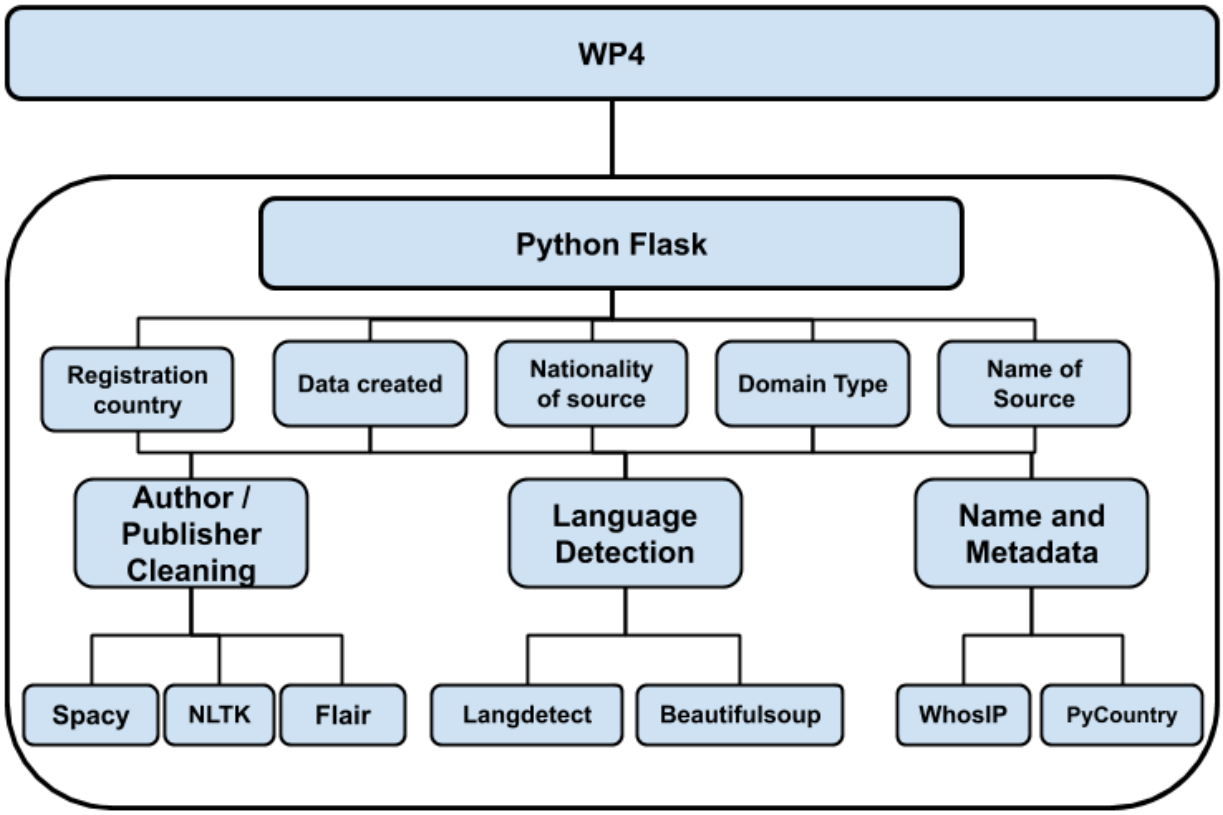

FIGURE 2 GENERAL BLOCK DIAGRAM SHOWING THE RELATIONSHIP, THE TECHNOLOGIES AND THE IMPACT OF THE PREPROCESSING SERVICE WITH WP4

3.2. OFFLINE PREPROCESSING OVERVIEW

It was previously introduced the necessity of a batch process to clean, organize and prepare the data collected from the crawlers for mitigate issues in the posterior analysis stage. In Big Data terminology, a batch processing consists in automatically analysing multiple transactions in groups or batches. Moreover, no user interaction is required once the batch processing has already started. Consequently, this scenario arises as a potential candidate in FANDANGO, where many articles are ingested to be analysed by the different artificial intelligence services implemented.

The offline service interacts with Kafka in order to receive, process and put back all the messages stored in a certain Kafka topic. In particular, the offline service reads the messages ingested in the Kafka topic *"input_raw",* where the raw articles were stored. Then, once the preprocessing is finished, the system puts the pre-processed article in a new Kafka topic named as *"input_preprocessed",* where the rest of the analysers can retrieve it and perform their corresponding procedures. The following table summarizes the stages of the offline or batch preprocessing method.

STAGE NUMBER	DESCRIPTION
1	Reads document from Kafka topic *input_raw*
2	Document is pre-processed
3	Puts document to Kafka topic *input_preprocessed*

TABLE 1 OVERVIEW OF THE STAGES INVOLVED IN THE OFFLINE PREPROCESSING PROCESSING

3.3. ONLINE PREPROCESSING OVERVIEW

In the online scenario, the procedure is quite similar to the offline where the main difference is that in this case, the transactions are performed during the user-interface and not automatically as in a batch processing. Moreover, in the online procedure, Kafka is not used as the intermediate between the different components. It is replaced by HTTP requests that are performed by the user interface. Therefore, in this case the communication of the service is established between the user interface and service via Rest API. The following table summarizes the stages of the offline or batch preprocessing method.

STAGE NUMBER	DESCRIPTION
1	User interface makes an HTTP requests with the raw document
2	Document is pre-processed
3	Yields pre-processed document to the user interface

TABLE 2 OVERVIEW OF THE STAGES INVOLVED IN THE ONLINE PREPROCESSING PROCESSING

3.4. REST API

To manage the different transactions of the service, a RESTful API has been developed for this service. Moreover, a Swagger has been included to provide the documentation of all the requests, as well as the data models involved in such service. In FIGURE 3, we present the documentation provided by the Swagger

regarding this service. As expected, there are two POSTS requests associated to both offline and online procedures.

In both cases, the preprocessing service yields an organized and cleaned document which has the same data model described in previous Deliverables.

Furthermore, FIGURE 4 shows the specifications of the POST request regarding the offline procedure. This request starts the batch processing and returns a status code of 200 whether the process has started properly. The starting process will be explained in the next section where the integration of the service is described.

On the other hand, FIGURE 5 depicts the specifications of the POST request regarding the online procedure. In this case, the request needs a body corresponding to the raw document or article in the proper format provided by the Crawlers. The output of the request consists in the cleaned document with the adequate data model.

We have included an example of the data model proposed for the article entity within the FANDANGO-Project. Such example is displayed in FIGURE 6, FIGURE 7 and FIGURE 8.

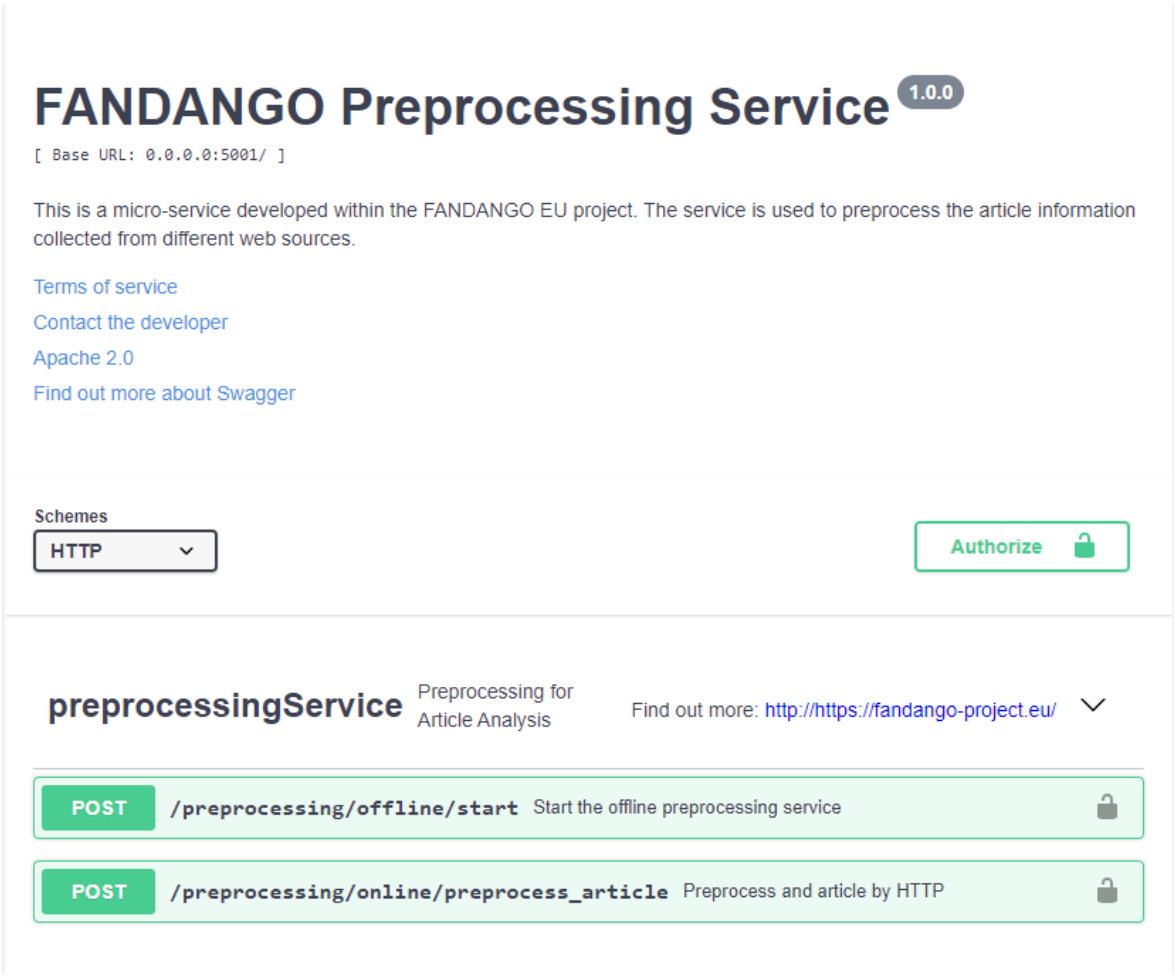

FIGURE 3 SWAGGER DEVELOPED FOR THE PREPROCESSING SERVICE

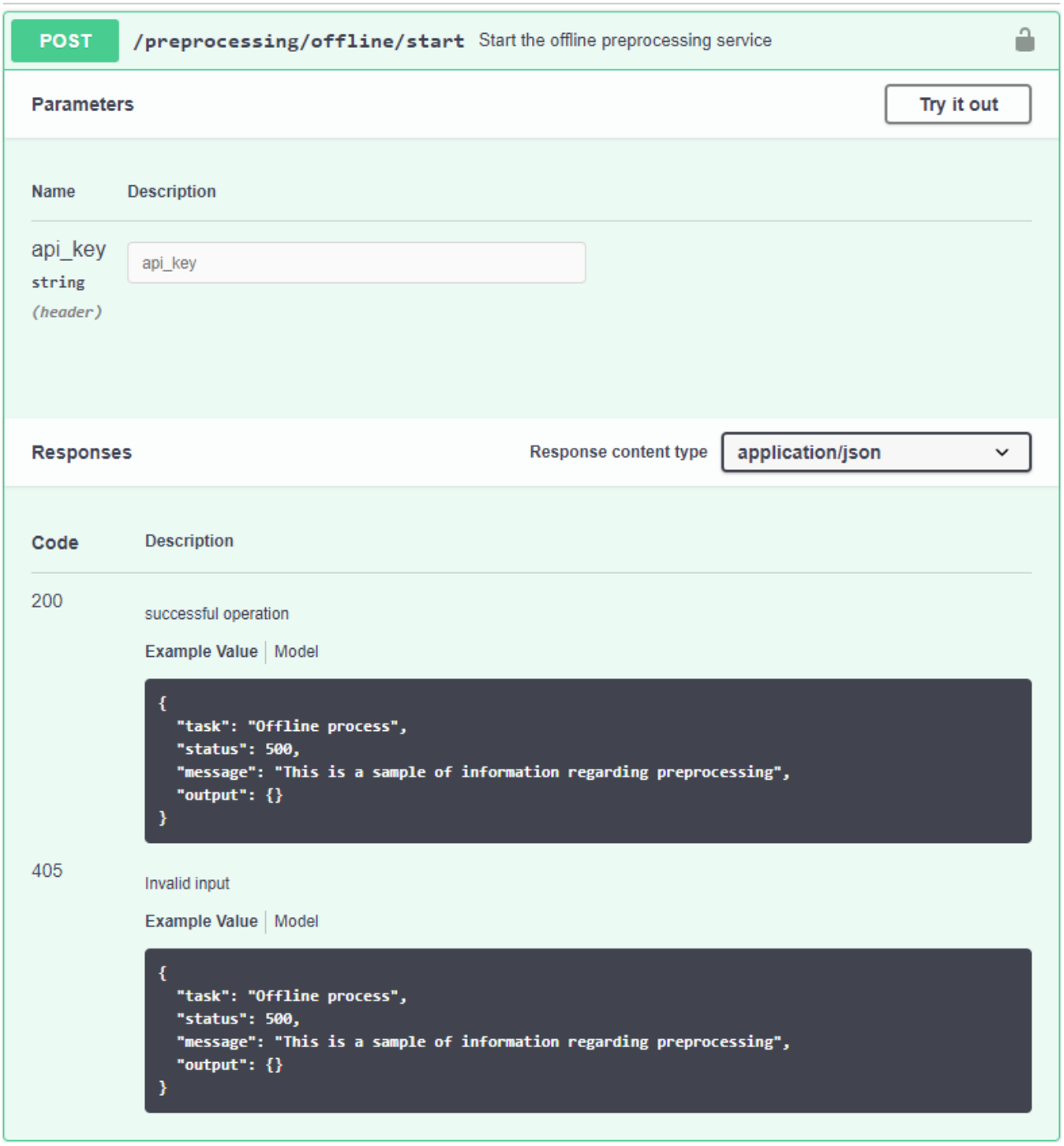

FIGURE 4 SPECIFICATIONS OF THE REQUEST FOR THE OFFLINE PROCESSING

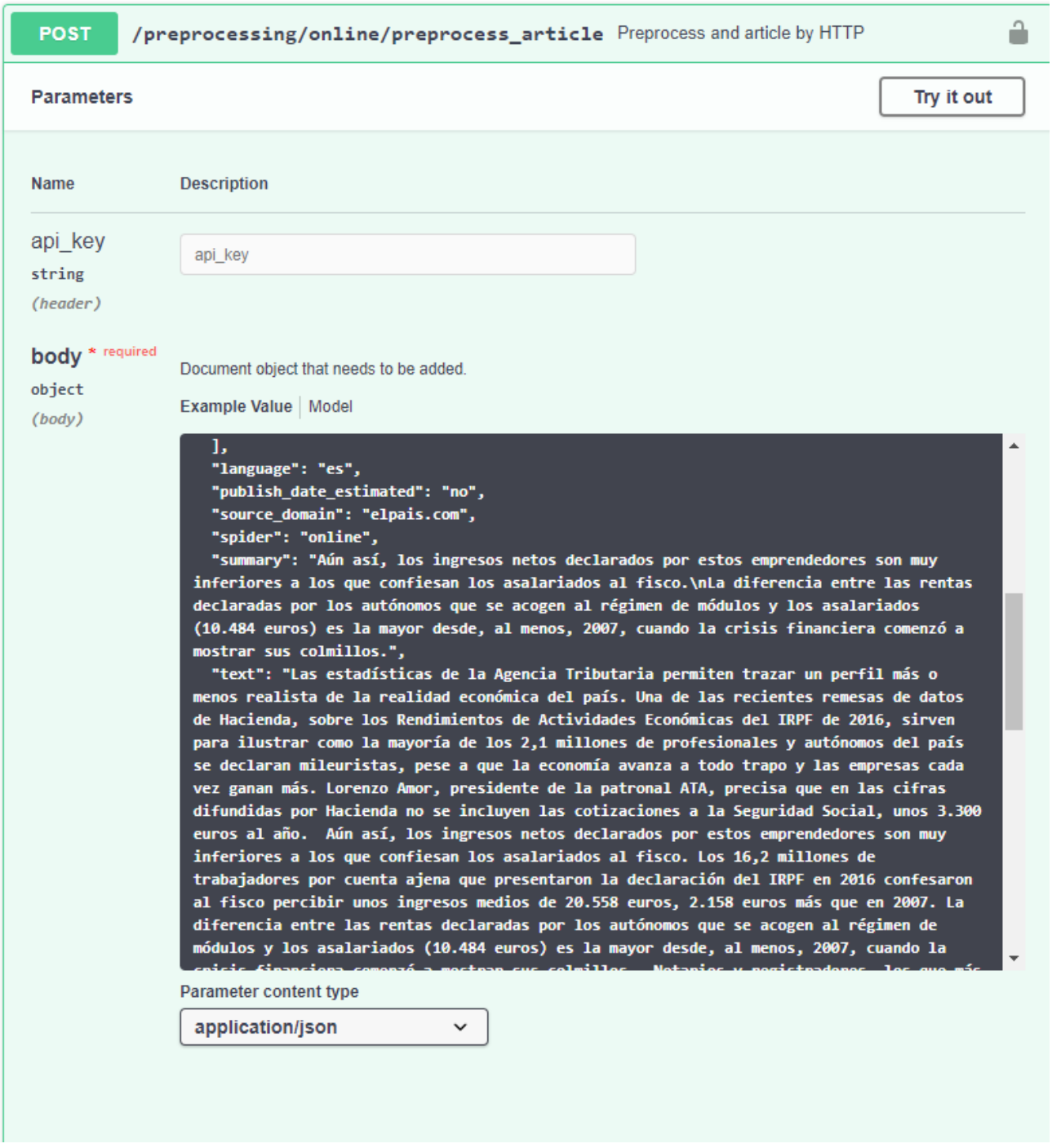

FIGURE 5 SPECIFICATIONS OF THE REQUEST FOR THE ONLINE PROCESSING

PreprocessingOutput ⌄ {

identifier*	string *example:* *92ce19eea1e0b639b790e1cf359c90acb8e0c30a999e73f59396c868b8a7bbb055e5071cb70b* 128 char identifier
headline*	string *example: Las estadísticas de la Agencia Tributaria permiten trazar un perfil país.* Article title
author*	› [...]
dateCreated*	string *example: 2019-11-05T13:45:35Z* Article date created (UTC)
dateModified*	string *example: 2019-11-05T13:45:35Z* Article date modified (UTC)
datePublished*	string *example: 2019-11-04T00:00:00Z* Article date published (UTC)
publishDateEstimated*	string *example: no* Estimated published date flag
publisher*	string *example: EL País*

FIGURE 6 PREPROCESSED ARTICLE I

images*	› [...]
country*	string *example: Kingdom of Spain* Country of the publisher
nationality*	string *example: Kingdom of Spain* Nationality of the publisher
language*	string *example: es* Article language
sourceDomain*	string *example: elpais.com* Source domain
articleBody*	string *example: Las estadísticas de la Agencia Tributaria permiten trazar un perfil más país. Una de las recientes remesas de datos de Hacienda, sobre los Rendimientos de sirven para ilustrar como la mayoría de los 2,1 millones de profesionales y autón que la economía avanza a todo trapo y las empresas cada vez ganan más. Lorenzo Am en las cifras difundidas por Hacienda no se incluyen las cotizaciones a la Seguri los ingresos netos declarados por estos emprendedores son muy inferiores a los qu millones de trabajadores por cuenta ajena que presentaron la declaración del IRPF ingresos medios de 20.558 euros, 2.158 euros más que en 2007. La diferencia entre acogen al régimen de módulos y los asalariados (10.484 euros) es la mayor desde, comenzó a mostrar sus colmillos. Notarios y registradores, los que más tributan E vislumbra que los notarios y registradores son los profesionales que más tributan rentas netas. Los auditores de cuentas y los entrenadores y jugadores de fútbol c pagan al fisco en su declaración de la renta. En el extremo opuesto, sobresalen l módulos, conocidos como moduleros. A diferencia de los anteriores que pagan por e gravamen en función de sus ingresos reales—, los autónomos que tributan el IRPF p en función de unos coeficientes relacionados con los metros cuadrados de su local eléctrico, entre otros indicadores. La estadística de Hacienda dice que apenas pa ellos, destacan los empresarios de transporte de viajeros (taxistas) y mercancías menos contribuyen los emprendedores del sector de servicios prestados a las empre seguridad...— que únicamente pagan al fisco el 1,82% de sus rentas. Aunque la exp ingresos pagan pocos impuestos, estos autónomos son uno de los colectivos que car fiscal. Limitar el régimen de módulos Son frecuentes los casos en los que algunos preguntan "¿Sin factura o con factura?". "Estos contribuyentes tienen más opcione legal su factura fiscal, pero también es conocido que en este colectivo hay algun Sequeiros, experto fiscalista y socio de Lean Abogados. "Mientras que toda la inf nómina, un autónomo puede obtener rentas de las que Hacienda no tiene conocimient decir, pueden no declarar al fisco todos sus ingresos.*

FIGURE 7 PREPROCESSED ARTICLE II

```
    calculateRating*        integer
                            example: -99

                            Rating of the article

    calculateRatingDetail*  string
                            example:

                            Detail of the rating

    url*                    string
                            example: https://elpais.com/economia/2019/05/23/actualidad/1558638881_652034.html

                            Article url

    videos*                  > [...]

}
```

FIGURE 8 PREPROCESSED ARTICLE III

3.5. SERVICE INTEGRATION

The scope of this section is to provide the specifications that were employed to integrate the Preprocessing service within the FANDANGO Big Data platform.

The distribution of the software corresponds to Software as a Service (SaaS), a distribution model where the third-party provider hosts a set of applications and programs and makes them available to customers over the internet. Thus, Docker arises as a potential candidate to integrate the Preprocessing service using the aforementioned software distribution.

The technology employed for this purpose is Docker which is a well-known virtualization solution based on Linux ecosystems that creates and runs different containers. Docker allows to define a batch of elements that will be dynamically run and configured during the deployment of the final application. All the required information is coded into a single text file which is named as *DockerFile* where all the commands that must be executed to deploy the application are included.

Basically, a Docker container includes a set of software resources and it allows to easily deploy the desired applications in such container. The deployed applications will be provided with all the corresponding dependencies, so that, its integration process is minimised. Thus, a container can be seen as a program with its own isolated set of hardware resources.

To set up the virtualization of the service, two main files have been configured:

- *DockerFile*: a template to generate the images of the application
- Docker-entrypoint.sh: a file with bash commands that start the required services for the application to start working automatically every time the docker starts.

The aforementioned files are presented below in its respective order.

```
FROM python:3.6
COPY . /app
WORKDIR /app
RUN apt-get update
RUN apt-get -y upgrade
RUN apt install -y openssh-server
RUN apt install -y openssh-client
#RUN apt-get -y install sshpass
RUN pip install requests
RUN pip install future
RUN pip install colorlog
RUN pip install flair
RUN pip install -r requirements.txt
RUN python -m spacy download en_core_web_sm
RUN python -m spacy download xx_ent_wiki_sm
RUN python -m spacy download it_core_news_sm
RUN python -m spacy download de_core_news_sm
RUN python -m spacy download es_core_news_sm
RUN python -m spacy download nl_core_news_sm
RUN apt-get install python3-lxml -y
# RUN sed -i -e "\$aStrictHostKeyChecking no" /etc/ssh/ssh_config
# RUN chmod a+x ./tunnel_kafka.sh
# RUN chmod a+x ./tunnel_elastic.sh

EXPOSE 5001
EXPOSE 9092
EXPOSE 9220

#ENTRYPOINT ./docker-entrypoint.sh

CMD [ "python", "./init_services.py" ]
```

```
#!/bin/bash
echo "Launching server..."
python /app/init_services.py
sleep 1m
echo "Docker started, activating stream service... (it lasts around 60 seconds)"
wget --server-response --spider --quiet "http://0.0.0.0:5001/preprocess/stream/start" 2>&1 | awk 'NR==1{print $2}'
if [ $? -eq 200 ]; then
    echo "Stream sevice not started"
else
    echo "Preprocessing service ready..."
fi
```

In FIGURE 9, a general schema shows the architecture of the service when it is virtualized. As one may observe, a Docker container is provided with a Python server that is implemented using Flask (Flask web development, one drop at a time, n.d.), which is a lightweight web application framework that is used as the backend of the Preprocessing service. Then, using the REST API, a client can access to the functionalities of the services in a simple way.

Furthermore, in the last version, it was decided to use Docker-compose (Docker, n.d.) which is a tool for defining and running multi-container Docker applications. It requires a YAML file in order to configure the different application's services. Then, using a simple command, it is possible to create and/or start all the services that were defined in the configuration. The following frame shows the Docker-compose file that is used in the Preprocessing service.

```
version: '3'
services:
    preprocessing :
      image: "tavitto16/fandango_preprocessing:latest"
      container_name: docker_fandango_pre
      ports:
        - "5001:5001"
```

```
command : bash -c "python /app/init_services.py & sleep 40 && curl -X POST http://0.0.0.0:5001/preprocessing/offline/start && tail -F anything"
```

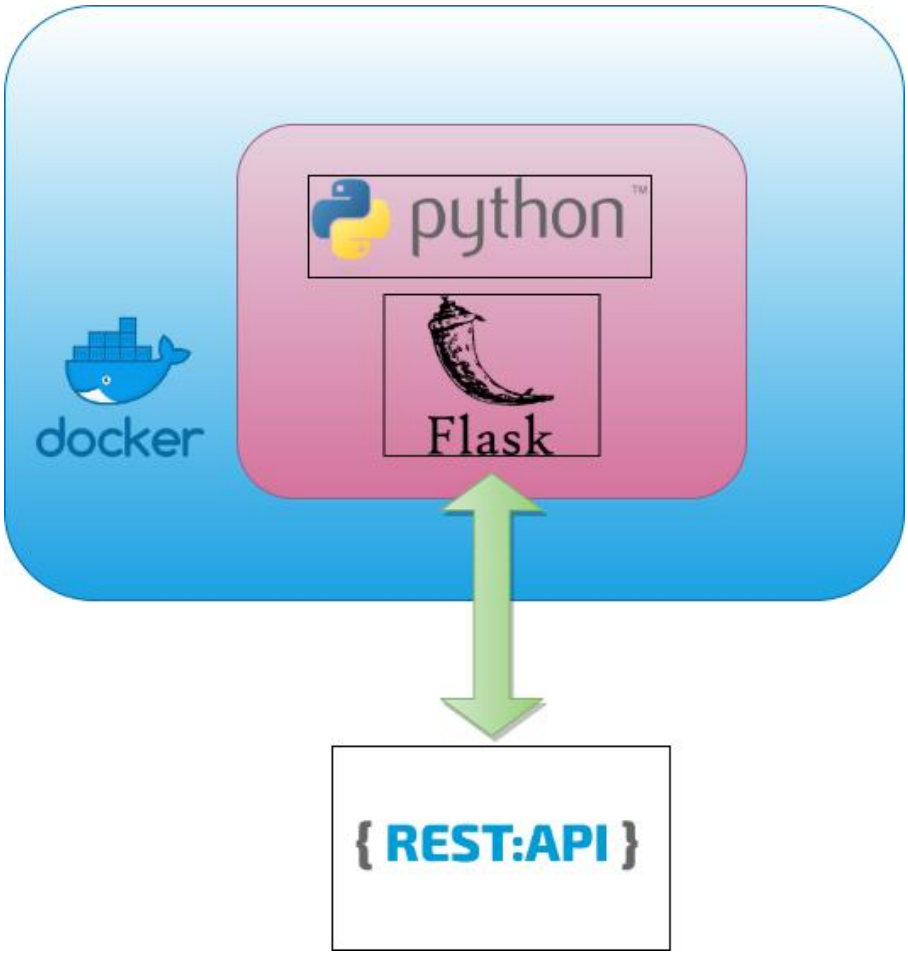

FIGURE 9 PREPROCESSING SERVICE VIRTUALIZATION

3.6. CODE RELEASE

All the code has been released in the production repository of the FANDANGO-project together with the documentation of all the methods, functionalities and setting ups to use properly the component.

The different versions of the preprocessing are released via Gitlab in the following link: https://github.com/fandangOrg/fandango/tree/upm/code/Preprocessing%20service. On the other hand, the services are available in the official Github of the project: https://github.com/fandangOrg. Finally, all the dockers with the full code that is used in the integrated version of the system are available at the official Dockerhub account of Fandango.

4. PREPROCESSING METHODS

The scope of this section is to provide the details of all the functionalities that have been implemented to preprocess the documents gathered by the Crawlers service.

Moreover, the main functionalities are related to the following topics:

- Language detection
- Author name cleaning
- Publisher extraction
- Image discarding
- Document discarding

4.1. LANGUAGE DETECTION

In many scenarios, the crawlers are not capable of retrieving the language of the document from the original HTML. The major reason for not detecting a language, or detecting the wrong language, is that websites do not follow a specific standard during the development of the site, therefore a language field might appear in different forms or not at all. Consequently, during the preprocessing stage, we need to extract the language of the document to be provided to the text analysers in a posterior stage.

Language detection solves the problem of identifying the language of the text using Natural Language Processing (NLP). The first procedure is to extract the language from the body of the article. To do so, state-of-the-art procedures have been employed including Deep learning approaches that are now in a production stage, so that, it can be used in Big Data Scenarios where the processes must be fast and efficient.

Moreover, the language identification stage is crucial in the preprocessing since it will trigger which models will be used to extract entities in the author name cleaning process that will be explained in the next section.

After several tests of different models, libraries and packages, we decided to use "*langdetect*" (Language-detection, 2016), a powerful library for language identification implemented by Google. It was originally developed in Java, but a Python wrapper was implemented for Python programmers. The model of the library reaches a 99% over precision in more than 50 languages.

FIGURE 10 shows a simple diagram to understand the flow at this step. As it is shown, the input of the model is both the body of the article as well as the headline and its output consist of the language code associated with the detected language.

FIGURE 10 LANGUAGE DETECTION BLOCK DIAGRAM

4.2. AUTHOR NAME CLEANING

In this section, the description of the processes that are employed to clean, modify or delete the authors named that are collected by the batch of Crawlers.

Why do we need this process? In the first place since one of the goals of FANDANGO is to measure the credibility of authors and publishers based on the available information, thus, if the names of the authors are not extracted properly, the analysis of their credibility will not be consistent. Secondly, due to the fact that every website may have its own schema and format and may differ from the standards (i.e 2-12-2019 Peter Cole or published by Laurie Marthin).

Many formats may be covered using standard regular expressions which are capable of working strings and remove, modify or isolate specific characters. However, it will not cover most of the formats and in many situations, it may arise issues when it faces a format that was not contemplated. Therefore, since this approach is semi-automatic because it requires a previous analysis of the formats where the names of the authors are embedded.

To address this issue, we decided to employ a Named Entity Recognition (NER) system to detect names associated to people. The exists many researches in this field such as the ones presented in (Akbik, 2018) (Yadav, 2019) (Akbik A. B., 2019).

More specifically, NER forms a core task in the process of building knowledge from unstructured sources such as text. These systems are based on three main approaches:

- **Statistical models,** throughout the use of Machine Learning (ML), and Deep Learning (DL) including Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).
- **Entity lists**, which are employed whenever the selected lists of entities is properly known and finite.
- **Regular expressions,** which are used when the entity can be defined as a certain pattern.

4.2.1. BRIEF INTRODUCTION TO NER SYSTEMS

The aim of this brief section is to provide a simple intuition about what the NER systems work using some mathematical theory (Sharma, 2018).

Firstly, we define an input sequence as a set of words that forms a particular sentence:

$$x = (x_1, x_2, \dots, x_n)$$

On the other hand, the output of the systems consists of a set of entity tags, that are denoted as output states:

$$s = (s_1, s_2, \dots, s_m)$$

The idea is that we can model the conditional probability of the output state given the input sequence using probability theory:

$$p(x_1, x_2, \dots, x_n)$$

To do so, we need to generate a feature map which is used to map an input sequence paired together with an entire state sequence to a certain feature vector in a d-dimensional space. This feature vector may be denoted as embedding. After that, we can model such probability as a log-linear model with a parameter vector denoted as *w*.

$$\Theta(x_1, x_2, \dots x_n, s_1, s_2, \dots, s_m) \in R^d$$

$$\mathrm{p}\,(\mathrm{x};\mathrm{w}) = \frac{exp\big(w\,\theta(x,s)\big)}{\sum_{s'} exp\big(w\,\theta(x,\,s')\big)}$$

In this case, *s'* ranges all possible output sequences. Using neural networks, we replace this linear scoring function by a non-linear representation. In FIGURE 11, an example of a schema regarding RNN's for NER systems is presented. The details of the model can be found at (Zhiheng Huang, 2015).

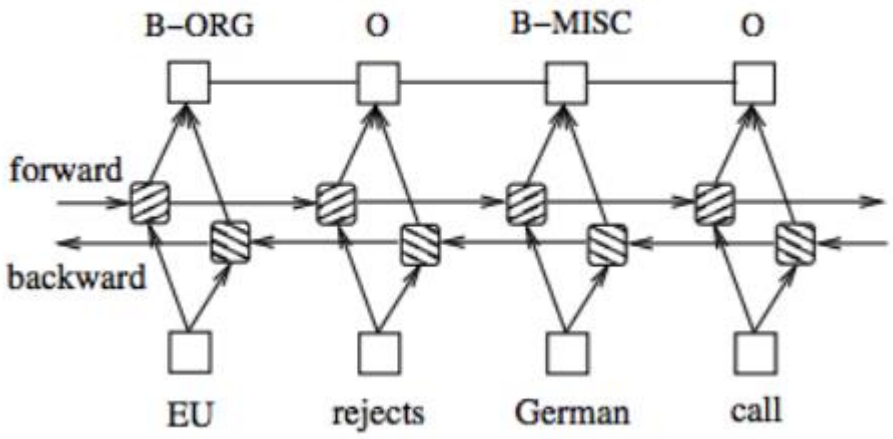

FIGURE 11 EXAMPLE OF AN RNN EMPLOYED IN NER SYSTEMS

4.2.2. NER SYSTEMS TO RETRIEVE THE NAME OF THE AUTHOR

In this section, an explanation of how a NER system is used to clean the name of the authors when they contain some artefacts or additional information that is not directly related to the name itself is presented.

To address this problem two state-of-art libraries in Natural Language Processing have been used: Spacy (Spacy, n.d.) and Flair (Research, n.d.).

Both libraries work for a large number of languages including the ones covering in FANDANGO: English, Spanish, Italian, Dutch and Germany. Moreover, they also have a multi-lingual approach to cover all the

languages using the same model in case it is necessary. In this scenario, it was decided to use the model associated with the language detected stage which was explained in a previous section.

To facilitate the understanding of the whole procedure, FIGURE 12 shows a general schema that is followed during the preprocessing of the document. As one may observe, once the language is detected, the proper model associated to it is launched using either Spacy or Flair. The selection of the library can be modified in a configuration file created for the service.

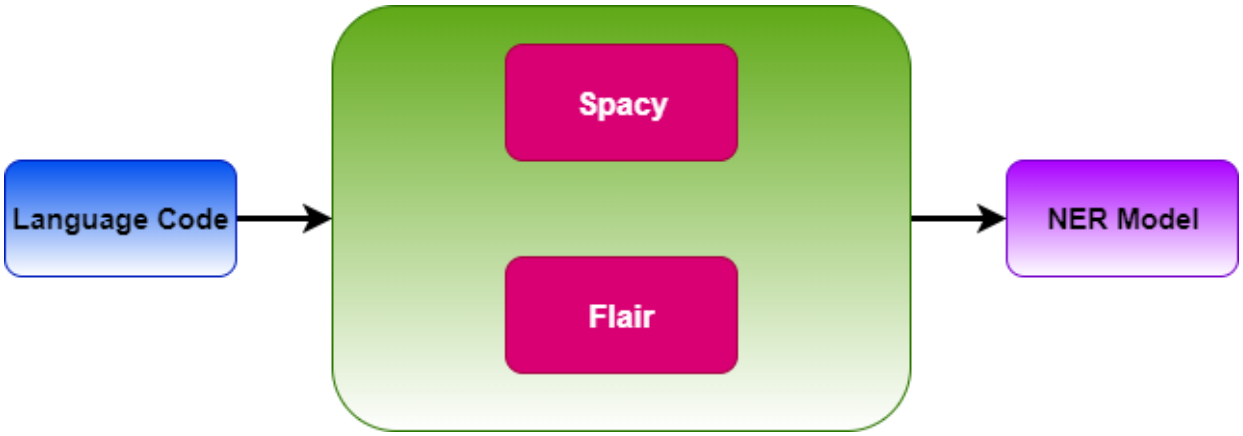

FIGURE 12 NAMED ENTITY RECOGNITION: LANGUAGE MODEL SELECTION

Then, the name of the author collected by the Crawlers is passed through the system and it returns the tags associated with it. The final output consists of the tags that are associated to either *PER* or *PERSON (it depends on the model).*

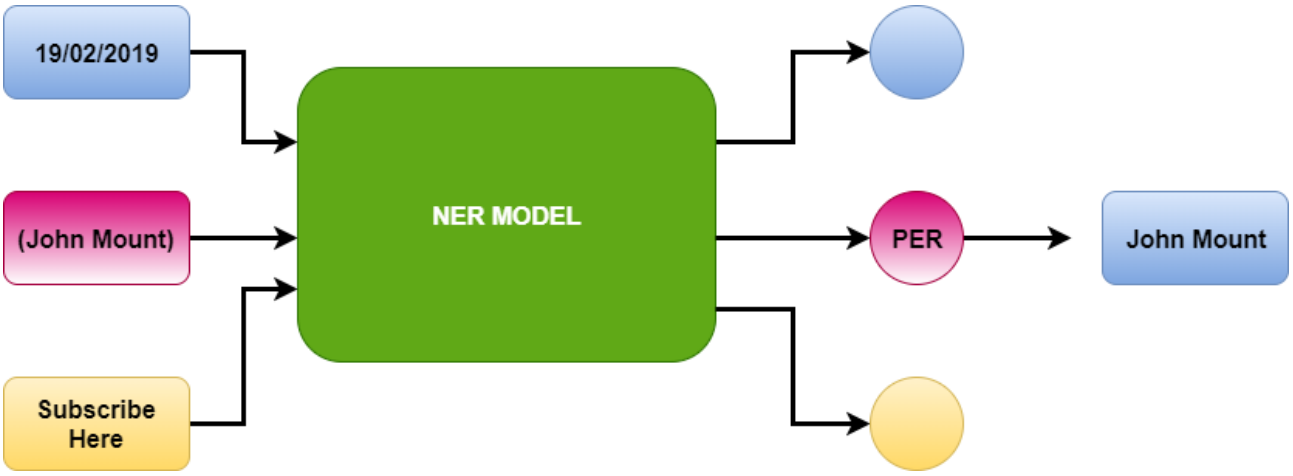

FIGURE 13 GENERAL SCHEMA FOR AUTHOR NAME CLEANING

Consequently, using this method, the system is capable of cleaning automatically the name of the author whether it is embedded with additional and useless information. A representation of this method is depicted in FIGURE 13.

On the other hand, due to the limitation of the model in terms of accuracy, precision and recall, the system may suffer from false positives and/or false negatives. However, even in these cases the system will be capable to mitigate this problem and the robustness of the presented models is continuously increasing and the performance will improve in new versions of the employed libraries.

4.3. PUBLISHER NAME EXTRACTION

In previous Deliverables, the data model of the project is described. Among the different entities involved in our model, one may observe the role of the publisher which arises as a potential entity in the whole process.

During the crawling stage where the information is collected from a large set of websites, the publisher is not retrieved directly. The unique data that can be used to get it relies on the URL of the article or the domain. Consequently, a preprocessing is required to ensure that the data model is satisfied.

There are two main approaches to address this issue:

- Automatic process using public API's such as WHOIS
- Semi-automatic process using a list with a large set of domains and publishers.

4.3.1. AUTOMATIC PROCEDURE

The aim of this automatic process is to directly extract the name of the publisher using public API's and well-known *URL-parsers*.

The main API employed to retrieve the publisher name given a domain consists in WHOIS (Whois Lookup, n.d.). It provides an API that is available in several programming languages including Python or Java. In this case, the Python wrapper (python-whois, 2019) has been used.

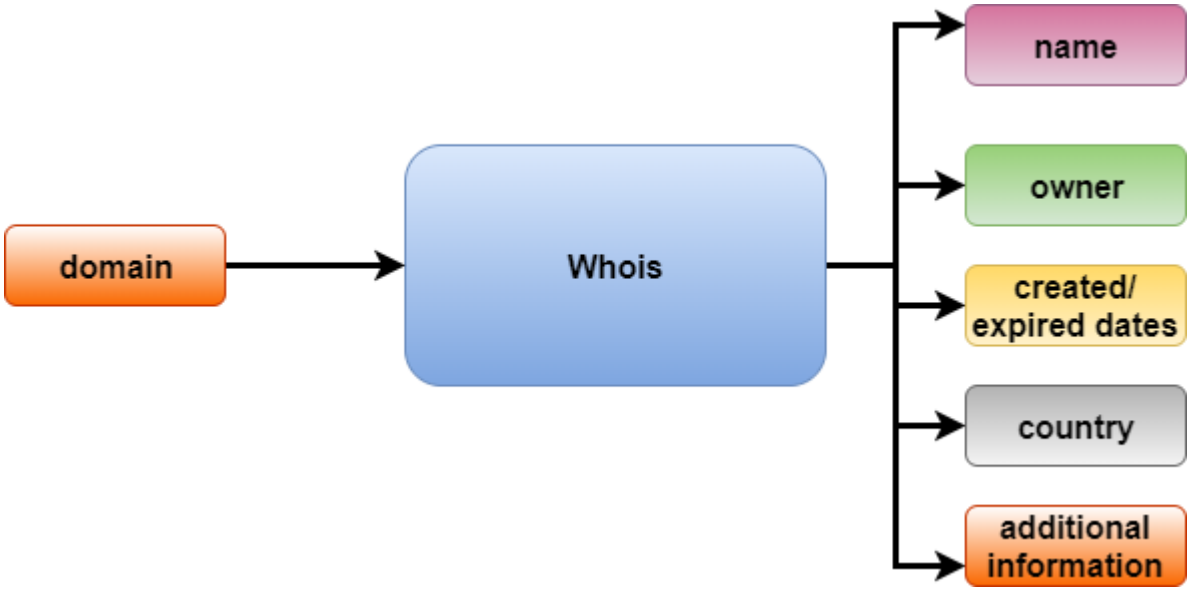

FIGURE 14 BLOCK DIAGRAM REPRESENTING THE RETRIEVAL OF THE DOMAIN INFORMATION

In FIGURE 14, a simple block-diagram schema of the information collected using WHOIS is depicted. As one may observe, given a domain, the system is capable of extracting very useful information including: the name of the domain, the owner, the creation and expiration dates, the country and many additional information that is not strictly necessary in this process but it can be used in other applications.

During this process, it was observed that in many situations, the domain is registered in countries that are not expected to be, and it can be used to track the information that this publisher is provided. Moreover, in other cases the name of the publisher is restricted or anonymous which can also be interpreted as not as reliable as one may believe with a first glance.

However, WHOIS does not cover all the domains as it is obvious and therefore, the automatic process cannot be used by its own since it will not cover all the cases that are required in the project. Consequently, a semi-automatic process arises to be essential to act in such cases where WHOIS cannot fulfil the needs of the project.

4.3.2. SEMI-AUTOMATIC PROCEDURE

In this second part, the explanation of the semi-automatic process to extract the publisher name is described. The main idea is to have a large list of domains and publishers and then, by the use of well-known distance metrics for text such as the so-called *Levenshtein* measure the level of similarity and retrieve the one that seems to have a higher score regarding the measurements.

Firstly, the generation of a large dataset of domains and publishers for the different countries that are covering along the project is required. In the first version of the service, a set of tools were developed to crawl and extract information from a very popular website named as *All Media Link* (All Media Link, n.d.). This site contains a large list of domains and websites very well-organized by countries. Consequently, it arises as a potential candidate to be employed for building the dataset. Specific crawlers were developed

to extract the data from the following countries: U.K, Spain, Italy, Greece, France, USA, Russian, Belgium and the Netherlands.

However, after a few months, due to possibly changes in the website, some of the links for the required countries were not available and it was necessary to use a distinct source to collect the list of domains and publisher names.

Consequently, it was finally decided to crawl ABYZ News Links (ABYZ News Links, n.d.) which has a similar organization in comparison with *All Media Link,* so that, all the websites are structured by countries. Similarly, as mentioned previously, the same set of websites from the aforementioned countries were analysed.

The dataset is collected only once, and it is always before the service starts working to mitigate the unnecessary computational cost. Once the dataset is available, the service safes it in the cache memory so that it can be used during its execution.

Secondly, a processing to extract the sub-domain from the original domain is necessary in order to compute the aforementioned distance metric. To do so, a Python library named as *tldextract* (tldextract, 2019) which is employed to accurately separate the gTLD (generic Top-Level Domain) or ccTLD (country code Top-Level Domain).

As an example, given the URL *http://forums.news.cnn.com/* , the output of this library consists of three main parts:

- Subdomain: i.e forums.news
- Domain: ie. cnn
- Suffix: com

Bearing all of this in mind, the process flows in two main steps:

1) Query the dataset using the original URL. If there exists a result, we retrieve the name associated with the domain and the process ends. If not, we go to step two.
2) Get the domain after using the aforementioned library and go to step three
3) Compute the similarity between the domain and the domains provided by the dataset.
4) Get the maximum value and compares it to a threshold. If the value is higher than the threshold, then the argmax is retrieved to get the name of the publisher associated to such value, otherwise the domain is used as the name of the publisher (i.e from *examplesite.com* we would get *Examplesite* whether *examplesite.com* is not found in the list of sources).

In this case, there is an important parameter denoted as threshold which lies in the range between 0 and 100 where 100 indicates that the similarity is perfect (both characters are the same) and 0 denotes the opposite. To avoid problems, this threshold must be close to 100, otherwise the system may retrieve a wrong publisher name which would have a bad consequence in the process of building the data model due to some articles would be associated to wrong publishers.

Moreover, the dataset can be updated with more countries and sources in order to be more robust when WHOIS is not capable of extracting the required information from publishers.

4.4. COUNTRY EXTRACTION

As it was introduced in the previous section, WHOIS provides with potential information regarding a certain domain. Among the different information one may find the country where the domain is supposed to be registered. These connections between the countries, the domains or the owner of the website may help security professionals profile attackers, guide online fraud investigations, and map cyber activity to attacker infrastructure. Consequently, the country of the publisher arises as a crucial role in posterior analysis such as the graph analysis defined in Deliverable D4.4 where the source credibility score is computed based on some features associated to publishers.

As in other sections, there are two main paths to be followed in this sense: and automatic procedure using WHOIS and a semi-automatic procedure using the aforementioned TLD python extractor library.

In FIGURE 15, the pseudocode of the process is depicted to illustrate and facilitate the interpretation of the algorithm. Basically, the first attempt is based on WHOIS as aforementioned described. However, in many cases this approach will not be enough. Therefore, the TLD library is required to extract the suffix of the original URL. If the suffix consists in a country code, a Python library named as *pycountry* (pycountry, 2019) is necessary to map the country code to the official name of the country.

Most of the cases will be covered by the aforementioned methods, however, a minority of websites will not be available for WHOIS and their suffix will not contain a country code. In this case, the output of the system will be a default unknown value to denote that it was not feasible to retrieve the desired country given such URL.

4.5. DISCARDING NON-RELATED IMAGES

Another preprocessing criteria that is used is discarding non-related images such as advertisement images or login ones. The aim is to reduce the computational cost that the image analysers needs to train the models using the images collected by the crawlers. Consequently, by filtering the images not related to the article itself, we are both optimizing the training computational cost and mitigating the propagation of noisy samples that may reduce the performance of the models.

As many other processes, it can be solved automatically or semi-automatically. In this case the automatic procedure consists in having some ML or DL model that has been trained to distinguish between normal images and non-related ones. However, there are no public datasets for this purpose so that, this automatic process was discarded as the initial approach.

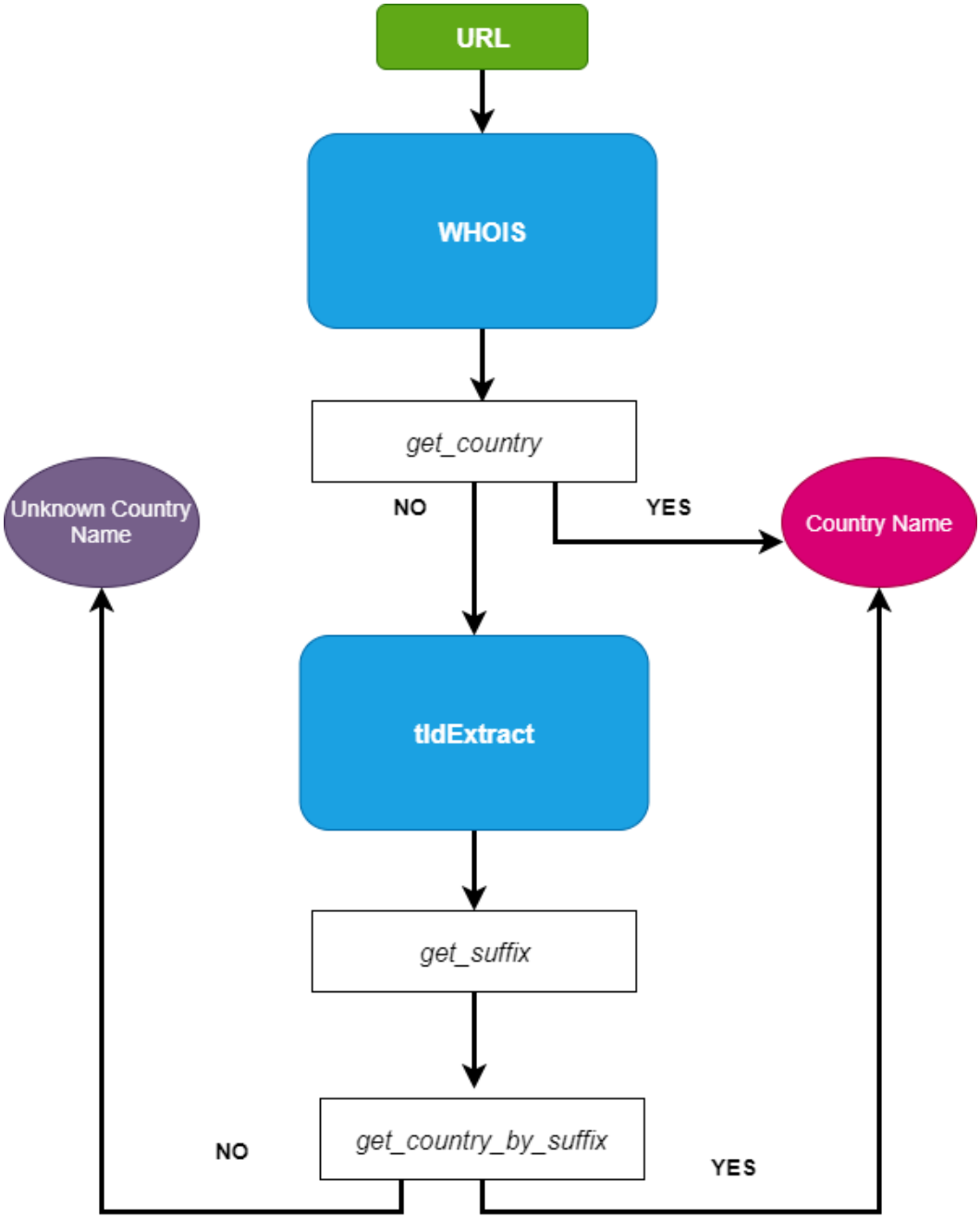

FIGURE 15 BLOCK DIAGRAM REPRESENTING THE PSEUDOCODE OF THE COUNTRY EXTRACTION PROCEDURE

On the other hand, the semi-automatic process relies on statistical modelling, where one tries to find patterns using a previous hypothesis. Then, a Hypothesis test is performed to accept or reject such criteria. In this case the criteria are the following:

- Small images are potential candidates to be non-related images such as icons, advertisements etc.

- Given a certain threshold regarding the aspect ratio of the images, non-related images can be distinguished from the related ones. However, such threshold needs to be found.

The first step consists in removing small-size images. To do so, 100x100 was used as the minimum number of pixels that the image must have. Therefore, images with a lower number of pixels than such threshold, are automatically discarded.

The second phase is used to discard those non-related images that are bigger than the aforementioned threshold but that has a very specific aspect ratio. More specifically, the image aspect ratio is basically the proportional relationship of the width to the height. As an example, a standard image of 1280x720 will have an aspect ratio of 16/9 (1.77:1) which means that the width is 1.77 higher than the height.

In many situations, the non-related images have a very high aspect ratio in terms of both height and width. The goal is to remove images that satisfies the following expression:

$$\frac{w}{h} \gg 1$$

Therefore, to find a threshold that satisfies the aforementioned expressions, a hypothesis was used. The 99% percentile of the aspect ratio was computed and then, the statistics for the remaining 1% of the distribution were employed to analyse the threshold. Finally, the final threshold was obtained using the percentile 85% of the distribution (around 5.2). Thus, all the images whose aspect ratio is above such value are candidates to be discarded.

For the other case, the goal is to delete images that satisfices:

$$\frac{w}{h} \ll 1$$

To do so, the steps are very similar as we introduced previously, but now the idea is to find the lower bound (lower threshold). In this case, we used the percentile 15% as the threshold of the analysis (around 0.3) which indicates that such images have much more height than width.

4.6. ADDITIONAL PREPROCESSING

Additionally, more processes were added in newest versions since it was detected that in some cases, the text analysers received empty articles which were wrong crawled and the preprocessing did not act to remove then. To mitigate this issue, an empty filtering was used to detect empty articles based on the body of the article as well as the headline.

5. GENERAL CONCLUSIONS

In previous sections, a description of the main functionalities included in the Preprocessing service are presented. Most of the methods and techniques employed during the preprocessing were obtained after analysing the problems that were firstly faced in FANDANGO such as the necessity of retrieving properly both the name of the authors and the publisher which were involved in the article. Therefore, the service must be continuously updated in order to work in future versions with better libraries and methods. Additionally, the service integration must also be evolved according to the needs of the final application of FANDANGO.

Furthermore, most of the processes described in this deliverable are addressed using well-known ML/DL methods together with traditional statistical techniques and public API'S. Using this fusion of information and techniques, it was shown that these methods are more robust and can be employed in multiple cases and scenarios, not only the ones considering in FANDANGO.

Due to the virtualization of the services throughout Docker, the service can be continuously updated and scaled according to the technical and user requirements without increasing any issue in terms of integration with other services.

On the other hand, the Preprocessing service can be used isolated to extract information from a crawled article thanks to its nature *Software as a Service* (SaaS). The API is well documented and has been created using the traditional Swagger so that, it is very intuitive and easy-to-use.

Regarding possible future functionalities, some processes such as detecting and discarding login or cookies pages may be considered to improve the performance of the text analysers models which can be affected when using this kind of pages during the training process.

Moreover, better methods for extracting automatically both authors and publisher can be incorporated to improve the robustness of the preprocessing.

6. REFERENCES

ABYZ News Links. (n.d.). Retrieved from ABYZ News Links: http://www.abyznewslinks.com/

Akbik, A. a. (2018). Contextual String Embeddings for Sequence Labeling. *27th International Conference on Computational Linguistics*.

Akbik, A. B. (2019). Pooled contextualized embeddings for named entity recognition. En A. B. Akbik, *Human Language Technologies.*

All Media Link. (n.d.). Retrieved from All Media Link: http://allmedialink.com/

Anandarajan, M. H. (2019). A Text Preprocessing Approach for Efficacious Information Retrieval. In D. &. Virmani, *In Smart Innovations in Communication and Computational Sciences.* Springer.

Denny, M. J. (2018). Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*.

Docker. (n.d.). Retrieved from Docker-compose: https://docs.docker.com/compose/

Flask web development, one drop at a time. (n.d.). Retrieved from Flask web development, one drop at a time: https://flask.palletsprojects.com/en/1.1.x/

Language-detection. (2016, Octubre). Retrieved from Language-detection: https://github.com/shuyo/language-detection

pycountry. (2019, August). Retrieved from pycountry: https://pypi.org/project/pycountry/

python-whois. (2019, September). Retrieved from python-whois: https://pypi.org/project/python-whois/

Research, Z. (n.d.). *Github*. Retrieved from Flair: https://github.com/zalandoresearch/flair

Sharma, R. (December de 2018). *Medium*. Obtenido de A complete tutorial for Named Entity Recognition and Extraction in Natural Language Processing using Neural Nets: https://medium.com/@rohit.sharma_7010/a-complete-tutorial-for-named-entity-recognition-and-extraction-in-natural-language-processing-71322b6fb090

Spacy. (n.d.). Retrieved from Industrial-Strength Natural Language Processing: https://spacy.io/

Srividhya, V. &. (2010). Evaluating preprocessing techniques in text categorization. *International journal of computer science and application*.

tldextract. (2019, October). Retrieved from tldextract: https://github.com/john-kurkowski/tldextract

Uysal, A. K. (2014). The impact of preprocessing on text classification. *Information Processing & Management*.

Vijayarani, S. I. (s.f.). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*.

Whois Lookup. (n.d.). Retrieved from Whois Lookup: http://whois.domaintools.com/

Yadav, V. &. (2019). A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. *COLING* .

Zhiheng Huang, W. X. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging.