D3.1 DATA MODEL AND COMPONENTS

Deliverable No.:	D3.1
Deliverable Title:	Data model and components
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):	ENG (Lead), LIVETECH, CERTH, UPM, SIREN
Date:	05.07.2019
Status:	V1.0
Deliverable type:	REPORT
Distribution:	PUBLIC

Revision History

Version	Date	Modified by	Comments
V0.1	03.06.2019	Massimo Magaldi (ENG)	First draft.
V0.2	14.06.2019	ENG, Siren, CERTH, UPM, LVT	Data model and Architecture update sections contributions.
V0.3	21.06.2019	Siren, CERTH, UPM, LVT	Component specification section contributions.
V0.4	28.06.2019	ENG, Siren, CERTH, UPM, LVT	Final consolidation and updates for complete version.
V1.0	05.07.2019	Massimo Magaldi (ENG)	Revised to match reviewers feedback.

TABLE OF CONTENTS

ABBREVIATIONS

ABBREVIATION	DESCRIPTION
H2020	Horizon 2020
EC	European Commission
WP	Work Package
EU	European Union

EXECUTIVE SUMMARY

This document is a deliverable of the FANDANGO project funded by the European Union's Horizon 2020 (H2020) research and innovation program under grant agreement No 780355.

In accordance with the consortium decision to adopt an iterative prototyping methodology in carrying out research and development activities, as reported in D1.4, this deliverable contains the updated revision of FANDANGO Architecture and Data Model as well as a first specification of the FANDANGO components (APIs and exchanged data specification).

After the submission of *D2.2 - Data Interoperability and data model design*, Data Model has been slightly modified - specifically the logical model of Elasticsearch - to meet analysis services' design details.

Similarly, after *D5.1 - FANDANGO Reference Architecture description* submission, the FANDANGO architecture has been revised according to actual processing needs. In particular, a streaming data pipeline has been implemented to handle continuous data ingestion and processing.

All data analysis services (trustworthiness scoring) as well as the FANDANGO User Interface were designed and implemented. All these components were specified in this deliverable.

This deliverable is also complemented by content defined in *D2.4 - Technical Requirements.*

As FANDANGO evolves so will its documentation, becoming more descriptive and precise during the lifespan of the project. Therefore, significant Data Model, Architecture and components revision will be integrated as annexes into next deliverables and/or *Project Progress Periodic Reports*.

1. INTRODUCTION

The Fandango project aims to address the aggressive emergence of fake news, post-truths, and misinformation, providing online web app and services that will support professionals with the following high-level features:

- Misinformation detection and trustworthiness scoring, based on Big Data analysis techniques (ML models and Graph Analysis), to understand if the news has been manipulated.
- Support user data investigation, through an interactive exploration of news, open data and verified claims databases.

More specifically:

- “trustworthiness” scoring

  Based on Big Data analysis techniques (ML and Graph Analysis) these features will provide a set of scores by analyzing the different components of news, i.e. text, authors, source, media:

  - “fake news” writing style (NLP)
  - media (video/image) manipulation
  - media (video/image) out of context
  - authors/publisher credibility
- data investigation
  - search for similar articles
  - search for similar claim (showing claim review details, if available)
  - search for claim related relevant open data
  - Data investigation with Siren Investigate (based on Siren Knowledge Graph)

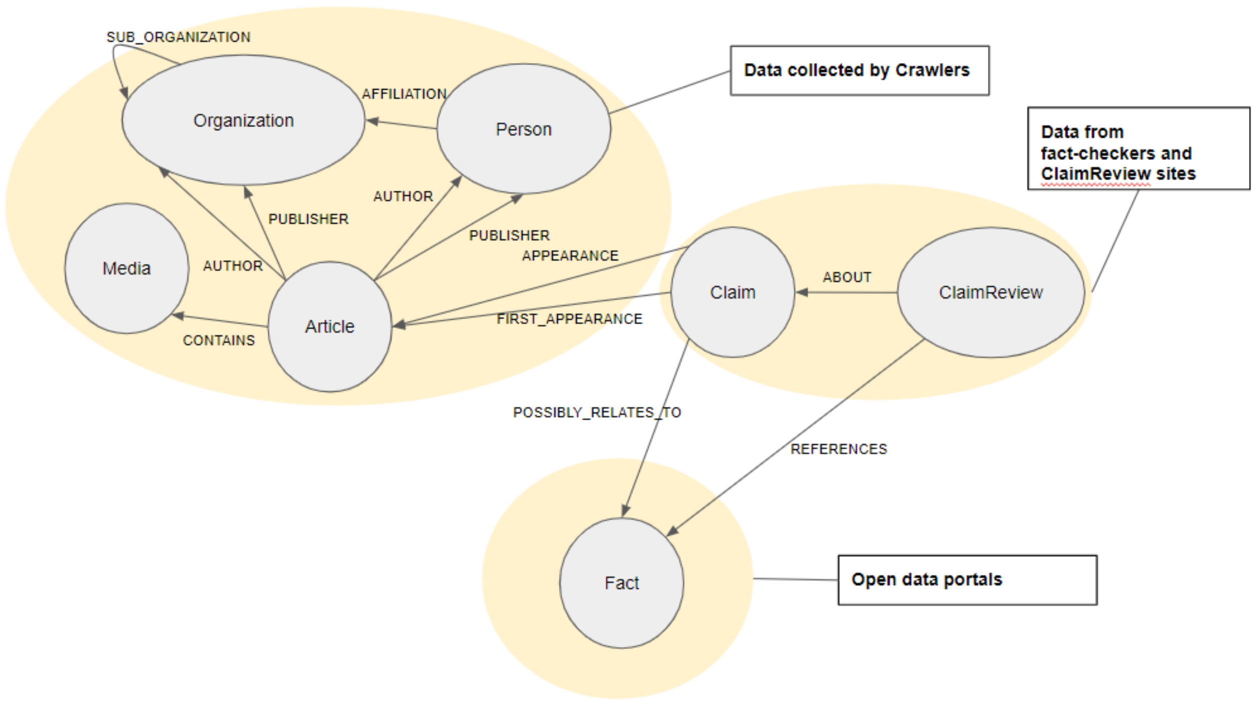
A first, prototypal version of all these features has been implemented in the last FANDANGO release (we’ve adopted an iterative prototype developing approach).

All implemented component, as well as the Data Model and Architecture updates are specified in the following sections of this deliverable.

2. Conceptual Data Model

FANDANGO's data contains several different entities that are used to store information about each of the concepts required to deliver the expected user functionality.

The conceptual data model defined in D2.2 and has not changed, but we report it here for clarity. Figure 3, seen below, contains the main concepts and the main relationships among them to represent the semantics of the data model.



c

Figure 1 - Conceptual Data Model

Since this conceptual data model represents how the users perceive and interact with the data in the real world, it drives the implementation of each logical data model across the system and ensures that the overall vision is kept, and data can be easily interoperated in the platform.

FANDANGO's conceptual data model was modelled to extend characteristics of standard implementations of schema.org. Each entity in the conceptual schema extends a schema.org entity to improve compatibility with external systems and make the platform scalable and reusable. The list of extended entities can be found in Table 2.

FANDANGO Entity	Extended schema.org entity
Organization	Thing > Organization
Person	Thing > Person
Article	Thing > CreativeWork > Article

Media	Thing > CreativeWork > MediaObject
Claim	Thing > CreativeWork > Claim
ClaimReview	Thing > CreativeWork > Review > ClaimReview
Fact	Thing > CreativeWork

Table 1 - Extensions of schema.org entities

Additionally, these structures were specially designed to address the user stories defined on D2.3, like "Search for data sources relevant to statements", "Related articles", "Who is the publisher?", "Who is the author of a news item?", "Is the photo/video new? Has it been tampered with?" and "Third-party fact-checking". These are the primary purpose of the functional design of FANDANGO and, therefore, carefully mapped into the data model to ensure user expectations are met.

2.1. LOGICAL DATA MODEL UPDATE

Logical Data Model changes are all related to Elasticsearch entities. All changes were included in the following updated section.

ELASTICSEARCH

In FANDANGO's architecture, Elasticsearch plays a central role as it is the main repository for the processed data that users will utilize through the web app.

Elasticsearch is the main data storage for Siren Platform, being an essential part of its functionality. Among the distinct capabilities leveraged by Siren platform are the relevance analysis, text search and analytical functions. Complementary, Siren Federate enhances Elasticsearch capabilities and include the possibility of relational analysis through its distributed join features, which is essential for FANDANGO's use cases.

As Elasticsearch is the main target data repository, where the final data is stored for user analysis after being processed, its data model resembles the conceptual data model in its entities as it is directly based on the way users will interact with FANDANGO's web app.

While entity-relationship structures are often not used on Elasticsearch implementations, since Elasticsearch does not natively provides a way of connecting data in different data sets, in this case, we can benefit from an ontological data model as Elasticsearch is enhanced by Siren Federate's join capabilities. Therefore, Figure 4 represents the logical data model implemented in Elasticsearch.

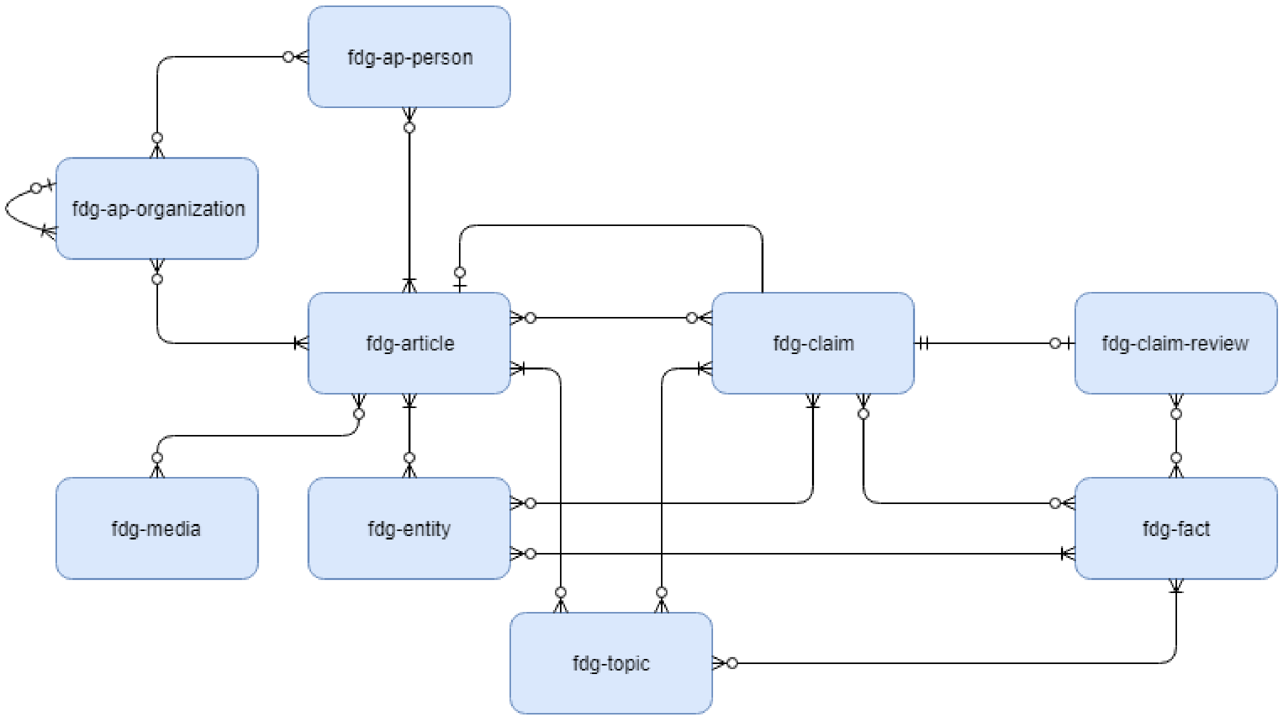


Figure 2 - Elasticsearch Logical Data Model

Differently from traditional relational databases, Elasticsearch allow data fields to contain arrays of different types of data, eliminating the need of joining tables or entities in case of many-to-many relationships when used in conjunction with Siren Federate. This dynamic simplifies the data model and makes it easier for systems to be implemented in a way that is intuitive to the users.

Table 3 describes each of the entities that will be available in Elasticsearch within the Siren data model.

ENTITY	OWNER	DESCRIPTION
fdg-ap-person	Siren	A person, such as a real individual or a fictional persona, like a pseudonym.
fdg-ap-organization	Siren	An organization, such as a school, a NGO, a newspaper, etc.
fdg-article	Siren	An article, such as a news article, piece of investigative report or social media publications. Newspapers, magazines, and social networks have articles of many different types and this is intended to cover them all.
fdg-media	Siren	A media object, such as an image, video, or audio object embedded in an article. Note that a creative work may have many media objects associated with it on the same item.

fdg-claim	Siren	A specific, factually-oriented claim that could be reviewed in a ClaimReview. Ideally, a Claim description includes enough contextual information to minimize the risk of ambiguity or inclarity. In practice, many claims are better understood in the context in which they appear or the interpretations provided by claim reviews.
fdg-claim-review	Siren	A fact-checking review of claims made (or reported) in some creative work.
fdg-fact	Siren	An Open Data creative work containing data, published by a recognized institution to make information available publicly.
fdg-entity	Siren	An entity that can be mentioned in articles, claims and facts, such as: a person, a place, a landmark or an event.
fdg-topic	Siren	A matter dealt with in an article, claim or fact publication.

Table 2 – Siren Elasticsearch Entities Description

ENTITY SPECIFICATION

A detailed view of each of the entities is provided below, identifying fields, types, primary keys, foreign keys. Additionally, a description of the purpose of the field and an example of the data contained is added facilitate comprehension.

Data types and structures are specific to Elasticsearch platform and, while the compatibility with schema.org standard is maintained whenever possible, some of the types were modified according to the needs of FANDANGO and the capabilities of the software in question.

ENTITY: FDG-AP-PERSON			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
Identifier	Text	“a1234f”	Unique Alfanumeric identifier of the item.
Name	Text	“John Smith”	The name of the person.
url	Text	"https://personpage.org/”	URL of the person.
Nationality	Text	”Swedish”	Nationality of the person.
Bias	Text	N.D. (for future use)	?
jobTitle	Text	”Financial Manager”	The job title of the person.
Gender	Text	“Not Specified”	Gender of the person.
Affiliation	array of text (FK fdg-ap-organization)	[10002, 356, 88, 8432]	An organization that this person is affiliated with.
trustworthiness	float	25.16	?

Table 3 – Entity specification

ENTITY: FDG-AP-ORGANIZATION			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**

identifier	Text	“a1234f”	The alfanumeric identifier of an organization.
name	Text	“New Org”	The name of the organization.
url	Text	"https://orgpage.org/”	URL of the organization.
nationality	Text	”irish”	Nationality of the organization.
country	Text	“Ireland”	Country of the organization.
bias	Text		?
parentOrganization	text (FK - AP_Organization)	“asd123”	The larger organization that this organization is a sub-organization of, if any.
trustworthiness	Float	50.84	?

ENTITY: FDG-ARTICLE			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
identifier	Text	“a123f”	The alfanumeric identifier of an article.
headline	Text	“News article headline”	Headline of the article.
articleBody	Text	“Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do	The actual body of the article.

		eiusmod”	
url	Text	“www.fandango.com”	The url of article
sourceDomain	Text	“fandango.com”	The source of article
inLanguage	Text	"en-GB"	The text that indicate the language
publishDateEstimated	Text	true/false	
dateCreated	Date	2017-12-15 14:45:13	The date on which the article was created, or the item was added to a DataFeed.
dateModified	Date	2017-12-29 14:45:13	The date on which the Article was most recently modified, or when the item's entry was modified within a DataFeed.
datePublished	Date	2017-12-28 14:45:13	Date of first broadcast/publication.
author	array of text (FK - fdg-ap-person or fdg-ap-organization)	[“topic27en”, “topic12en”]	The author of this content.
publisher	array of text (FK - fdg-ap-person or fdg-ap-organization)	[“topic27en”, “topic12en”]	The publisher of the article.
calculatedRating	Number	0.898	Rating of trustworthiness calculated by FANDANGO for the article.
calculatedRatingDetail	JSON ({ string:number, ...})	{“publishRating”: 0.99, “authorRating”: 0.873}	Rating of trustworthiness calculated by FANDANGO broken down into different evaluated categories.

about	array of text (FK – Topic)	[“topic27en”, “topic12en”]	Topics to which the article talks about.
mentions	array of text (FK - Entity)	[“topic27en”, “topic12en”]	Entities mentioned by the article.
contains	array of text (FK - Media)	[“topic27en”, “topic12en”]	Media files like audio, videos and images that are part of the article.

Table 4 – Entity specification

ENTITY: FDG-MEDIA			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
identifier	text	“a1234f”	The alfanumeric identifier of a media object.
contentUrl	text	"https://orgpage.org/”	URL for actual bytes of the media object, for example the image file or video file.
contentSize	text	“15.3 MB”	File size in (mega/kilo) bytes.
uploadDate	date	2017-12-29 14:45:13	Date when this media object was uploaded to this site.
encodingFormat	text	video/mp4	Media type typically expressed using a MIME format.
type	text	77	Type of the media. Valid values are “audio”, “video” or “image”.

calculatedRating	number	0.898	Rating of trustworthiness calculated by FANDANGO for the media.
calculatedRatingDetail	object	{“analyzer”: “faceForansics”, “bboxes”: [[0,0,0.7,0.7]], “scores”: [0.78]}	Some detail of the calculated rating

Table 5 – Entity specification

ENTITY: FDG-CLAIM			
FIELD	TYPE	EXAMPLE	DESCRIPTION
identifier	text	“a1234f”	The alfaNumeric identifier of a claim.
appearance	array of text (FK - Article)	[9765, 34]	Indicates an occurence of a claim in some article.
firstAppearance	text (FK - Article)	387487	Indicates the first known occurence of a claim in some article.
text	text	“”	The textual content of this claim.
dateCreated	date	2017-12-15 14:45:13	The date on which the claim was created, or the item was added to a DataFeed.
dateModified	date	2017-12-29 14:45:13	The date on which the claim was most recently modified, or when the item's entry was modified within a DataFeed.

datePublished	date	2017-12-28 14:45:13	Date of first broadcast/publication.
calculatedRating	number	0.898	Rating of trustworthiness calculated by FANDANGO for the claim.
about	array of text (FK - Topic)	[“topic27en”, “topic12en”]	Topics that the claim talks about.
mentions	array of text (FK - Entity)	[283, 18289, 84933]	Entities mentioned by the claim.
possiblyRelatesTo	array of text(FK - Fact)	[9879, 15378]	Facts that are possibly related to the claim.

Table 6 – Entity specification

ENTITY: FDG-CLAIM-REVIEW			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
identifier	text	“a1234f”	The alfaNumeric identifier of a claim review.
claimReviewed	text	“Claim made by X about the increase in Y and Z”	A short summary of the specific claims reviewed in a ClaimReview.
reviewAspect	text	“Segment X of the claim”	This review is relevant to this part or facet of the claim reviewed.
reviewBody	text	“This claim seems accurate because facts on the publication XPTO corroborates the statement X.”	The actual body of the review.
dateCreated	date	2017-12-15 14:45:13	The date on which the claim was created, or the item was added to a

			DataFeed.
dateModified	date	2017-12-29 14:45:13	The date on which the claim was most recently modified, or when the item's entry was modified within a DataFeed.
datePublished	date	2017-12-28 14:45:13	Date of first broadcast/publication.
aggregateRating	number	0.567	The overall rating of this review, based on a collection of reviews or ratings, of it.
reviewRating	number	0.348	The review rating value
itemReviewed	text (PK - Claim)	7646	The claim that is being reviewed/rated.
references	array of text(FK - Fact)	[5476, 976, 8754]	Facts referenced in this review.
url	text	"http://someurl/claimreview.html"	The url of claim

Table 7 – Entity specification

ENTITY: FDG-FACT			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
identifier	text	"a1234f"	The alfanumerical identifier of a fact.
name	text	"Climate change report 2018"	The name of the fact.
text	text	"Reports show increase in temperatures within	The textual content of this fact.

		the regions …”	
url	Text	“https://opendata.org/a”	URL of the fact.
about	array of text (FK - Topic)	[“topic27en”, “topic12en”]	Topics that the fact talks about.
mentions	array of text (FK - Entity)	[283, 18289, 84933]	Entities mentioned by the fact.
dateCreated	date	2017-12-15 14:45:13	The date on which the fact was created, or the item was added to a DataFeed.
dateModified	date	2017-12-29 14:45:13	The date on which the fact was most recently modified, or when the item's entry was modified within a DataFeed.
datePublished	date	2017-12-28 14:45:13	Date of first broadcast/publication.
temporalCoverageStart	date	2017-01-01	The start date and time of the fact.
temporalCoverageEnd	date	2017-12-31	The end date and time of the
spatialCoverage	text	“Brooklyn, NY, USA”	The spatialCoverage of a fact indicates the place(s) which are the focus of the content.

Table 8 – Entity specification

ENTITY: FDG-ENTITY			
FIELD	TYPE	EXAMPLE	DESCRIPTION

identifier	text	“a1234f”	The alfanumeric identifier of an entity.
name	text	“Mary’s Status”	The name of the entity.
type	text	"landmark”	Type of the entity. Possible values are “place”, “person” or “landmark”, but are not limited to it.

Table 9 – Entity specification

ENTITY: FDG-TOPIC			
FIELD	**TYPE**	**EXAMPLE**	**DESCRIPTION**
identifier	text	“topic27en”	The identifier of the topic
name	text	“elections”	The name of the topic.
keywords	array of text	[“some”, “words”, “define”, “topic”]	Some keywords that define the topic

Table 10 – Entity specification

3. Revised Architecture

The adoption of an iterative prototypingmethodology in carrying out research and development activities has naturally lead us to an architectural revision.

The revised architecture reflects the needs collected by the developments of different analysis provided by each partner. In order to satisfy specific requirements expressed by an heterogenous team with different backgrounds and practices, the overall system has slightly changed its components, keeping the same logic and philosophy.

The platform relies on a Cloud Infrastructure, in order to easily scale up or out in terms of HW resources. It is indeed composed by modules, that work in a distributed fashion, allowing to manage huge amounts of data and serve a great number of advanced analytics.

The core of the system run analytical Python models deployed on Docker containers orchestrated by Kubernetes, that was initially involved for the scalability of the front end web app only. Each model can be seen as a single service comprehensive of its own data and interface. In particular the communication between services is kept in an asynchronous way that can be managed by a Kafka broker. Kubernetes coordinates the services and manage their lives, spikes of requests instantiating more instances in an elastic mode and load balancing too. The graph processing model container embeds Neo4J noSql database as planned for the previous scenario, but the instance is volatile as the container itself.

Apache Nifi is going to play the role it was supposed to, meaning that it is configured to run the ingestion crawlers through bash scripts invoking the Docker containers hosting the crawling functionalities. At the moment Nifi is also responsible for indexing the collected contents into Elasticsearch for Siren tools.

Kafka is the data broker connecting all of the inputs and outputs from the different containers. Plus, it helps in aggregating analysis results, asynchronously provided by the different analysis module, thanks to KSQL capability, that queries Kafka topics according to a sliding time window, and finally delivering aggregated results into ES.

Hadoop and Spark originally chosen as the main components for the storage and processing respectively, have left the roles to Elasticsearch for the indexed contents and Python code for the algorithms.

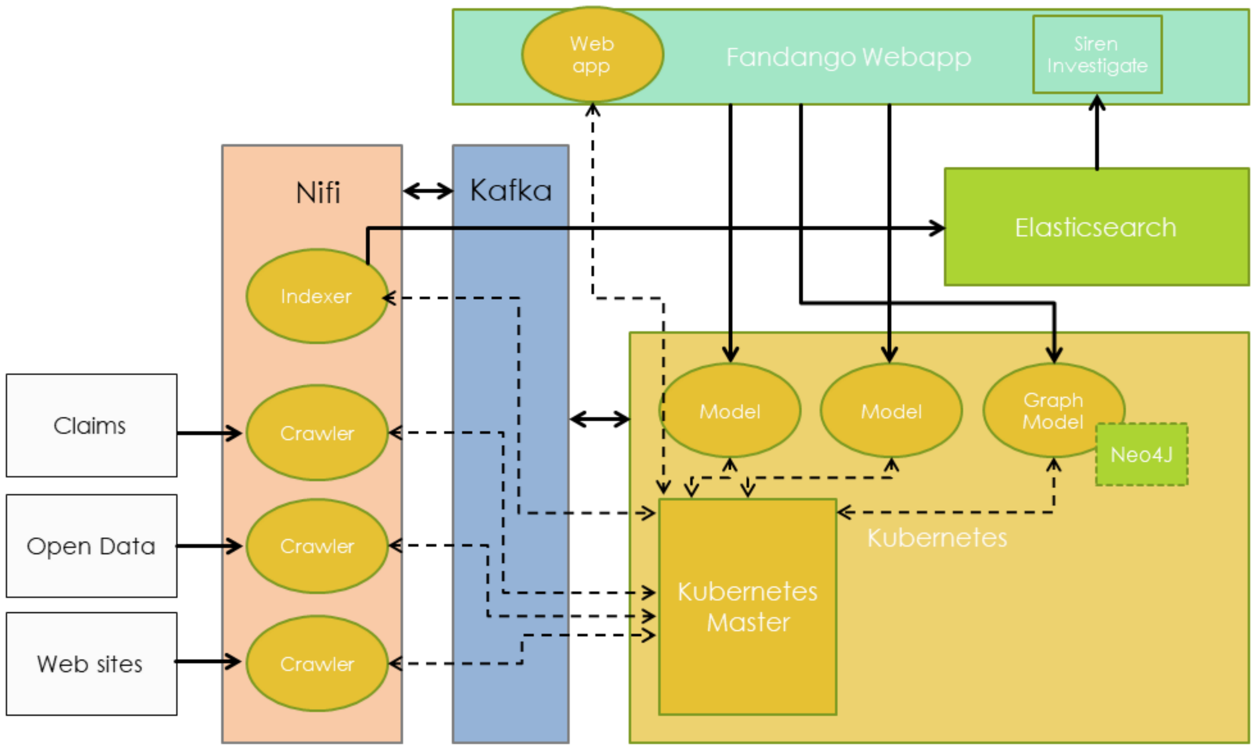


Figure 3 - Architecture Overview

FANDANGO's features to support journalist in fake-news detection and verification, as well as scoring the news with different trustworthiness scores, requires the development of several different big data processing and analyzing techniques. To optimize the solution and better comply to software quality standards, such as: Functional Suitability, Reliability, Operability, Performance Efficiency, Security, Compatibility, Maintainability and Transferability, FANDANGO relies on well-established products that were brought together to form the proposed architecture. The components of the architecture, which needs to be integrated are described on Table 1 – Architecture Components Overview.

SOFTWARE	DESCRIPTION
NiFi	Data flow ingestion tool, open source, distributed and scalable, to model real-time preprocessing workflow from several different sources. It hosts the crawlers as well as the indexing processor.
Kafka	Publish-subscribe distributed messaging system, that grants high throughput and back pressure management. This is the tool FANDANGO uses to connect the different components
Elasticsearch + Siren Federate	Distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Siren Federate plugin is added to Elasticsearch to allow data set semi-joins and seamless integration with different data sources.
Siren	Investigative Intelligence UI with connectivity to Elasticsearch, whose aim is to allow reporting, investigative analysis and alerting to users based on the

	indexed contents.
Rest APIs, RSS, Web Sites, Open Data, Social network	Data sources of the FANDANGO project. Specific crawlers will connect to these sources of data to get the information needed to verify the news.
FTP	The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network. In our Architecture it is where Users can place files that will be than ingested in the data lake.
Web App	Access point to FANDANGO Infrastructure. The journalist will use the FANDANGO Web application to insert news and verify the trustworthiness of specific news or media.
Docker	Docker is used to run software packages called "containers". Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels.
Kubernetes	Kubernetes is an open-source container-orchestration system for automating deployment, scaling and management of containerized applications
Neo4J	Graph NoSQL database embedded in the Graph Analysis Container as single instance

Table 11 – Architecture Components Overview

While the overall FANDANGO solution requires all aforementioned components to work in conjunction, only some of those components will store and share structured data across the platform, thus requiring a clear data model structure in order to achieve such goals.

The components highlighted, in green, on Figure 1 identify the specific parts of FANDANGO architecture where structured data will be stored and shared, either with temporary or long-term persistence. These components are essential for long-term sustainability of the solution and will handle the majority of the data required for machine learning processes, knowledge-graph analysis, semantic interpretation and user interaction.

FANDANGO Data Model was defined in D2.2. Due to the adoption of an iterative prototyping approach, section 3 of this document contains all data model updates.

3.1. STREAMING PIPELINE

To meet FANDANGO's user requirements, a streaming data pipeline has been implemented to continuously ingest and process (i.e. trustworthiness scores evaluation) articles crawled from a list of "monitored" websites. Actually, to build ground truth, graph models and data lake, FANDANGO have to support the ingestion of different types of data: articles (including embedded images and videos); open data metadata and claim reviews. End users selected and defined a list of specific News websites to ingest articles; a list of Open Data repositories to gather and indexing information about the available open data dataset, in order to offer fast access to factual information about important topics, like climate change, immigration, and European politics; a list of fact checking sites to gather claim reviews in order to allow searching capabilities for claim verification.

The streaming pipeline dataflow is represented in the following diagram.

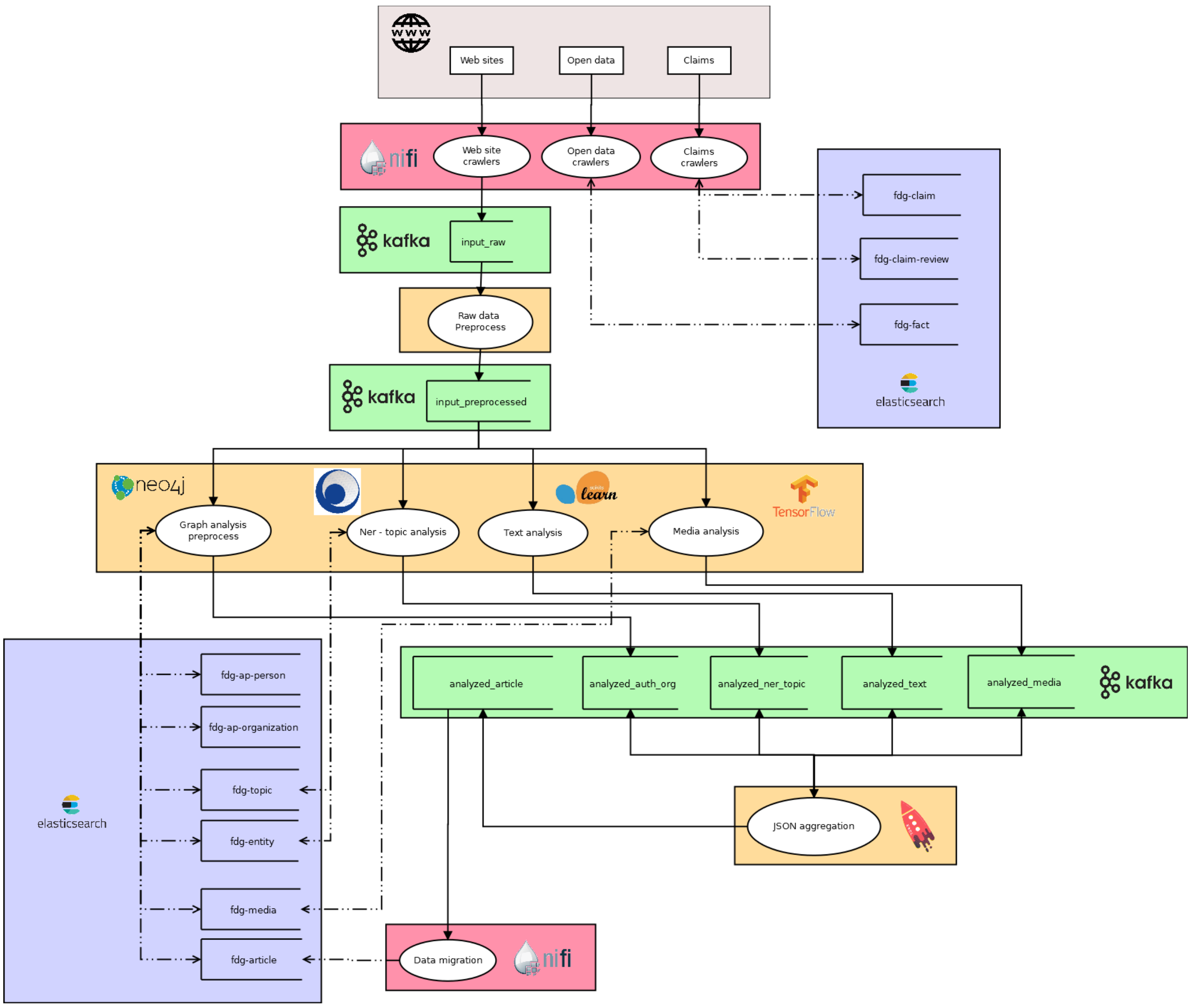


Figure 4 – Pipeline DataFlow

There are four data analysis services: Text analysis, Entities (NER) and topics analysis, Graph analysis and Media analysis. Since images and videos analysis are much more time consuming than text and graph analysis, Kafka has been used to handle asynchronous analysis results, defining the following topics:

Kafka Topic	**Written by**
input_raw	Crawlers

input_preprocessed	Raw data preprocessor
analyzed_text	Text body analyser
analyzed_auth_org	Author, organization graph analysis
analyzed_ner_topic	Entities and topic extractor
analyzed_media	Image and video analyser
analyzed_article	KSQL aggregation

All analysis results are joined and aggregated in the topic analyzed_article using KSQL, the streaming SQL interface for Kafka. The topics involved in the join are:

- analyzed_text
- analyzed_auth_org
- analyzed_media
- analyzed_ner_topic

KAFKA TOPICS DATA STRUCTURE

Note: Data are exchanged in JSON format.

topic: input_raw

Field	Type	Example
identifier	String	j123gjkclsa92kdjlsi0c
source_domain	String	www.repubblica.it
title	String	Economia e Finanza con Bloomberg
summary	String	Lorem Ipsum ...
description	String	Lorem Ipsum ...
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://...”]

url	String	www.newssite.com/economia
texthash	String	j123gjkclsa92kdjlsi0c
fakeness	String	good
keywords	List of String	[“repubblicait”, ...]

topic: input_preprocessed

Field	**Type**	**Example**
identifier	String	j123gjkclsa92kdjlsi0c
headline	String	Economia e Finanza ...
articleBody	String	Lorem Ipsum ...
images	List of String	[“http://...”, “http://...”]
videos	List of String	[“http://...”, “http://...”]
dateCreated	Date	2019-02-19 14:45:13
dateModified	Date	2019-02-19 14:45:13
datePublished	Date	2019-02-19 14:45:13
publishDateEstimated	String	yes
author	List of String	[“Name Surname”]
publisher	List of String	repubblica
language	String	it
sourceDomain	String	www.repubblica.it
country	String	Italy
nationality	String	Italian
url	String	www.repubblica.it/economia
calculateRating	Double	-99
calculateRatingDetail	String	“”
fakeness	String	good

topic: analyzed_text

Field	**Type**	**Example**
identifier	String	j123gjkclsa92kdjlsi0c

headline	String	Economia e Finanza ...
articleBody	String	Lorem Ipsum ...
dateCreated	Date	2019-02-19 14:45:13
dateModified	Date	2019-02-19 14:45:13
datePublished	Date	2019-02-19 14:45:13
publishDateEstimated	String	yes
textRating	String	95.4
url	String	http://www.repubblica.it/economia
sourceDomain	String	www.repubblica.it
features_text	List of Object	[{'title_AVGWordsCounter': 0.1940298507, 'title_PunctuationCounter': 0.0298507463....}]
inLanguage	String	it

topic: analyzed_auth_org

Field	**Type**	**Example**
identifier	String	j123gjkclsa92kdjlsi0c
author	List of String	["asdaij21231", "12i3oisakjasd"]
publisher	List of String	["sjduahu123", "12kjnjijdoadaaa"]
authorRating	List of Double	[99.0, 21.25]
publisherRating	List of Double	[10.55,12.44]

topic: analyzed_ner_topic

Field	**Type**	**Example**
identifier	String	j123gjkclsa92kdjlsi0c
about	List of String	["topic09it", "topic41it"]
mentions	List of String	["sda123sda", "123313dsff", "231sdasd"]

topic: analyzed_media

Field	Type	Example
identifier	String	j123gjkclsa92kdjlsi0c
contains	List of String	[“a231uha231”, “213khhias”, ...]
mediaRating	List of Double	[68.0, 23.0, ...]

topic: analyzed_article

Field	Type	Example
identifier	String	j123gjkclsa92kdjlsi0c
headline	String	Economia e Finanza ...
articleBody	String	Lorem Ipsum ...
inLanguage	String	it
url	String	http://www.repubblica.it/economia
sourceDomain	String	www.repubblica.it
dateCreated	Date	2019-02-19 14:45:13
dateModified	Date	2019-02-19 14:45:13
datePublished	Date	2019-02-19 14:45:13
publishDateEstimated	String	yes
author	List of String	[“asdaij21231”, “12i3oisakjasd”]
publisher	List of String	[0.55, 0.44]
calculatedRatingDetail	Dictionary	{ “textRating”: 77.0, “mediaRating”: [44.5, 22.3], “authorRating”: [88.3, 99.0], “publisherRating”: [99.0, 88.0] }
calculatedRating	Double	98.55
about	List of String	[“djnad8asd1”, “sdkj129kskd”]

mentions	List of String	[“sda123sda”, “123313dsff”, “231sdasd”]
contains	List of String	[“a231uha231”, “213khhias”, ...]
processType	String	offline

4. COMPONENT SPECIFICATIONS

This section contains the description and specification of all FANDANGO’s components.

4.1. DATA CRAWLERS

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. There are even cases where parts of the news source’s website is structured differently than the rest of the website.

Even though some type of information is useful for the Fandango project, it does not mean that the website that is crawled offers that kind of information. Additionally while a specific structure is followed by a website, there is no assurance that a web page from the site uses includes all the information. The Fandango crawlers handle unfound information by using default empty strings and lists with the exception of the publishing date. The publishing date of an article is so important for detecting temporal flow of information that a publishing date is always provided. If the date is not explicitly found in the webpage, the crawling date is used as publishing date as one can be certain that at this date the news item was definitely published. In order to clarify to the end user if the date was explicitly found or inferred, an additional field is populated with a notification.

Many web sites include advertisements in their web pages. In order to avoid following links to these advertisements we added a list of regular expressions to filter outgoing links. Additionally, content that is included in the web page through links is also filtered using the same regular expressions. This allows to filter most of the advertisements in image format. A second filter for advertisement images is applied to remove images that are too small to show actual content or have very disproportionate width to height ratios like banners.

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 offer a summary of the review also in a crawler friendly json format that follows schema.orgs claimReview schema. Fandango’s crawlers gather the claim reviews that follow the specification so that users can quickly find relevant to their query 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 that aim to extract the type of information that is provided in each data source of each database and the spatial and temporal scope of the data. Thus, the user can quickly detect factual data relevant to the query they are performing.

4.1.1. Component Crawlers

The crawlers component iteratively gathers information for all three entities: articles, claim reviews and facts. The components use Scrapy, a python scraping and web crawling framework. For each entity type a different scrapy project has been created and integrated in a Docker container to make easier the integration in the Fandango Platform.

For website sources, articles are parsed using the newspaper3k library. Newspaper3k is a python library for extracting and curating articles. This library can extract information such as authors, publishing date, article title and body, the language the article is written in, images and videos. A general case spider was created to handle web pages from sources where the newspaper library could extract all relevant information. For the sources where the newspaper library fails, a source specific spider was defined to handle the requests.

The general spider starts crawling using a user provided seed of starting urls for the desired sources. The spider then follows all links that are extracted from those seed pages that point inside the desired source domains. The crawling is performed in a breadth-first, first-in-first-out order so that at all times there is a relative balance in the number of articles from each source.

For each article we create hashes based on the url to use as article identifiers and of the article text so that we can check if a web page has been edited since the last visit. To better take advantage of the elasticsearch storage of the articles we have developed middlewares for the scrapy framework. These middlewares use an elasticsearch index to store the requests of the scrapy scheduler and also use the same index to detect if a request is already scheduled to be served. Additionally the same index can be used to iterate through all the requests subsequent times and keep a count of how many times each url has been visited.

For claim reviews the pages are parsed for data that follow the well defined claimReview shema. The 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 syntaxes 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 reviewed, 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. This way we can associate claim reviews and therefore claims, to articles

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.

4.1.2. Service Specification

The offline data crawling components include a list of default starting urls for the spiders to crawl. These starting urls are the root urls of the domains that the users of the Fandango platform have suggested to be crawled.

From inside the docker container, the crawling process starts by calling the command:

```
scrapy crawl <name_of_spider> -a domain=<url> -s <setting>=<value>
```

As is expected, the crawler adds the url provided by the argument to the request scheduler and starts processing requests. If the crawling process is stopped, the next time that it is started it will continue crawling using also the requests stored in the scheduler. In order to clear the scheduler the setting SCHEDULER_FLUSH_ON_START must be set to False, either by changing the value in the settings file or by adding the setting in the command call.

Each request that is processed outputs a document in the specified kafka topic, which is named input_raw. The format of the document is described in the Batch Elaboration section in the **input_raw** Kafka topic.

The online data crawling component is a REST service that allows the parsing of a single web page. The specifics of this service are provided in the following table.

Call method	**POST**
Headers	Accept: application/json Content-type: application/json
Entry point	/api/retrieve_article
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	url

Output (Json format)	authors data_created date_modified data_published publish_date_estimated description identifier images videos keywords language source_domain spider summary text texthash title top_image url
Json example output	{"**authors**":["Juan Diego Quesada","Álvaro García / EP"],"**date_created**":"2019-07-01T15:58:16Z","**date_modified**":"2019-07-01T15:58:16Z","**date_published**":"2019-05-29T00:00:00Z","**description**":"Pepu Hernández dice que trabaja para aislar a la extrema derecha y ha puesto en el tejado de Cs conformar una mayoría equilibrada en la ciudad","**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**keywords**":["que","como","la","el","y","madrid","los","carmena","alcaldesa","intentará","ha","repetir","más","su","se","en"],"**language**":"es","**publish_date_estimated**":"no","**source_domain**":"elpais.com","**spider**":"online","**summary**":"La exjueza de 75 años reapareció en el acto de la colocación de una placa en la calle Argensola en recuerdo del pintor Eduardo Arroyo (1937-2018).\n\"En este momento estamos en una situación e","**text**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram","**texthash**":["4a8da3193d0c42343b3395e086294bd87b80cbef075c91fb87fcf15cbb0d3c81"],"**title**":"Carmena intentará repetir como alcaldesa de Madrid","**top_image**":"https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_

	normal.jpg","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html", "**videos**":[]}

4.2. Data Preprocessing

The main goal of this service is to both clean the data provided by the crawlers and provide an adequate format to the collected information before being ingested in Elasticsearch. To do so, several procedures including machine/deep learning techniques are required to ensure the quality of different input fields such as authors, source domain, language or dates.

More specifically, the set of crawlers collect all the authors that are assumed to be in the article. However, in many scenarios, they are not capable to distinguish between the name of the author and other useless tags that are associated to them in the website. Hence, this service contains a deep learning model that is capable of distinguishing between human names in a particular sentence. This family of models is well-known as Named Entity Recognition (NER). Despite the potential of these algorithms, they are limited by the language to be applied in the sense that, if one desires to detect human names in Spanish, the model is needed to be trained with spanish information and so on. To avoid this situation in the project, we use a Multilingual model to support this task. Moreover, a previous cleaning stage is also required to avoid ingesting noise in the model. However, in some articles, the authors may be accomplished by additional information, for example: "Andrea Di Matteo (Support Andrea Di Matteo)". In this case, the NER model will not work if we directly introduce the name into it. To solve the problem, we use regular expressions to remove dirty information from the author name. Doing this step, we will have "Andrea Di Matteo" as the input of the model, and finally, it will return that this entity is a person.

Detecting the language is also mandatory since in many occasions the crawlers cannot capture it directly from the website, so that, the preprocessing is required to check this problem. To do so, the service employs a deep learning model for Automatic Language Identification (ALI) based on the title and the text of the article in such cases where the crawlers are not capable of collecting it.

Next, the name of the publisher is also required in order to have more information about the source of the article, for instance: elpais.com. In this case, we have made use of Whois IP which is a powerful tool to get information of a particular domain. or IP address. By employing Whois IP, we collect the country code where the domain is registered , the name of the website, the postal code etc. In our case, we are interested in the name and the country for the posterior analysis. On the other hand, the main limitation of this tool lies in the lack of information regarding certain domains. In such scenarios, we have implemented a robust method to complete the required information. This procedure analyses the so-called AllMediaLink website (https://allmedialink.com/), where all the media sources for many countries is available. In this case, we collect all the media sources from Spain, Italy, Greece, United States, Belgium and the United Kingdom. However, it can be augmented whether more data is needed. The second key factor in this stage is the Levenshtein distance which is employed to measure the similarity between the input source domain and the media sources extracted from AllMediaLinks. Finally, taking the maximum argument and using a threshold, we are capable of extracting properly the name of the publisher. Finally, to get the nationality of the domain, we use the REST Country API which receives a country code and yields all the information associated to it (country name, nationality etc).

Additional methods are used to discard those articles whose text or title fields are empty probably because of they are not articles indeed but other sorts of sources that the crawlers collected.

Finally, the dates are also reviewed in order to avoid unexpected formats. The format that is used in this case is based on Coordinated Universal Time (UTC) to mitigate problems associated with the time zone of each country.

4.2.1. SERVICE SPECIFICATION

The preprocessing service is developed in a Python Web Framework named as Flask http://flask.pocoo.org/() and has been integrated in a Docker file to make easier the integration in the Fandango Platform.

The main breakthroughs of this service is the management of the author names, the publisher name extraction and the incorporation of both country and nationality regarding the publisher. On the other hand, a set of unused fields are removed along the process.

In the following table, the GET request that launches the offline service is presented together with a specific example of the input and the output of such service.

Call method	**GET**
Headers	Accept: application/json Content-type: application/json
Entry point	/preprocess/batch/start
(Input) Read From	Kafka queue *input_raw*
(Output) Writes Into	Kafka queue *input_preprocessed*
Json example input	{"**authors**":["Juan Diego Quesada","Álvaro García / EP"],"**date_created**":"2019-07-01T15:58:16Z","**date_modified**":"2019-07-01T15:58:16Z","date_published":"2019-05-29T00:00:00Z","**description**":"Pepu Hernández dice que trabaja para aislar a la extrema derecha y ha puesto en el tejado de Cs conformar una mayoría equilibrada en la ciudad","**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**keywords**":["que","como","la","el","y","m

	adrid","los","carmena","alcaldesa","intentará","ha","repetir","más","su","se","en"],"**language**":"es","**publish_date_estimated**":"no","**source_domain**":"elpais.com","**spider**":"online","**summary**":"La exjueza de 75 años reapareció en el acto de la colocación de una placa en la calle Argensola en recuerdo del pintor Eduardo Arroyo (1937-2018).\n\"En este momento estamos en una situación e","**text**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram","**texthash**":["4a8da3193d0c42343b3395e086294bd87b80cbef075c91fb87fcf15cbb0d3c81"],"title":"Carmena intentará repetir como alcaldesa de Madrid","**top_image**":"https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html","**videos**":[]}
Json example output	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**headline**":"Carmena intentará repetir como alcaldesa de Madrid","**articleBody**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram\". ","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html","**language**":"es","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**videos**":[],"**dateCreated**":"2019-07-01T16:09:16Z","dateModified":"2019-07-01T16:09:16Z","**datePublished**":"2019-05-29T00:00:00Z","**publishDateEstimated**":"no","**author**":["Alvaro Garcia","Juan Diego Quesada"],"**publisher**":["El Pais"],"**sourceDomain**":"elpais.com","**country**":"Spain","**nationality**":"Spanish","**calculateRating**":-99,"**calculateRatingDetail**":""}

The online preprocessing service is employed when using the web interface designed by FANDANGO's partners. The main difference in this case lies in the fact that instead of using Kafka for communication purposes, the service receives the expected input in a POST request that is made by the web interface. As expected, the output of the online preprocessing service is the same as the offline since the functionalities and the purposes are exactly the same. In the following table, a description of the service specification is presented.

Call method	POST
Headers	Accept: application/json Content-type: application/json
Entry point	/preprocess/article
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	authors data_created date_modified data_published publish_date_estimated description identifier images videos keywords language source_domain spider summary text texthash title top_image url
Json example input	{"**authors**":["Juan Diego Quesada","Álvaro García / EP"],"**date_created**":"2019-07-01T15:58:16Z","**date_modified**":"2019-07-01T15:58:16Z","**date_published**":"2019-05-29T00:00:00Z","**description**":"Pepu Hernández dice que trabaja para aislar a la extrema derecha y ha puesto en el tejado de Cs conformar una mayoría equilibrada en la ciudad","**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**keywords**":["que","como","la","el","y","madrid","los","carmena","alcaldesa","intentará","ha","repetir","más","su","se","en"],"**language**":"es","**publish_date_estimated**":"no","**source_domain**":"elpais.com","**spider**":"online","**summary**":"La exjueza de 75 años reapareció en el acto de la

	colocación de una placa en la calle Argensola en recuerdo del pintor Eduardo Arroyo (1937-2018).\n\"En este momento estamos en una situación e","**text**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram","**texthash**":["4a8da3193d0c42343b3395e086294bd87b80cbef075c91fb87fcf15cbb0d3c81"],"**title**":"Carmena intentará repetir como alcaldesa de Madrid","**top_image**":"https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html", "**videos**":[]}
Json example output	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**headline**":"Carmena intentará repetir como alcaldesa de Madrid","**articleBody**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram\". ","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html","**language**":"es","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**videos**":[],"**dateCreated**":"2019-07-01T16:09:16Z","dateModified":"2019-07-01T16:09:16Z","**datePublished**":"2019-05-29T00:00:00Z","**publishDateEstimated**":"no","**author**":["Alvaro Garcia","Juan Diego Quesada"],"**publisher**":["El Pais"],"**sourceDomain**":"elpais.com","**country**":"Spain","**nationality**":"Spanish","**calculateRating**":-99,"**calculateRatingDetail**":""}

4.3. Multimedia Analysis

The multimedia analysis components are tasked with analyzing the multimedia modalities contained inside an article that is being analyzed. The analysis that is being performed can be separated into two main categories per modality, i.e. descriptors and manipulation detectors for images and videos. The descriptors aim to provide the other components of the platform with information about the visual content of media found in the articles, for example objects that appear in an image or the location where a video was shot. On the other hand, the manipulation detectors examine the available media in order to detect their authenticity and predict a probability that the media object was altered from its original form. The prediction for an image has per pixel detail and for a video per frame and per pixel.

In the world of deep learning multiple libraries exist to train and use a classifier or feature extractor. Additionally state of the art methodologies for different functions might be available using different version of a library. Therefore each of the modules needs to be packed in a separate docker container specifically composed to accommodate an individual module.

In order to provide a versatile system, we implemented three media analysis components that forward requests for analysis to the appropriate individual module; a component to call all image analyzers and descriptors, a component to call all video analyzers and descriptors and a component that can find images stored in the system data lake with semantic similarity to an input image.

The first two components offer services working in an asynchronous way. The services, through one endpoint accept the url to an image or video and return the identifier of the media object stored inside the Fandango data lake. Through a second endpoint they return the analysis given the media identifier, when it is completed. In order to avoid analysing the same media file multiple times, the identifier is based on the media uri.

** Based on the user requirements described in deliverables D2.3 and D2.4, the analysis results are drawn on top of the original media object

4.3.1. Component Multimedia Analysis Service Specification

All of the provided services for the multimedia components are RESTful and are built using the Flask python web framework. A detailed description of each endpoint is presented in the following tables.

Call method	**POST**
Headers	Accept: application/json Content-type: application/json
Entry point	/api/media_analysis
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	identifier images videos

Json example input	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**videos**":["https://www.youtube.com/watch?v=8sJBWIUrWfE"]}
Json example output	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa", "**contains**": ["fc3ce5ddcf6ca40e6e38e0ef30b3d2802ed2961be4599b", "3d2802ed2961be4599b22ac8ed3526a661be459ac8ed3", "ce5ddcf6ca46e30ef30b3d28061be4592ed2961be4599b"]}}

Call method	**POST**
Headers	Accept: application/json Content-type: application/json
Entry point	/api/analyze_image
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	url

Json example input	{"**url**":"https://ep0.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"}
Json example output	{"**identifier**":"fc3ce5ddcf6ca40e6e38e0ef30b3d2802ed2961be4599b","**status**": "started"}

Call method	**GET**
Headers	Accept: application/json Content-type: application/json
Entry point	/api/analyze_image/<image_id>
(Output) Writes Into	Yields a JSON File

Json example output	{"**identifier**":"fc3ce5ddcf6ca40e6e38e0ef30b3d2802ed2961be4599b", "**mediaRating**": 57.14, "**mediaRatingDetails**": {"analyzer": "faceForensics", "bboxes": [[0.2, 0.3, 0.7, 0.4]], "scores": [0.4286]}, "**display**": [{"analyzer": "faceForensics", "display": <base64 encoded image>}, …, {"analyzer": "busterNet", "display": <base64 encoded image>}]}

Call method	**POST**
Headers	Accept: application/json Content-type: application/json
Entry point	/api/analyze_video
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	url
Json example input	{"**url**":"https://www.youtube.com/watch?v=8sJBWIUrWfE"}

Json example output	{"**identifier**":"ce5ddcf6ca46e30ef30b3d28061be4592ed2961be4599b","**status**": "started"}

Call method	GET
Headers	Accept: application/json Content-type: application/json
Entry point	/api/analyze_video/<video_id>
(Output) Writes Into	Yields a JSON File
Json example output	{"**identifier**":"ce5ddcf6ca46e30ef30b3d28061be4592ed2961be4599b","**status**": "done", "**mediaRating**": 57.14, "**mediaRatingDetails**": {"analyzer": "faceForensics", "bboxes": [[0.2, 0.3, 0.7, 0.4]], "scores": [0.4286]}, "**display**":[{"analyzer": "faceForensics", "display": "http://localhost/videos/ce5ddcf6ca46e30ef30b3d28061be4592ed2961be4599b.mp4"}, …, {"analyzer": "busterNet", "display": "http://localhost/videos/ce5ddcf6ca46e30ef30b3d28061be4592ed2961be4599b.mp4"}]}

Call method	POST
Headers	Accept: application/json Content-type: application/json
Entry point	/api/similar_images

(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	url
Json example input	{"**url**":"https://ep0.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"}
Json example output	{"**identifier**":"fc3ce5ddcf6ca40e6e38e0ef30b3d2802ed2961be4599b","**similar**": ["started"]}

All five services are available to the online pipeline and are included in a single docker container. From inside the container the command to activate the service is:

```
python run_online_service.py --host <host ip> --port <port number>
```

The above described services use a settings file for the definition of the individual descriptors and classifiers that are being used by the system. For each analyzer that is being used, the appropriate host and internal endpoint is defined. By using this modular system of request forwarding, we allow the system to transparently exchange classifiers when better state of the art models are available. Additionally, by setting a reverse proxy to handle the endpoints, one can deploy multiple instances of the services and perform load balancing between them.

For the offline pipeline we have developed a python script, contained in docker, that reads the appropriate kafka topic, i.e. input_preprocessed, calls the online service and writes the reply to the appropriate kafka topic, i.e. analyzed_media.

4.4. COMPONENT GRAPH ANALYSIS

This service is in charge of investigating the fake news detection by providing a trustworthiness score to both authors and organizations that are involved in the release of a piece of news. The higher the score indicator is, the higher the level of confidence for such entity will be. The objective of the service is therefore, to promote the fake news detection by incorporating the adding value of measuring the credibility of the entities that are involved in the media industry. This service has been deployed using parallelize processes to speed up the distinct operations.

Firstly, the service reads documents from Kafka queue *input_preprocessed* and it first ingests the authors and organizations in Elasticsearch ensuring that they are not duplicates. Elasticsearch provides the identifiers for all the elements that were ingested. Once the identifiers are available, they are stored in a queue to be processed by another thread that will perform the graph analysis operations.

This second thread starts reading from such queue and builds the graph including nodes and relationships throughout CYPHER, the query language that is wide employed in NEO4J.

Regarding the graph analysis, there are different families of algorithms that are investigated for the purpose of the analysis: *Centrality algorithms* and *Similarity* methods. Centrality procedures are used to determine the level of influence of the node in FANDANGO's network. This indicator measures how important a certain node is in a particular network and in this case, the ArticleRank was selected to be used in this calculation. On the other hand, the service determine the Eigenvector Centrality algorithm to consider the transitivite importance of a node in a graph, rather than only considering its direct importance as the ArticleRank does. Next, Similarity algorithms such as the Jaccard or Cosine are used to find similar patterns between publishers and authors and are used to infer in cases where the information available for a certain author or publisher is limited and a score is provided based on similar entities to them.

Finally, additional features such as anonymous authors or unknown country regarding the source domain are also considered in the final score since they can provide useful information to the result. Hence, the total score of a node is determined as a weighted calculation of the aforementioned procedures.

Once the scores are calculated, the service update them into Elasticsearch in the adequate index to be available in future analysis as well as in the user interface.

4.4.1. Graph Analysis Service Specification

The graph analysis service is developed in Flasck, a Python Web Framework and has been integrated in a Docker file to make easier the integration in the Fandango Platform as in the aforementioned services.

Moreover, for automatically scheduling the service, a launch file has been developed to start the docker every day without a manually implication of any technical partner. In addition, a complete folder with the logs of the process is also stored in the docker for tracking unexpected behaves or errors during the process of the service. This launch file together with the docker are stored in the FANDANGO's platform. More specifically, this service executes a HTTP GET request in order to start the procedure.

Once, you are in the proper directory where the docker is stored, the command to start the docker where the graph analysis service (both online and offline) are integrated is the following:

```
launcher.sh --start
```

Using this command, we ensure that both services are ready. Internally, the launcher file contains the following HTTP POST request to start the process.

As in the rest of the offline services, the Graph analysis offline service reads from Kafka the documents that are cleaned by the preprocessing offline service and after its performance, the service puts the data into the corresponding kafka topic, which is named as *analyzed_auth_org*. Thus, as soon as a new document is ingested in the queue, the offline service starts reading it and applying its corresponding methods. As a result, the input of the service is a document stored in the Kafka queue *input_preprocess* and whose format is defined in the table below.

As it is expected, the format corresponds to the one explained in the *Batch Elaboration section*. On the other hand, the output specification of the service is described in the aforementioned section related to the *input_preprocessed* Kafka topic. An example of the output of the service is provided in the table below.

In the following table, the GET request that launches the offline service is presented together with a specific example of the input and the output of such service. The output of the service contains the identifier of the article, as well as the identifiers of the authors as well as the organization. Additionally, it provides the trustworthiness score for the entities involved in the creation of the article. These scores are float numbers between 0 and 100 where 0 indicates the lowest credibility whereas 100 refers to the opposite.

Call method	GET
Headers	Accept: application/json Content-type: application/json

Entry point	/graph/batch/start
(Input) Read From	Kafka queue *input_preprocessed*
(Output) Writes Into	Kafka queue *analyzed_auth_org*
Json example input	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa","**headline**":"Carmena intentará repetir como alcaldesa de Madrid","**articleBody**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram\". ","**url**":"https://elpais.com/ccaa/2019/05/29/madrid/1559125923_282323.html","**language**":"es","**images**":["https://ep00.epimg.net/t.gif","https://ep00.epimg.net/ccaa/imagenes/2019/05/29/madrid/1559125923_282323_1559140930_rrss_normal.jpg"],"**videos**":[],"**dateCreated**":"2019-07-01T16:09:16Z","dateModified":"2019-07-01T16:09:16Z","**datePublished**":"2019-05-29T00:00:00Z","**publishDateEstimated**":"no","**author**":["Alvaro Garcia","Juan Diego Quesada"],"**publisher**":["El Pais"],"**sourceDomain**":"elpais.com","**country**":"Spain","**nationality**":"Spanish","**calculateRating**":-99,"**calculateRatingDetail**":""}
Json example output	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa", "**author**": ["MICT-GoBu09LVEn7SCPg"], "**publisher**": ["MYCT-GoBu09LVEn7SiNn"], "**publisherRating**": [10.25], "**authorRating**": [21.45]}}

ONLINE PROCESS

The graph analysis service is also called by web app designed by FANDANGO's partners to provide a credibility scoring to both authors and publishers associated to the selected article. In this case the service receives the expected input in a POST request that is made by the web interface. However, in order to avoid time delays in the online process, the scores that are provided in the output are the ones corresponding to the last update in Elasticsearch. Otherwise, the service should wait until the computation of the graph analysis is finished. Finally, once the graph analysis is done, the new scores are updated in Elasticsearch so that, the next time the user wants to analyse the same author or publisher, the score will have changed.

4.5. Component Topics and NER

It is composed by Topic (string) and Entity (string)

The topics and named entities component is tasked with providing the pipeline with information about the content that is included in the text modality of an article, claim or fact. The component offers a REST service built with the Flask python framework. For fast and easy deployment the component is dockerized. Once inside the docker container the service can be activated by:

```
python run_service --host <host ip> --port <port number>
```

The necessary input to the service is the cleaned text that is used in the article as well as the detected language the article is written in. The output of the service is two lists of identifiers, the one corresponding to all the entities that were detected in the text and the other to the top topics that are discussed in the text based on the machine learning algorithm that was developed and is described in deliverable D4.1.

If the identifier field, which is the id of the article, is included in the input JSON it is copied back to the output JSON. This way the caller does not have to remember which article is being analyzed and can schedule an asynchronous call plan.

4.5.1. Topics and NER Service Specification

The service is deployed offering one endpoint which is described in the following table.

Call method	POST
Headers	Accept: application/json Content-type: application/json
Entry point	/api/extract_topics
(Input) Reads From	JSON File provided in the POST body
(Output) Writes Into	Yields a JSON File
Body (Json format)	articleBody language

Json example input	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa", "**articleBody**":"Pepu Hernández (izquierda) y Manuela Carmena, este miércoles durante la colocación de una placa en homenaje al pintor madrileño en Instagram\\". ","**language**":"es"}
Json example output	{"**identifier**":"ab49ec778314099e0fc3ce5ddcf6c2ca40e6e38e0ef30b3d2802ed2961be4599b22ac8ed3526a609b18dabc203f7b5fbcf3504ac4192f31cdd385c8431fbcbaa", "**about**": ["topic12es", "topic31es", "topic02es", "topic27es", "topic29es"], "**mentions**": ["cf3504ac4192f31cdd385c8431fbcbaa", ..., "783199e0fc3ce5ddcf6c2ca40e6e3cb8"]}

For the offline pipeline we have developed a python script, contained in docker, that reads the appropriate kafka topic, i.e. input_preprocessed, calls the online service and writes the reply to the appropriate kafka topic, i.e. analyzed_ner_topic.

4.6. Component Text Analysis

This component provides a prediction, whenever an article is reliable or not, analyzing its text and title. In order to achieve it, a machine learning classifier has to be trained and validated with thousand of labeled articles consider as a sample set. To support multilinguality, different ground truth have been built to train an ML classifier for each language.

The prediction will be a probability score (trustworthiness) given a new article that has never been seen in the training phase. In order to predict a result, the ML model extracts a set of linguistic features not related directly with words meaning as described in detail in D2.4.

4.6.1. TEXT ANALYSIS SERVICE SPECIFICATION

As said previously, the main purpose of this service is to analyze an article syntax and stylographic structure to get as much as possible accurate classification result of the article, identifying its truthful or false nature.

This component text analysis (on line) is used by web app to answer to end-user needs. In other words, it's used to analyze the news, inserting a url into the interface.

It must be query using the following structure:

Call method	POST
Headers	Accept: application/json Content-type: application/json
Entry point	/textanalysis
Json Format	url headline articleBody dateCreated dateModified datePublished author publisher images video sourceDomain calculateRatingDetail calculateRating identifier language country nationality publishDateEstimated

Json example	{"**url**":"www.theguardian.com/world/2019/jul/02/chinese-media-calls-for-zero-tolerance-after-violent-hong-kong-protess", "**identifier**": "05353d6f40168e069ea3636ba8e1798f", "**headline**": "Joanna Yeates murder: Landlord Chris Jefferies could hold key?", "**articleBody**": "Could landlord hold the key to Joanna's murder? 'I saw her leave with two others and talking in hushed tones' In custody: Chris Jefferies, the landlord of Joanna Yeates, pictured yesterday. The landlord of murdered architect Jo Yeates watched as she left her flat with two people on the night she disappeared, it was claimed yesterday. ... I have full confidence in the police, father insists. The father of Jo Yeates yesterday backed the way police are conducting their inquiry, adding he feels sure they will catch her killer. David Yeates, 63, said he had complete faith in the investigation into his daughter\u2019s death. At his \u00a3600,000 home in Ampfield, Hampshire, he said: \u2018The police have been really helpful in this investigation and we have every faith in them. \u2018We all want to find whoever is responsible for Jo\u2019s death.\u2019", "**language**":"en","**images**": ["https://i.dailymail.co.uk/1s/2018/11/09/12/5979502-0-image-a-16_1541765781497.jpg", "https://i.dailymail.co.uk/1s/2018/11/09/17/5989332-0-image-a-13_1541782973758.jpg"], "**video**": ["https://www.youtube.com/v/null&hl=en&fs=1&rel=0&color1=0x3a3a3a&color2=0x999999"], "**dateCreated**": "2019-03-16 15:07:26", "**dateModified**": "2019-03-16 15:07:26", "**datePublished**": "2019-03-16 15:07:26", "**author**": ["Mark Prigg"], "**publisher**": ["Daily Mail"], "**sourceDomain**": "www.dailymail.co.uk", "calculateRating": -99, "**calculateRatingDetail**": "", "**publishDateEstimated**:True, "**country**" :"England","**nationality**":"english" }

The result output will have the same structure:

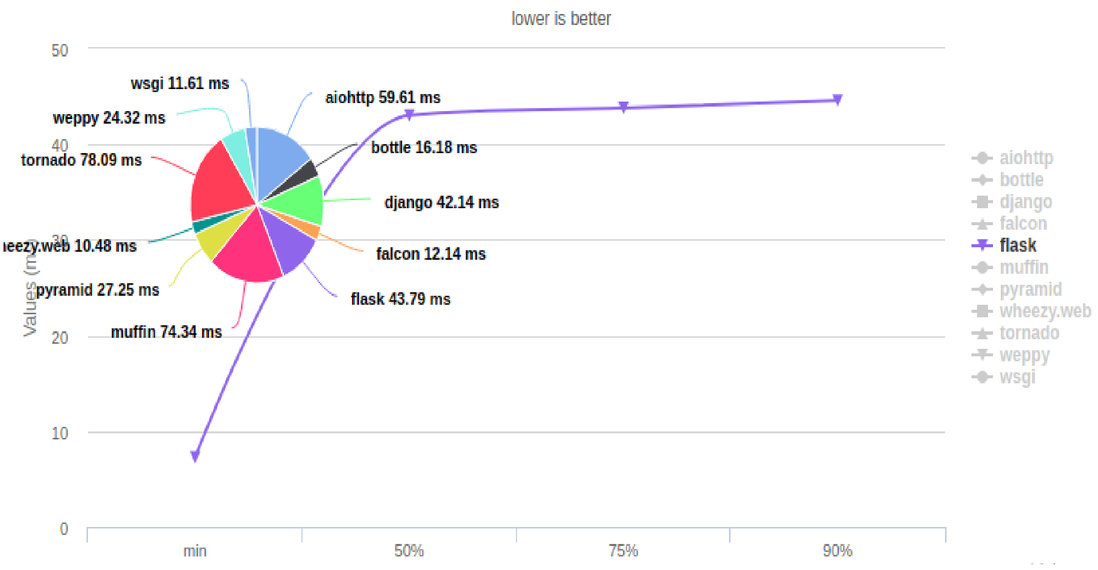
Json Format	identifier textRating
Json Example	{"identifier":"05353d6f40168e069ea3636ba8e1798f ", "textRating":"75.9"}

Communication specifications are fully related with Flask, the web framework used to interconnect all the modules in Fandango's user interface.

Some important information about the throughput of Flask are underlined in the following 2 images:

The average time to encode an object to json and return a response is 43.76 ms with 4630 supported requests, while the time on average to load a response from remote server and return a response is 3344.27ms with time with supported requested 18.15.

In the offline process this module won't be query by the web interface, but it use the kafka queue as broker manager. It implements a kafka consumer to retrieve the news to analyze. Data into kafka will have a similar structure as the ones called by the web application(on line), and the ML models are the same as the online part. The result of the analysis is put in an other kafka queue.



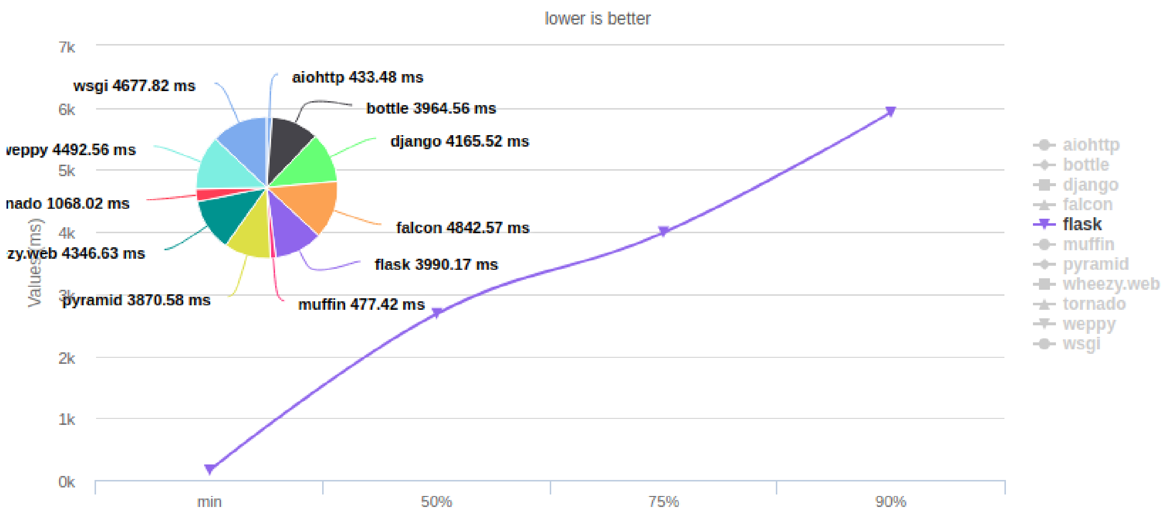


Figure 5 – Data view info

4.7. Component Fandango web APP

Web interface is the main tool for final users and whereas all the modules/components are integrated and connected among each other. The main features provided by Fandango web app are:

#	FEATURE	DESCRIPTION
1	News verification	Given an article, this service provides a full news analysis, returning scores to support or not its trustworthiness
2	Claim verification	It provides a list of claims related to a given statement
3	Image Verification	It verifies if an image has been modified
4	Video verification	It provides links to open data references related to a statementIt verifies if a video has been modified
5	Dashboard dynamic analysis	It supports a dynamic investigation of a news into Siren platform It verifies if a video has been modified

Table 12 –FANDANGO main features

All these features are implemented by services and their functionalities are fully described in this section of the document as well as in D2.4.

4.7.1. FANDANGO WEB APP SERVICE SPECIFICATION

Fandango user interface is composed by 5 different pages connected with a specific function, it has been thought and designed, reminding to some of the most famous search engines interface, in order to make it easy to navigate for final users.

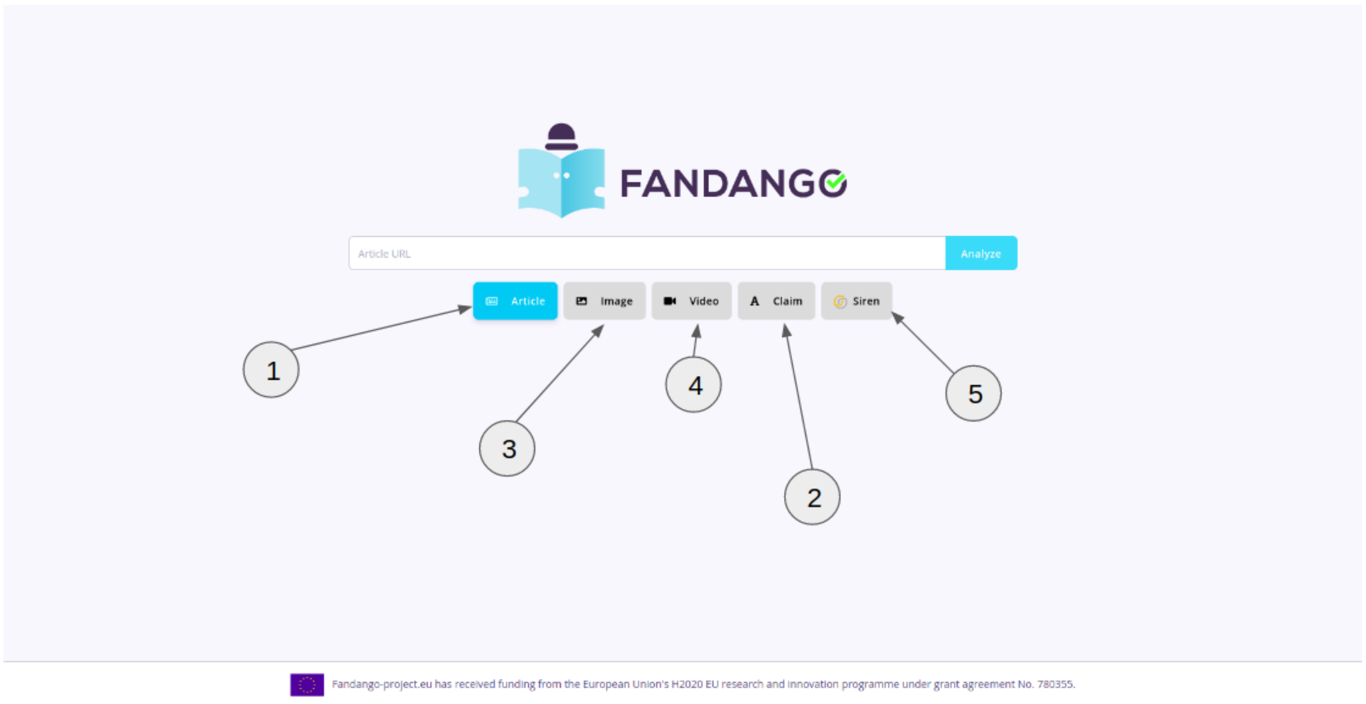
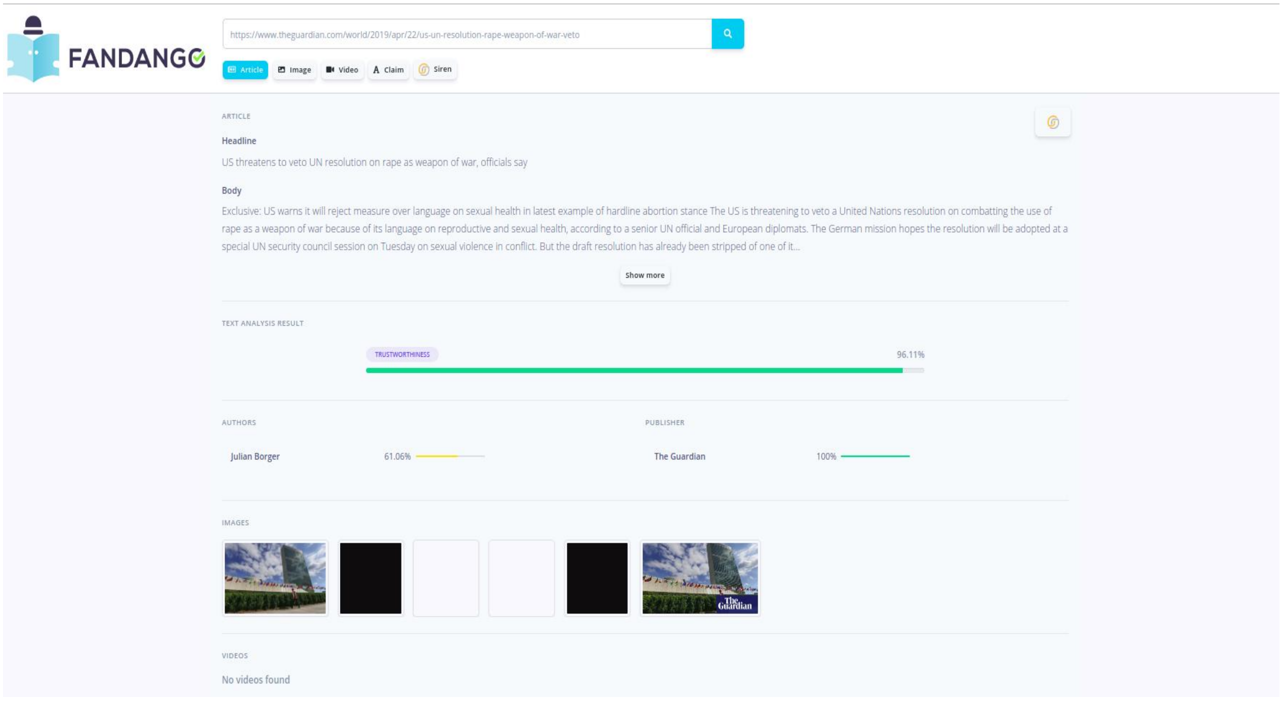

FANDANGO
Article URL
Analyze
Article
Image
Video
Claim
Siren
1
2
3
4
5
Fandango-project.eu has received funding from the European Union's H2020 EU research and innovation programme under grant agreement No. 780355.

Figure 6 – Web App

Fandango web interface communicates with the back-end side which in turn manages information exchanges among micro-services. This configuration makes easier integration since a service script is visible only to its owner and just an output and input format has to be defined a priori.

By clicking one of the buttons on the above image, a different input can be inserted into the white text area, for example pressing Article, the input has to be an article URL or pressing Claim, the output accepted is text.

Once a user's inserted a url and press analyze, the following page with results will appear:



https://www.theguardian.com/world/2019/apr/22/us-un-resolution-rape-weapon-of-war-veto

ARTICLE

Headline

US threatens to veto UN resolution on rape as weapon of war, officials say

Body

Exclusive: US warns it will reject measure over language on sexual health in latest example of hardline abortion stance The US is threatening to veto a United Nations resolution on combatting the use of rape as a weapon of war because of its language on reproductive and sexual health, according to a senior UN official and European diplomats. The German mission hopes the resolution will be adopted at a special UN security council session on Tuesday on sexual violence in conflict. But the draft resolution has already been stripped of one of it...

Show more

TEXT ANALYSIS RESULT

TRUSTWORTHINESS 96.11%

AUTHORS

Julian Borger 61.06%

PUBLISHER

The Guardian 100%

IMAGES

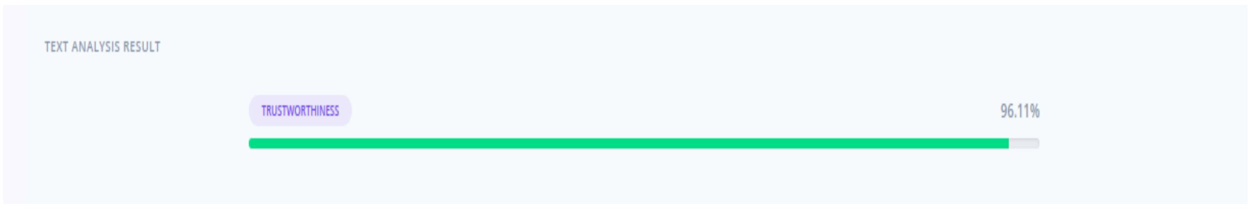
VIDEOS

No videos found

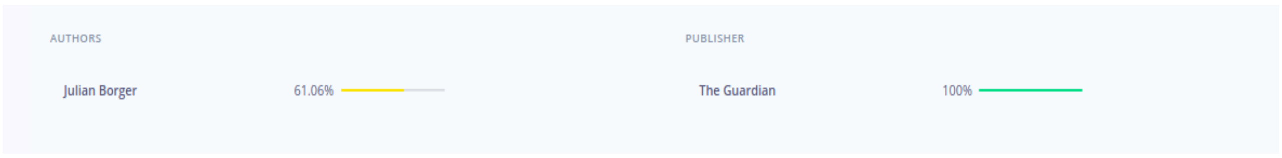
Figure 7 – Processing result page

It will provide a first of all, a result from the text analysis; then a score for each author and publisher, a list with videos and images related to the article that can be analyzed selecting one by one with a click and a table containing the most similar news related with the one in evaluation.

In particular, text analysis service provides a bar whereas report a probability of a news to be legitimate.

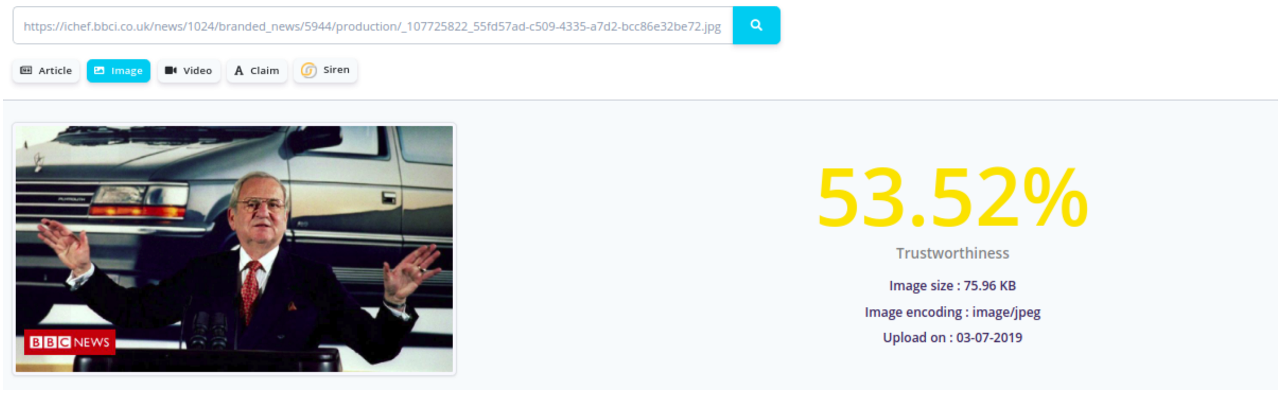


Same meaning are attributed to the authors and publisher bar.



Images and videos lists are shown as below, each object can be analyzed by clicking on it.

After clicking on an image or video, a new window will open and a screen as the following will show the image and a percentage, in the future, there are going to implement other details from images and videos analysis.


https://ichef.bbci.co.uk/news/1024/branded_news/5944/production/_107725822_55fd57ad-c509-4335-a7d2-bcc86e32be72.jpg
Article
Image
Video
Claim
Siren
BBC NEWS

53.52%

Trustworthiness

Image size : 75.96 KB

Image encoding : image/jpeg

Upload on : 03-07-2019

CONCLUSION

In accordance with the consortium decision to adopt an iterative prototyping methodology in carrying out research and development activities, this deliverable contains the updated revision of FANDANGO Architecture and Data Model as well as a first specification of the FANDANGO components.

It is worth to notice that this approach has led the consortium to the definition, design, implementation, testing and, most importantly, the evaluation of FANDANGO releases by end users.

As expected, Architecture and Data Model has been slightly changed to meet technical requirements.

Specifically, while conceptual Data Model has not changed, logical data model has been revised to meet analysis services' design details. Architecture has also been revised to meet the needs and technical requirements of the different analysis component developer partners. In order to satisfy specific requirements expressed by such a heterogenous team, with different backgrounds and practices, the overall architecture has slightly changed, keeping the same logic and philosophy. The most significant changes is the definition of a streaming pipeline using Kafka.

All FANDANGO components where specified - including the web app that implement FANDANGO's UI, with the exception of SIREN platform, that need just Data Model and Dashboard configuration.

Due to the iterative method adopted, we're expecting several changes in analysis services (trustworthiness scores) due to end users feedback on analysis results effectiveness.

Other changes will be reported in following deliverables (or interim report, if needed).