This project has received funding from the European Union's Horizon 2020 research and innovation programme under **grant agreement No 780355**

FANDANGO DELIVERABLE

Deliverable No.:	D3.2
Deliverable Title:	Lightweight data shipping components development
Project Acronym:	Fandango
Project Full Title:	FAke News discovery and propagation from big Data and artificial inteliGence Operations
Grant Agreement No.:	780355
Work Package No.:	3
Work Package Name:	Data gathering and preprocessing for Data Lake population
Responsible Author(s):	CERTH
Date:	30.12.2019
Status:	v1.0 - Final Version
Deliverable type:	DEMONSTRATOR
Distribution:	PUBLIC

REVISION HISTORY

VERSION	DATE	MODIFIED BY	COMMENTS
V0.1	04.12.2019	Panagiotis Stalidis	Table of Contents
V0.2	09.12.2019	Panagiotis Stalidis	Data sources section contributions
V0.3	13.12.2019	Panagiotis Stalidis	Methodology section contributions
V0.4	18.12.2019	Panagiotis Stalidis	Introduction and conclusions contributions
V0.5	19.12.2019	Panagiotis Stalidis	Formatting of content
V0.6	20.12.2019	Theodoros Semertzidis	Executive summary contributions and final consolidation for Internal Review version
V0.7	27.12.2019	Panagiotis Stalidis	Revised according to Internal Review feedback
V0.8	29.12.2019	Theodoros Semertzidis	Quality check
V1.0	30.12.2019		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

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 documents the work done in WP3 - Data gathering and preprocessing for Data Lake population regarding the gathering of data for populating the data lake. It expands on the project's ingestion architecture, reported in D3.1 - Data model and components, by documenting the development of the data shipping components.

The first two sections briefly introduce the concept of crawling websites for gathering data and the strategies that were adopted for selecting sources of information for the data types defined in the Fandango project.

Then, the next section describes how the developed crawling modules are integrated in the project architecture. A vital part for integrating the crawling modules in the end-to-end data workflow of the project is the development of a custom scheduler that shares the project database for persisting the requests.

Finally, the last section analyzes the internal workings of the crawling modules, the architecture and the development process that was followed to produce the software. Special note is given to handling missing data and also in detecting and ignoring irrelevant content such as advertisements.

1. INTRODUCTION

The Fandango project aims to provide journalists, reporters and news agencies with the proper tools and information for them to properly assess the validity of stories that they want to report. Information can originate in many different forms. A major source of information comes from news that have been reported in the near past from other journalists and agencies. The number of news providing websites is nowadays so huge that a person can not possibly monitor even the most prominent of them. A major drawback of sourcing information to other journalists is the uncertainty of how valid their sources are. Therefore, other sources of information that can not be disputed are also necessary. Such information comes from official European, government and scientific data repositories.

Having these sources of information enables a journalist to verify the story to be reported by searching through relevant past articles and facts that exist in data repositories. Fandango gathers and stores these data in an internal data lake, thus providing faster access and durability. The major problem that arises in the data gathering process is that news websites do not follow any common format in how they distribute information. Therefore, different specialized web page parsers have to be implemented in order to consistently gather data from each source.

Finally, another source of data has been identified in fact checking agencies. These agencies use journalists for assessing the validity of circulating claims and report their results in their websites. Many of these agencies have created a network and defined a common data schema (claimReview) to use. In Fandango we have focused on gathering, as many claim reviews as possible, that follow this schema.

2. DATA SOURCES

The internet hosts a huge amount of information through news sites, social applications, open databases and other web sources. During the user requirement task conducted in Fandango, the end-users provided lists of sources for the various types of data that are interesting to them. These lists have been limited to the providers of content in the languages that are going to be handled by the project. Also a prioritization has been provided depending on the pilots that are being implemented and other needs of the project (i.e. user annotation).

The data that are being gathered are of three categories: articles, which are news items gathered from various sources on the internet, facts, which are verified pieces of information and claims/claim reviews which are analyses of claims performed by specialized journalists.

2.1. NEWS ARTICLES

The largest part of information coming into the system originates from web newspapers and websites of news agencies. While the process of gathering news articles can be extended indefinitely, for the scope of this project the end-user partners selected a considerable number of electronic publications to crawl. These publications were limited to 15 sources per language targeted by the project (i.e. English, Italian, Flemish and Spanish). Furthermore, this list is separated into: sources that are generally trusted by the public, sources that should not be trusted by the public and sources to be used for validating the effectiveness of the machine learning algorithms. The starting urls for the 40 trusted and untrusted sources are listed in the following Table 1 and Table 2.

In order to speed data gathering for the user annotation process described in deliverable D3.4, a further specification was defined for the validation sources by selecting specific subdomains/categories to limit the crawlers. The list of allowed domains for data gathering is in Table 3.

LANGUAGE	START URL
IT	http://www.ansa.it/
IT	https://www.repubblica.it/
IT	https://www.corriere.it/
IT	https://www.ilsole24ore.com/
IT	https://www.rainews.it/
ES	http://elpais.com/
ES	http://elmundo.es/
ES	http://elconfidencial.com/
ES	http://www.rtve.es/noticias/
ES	https://cadenaser.com/
NL	https://www.vrt.be/vrtnws/nl/
NL	http://www.standaard.be/
NL	https://www.tijd.be/
NL	https://nos.nl/nieuws/
NL	https://www.nrc.nl/
EN	https://www.apnews.com/
EN	https://www.washingtonpost.com/
EN	https://www.wsj.com/
EN	https://www.theguardian.com/

Table 1: Starting URLs of trustworthy news article sources.

LANGUAGE	START URL
IT	https://www.tg24-ore.com/
IT	https://tg-news24.com/
IT	http://il-quotidiano.info/
IT	https://www.lavocedelpatriota.it/
IT	https://www.semplicipensieri.it/
ES	https://www.mediterraneodigital.com/
ES	https://casoaislado.com/
ES	http://www.elespiadigital.com/
ES	http://www.alertadigital.com/
ES	http://latribunadecartagena.com/
NL	https://www.ninefornews.nl/
NL	https://www.dagelijksestandaard.nl/
NL	https://fenixx.org/
NL	https://jdreport.com/
NL	https://revolutionaironline.com/
EN	https://www.infowars.com/
EN	https://sputniknews.com/
EN	https://russia-insider.com/en/
EN	http://en.news-front.info/

Table 2: Starting URLs of untrusted news article sources.

LANGUAGE	START URL
NL	https://www.hln.be/nieuws
NL	https://www.nieuwsblad.be/nieuws/binnenland
NL	https://www.nieuwsblad.be/nieuws/economie
NL	https://www.nieuwsblad.be/nieuws/buitenland
NL	https://newsmonkey.be/news/buitenland/
NL	https://newsmonkey.be/news/politiek/
NL	https://newsmonkey.be/news/politiek/verkiezingen/
NL	https://newsmonkey.be/planet/planet-planet/global-warming/
NL	https://newsmonkey.be/planet/science/
NL	https://newsmonkey.be/news/binnenland/
NL	https://www.gva.be/nieuws/binnenland
NL	https://www.gva.be/nieuws/buitenland
NL	https://www.hbvl.be/nieuws/binnenland
NL	https://www.hbvl.be/geld-en-zaken
NL	https://www.hbvl.be/nieuws/buitenland
NL	https://www.hbvl.be/nieuws/wetenschap
NL	https://www.nd.nl/nederland
NL	https://www.nd.nl/buitenland
NL	https://www.nd.nl/politiek
NL	https://www.nd.nl/politiek
NL	https://www.rtlnieuws.nl/nieuws/nederland
NL	https://www.rtlnieuws.nl/nieuws/buitenland
NL	https://www.rtlnieuws.nl/nieuws/politiek
NL	https://www.rtlnieuws.nl/nieuws/opmerkelijk
NL	https://www.rtlnieuws.nl/tech
NL	https://www.rtlnieuws.nl/economie

NL	https://www.nu.nl/algemeen
NL	https://www.nu.nl/binnenland
NL	https://www.nu.nl/buitenland
NL	https://www.nu.nl/politiek
NL	https://www.nu.nl/klimaat
NL	https://www.nu.nl/dvn
NL	https://www.nu.nl/economie
NL	https://www.nu.nl/tech
NL	http://www.dewereldmorgen.be
NL	https://nl.metrotime.be/news/general/
NL	https://nl.metrotime.be/news/world/
NL	https://nl.metrotime.be/business/
NL	https://nl.metrotime.be/over/green/
NL	https://www.mo.be/category/thema/milieu/klimaat
NL	https://www.mo.be/category/thema/politiek/conflict
NL	https://www.mo.be/theme/mondialisering
NL	https://www.mo.be/category/thema/maatschappij/sociaalmaatschappelijk/migratie
NL	https://www.mo.be/category/thema/ontwikkeling
NL	https://www.mo.be/category/thema/sociaal/ongelijkheid
NL	https://nl.express.live/economie/
NL	https://nl.express.live/science/
NL	https://nl.express.live/tech/
NL	https://nl.express.live/politiek/
NL	https://www.apache.be/category/nieuws/
NL	https://www.ad.nl/binnenland/
NL	https://www.ad.nl/buitenland/
NL	https://www.ad.nl/politiek/
NL	https://www.ad.nl/economie/

NL	https://www.ad.nl/gezond/
NL	https://www.ad.nl/wetenschap/
NL	https://www.ad.nl/tech/
NL	https://doorbraak.be/onderwerp/binnenland/
NL	https://doorbraak.be/onderwerp/buitenland/
NL	https://doorbraak.be/onderwerp/multicultuur-samenleven/
NL	https://doorbraak.be/onderwerp/economie/
NL	https://doorbraak.be/onderwerp/europa/
NL	https://doorbraak.be/onderwerp/geschiedenis/
NL	https://doorbraak.be/onderwerp/klimaat/
NL	https://doorbraak.be/onderwerp/politiek/
NL	https://doorbraak.be/onderwerp/vlaamse-beweging/
NL	https://www.krapuul.nl/category/wereld/
NL	https://www.krapuul.nl/category/politiek-blog/
NL	https://www.krapuul.nl/category/natuur-en-milieu-2/
ES	http://okdiario.com
ES	https://www.huffingtonpost.es
ES	http://publico.es
ES	http://libertaddigital.com
ES	http://periodistadigital.com
ES	http://vozpopuli.com
ES	http://elindependiente.com
ES	http://elplural.com
ES	http://lamarea.com
ES	https://www.elsaltodiario.com
ES	http://europapress.es
ES	http://efe.com
ES	https://www.cope.es

ES	http://lavozdegalicia.es
ES	http://elperiodico.com
ES	http://elcorreo.com
ES	https://www.lne.es
ES	http://farodevigo.es
ES	http://heraldo.es
ES	http://levante-emv.com
ES	http://diariovasco.com
ES	http://diariodenavarra.es
ES	https://www.eldiariomontanes.es
ES	http://elnortedecastilla.es
ES	http://laverdad.es
ES	https://okdiario.com/espana/
ES	https://okdiario.com/ultimas-noticias
ES	https://okdiario.com/investigacion/
ES	https://okdiario.com/sociedad/
ES	https://www.huffingtonpost.es/politica/
ES	https://www.huffingtonpost.es/internacional/
ES	https://www.huffingtonpost.es/economia/
ES	https://www.huffingtonpost.es/virales/
ES	https://www.publico.es/politica
ES	https://www.publico.es/internacional
ES	https://www.publico.es/sociedad
ES	https://www.libertaddigital.com/espana/
ES	https://www.periodistadigital.com/politica/
ES	https://www.periodistadigital.com/mundo/
ES	https://www.periodistadigital.com/gente/
ES	https://www.periodistadigital.com/economia/

ES	https://www.vozpopuli.com/economia-y-finanzas/
ES	https://www.vozpopuli.com/espana/
ES	https://www.vozpopuli.com/gritos/
ES	https://www.elindependiente.com/politica/
ES	https://www.elindependiente.com/espana/
ES	https://www.elindependiente.com/sociedad/
ES	https://www.elindependiente.com/sociedad/sucesos/
ES	https://www.elindependiente.com/economia/
ES	https://www.elplural.com/politica
ES	https://www.elplural.com/economia
ES	https://www.elplural.com/sociedad
ES	https://www.lamarea.com/secciones/internacional/
ES	https://www.lamarea.com/secciones/cultura/
ES	https://www.lamarea.com/secciones/politica/
ES	https://www.lamarea.com/secciones/sociedad/
ES	https://www.lamarea.com/secciones/cambio-climatico/
ES	https://www.lamarea.com/secciones/economia/
ES	https://www.elsaltodiario.com/el-salmon-contracorriente
ES	https://www.elsaltodiario.com/actualidad
ES	https://www.elsaltodiario.com/global#normal
ES	https://www.europapress.es/nacional/
ES	https://www.europapress.es/internacional/
ES	https://www.europapress.es/economia/
ES	https://www.europapress.es/sociedad/
ES	https://www.europapress.es/epsocial/
ES	https://www.efeminista.com/
ES	https://www.efe.com/efe/espana/efeverifica/
ES	https://www.efe.com/efe/espana/economia

ES	https://www.efe.com/efe/espana/sociedad/
ES	https://www.cope.es/actualidad/espana
ES	https://www.cope.es/actualidad/sociedad
ES	https://www.cope.es/actualidad/economia
ES	https://www.cope.es/actualidad/internacional
ES	https://www.lavozdegalicia.es/
ES	https://www.lavozdegalicia.es/galicia/
ES	https://www.lavozdegalicia.es/localidades/
ES	https://www.lavozdegalicia.es/economia/
ES	https://www.lavozdegalicia.es/espana/
ES	https://www.lavozdegalicia.es/internacional/
ES	https://www.lavozdegalicia.es/sociedad/
ES	https://www.elperiodico.com/es/ultimas-noticias/
ES	https://www.elperiodico.com/es/sociedad/
ES	https://www.elperiodico.com/es/politica/
ES	https://www.elperiodico.com/es/temas/sentencia-proces-43043
ES	https://www.elcorreo.com/politica/
ES	https://www.elcorreo.com/internacional/
ES	https://www.elcorreo.com/sociedad/
ES	https://www.elcorreo.com/sociedad/salud/
ES	https://www.elcorreo.com/sociedad/microfeminismos/
ES	https://www.lne.es/espana/
ES	https://www.lne.es/sucesos/
ES	https://www.lne.es/asturias/
ES	https://www.lne.es/sociedad/
ES	https://www.farodevigo.es/gran-vigo
ES	https://www.farodevigo.es/galicia/
ES	https://www.farodevigo.es/espana/

ES	https://www.heraldo.es/noticias/aragon/portada/
ES	https://www.heraldo.es/noticias/nacional/portada/
ES	https://www.heraldo.es/noticias/internacional/portada/
ES	https://www.heraldo.es/noticias/economia/portada/
ES	https://www.levante-emv.com/comunitat-valenciana/
ES	https://www.levante-emv.com/espana/
ES	https://www.levante-emv.com/sociedad/
ES	https://www.levante-emv.com/sucesos/
ES	https://www.levante-emv.com/tags/medio-ambiente.html
ES	https://www.diariovasco.com/politica/
ES	https://www.diariovasco.com/sociedad/
ES	https://www.diariovasco.com/gipuzkoa/
ES	https://www.diariovasco.com/economia/
ES	https://www.elespanol.com/espana/
ES	https://www.elespanol.com/mundo/
ES	https://www.elespanol.com/economia/
ES	https://www.elespanol.com/social/
ES	https://www.elespanol.com/corazon/
EN	https://www.thesun.co.uk/
EN	https://www.dailymail.co.uk/
EN	http://www.breitbart.com/
EN	http://cen.at/
EN	https://www.defendevropa.com/2017/news/electoral-commission-arron-banks/
EN	https://voiceofeurope.com/
EN	https://gellerreport.com/
EN	https://www.zerohedge.com/
EN	http://v4report.com/
IT	http://www.ilgiornale.it

IT	https://www.liberoquotidiano.it
IT	https://www.ilfattoquotidiano.it

Table 3: Starting URLs of news article sources for validation.

2.1.1. Social Media

As it is described in D4.4, there is an interesting task in analysing social media content such as Twitter in order to find specific patterns that might be employed to detect non-trusted sources based on different features such as the profile of the account, the number of tweets posted, retweeted etc.

To do such purpose, the well-known easy-to-use Python library named as *tweepy* [5] was employed. Moreover, a Developer account for Twitter is mandatory in order to access the desired information.

Then, three main elements are required:

- The set of keywords or Hashtags to extract only tweets within this criteria.
- The language of the tweets.
- An interval of dates that wanted to be analysed.
- The number of tweets.

As it was already mentioned in D4.4, the main limitation in Twitter lies in the number of tweets that can be collected per query as well as the total number of queries that can be performed. Due to this drawback, it was defined that in FANDANGO, this process is only performed in the online part of the platform throughout the user interface devoted to analyse twitter. All the details of the set of features collected during the Twitter analysis can be found in D4.4.

2.2. Facts from Open Data portals

Factual data are very important for journalists because they can verify if a claim is true or not. The end user partners of the Fandango project, based on the planned pilots, have selected specific portals of open data to be imported into the project data lake. The dataset name and index endpoints are listed in Table 4.

Since the data already exist in well defined databases, the Fandango project will not curate this data by integrating them into the project data lake for two main reasons: the existing data t in each database are heterogeneous and cannot be brought to the same database schema, and this task would be big enough for a disting project for each one of these data portals. Therefore, we identify the location where the data can be accessed and a description of what data are contained, and then insert this information to our data lake.

NAME	START URL
Eurostat	https://ec.europa.eu/eurostat/search
Eurobarometer	https://ec.europa.eu/commfrontoffice/publicopinion/index.cfm
EU ODP	https://data.europa.eu/euodp/data/dataset

Table 4: Starting URLs of open European data portals.

2.3. CLAIMS AND CLAIM REVIEWS

Fact checking is the act of checking factual information in news articles to determine the veracity and correctness of the factual statements in the text. While this internal process has been a common practice in news agencies for a long time, lately, several organizations have devoted to checking already published articles and rumors. Post hoc fact checking is usually followed by a written report published in websites and can be accessed freely by the public.

The Reporters' Lab [1] at Duke University maintains a database of fact-checking organizations that is managed by Mark Stencel and Bill Adair. The database tracks more than 180 non-partisan organizations around the world. The Lab's inclusion criteria is based on whether the organization:

- examines all parties and sides;
- examines discrete claims and reaches conclusions;
- tracks political promises;
- is transparent about sources and methods;
- discloses funding/affiliations;
- and whether its primary mission is news and information.

The claimReview data type was created in 2015 after a conversation between staff at Google and Glenn Kessler, the Washington Post fact-checker. Kessler wanted Google to highlight fact-checks in its search results. Bill Adair, director of the Duke Reporters' Lab, was soon brought in to help.

Dan Brickley from Schema.org [2], Justin Kosslyn from Google and Adair developed a tagging system based on the schemas maintained by Schema.org, an organization that develops structured ways of organizing information. They created a universal system for fact-checkers to label their articles to include the claim checked, who said it and a ruling on its accuracy.

In order to make the existing reviews available to the users of the Fandango platform, a local copy has been created with as many claim reviews as we could scrap. Regarding the scraping processes, we used the *scrapy* python library [3]. Scrapy uses a list of initial seed urls and parses the equivalent pages for urls to other pages. Each downloaded page is also parsed for information, following specific rules.

A list of fact checking sites (Table 5), provided by the end-user partners of Fandango was used in order to focus the scraping process to well known fact checkers that provide analyses for European affairs.

LANGUAGE	START URL
NL	https://www.faktistfakt.com/faktenchecks/
ES	https://maldita.es/
EN	https://www.factcheck.org/archives/
EN	https://www.politifact.com/truth-o-meter/statements/
EN	https://www.snopes.com/archive
NL	http://nieuwscheckers.nl/
IT	https://pagellapolitica.it/
EN	https://factcheck.afp.com/
NO	https://www.faktisk.no/
DK	https://www.mm.dk/tjekdet/

Table 5: Starting URLs of selected fact checking sites.

Starting from these home pages, an iterative procedure of following links and extracting claim reviews that follow the specification was engineered. The document that is pointed at by each url is downloaded to the server. All of the urls that are mentioned in the document are detected and extracted then appended to a scheduling queue. After following every url from the set of gathered urls, all the published pages of the sites have been visited.

At the same time, every page that is downloaded is also parsed for included data in the form of json-ld and microdata. The data are extracted using the *extruct* [4] python library. All the data instances that follow the claimReview specification are extracted and stored in the Fandango data lake.

Even though ClaimReview has also produced code for publishers to easily incorporate the schema in their websites, not all of the fact checkers have complied to the specification. After the latest iteration, almost 16000 unique claim reviews had been gathered by the Fandango crawlers, from the 24000 claim reviews that the ClaimReview database claims that exist. The crawlers are always running on the Fandango server, checking for new and updated content enriching the database with more recent content.

3. INTEGRATION TO THE PLATFORM

The Fandango platform relies on a cloud infrastructure, in order to be easily scalable depending on the available hardware resources. It is composed of modules that work in a distributed fashion, allowing to manage huge amounts of data and serve a great number of advanced analytics services.

For data ingestion, the platform uses Apache Nifi to control the execution of multiple independent crawlers, one per data source. Apache NiFi [6] is a software project designed to automate the flow of data between software systems. The software design is based on the flow-based programming model and offers features which prominently include the ability to operate within clusters, security using TLS encryption, extensibility (users can write their own software to extend its abilities) and improved usability features like a portal which can be used to view and modify behaviour visually. In Figure 1 we present the flow architecture of the crawling modules in the Nifi platform.

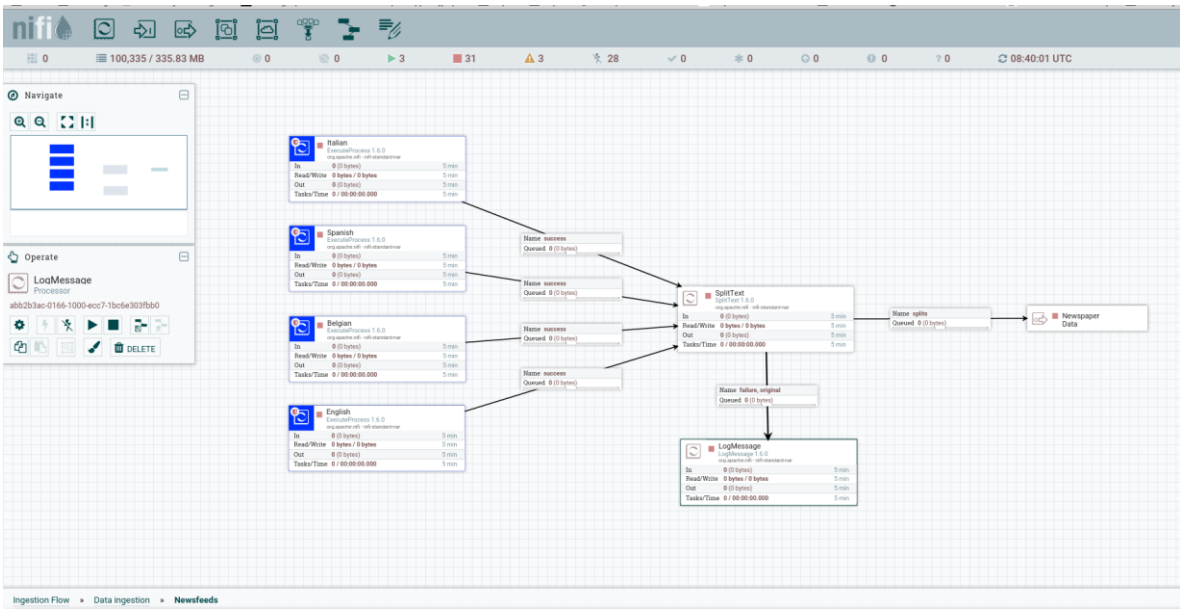

Figure 1: Architecture of Nifi crawler execution modules.

In order to allow maximum portability, the crawlers are packaged in Docker containers. Docker [7] is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages, called containers, that makes it easier to create, deploy, and run applications. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. While containers are isolated from one another they can communicate with each other through well-defined channels. Because all containers are run by a single operating-system kernel they are more lightweight than virtual machines. The compiled docker images are provided in the projects Docker account.

The data that are ingested by each crawler are pushed in a Kafka queue/topic to await preprocessing and analysis. Kafka [8] stores key-value messages that come from arbitrarily many processes called producers. Kafka runs on a cluster of one or more servers, distributing topics across the cluster nodes. This architecture allows Kafka to deliver massive streams of messages in a fault-tolerant fashion. Any number of processes can ask for messages stored in a topic. Each process belongs to a group. Different groups receive copies of the same messages but processes belonging to the same group will receive messages that have not been read by processes belonging to the same group. This feature allows for different analyzers to read the same

stream of data without interrupting each other. Additionally, it allows many instances of the same analyzer to exist, processing different data.

4. DATA GATHERING

The main goal of the data crawler components is to gather published data from various sources in the world wide web. These data come from three pools of sources: articles from news sites, claim reviews from fact checking agencies and facts from open data repositories.

Data retrieved from news sites come in html formatted text which is an unstructured source of information. One of the jobs of the data crawlers is to extract us much structured information as possible from the web page such as the text, authors, media and publication date. Unfortunately different sources use totally different structures in html code to present the relevant information. While a universal crawler can extract information from most of the sources, a number of source specific crawlers had to be developed to handle some cases. Note that there are even cases where parts of a single news source's website is structured differently than the rest of the website.

The crawling process can be roughly divided into two functionalities. The first functionality is the discovery and download of new documents from the sources. The second functionality is parsing these documents in order to extract useful information and put it into structured data items.

4.1. DISCOVERY AND DOWNLOAD

Scrapy [3] is a free and open-source web-crawling framework written in Python. Originally designed for web scraping, it can also be used to extract data using APIs or as a general-purpose web crawler. Following the spirit of other don't repeat yourself frameworks, it makes easier to build and scale large crawling projects by allowing developers to reuse their code. Scrapy project architecture (Figure 2) is built around "spiders", which are self-contained crawlers that are given a set of instructions.

The Scrapy engine is responsible for controlling the data flow between all components of the system. The engine starts the flow by picking the next request from the scheduler. The scheduler receives requests from the engine and enqueues them for being fed to the engine when requested. The request that was received by the engine is sent to the downloader, passing through the downloader middlewares. After the downloader fetches the content from the internet, it creates a response that is returned back to the engine. The downloader middlewares are responsible for processing both the request and the response and define the behaviour of the downloader based on the content. Useful applications include handling flash pages and dropping duplicate requests.

The received response, with the page content, is then forwarded by the engine to the spiders. The spiders are custom user defined classes for parsing responses and extracting scraped items (useful information) and additional requests to follow (urls). Spiders have a series of middlewares for incoming responses and outgoing items and requests for handling exceptions and post processing spider callbacks. The extracted requests are pushed to the scheduler, through the engine, for future downloading.

On the other hand, items are pushed to the item pipeline for processing, which is the last step of the data flow. Typical tasks performed in the item pipeline is cleaning, validation and persistence of the extracted information.

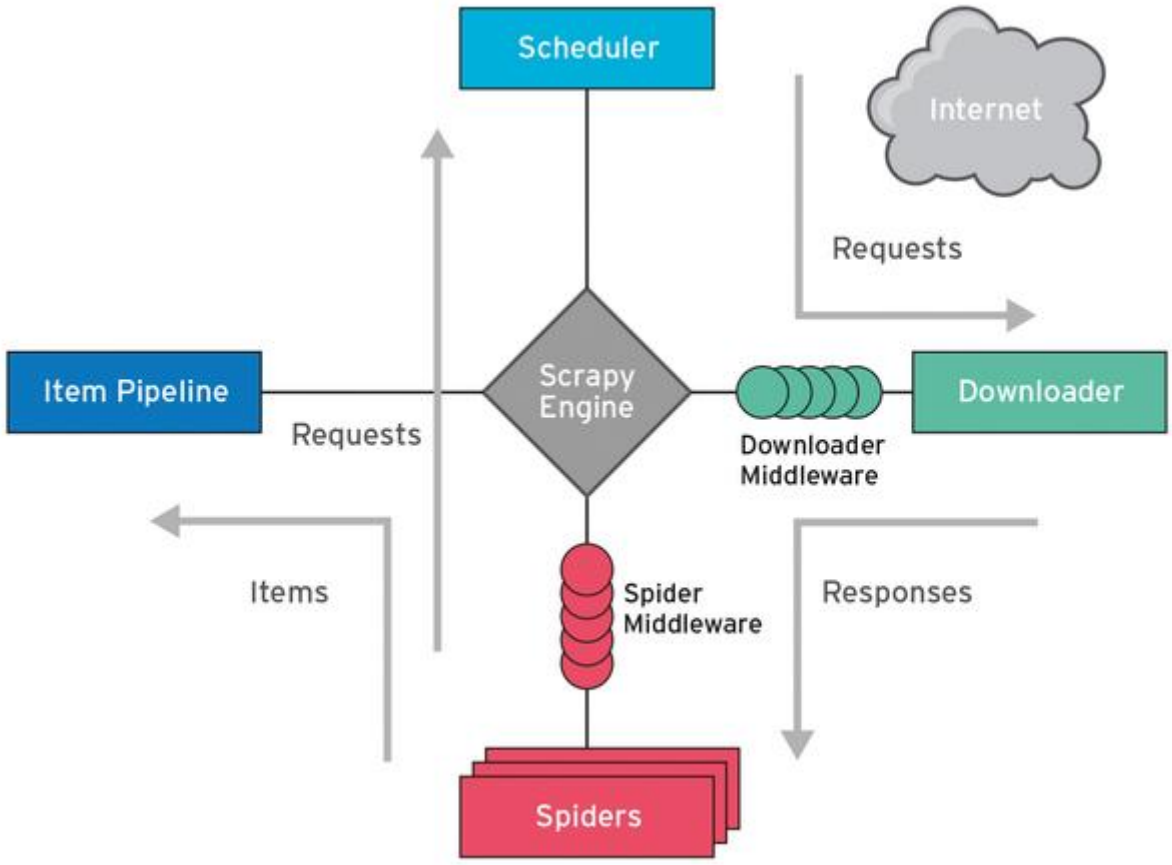

Figure 2: Architecture of Scrapy crawler software.

Scrapy includes two schedulers in the default installation. The first keeps the request queue in RAM and the second uses the filesystem for persistence. Using the first version is an ill fit for the Fandango project because the volume of the requests is huge for storing in RAM. Although persisting the queue in the filesystem is a viable solution, a better solution exists in storing the requests in an elasticsearch index.

On the one hand, storing information into a database is easier to manage and faster to search through. On the other hand, we have an elasticsearch database already in the cluster. Third, using the same identifier for storing the requests as the analyzed article, there is no need for a separate duplicate check. The requests can be the first step regarding the article analysis life cycle. This is possible since the identifier of every single article is calculated based on the *url* of the page which is available to the scheduler.

By designing a custom scheduler for Scrapy we can add additional fields for prioritising requests. In general, a first-come first-serve schedule regime is a logical default behaviour and can be easily implemented by returning the oldest existing not processed request. This is achieved by using a timestamp when a request to a url is first added.

In Fandango, the main need consists in continuously iterating all the sources for new content. This allows the database to be constantly updated. Therefore, instead of using a binary value for checking if a request has been served, a counter of how many times it has been served is preferred. In some cases, the user might want to prioritize specific pages over other based on unpredicted conditions. In order to serve this need we incorporated a priority field that can be edited by the user to increase the priority of specific pages. This modifies the scheduler sorting criteria to sort first by the priority field, then by the number of visits and finally by the timestamp.

4.2. DATA PARSING

In Scrapy, data parsing is performed by the spider instances. In order to define the parsing methodology, one has to inherit the scrapy spider class. In the derived class each downloaded response triggers the *overloaded parse_item* method. The *parse_item* method should either return a structured item or None.

4.2.1. NEWS ARTICLES

While schema.org has well defined data types for posting a news article on the web, almost none of the news providing sites employes such a protocol to allow machine readable information but offer content in human readable text, formatted with HTML. While HTML is adequate for marking text with style information, the semantics of the text are missing. In order to extract structured information, an analysis of the HTML code to detect common patterns is necessary. Most programming languages offer tools for this and python is no exception. The fastest and most extensive python library for parsing XML and HTML documents is *lxml*. Using *lxml* one can parse the HTML document for tags and build a tree of elements. An analysis of the tree structure can reveal the elements where the data are stored. Then, the data can be both extracted and inserted in a new data structure containing the information that needs to be kept.

Newspaper3k [9] is a python software library for parsing newspaper sites. The information that this library can extract includes the article title, summary and keywords, the names of the authors, the publication date, a list of included images and videos and the full body of the article. Additionally the language that the article is written in can also be detected. In short, all the desired fields defined in the data model (Table 6) for raw articles can be filed with information extracted with this library.

In the cases where some information is non existent default values are returned, when the type is: list, an empty list; string, an empty string; date, the current date and time in UTC. This practice might create some confusion about the publication date if the user does not know if this is the actual publication date or a default value, therefore the field “publish_date_estimated” which is filed with a “yes” when the default value is used.

One of the problems regarding the *newspaper3k* library arises when an image is not included by url but the image contents are included in the HTML. While the image is displayed properly to the browser, the library considers the content as a url, creating multiple problems both in the persistence of the extracted items and in the subsequent analyzers. A modification of the code of the library was proposed and pushed to fix this issue by checking if the src of the image tag starts with “http” or not. When it is not a url we store the image content in the project server and insert to the item the url to this object.

A more general problem is handling advertisements in web pages. Many of the images that are detected in an article have nothing to do with the article itself but are advertisements. In order to filter out advertisements we use an adblock rule library [10] that can detect most of the well known advertisement hosts. The same list of adblock rules is also used to filter out requests for advertisements.

FIELD	TYPE	EXAMPLE
identifier	String	5c03fbd2cdd66f1c306416f19895aab5f48d0faf7b2447
url	String	www.newssite.com/economia
source_domain	String	www.newssite.com
title	String	Economia e Finanza con Bloomberg
summary	String	Lorem Ipsum ...
description	String	Lorem Ipsum ...
keywords	List of String	[“repubblicait”, ...]
language	String	it
text	String	Lorem Ipsum ...
date_modified	Date	2019-02-19 14:45:13
date_created	Date	2019-02-19 14:45:13
date_published	Date	2019-02-19 14:45:13
publish_date_estimated	String	yes/no
authors	List of String	[“Name Surname”]
images	List of String	[“http://...”, “http://...”]
videos	List of String	[“http://...”, “http://...”]
texthash	String	d6583be6a3e2a4f268c8ab6b6c776fbf738bf79e6e4c

Table 6: Data structure of raw articles.

4.2.2. CLAIMS AND CLAIM REVIEWS

The second source of data for the Fandango platform is the claim reviews that are provided from fact checking sites. While the information provided by fact checkers in their web sites is also unstructured, a number of the sites have followed Google’s initiative for verified news. These fact checkers additionally offer a summary of the review in a crawler friendly *json* format that follows schema.org’s claimReview schema.

Various web sites can use different protocols to embed the data in the web page, therefore we search for and extract data in the HTML Microdata, JSON-LD, Microformat, Open Graph and RDFa syntax using the *extruct* python library.

For each claim review entity that is extracted from the crawlers, a corresponding claim entity is also created that represents the claim that is reviewed. When the review references the items that are revised , they are passed on to the claim entity. Additionally, a request is added to the articles crawler to fetch and parse the referenced items.

The web page that hosts the claim review is also analyzed in the same manner as an article to extract topics and named entities that might appear in the review text. In this way, we can associate claim reviews and thus, claims, to articles.

4.2.3. FACTS FROM OPEN DATA PORTALS

Finally, the open data repositories are crawled in order to create an index of the contents in each database. For each entry in the referenced database we try to extract a description of the contents. This description is also parsed for topics and entities to allow for association with articles and claim reviews.

The third source of information for the Fandango platform is an index to open datasets. Open data are silos of raw information offered to the public. Each of the databases follows an internal structure for the data they provide, therefore a common analysis of data from different sources is practically impossible.

The index creation is performed by a number of source specific crawlers, aiming to extract information about the semantics of the contained data in each source. Additionally, the spatial and temporal scope of the data is also determined and extracted. Thus, the user can quickly detect factual data relevant to the query they are performing.

REFERENCES

[1] Shin, Jieun, and Kjerstin Thorson. "Partisan selective sharing: The biased diffusion of fact-checking messages on social media." *Journal of Communication* 67.2 (2017): 233-255.

[2] "ClaimReview - schema.org Type." https://schema.org/ClaimReview. Accessed 20 Dec. 2019.

[3] "Scrapy." https://scrapy.org/. Accessed 18 Dec. 2019.

[4]"scrapinghub/extruct: Extract embedded metadata ... - GitHub." https://github.com/scrapinghub/extruct. Accessed 20 Dec. 2019.

[5] "Tweepy." https://www.tweepy.org/. Accessed 27 Dec. 2019.

[6] "Apache NiFi." https://nifi.apache.org/. Accessed 18 Dec. 2019.

[7] "Docker." https://www.docker.com/. Accessed 18 Dec. 2019.

[8] "Apache Kafka." https://kafka.apache.org/. Accessed 18 Dec. 2019.

[9] (2019, May 18). Building an Article Extraction Python API with newspaper3k .... Retrieved December 18, 2019, from https://shekhargulati.com/2019/05/18/building-an-article-extraction-python-api-with-newspaper3k-and-flask/

[10] "scrapinghub/adblockparser: Python parser for ... - GitHub." https://github.com/scrapinghub/adblockparser. Accessed 27 Dec. 2019.