Grant Agreement No.: 761488

D3.1: Initial Design and APIs of Technology Bricks

This deliverable provides the first view of technology bricks connecting the CPN available technology to the user requirements on the one side and the planned platform prototypes on the other. This provides the first step for the planning of the CPN platform prototypes with the first one expected in the next three months.

The prioritization of the features to implement for the first prototype have been carried out in such a way as to respect the expected timing, but at the same time release meaningful functionalities, that will be improved and extended in the next prototypes.

The next step is the initiation of an Agile prototyping process which is expected to provide rapid results through two-week Sprint intervals.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Work package	WP 3
Task	T3.1
Due date	31/3/2018
Submission date	31/3/2018
Deliverable lead	ATC
Version	1.0
Authors	Marina Klitsi, Nikos Sarris, Efstratios Tzoannos, Stamatis Rapanakis (ATC)
Reviewers	Vincenzo Croce , Bosco Ferdinando (ENG)
Keywords	Technology Bricks, APIs

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.1	24/01/2018	Table of Contents	Marina Klitsi (ATC)
V0.2	23/3/2018	1st completed version including partners' contribution	Nikos Sarris, Stamatis Rapanakis, Marina Klitsi (ATC), Matthias Strobbe (IMEC), Michele Nati(DIGICAT), Bosco Ferdinando (ENG), Fulvio D'Antonio (LiveTech)
V0.3	25/3/2018	Edited final version including prioritisation of requirements	Nikos Sarris, Marina Klitsi (ATC)
V0.4	28/3/2018	Internal Review	Ferdinando Bosco, Vincenzo Croce (ENG)
V1.0	29/3/2018	Final version	Nikos Sarris, Marina Klitsi (ATC)

Co-funded by the Horizon 2020 Framework Programme of the European Union

DISCLAIMER

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 761488.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		**R**
Dissemination Level		
PU	Public, fully open, e.g. web	**X**
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to CPN project and Commission Services	

Co-funded by the Horizon 2020 Framework Programme of the European Union

EXECUTIVE SUMMARY

The purpose of this deliverable is to enumerate the components and services foreseen in the project with a focus on those to be included in the first prototype of the platform. For each component, a brief description of its functionality is provided, along with parameters, input, output, and API examples (where applicable).

The primary focus of the third work package is to provide the needed infrastructure for supporting the development of the CPN platform, and to provide the set of components for supporting the implementation of the CPN pilots. The CPN platform is conceptually composed of several layers, which in turn consist of the different modules that will enable the system functionalities. The CPN platform foresees the integration of three different layers of modules:

1. The Content Layer: will focus on the extraction of relevant information from the different content sources. This layer is composed of two types of services: (1) content procurement (for structured and unstructured heterogeneous data stream gathering) and (2) knowledge extraction (for user personalisation such as relations identification and clustering).

2. The Mapping Layer: The goal of this layer is to provide services which map content onto users. The services in this layer will make content available to the users through personalisation and contextualization processes. Further, the layer also includes permission and contract aspects, which will assist with overcoming legal and ethical issues, as well as preserving privacy criteria and copyright.

3. The User Layer: this will offer specific services to deal with the user data itself. It will contain modules that create user profiles which potentially include preferences, socio-demographical information, history etc., as well as services to appropriately handle user context.

At this stage we foresee the CPN platform in need of the following 13 technology bricks, as described in this deliverable:

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Sentiment
Users Technology Bricks	User Modelling
	Reader's App - TRULY MEDIA
	Personal Data Receipts
Mapping Technology Bricks	Producer's App - CUTE4LE
	Reward Framework
	Twitter Analytics - TRUTHNEST
	Recommender

Co-funded by the Horizon 2020 Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020 Framework Programme of the European Union

Co-funded by the Horizon 2020 Framework Programme of the European Union

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

API	Application Programming Interface
AWS	Amazon Web Services
CPN	Content Personalisation Network
CRUD	Create, Read, Update, Delete
CSV	Comma-separated values
GDPR	General Data Protection Rules
JDBC	Java DataBase Connectivity
JSON	JavaScript Object Notation
NLP	Natural Language Processing
OCR	Optical Character Recognition
PDF	Portable Document Format
RDF	Resource Description Framework
REST	Representational State Transfer
SM	Social Media
XML	eXtensible Markup Language

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

This Deliverable contains the basic description of the conceptual architecture and the technological infrastructure of the CPN platform which is composed by what we call 'technology bricks'. Although some technology bricks were defined even in the CPN project proposal, it is expected that changes, adjustments and additions will be needed during the implementation phase of the project, in order to refine and finalize all the bricks that have to be offered within the CPN platform.

The CPN user requirements have now been collected through numerous workshops, interviews and user surveys and are explained in D1.1 "User Requirements Model". Following this work the user partners cooperated with the technical partners to agree on a prioritisation, classifying the requirements into one of the three expected prototypes (of the CPN platform). Of course, there are also some requirements for which it is still unclear how and if the CPN platform will manage to satisfy, as they seem to be beyond the ambition of the project, given the current (relevant) technology state of the art. This classification is included in this deliverable and accompanied by a plan that describes which technology bricks are needed to meet the requirements of each prototype, paying close attention to the first one that is easier to visualise at this point. The conceptual workflows that are necessary to satisfy the requirements of at least the first prototype are also described in this deliverable.

One of the main goals of the document is to enumerate the technology bricks that are foreseen at this point of the project necessary to satisfy the user requirements. For each technology brick, a brief description of its functionality is provided, along with parameters, inputs, output, and API examples (whether applicable).

The structure of the deliverable is organized as follows: Section 2 provides an overview of the prioritised requirements. Section 3 provides details on the conceptual architecture and workflows of the technology bricks while Section 4 describes the first versions of the CPN technology bricks infrastructure. Finally, section 5 concludes this document.

Co-funded by the Horizon 2020
Framework Programme of the European Union

2 SUMMARY OF REQUIREMENTS

As explained in the introduction following the definition of the user requirements we have gone through the exercise of prioritising the requirements, based on what is technically possible in the time plan foreseen for each one of the CPN platform prototypes.

We therefore enlist here the requirements we will try to satisfy in each one of the three prototypes, including also a list of requirements that need to be further analyzed and which we currently do not see how they will satisfied in the course of this project. At the end of each prototype category we also list the modules that we see necessary for satisfying all requirements.

PROTOTYPE 1

Requirement category	Requirement ID	Requirement description
UR-UP1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.2	The system should create/refine interests based on the user's consumption habits
	UR-UP 1.4	The system should refine the user's interests through frequent interaction with the user (talkback)
	UR-UP 1.6	The system should assign preferences (1-5) to categories based on the users behaviour
	UR-UP 1.7	The system should allow users to assign and change preferences (1-5) to categories themselves
	UR-UP 1.8	The system must allow users to completely turn off the personalisation algorithm and receive content as is and vice versa
UR-UP2: Network: Making use of connections the user already has through social media.	UR-UP 2.7	The system should allow users to share content from the CPN system to social networks
UR-UP3: Time & Length: When does the user prefer to consume content and for how long?	UR-UP 3.1	The system must allow the user to choose a preferred time frame or frames to consume content
	UR-UP 3.2	The system should create/refine time frames based on the user's consumption habits
	UR-UP 3.3	The system should refine the user's time frames through frequent interaction with the user (talkback)
	UR-UP 3.5	The system must allow the user to postpone a time frame for a chosen amount of time.
	UR-UP 3.6	The system must allow the user to ignore a time frame completely
UR-UP5: Location & Surroundings: Where is the user and what's going on around him/her?	UR-UP 5.2	The system should allow the user to set a home/main interest location
UR-UP9: User Profile Management: Giving the user transparency and control over their data	UR-UP 9.1	The system must provide transparent, simple and easy-to-understand information on what user data are collected, for what purpose and how they are stored
	UR-UP 9.2	The system should require informed and explicit consent for processing of personal user data, beyond those required for the provisioning of the agreed service

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Requirement description
UR-AF1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.5	The system should allow users to choose favourite sources
UR-AF2: Avoiding FOMO: How to ensure people think they know everything there is to know	UR-AF 2.4	The system should show users only a limited number of items at once
	UR-AF 2.5	Once all articles proposed have been consumed, the system should only offer more content upon request by the users
UR-AF3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.4	The system should be able to offer both news content and entertainment
	UR-AF 3.5	The system should be able to offer both locally and globally relevant content
	UR-AF 3.8	The system should allow users to filter content by language
UR-AF4: Sources: Where does the necessary content come from?	UR-AF 4.1	The system should be able to personalise news from/for the CPN media partners (VRT, DIAS, DW)
	UR-AF 4.2	The system should allow for additional content sources, outside the consortium
UR-AF7: User Feedback: Asking users to help improve the system	UR-AF 7.2	The system should include guided feedback for specific elements of the system, allowing users to (help) improve it

Table 1: First prototype requirements

Foreseen necessary technology bricks

The foreseen technology bricks necessary for this prototype are illustrated in the shaded cells of the table below.

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Sentiment
Users Technology Bricks	User Modelling
	Reader's App - TRULY MEDIA
	Personal Data Receipts
Mapping Technology Bricks	Producer's App - CUTE4LE
	Reward Framework
	Twitter Analytics - TRUTHNEST
	Recommender

Table 2: First prototype technology bricks

Co-funded by the Horizon 2020 Framework Programme of the European Union

PROTOTYPE 2

Requirement category	Requirement ID	Requirement description
UR-UP1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.5	The system should refine the interests based on the user's behaviour on social networks (through data upload or connection of the networks)
UR-UP2: Network: Making use of connections the user already has through social media.	UR-UP 2.1	The system should allow for social media integration to recommend content based on what connections like, read and share
	UR-UP 2.2	The system should offer a recommendation of articles based on most liked/most shared numbers from a users network and beyond that. (Nuzzle-Feature)
	UR-UP 2.3	The system should allow for social media integration to keep track of what the user has already seen elsewhere.
	UR-UP 2.4	The system should be able to analyse whom a user has been most interacting with on social media to prioritize the users for the personalisation on social media to prioritize the users for the personalisation
	UR-UP 2.5	The system should allow the user to down-/upload their network connections through user account.
	UR-UP 2.6	The system should allow users to search for other users on social media to build direct connections
UR-UP 3: Time & Length: When does the user prefer to consume content and for how long?	UR-UP 3.4	The system should use the time frames in order to decide how many items of what length and of what format it offers to the user length and of what format it offers to the user
	UR-UP 3.7	The system should learn from these user responses and adjust its offerings accordingly
UR-UP 5: Location & Surroundings: Where is the user and what's going on around him/her?	UR-UP 5.1	The system should make use of the location data of the user (permission of the user granted) to choose the right content for the user
	UR-UP 5.3	The system should make use of the location data of the user to determine the best point in time to offer content
	UR-UP 5.4	The system should try to determine the surroundings of the user based on either just location data or location data and direct interaction with the user (talkback)
	UR-UP 5.5	The system must give the user an easy option to agree to or withdraw from using location data for personalised offers
UR-UP 6: Knowledge (Management): What does the user already know?	UR-UP 6.1	The system must keep track of what content the user has already consumed on a piece and on a content basis within CPN and beyond
	UR-UP 6.2	The system must keep track of how much of each item users consume, where they stop, continue and what they skip
	UR-UP 6.3	The system should interact with the user in order to refine user interests in regards to why

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Requirement description
		something was skipped or something was consumed completely
UR-UP 8: Importance for user: What is relevant for the user, outside their given interests?	UR-UP 8.2	The system should always offer content that has a direct influence on the users (e.g. life-threatening), overruling other interest settings
UR-UP 9: User Profile Management: Giving the user transparency and control over their data	UR-UP 9.3	The system must give the user a full overview of his/her data and allow them full control, including update and removal of data
	UR-UP 9.4	The user must be able to change and overwrite settings in their profile
	UR-UP 9.5	The user must be able to download their profile data in CPN in a machine readable format and a user friendly format
UR-AF 1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.6	The system should offer the user a random news selection upon request based on certain data and preferences of the users profile, which the user can choose
UR-AF2: Avoiding FOMO: How to ensure people think they know everything there is to know	UR-AF 2.1	The system should show users who else from their network has consumed the same content item.
	UR-AF 2.2	The system should show users what else their network has shown, if there are differences
	UR-AF 2.3	The system should be able to show users the content item from another user (anonymously)
UR-AF 3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.7	The system should be able to give the user a timeline overview of events regarding a specific topic
UR-AF 5: Transparency: Giving the user control & understanding over the content he sees.	UR-AF 5.1	The system must offer the user an easy to access and easy to understand overview of their profile
	UR-AF 5.2	The system must offer users easy access to their profile in order to change settings and data
UR-AF 6: Archive: Making content available beyond the moment	UR-AF 6.1	The system must allow users to access content again that they have already opened before
	UR-AF 6.2	The system should allow users to consume content beyond their predefined timeframe after an interaction with the user (talkback)
	UR-AF 6.3	The system should allow users to actively save articles for later consumption
UR-AF 8: Temporary Categories: Users can temporarily change the personalisation algorithm	UR-AF 8.1	The system should allow users to search for specific topics they are temporarily interested in
	UR-AF 8.2	The system should allow users to add this search as a temporary personalisation category
	UR-AF 8.3	The system should allow users to define a specific time frame for this temporary change
UR-AF 9: Mute topics: Exclude topics from the personalisation for a certain time	UR-AF 9.1	The system should allow users to define keywords and logical combinations of them to exclude content from their personalisation
	UR-AF 9.2	The system should allow users to define a time frame per keyword/logical combination

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Requirement description
	UR-AF 9.3	The system should be able to overwrite this exclusion for important breaking rules
UR-PS 1: Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience	UR-PS 1.1	The system should show the access to items through users by numbers (who, when, how long)
	UR-PS 1.3	The system should show which topics were most interesting to users

Table 3: Second prototype requirements

Foreseen necessary technology bricks

The foreseen technology bricks necessary for this prototype are illustrated in the shaded cells of the table below.

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Sentiment
Users Technology Bricks	User Modelling
	Reader's App - TRULY MEDIA
	Personal Data Receipts
Mapping Technology Bricks	Producer's App - CUTE4LE
	Reward Framework
	Twitter Analytics - TRUTHNEST
	Recommender

Table 4: Second prototype technology bricks

PROTOTYPE 3

Requirement category	Requirement ID	Requirement description
UR-UP 1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.3	The system should be able to offer personalised content on the basis of the users mood or values
UR-UP 4: Preferred Media: Which type of content does the user prefer?	UR-UP 4.1	The system must allow the user to choose preferred types of content
	UR-UP 4.2	The system should set/refine preferred types of content based on the user's consumption habits and the timing
	UR-UP 4.3	The system should refine the user's preferred types of content through frequent interaction with the user (talkback)
UR-UP 6: Knowledge (Management): What does the user already know?	UR-UP 6.4	The system should be able to offer insights and advice based on what it learn about what a user consumed in relation to a certain entity (e.g. a place)
	UR-UP 6.5	The system should allow the user to delete part of the systems knowledge for specific

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Requirement description
		time frames back in time from the moment of viewing
UR-UP 7: Devices: On what device is the user consuming content?	UR-UP 7.1	The system should check on what device the user is consuming the content
	UR-UP 7.2	The system should adjust its content offering based on the type of device the user is using
	UR-UP 7.3	The system should try to make smart use of device data to determine the surroundings of the user and adjust the content strategy accordingly
UR-UP 8: Importance for user: What is relevant for the user, outside their given interests?	UR-UP 8.1	The system should combine reading habits and knowledge about the user to provide smart updates on things the user could be interested in, even if this doesn't fit his/her set interests
	UR-UP 8.3	The system should be able to surprise the user with content, he/she would not have chosen themselves
UR-UP 9: User Profile Management: Giving the user transparency and control over their data	UR-UP 9.6	The system should allow the user to add external data to update their profile
UR-AF 1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.1	The system should offer users an overview of other sources, covering the same topic
	UR-AF 1.3	The system should offer the user an easy overview of what content from which sources he has consumed over a certain period of time
UR-AF 3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.2	The system should offer the user a short overview of all important headlines at a specific point in time with access to more details upon request
UR-AF 5: Transparency: Giving the user control & understanding over the content he sees	UR-AF 5.3	The system must make it transparent to the users why they are shown certain content, based on an item level
UR-AF 6:Archive: Making content available beyond the moment	UR-AF 6.4	The system should be able to memorize where a user left off and restart at the same point
UR-AF 7: User Feedback: Asking users to help improve the system	UR-AF 7.1	The system should offer user feedback requests in a playful/entertaining way
	UR-AF 7.3	The system should allow users to assign both existing or new attributes (categories, moods etc.) to a content item
	UR-AF 7.4	The system should be able to offer a feedback interaction to determine the

Co-funded by the Horizon 2020 Framework Programme of the European Union

Requirement category	Requirement ID	Requirement description
		ground level of personalisation based on mood, time and interest
UR-PS 1: Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience	UR-PS 1.4	The system should be able to show these numbers during the creation process of the content
UR-PS 2: Integration: How should CPN be connected to the production side?	UR-PS 2.1	The system should allow for an easy integration into the producers workflow
	UR-PS 2.2	The system should provide contract templates to allow freelancers to easily work together and with editors, to define and track the scope of individual contributions and expected revenues
	UR-PS 2.3	The system should allow producers to transparently see how often their contributions are used and distributed to readers
	UR-PS 2.4	The system should allow producers to export the record of their publications through standardized and interoperable formats
	UR-PS 2.5	The system should allow for an easy contribution of content from different publishers through standardised interfaces
	UR-PS 2.7	The system should allow editors to easily add missing attributes to articles manually

Table 5: Third prototype requirements

Foreseen necessary technology bricks

The foreseen technology bricks necessary for this prototype are illustrated in the shaded cells of the table below.

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Sentiment
Users Technology Bricks	User Modelling
	Reader's App - TRULY MEDIA
	Personal Data Receipts
Mapping Technology Bricks	Producer's App - CUTE4LE
	Reward Framework
	Twitter Analytics - TRUTHNEST
	Recommender

Table 6: Third prototype technology bricks

Co-funded by the Horizon 2020 Framework Programme of the European Union

REQUIREMENTS UNDER QUESTION

At this point of the project it is not very clear how we will be able to satisfy these requirements. For this reason, we have classified them in this 'pending' category until we have a more mature view of the CPN platform and manage to discover ways in which these requirements can be satisfied.

Requirement category	Requirement ID	Requirement description
UR-UP 1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR-UP 1.1	The system must allow the user to manually choose their interests that later define the personalisation
UR-AF 1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers?	UR-AF 1.2	The system should highlight differences between the perspectives of different sources on a similar topic
	UR-AF 1.4	The system should make it easy for the user to see a bias of a content item or source
UR-AF 3: Content/Format: In which way do we have to prepare content for the user?	UR-AF 3.1	The system should offer content items in small, easy to consume and logical packages, allowing the user to consume them bit by bit
	UR-AF 3.3	The system should allow users to choose whether they prefer an overview or all content at once
	UR-AF 3.6	The system should be able to put global news in a local relevance context for users
	UR-AF 3.9	The system should allow users to filter content by complexity within a language
UR-PS 1: Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience	UR-PS 1.2	The system should show which parts (paragraphs, entities) of an item were most interesting to users
UR-PS 2: Integration: How should CPN be connected to the production side?	UR-PS 2.6	The system should give feedback on what attributes are best used on content to improve the personalisation performance

Table 7: List of requirements under question

Co-funded by the Horizon 2020 Framework Programme of the European Union

3 CONCEPTUAL ARCHITECTURE & WORKFLOWS

Considering the expected CPN platform at a conceptual level, we can separate three basic parts in its functionality: the two front-end applications (for content producers and consumers) and the user modelling process. Based on these we can define 3 respective important workflows: the content creation workflow, the client apps (mobile, web) serving content and the user profile workflow.

The **content creation workflow** is mainly served by the web application for producers (DW, VRT and Dias). This will be based on Cute4LE which will be suitably modified to serve the purposes of pushing content into the platform. The content will be published to the message queue. The web application will also interact with the reward framework in order to allow producers to form collaborative teams for enriching or creating new content.

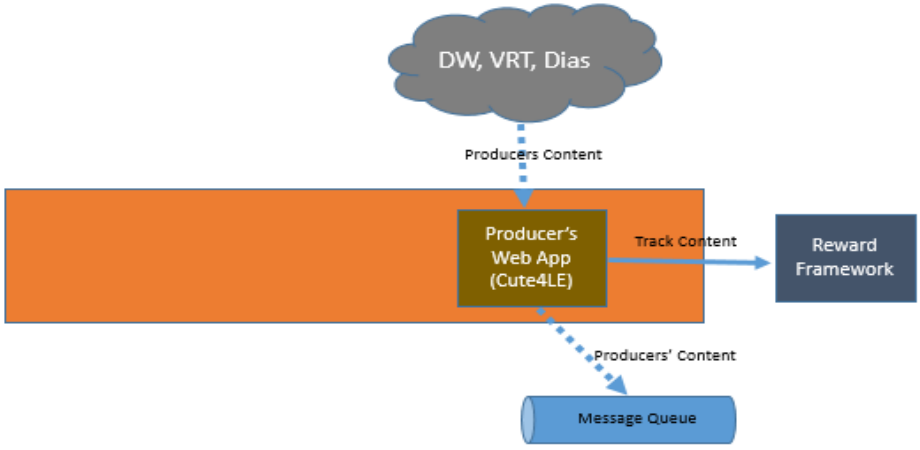

Figure 1: Content creation workflow

The front end application for content consumers (**readers workflow)** will be based on a customized version of TrulyMedia which already provides many useful functionalities for browsing through contents and organising it in thematic collections.

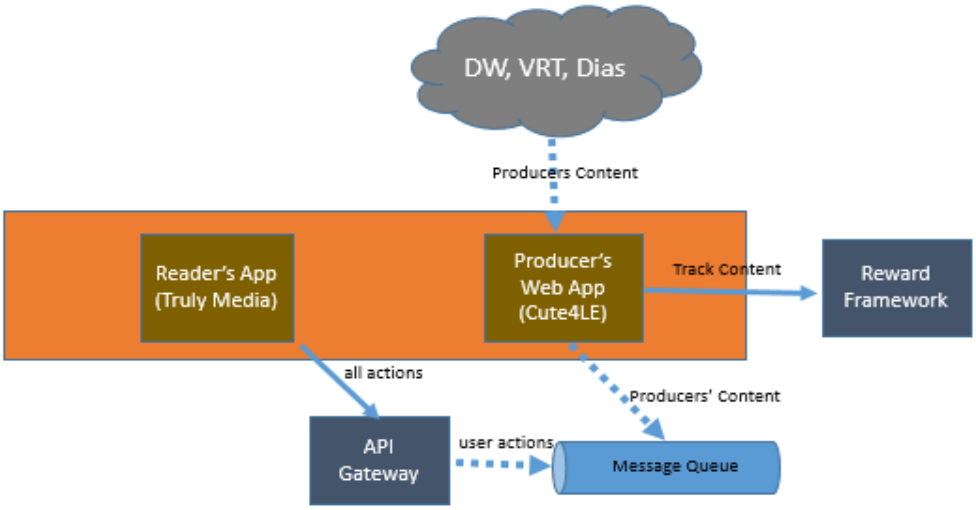

Figure 2: Reader's workflow

Co-funded by the Horizon 2020 Framework Programme of the European Union

This 'reader's app' will communicate through the API gateway with the CPN platform. All calls to the CPN API will be directed to the API gateway. The API gateway will be responsible for the distribution of all calls to the related modules, based on the workflow. A message queue will be used to implement a publish/subscribe protocol in order to enable modules fetch and analyse content. The reader's app will be both pulling and pushing content to the platform, as it will be the front end for the users but will also be bringing content from Social Media sources, following user queries as well as queries automatically generated by the recommender module in search of SM content that matches the interests of the user. The plan is to avoid centralized databases. Therefore, all modules are going to follow a Microservices approach and store data internally on their own repository. The "publish – subscribe" pattern covers the need for asynchronous communication and provides greater network scalability and a dynamic network topology.

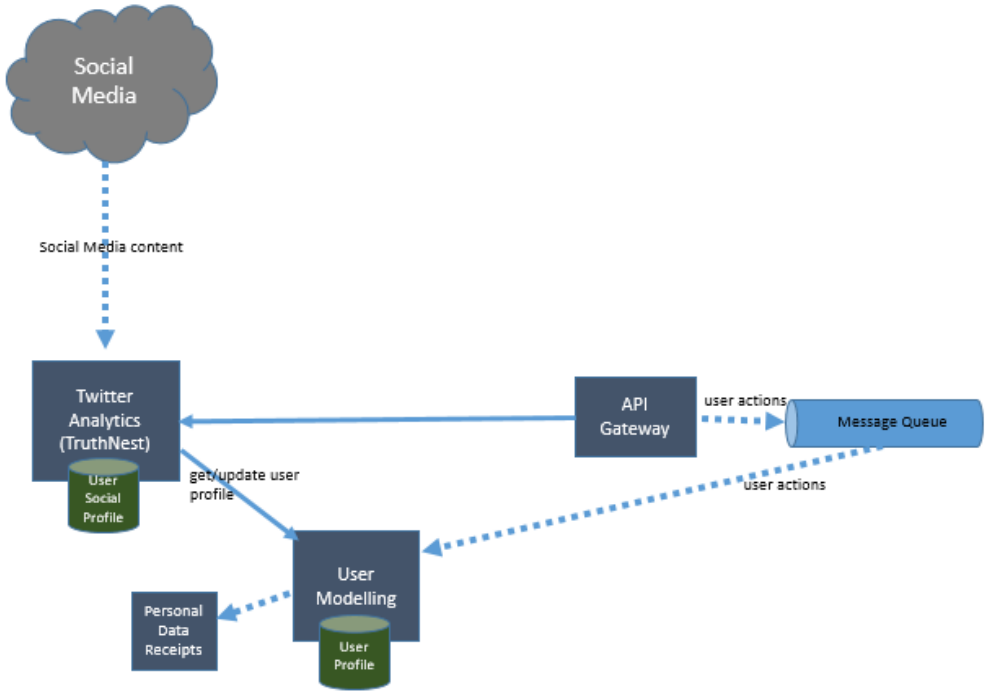

Figure 3: User profile workflow

The **user profile workflow** refers to the user model creation and maintenance. User Modelling will listen (through the message queue) to all user actions in order to modify the user profile accordingly. It will also be retrieving input from the Twitter Analytics module (to be based on Truthnest) providing useful information regarding the user activity, network and preferences in Twitter. Twitter Analytics will also be responsible for providing trending news on Twitter. Finally, User Modelling will also collaborate with Personal Data Receipts to allow the user to have a transparent view to all information we keep and utilise around his/her personal profile.

All analysis modules (abstracted here under Semantic Indexing) will be retrieving incoming content to process and index it in a Triple Store so that it can be retrieved and pushed to the user following commands of the Recommender, which has direct access to all user models.

The overall depiction of the workflow and the interactions of the components is shown in the following diagram

Co-funded by the Horizon 2020 Framework Programme of the European Union

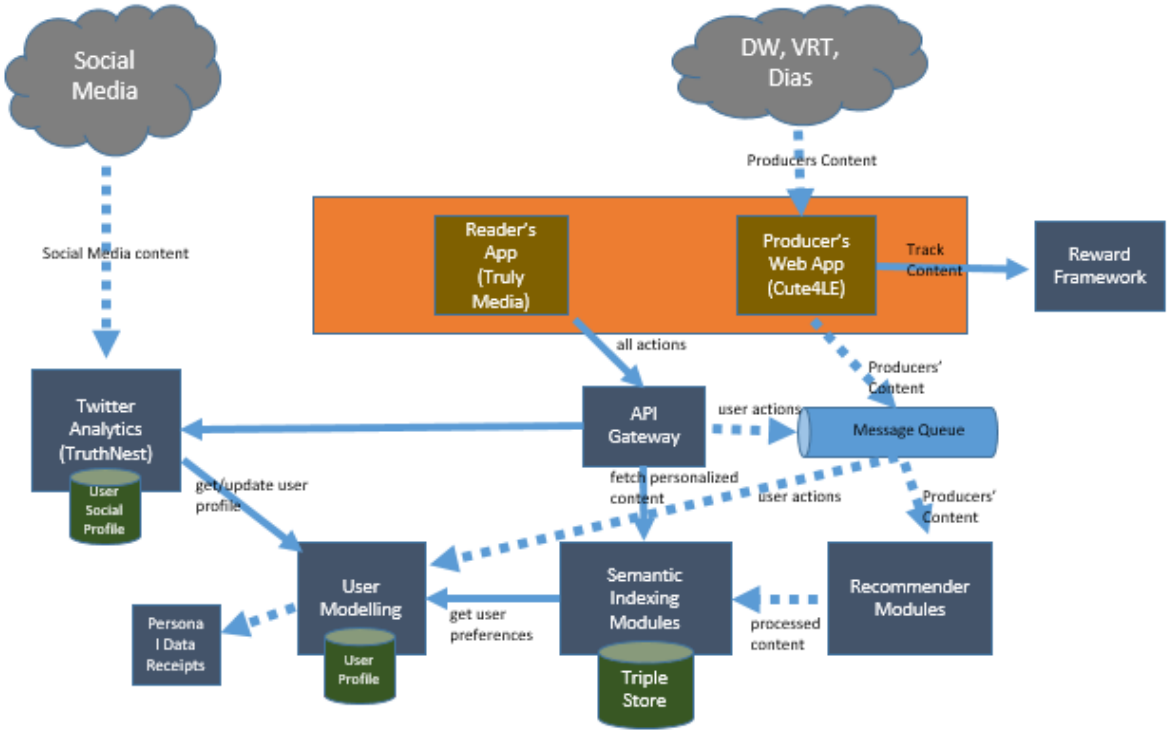

Figure 4: Overall conceptual architecture

Co-funded by the Horizon 2020 Framework Programme of the European Union

4 MODULES DESCRIPTION

This section describes the modules needed to implement within the CPN platform all the features that will meet the user requirements.

4.1 PRODUCER'S APP – CUTE4LE

4.1.1 Tool Overview

Cute is a content curation system that integrates search, storytelling, analysis and monitoring features to support content marketing strategies. The curator can create stories aggregating and integrating content from all major web sources including social networks.

Cute allows you to define content marketing strategies through research planning, integration with analytics systems and social content analysis, enabling you to discover new trends, monitor social reputation and the effectiveness of promotional campaigns.

Cute for Large Events (Cute4LE) is the evolution of original Cute tool with the specific aim to tailor contents curation to address large events specific needs (pre-event, event, post-event timeframes).

4.1.2 Tool application Programming Interfaces

Cute4LE offers a set of front-end features, through web interface and some back-end services as REST APIs.

Front-End Features

The two main macro-features of Cute4LE are storytelling and monitoring.

Storytelling: is the core feature and it is composed of several steps: search, aggregate, editing and publish

Search: It integrates all the major social networks APIs and different search methods (keywords, users, locations). It allows to find user generated contents in different sources at the same time. The system allows also to schedule periodic researches and include new content providers.

Aggregate: The user can collect different contents and store them in a personal archive. The archive can be temporary, waiting to create a story (as a shop cart), or permanent.

Edit: A collaborative editing system that allows to create a story with user generated contents and/or personal contents

Publish: The user can publish or share a story on websites and social networks.

Monitoring: The are two principal monitoring mechanisms: livestream and territory monitoring

Livestream: A real-time monitoring mechanism developed to support events stories creation. It allows to expose real-time monitoring of hashtags from major Social Networks. In particular it includes: continuous monitoring based on hashtags (Twitter and Instagram), search on Facebook pages and search public contents on all other social networks.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Territory monitoring: A monitoring mechanism to cluster contents by a geo-referred position. It supports events stories creation focussing the search on a relation of geographic position of the contents produced by users. This mechanism allows to set a geo-referred position and monitor events arranged and planned in the specific territory. The analysis of stream of contents injected results in the identification of three main analytics categories: influencers, trending topics, most popular contents

REST APIs

All the APIs must be authenticated and the output format is JSON.

Search: Search contents on an external content provider or a social network.

Needs -> External content providers

Input -> Search parameters (keywords, location …)

Output -> Filtered contents based on search parameters

Method	HTTP Request	Description
GET	/api/search/<source_name>	Generic search for all content providers

Stories: Stories are created through web interface

Needs -> User generated contents and/or personal contents uploaded on platform (in Storybrick format)

Input -> No

Output -> Stories

Method	HTTP Request	Description
GET	/api/story/find	Return all stories
GET	/api/story/get	Get story by id

Storybricks: All the contents retrieved from external sources or uploaded by users, are formatted uniformly into a Storybrick. A storybrick is the smallest unit that can be included into a story.

Needs -> No

Input -> No

Output -> Contents formatted uniformly and saved on platform

Method	HTTP Request	Description
GET	/api/storyBrick/get	Get content by id
GET	/api/storyBrick/getByOriginId	Get content by original source id

Livestream: Livestream is a frequently automatic search on different sources

Co-funded by the Horizon 2020 Framework Programme of the European Union

Needs -> External content provider

Input -> Frequency, search parameters (source, keyword)

Output -> Auto-updated results (livestream)

Method	HTTP Request	Description
POST	/api/liveStream/save	Create a new livestream
GET	/api/liveStream/get	Get livestream by id and its contents

Analytics: Analytics data related to the different objects of Cute4LE

Needs -> Objects to analyze (livestream, live-search, automatic-search)

Input -> the object to analyze

Output -> Aggregated analytics data

Method	HTTP Request	Description
GET	/api/analytics/analyzeLiveStream	Analyze a livestream
GET	/api/analytics/analyzeLiveContents	Analyze a search
GET	/api/analytics/analyzeScheduledContents	Analyze a scheduled or a geo-referred search

4.1.3 Technologies used

Cute4LE is realized exploiting an Open Source stack optimized for scalability and Data Input /Output effectiveness (MEAN stack).

MEAN is a full javascript stack composed of:

- MongoDB, NoSQL and Document Model Database
- ExpressJS, a framework to create web applications
- AngularJS, frontend web framework to create single page applications
- Node.js, a javascript Event Driven framework

4.1.4 3rd party dependencies & hosting environment

- Node.js > v6.x
- MongoDB > v3.2

4.1.5 Hardware specifications

Not available

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.1.6 Packaging

Not available

4.2 REWARD FRAMEWORK

4.2.1 Tool Overview

The Reward Framework provides a platform for content producers to form ad hoc teams needed for the creation of a given item of media content, e.g., requiring multidisciplinary expertise, including video production and editing capabilities, text writing skills, photography experience, etc. A number of pre-defined contracts are available for selection and agreement among team members and content distributors. Additional tools allow digitisation of different team members' contributions, to register them as digital assets and share rewards following their use by the content distributors. Smart contract and distributed ledger technologies are used to independently enforce such contracts and transparently and automatically account for generated revenues.

4.2.2 Tool application Programming Interfaces

APIs for:

- accessing information on distributed content from Recommendation module and store on blockchain
- selecting contracts for use of digital assets in the Producer's Web App

4.2.3 Technologies used

Several Docker containers which include services implemented using: Javascript, node.js, go Blockchain platform

4.2.4 3rd party dependencies & hosting environment

AWS

4.2.5 Hardware specifications

Not available

4.2.6 Packaging

Docker image

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.3 TRULY MEDIA

4.3.1 Tool Overview

Truly Media is a collaborative platform that helps users' aggregate, curate and verify news content. As Truly Media provides useful functionalities for organising and browsing through news related content, we expect to use it as a basis for the front end application CPN users will have for consuming the content deemed (by the personalisation services) suitable for them.

4.3.2 Tool application Programming Interfaces

Truly Media operates as a front end, allowing users to organise and browse through content and is connected with APIs of Social Media as Twitter, Facebook and Youtube to receive content in real time.

4.3.3 Technologies used

The front end is based on the Angular Fuse template. Java 8(Sping boot) and Node.js are used for the backend. The front end is based on AngularJS 1.5.x and in particular on the Angular Fuse template.

4.3.4 3rd party dependencies & hosting environment

The project is hosted on a cloud platform. We use MongoDB database hosted on mLab, the frontend is hosted on Amazon CloudFront, the backend on Heroku and we store the data on Amazon S3.

4.3.5 Hardware specifications

Truly.Media requires 2 GB of RAM and a dual core CPU environment. It is fully cloud hosted (application and data).

4.3.6 Packaging

We use maven for packaging. It is configured with Jenkins for deployment management

4.4 USERMODELLING

4.4.1 Tool Overview

The user modelling is used to create and update user profiles for CPN users. Profiles contain an up-to-date history of the user news' consumption by collecting click-streams, topics of interests (automatically extracted from news articles), demographics information and location data (when available). User profiles play an important role in the success of the recommendation process since the profiles represent the users' information needs. The accuracy of each user profile affects the performance of the entire recommender system.

4.4.2 Tool application Programming Interfaces

The module offers the following APIs.

Co-funded by the Horizon 2020
Framework Programme of the European Union

User-profile API: it is used to manage profiles (CRUD operation) for existing users.

4.4.3 Technologies used

Python, MongoDB (for storing user profiles)

4.4.4 3rd party dependencies & hosting environment

MongoDB

4.4.5 Hardware specifications

The amount of RAM needed is typically around 4-8gb per node.

4.4.6 Packaging

Docker image

4.5 TWITTER ANALYTICS - TRUTHNEST

4.5.1 Tool Overview

TruthNest is a platform that provides in real time very detailed analytics around any information relating to Twitter. TruthNest returns detailed information around the activity, network and influence of any user. It can also provide analytics on the engagement and virality of specific posts, as well as timelines illustrating the unfolding of specific events as reported by posts with the greater engagement.

4.5.2 Tool application Programming Interfaces

TruthNest may receive as input either the url of a Twitter post, the identifier of a specific Twitter user, or a free text query representing a topic under investigation. Twitter can provide responses in json format encapsulating all types of supported analytics.

4.5.3 Technologies used

The backend in written in Java 8. The frontend uses PHP and Javascript libraries (jQuery, D3 etc).

4.5.4 3rd party dependencies & hosting environment

We use MongoDB and MySQL databases. The project is hosted on ATC premises on an Ubuntu 14.04.5 LTS server.

4.5.5 Hardware specifications

The platform requires at least 4 GB of RAM and a dual core processing environment to run smoothly.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.5.6 Packaging

We use maven for packaging. It is configured with Jenkins for deployment management.

4.6 PERSONAL DATA RECEIPTS

4.6.1 Tool Overview

Personal Data Receipts are a tool compliant with GDPR Articles on Information Notice and aiming to simplify users' understanding of privacy policies, while providing them with a human-readable record on what personal data are collected, the purpose of use they have consented to, and for how long given data will be stored. PDRs are an instrument to allow users to ask for data removal or for executing other digital rights. Integration with blockchain is leveraged to provide a non-repudiable receipt record, useful for future verification that personal data are used according to the user's wishes. This tool will be tested during the project to gather user feedback for further refinement and with adopters (e.g., content distributors) in order to derive recommendation for standardized PDRs.

4.6.2 Tool application Programming Interfaces

APIs for:

- create PDRs, interact with User Modelling module to fetch user profile
- store PDR hash on a blockchain
- search given PDR on a blockchain
- revoke PDRs

4.6.3 Technologies used

Several Docker containers which include services implemented using: Javascript, node.js, go Blockchain platform

4.6.4 3rd party dependencies & hosting environment

AWS

4.6.5 Hardware specifications

Not available

4.6.6 Packaging

Docker image

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.7 RECOMMENDER

4.7.1 Tool Overview

The current state of the art techniques for recommending items are based on two main areas: content based (that relies on good semantic modelling/feature extraction and selection on the items to be recommended) and collaborative filtering techniques (that are essentially domain-independent and take into account network metrics based on emerging similarity graphs of users and items). Our system uses a hybrid approach that uses variable proportions of the mentioned techniques for each user learning (using Machine Learning techniques) from explicit and implicit feedback given by the users themselves: clicks, ratings, sharings, etc. The system is customizable for including content-delivery strategies' optimization: multichannel and date/time optimization (predicting the probability of interests at a given time on a given channel) and includes mechanisms for fostering "serendipitous" discoveries.

The recommender exploits the features extracted from the document enriching modules: sentiment extraction, user modelling, user feedbacks, and topic annotation.

4.7.2 Tool application Programming Interfaces

The module offers the following APIs.

Recommendation API: it is used to retrieve recommendation for a given user drawn from a catalog of news items.

4.7.3 Technologies used

Python, MongoDB (for storing recommendation)

4.7.4 3rd party dependencies & hosting environment

MongoDB

4.7.5 Hardware specifications

The amount of RAM needed is typically around 16-32gb per node.

4.7.6 Packaging

Docker image

4.8 SEMANTIC LIFTING

4.8.1 Tool Overview

This module semantically annotates (semi-) structured data (namely data in tabular, hierarchical, or attribute-value pair structure) and generates the corresponding Linked Data (in RDF format).

Co-funded by the Horizon 2020 Framework Programme of the European Union

The module consists of an editor to define the rules that specify how Linked Data is generated, a processor which actually generates the Linked Data and a validator that validates both the mapping rules and the implementation and makes sure that high quality Linked Data is generated.

4.8.2 Tool application Programming Interfaces

The semantic lifting module may take as input a (relational) database (O/JDBC), a Web API or a file which contain data in a table, CSV, XML, JSON or wikitext format. Other formats may be supported but it has to be known so the implementation can be extended.

One can find examples of input data on :

- the website, http://rml.io/RML_examples.html
- the specification, http://rml.io/spec.html , or
- the unit tests of the implementation, https://github.com/RMLio/RML-Processor/tree/ab26dac414692b3235164b271b376304869225ca/src/test/resources

One can run the module by providing 1. the data sources and 2. a mapping file (with the semantic annotation rules described in RML). One can use the command-line-based version or the Web API. In the latter case, the data and the mapping file are provided in a POST request. Afterwards, via a separate request, one can get the results.

The mapping file may be edited manually or using the RMLEditor. The RMLEditor is a Web service which one may use after requesting access to it. The consistency of a mapping file may be checked using the RMLValidator. The RMLValidator is available as a command-line tool and takes a mapping file as input and has a list of violations (if any) as output.

4.8.3 Technologies used

The implementation is in JAVA and it is available via the command line. There is also a wrap-up of the JAVA implementation in Node.js which provides a Web API around the implementation. Both of them are available with their source code, releases and Docker images.

4.8.4 3rd party dependencies & hosting environment

One should have Java (version 7 or 8) and maven (version 3) installed.

4.8.5 Hardware specifications

Not available

4.8.6 Packaging

There are docker images and the implementation can be used as a Java library.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.9 GENERIC TRAINING MODEL FOR RELATION EXTRACTION

4.9.1 Tool Overview

This module allows predicting slots to complete tuples of the form (subject, relation, and object), with subject or object missing, and for a fixed set of relations (our current system is designed for those relations defined in http://surdeanu.info/kbp2014/TAC_KBP_2014_Slot_Descriptions.pdf , but we intend to extend that set, depending on the need of the content providers).

The module will use English data from Deutsche Welle as basis for the extractions.

Given that named entities are essential for the slot filling component, an additional component for automated named entity extraction (based on publicly available training data) will be provided.

4.9.2 Tool application Programming Interfaces

API access to

- Enrich an input article with various standard NLP elements (named entities, POS tags, etc.), by means of a Stanford CoreNLP server.
- Provide (subject, relation) as input, returning the matching object with confidence score, as well as the passage in the article archive, based on which the prediction was made. Similarly for input (relation, object), with subject as output.

4.9.3 Technologies used

Python, (sklearn for the prediction component, with a flask API)

Stanford CoreNLP

4.9.4 3rd party dependencies & hosting environment

Standard

4.9.5 Hardware specifications

Standard

4.9.6 Packaging

The code will consist of python packages; trained models will be available as binary files, and enriched dataset will consist of json files.

4.10 TOPIC EXTRACTOR

4.10.1 Tool Overview

Domain independent terminological, taxonomical and ontological extraction from unstructured sources using metrics and strategies based on statistical, linguistics and extra-linguistic features (e.g. text

Co-funded by the Horizon 2020 Framework Programme of the European Union

tagging, position, etc.). The text-extraction sub-module is able to extract text from heterogenous documents including PDFs, web pages, ms-office files, xml, etc. It also provides the possibility of extracting text from scanned PDF documents via state-of-the-art OCR technologies (tesseract v4.x). The tool can be used to automatically build a domain terminology/ontology from unstructured sources that can be later used for document annotation and retrieval; the terminological candidates are multiword expressions that are filtered through different stages of scoring.

In the context of CPN it will be used to constantly enrich a list of emerging topics from news corpora and annotating incoming news according to this topic list

4.10.2 Tool application Programming Interfaces

The module offers the following APIs.

Document API: it used to create document corpora by submitting text documents (docs, pdfs, etc.) or a link to an online document (URL). The corpus will be then processed to extract the text from each documents and store it for later elaborations.

Topic Extraction API: it is used to launch a topic extraction task on a previously created corpus. The API will return an "OK" if the task is correctly submitted. The list of ongoing extraction tasks and their completion status can be subsequently retrieved.

Topic API: it is used to manage (CRUD operations) the results of topic extraction tasks.

4.10.3 Technologies used

Python, MongoDB (for storing the text extracted from documents, list of topics and documents to topic associations)

4.10.4 3rd party dependencies & hosting environment

MongoDB, Tesseract (if OCR is needed)

4.10.5 Hardware specifications

The module is thought for processing large quantities of data in an efficient way. However for very large corpora it is recommended to run different containers with at least 8-16gb RAM.

4.10.6 Packaging

docker image

4.11 UPLIFTING/DEPRESSING ARTICLE CLASSIFIER

4.11.1 Tool Overview

Original definition: Classifying the stance of the view expressed in an article on a certain topic (e.g., political view, left- vs right-wing; or positive/neutral/negative in a more basic setting). This can be used for the unsupervised perspective extraction.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Given that no (or very little) opinionated text is available (articles themselves are fairly neutral; user comment feature is minimally used), it was decided to abandon this module: without data, it would be nearly impossible to train useful models, and if data is not generated in practice, applicability of such models would be limited anyway.

The eventual application of this module is in balancing lists of suggested articles: a substantial fraction of news items are typically viewed negatively (e.g., conflicts, disasters, accidents) and thus "depressing". To balance this, we need positive, or "uplifting" articles to be interspersed in a suggested reading list.

The idea of this component is to build a text classifier, which takes as input an article, and classifies (or potentially scores) the article as "depressing" vs "uplifting". The definition of what exactly is "uplifting" and "depressing" is part of the study that will be performed by developing this component. Given that without a decent analysis of a first dataset, these definitions are hard to specify upfront, we will continue to refine/define the exact component to be delivered as we go, to maximize both (a) feasibility as well as (b) relevance for the content providers. The latter relevance was validated during the Brussels meeting in March and follow-up calls. The feasibility largely depends on data availability, and particularly labeled data. Since the latter is not yet available, we foresee a phased approach including collection of annotation. Both task and dataset definition are currently under further investigation with Deutsche Welle.

First step will be defining a development data set, to gather first uplifting/depressing annotations, thus assessing the feasibility of the task in terms of data. In parallel, baseline text classifiers will be developed.

4.11.2 Tool application Programming Interfaces

The prediction module will be accessible through a REST API, where a sentence will be provided as input, and JSON output with the predictions returned (with a confidence score for each label, depending on the final definition of the task).

4.11.3 Technologies used

Python (pytorch or tensorflow for the prediction component, with a flask API)

4.11.4 3rd party dependencies & hosting environment

Standard

4.11.5 Hardware specifications

Depending on the size of the dataset, training and prediction may require GPUs.

4.11.6 Packaging

The code will consist of python packages; trained models will be available as binary files.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.12 FRAME BASED SLOT-FILLING SYSTEM

4.12.1 Tool Overview

The eventual application of this module is to extract key features from articles in terms of the event(s) they cover. For example, the system would identify whether an article talks about a "natural disaster", or a "political event", or a "sports event", etc. In addition, the important entities/organizations/locations involved and what role they play would be extracted. This extraction will essentially fill the slots answering 'who?', 'what?', 'where?', 'when?' questions. Having this information extracted for a series of articles (potentially from different publishers) will help to identify whether articles talk about the same or different events, and potentially allow to link articles together. This will prove useful in aggregating news article into a feed of articles to read without significant overlap (which is especially a risk if articles are coming from multiple publishers). It thus serves a building block for news diversification and/or clustering into news stories.

The idea of this component is to build a so-called frame based system, which takes as input an article, essentially takes a down-to-earth approach to understand what the article is about and capture it in a 'frame', which has a well-defined type/class and a set of slots that have associated a set of slot values. Example of frame classes: events (deaths, awards and prizes, strikes, accidents, extreme weather), recurrent_event (solar eclipse, charity campaigns, summer/winter time transition, expos, tournament), with the following sample slots:

- event_death: slot_who, slot_how, slot_when, slot_where
- event_sportmatch: slot_team, slot_person, slot_sport, …
- event_charity: slot_name (e.g., "De Warmste Week"), slot_organizer, …
- event_strike: slot_company

The first step will be defining a development data set, to gather first annotations (esp. focusing on the event types and associated slots), thus assessing the feasibility of the task in terms of data. In parallel, baseline text classifiers will be developed.

4.12.2 Tool application Programming Interfaces

The prediction module will be accessible through a REST API, where a text document (an article) will be provided as input, and JSON output with the extracted frame info (type of 'event' and the relevant slots).

4.12.3 Technologies used

Python (Pytorch or Tensorflow for the extraction component, with a flask API)

4.12.4 3rd party dependencies & hosting environment

Standard

4.12.5 Hardware specifications

Depending on the size of the dataset, training and prediction may require GPUs.

Co-funded by the Horizon 2020 Framework Programme of the European Union

4.12.6 Packaging

The code will consist of python packages; trained models will be available as binary files.

4.13 SENTIMENT

4.13.1 Tool Overview

DS4Biz-Sentiment is a multiplatform, microservice based, trainable classification system aimed at:

- Polarity detection
- Complex opinion and sentiment extraction (via unsupervised techniques)
- Spam, hate-speech and flames detection

It automatically select the best performing machine-learning module from a (configurable via API) family of models. It also offers an API to update the models incrementally by submitting new polarity examples.

In the context of CPN it will be used to add meta-level information about the news articles to be later exploited in the context of recommendation (e.g. recommend the right balance of negative/positive articles on a given topic)

4.13.2 Tool application Programming Interfaces

The module offers the following APIs.

Training API: it is used to create and update sentiment extraction models. Models can be created from scratch by submitting whole training sets or can be updated incrementally

Classification API: classifcation of documents by using one of the models created with the Training API.

SentimentExtraction API: it is used to extract sentiment expression, in an unsupervised way, from given documents.

4.13.3 Technologies used

Python, MongoDB (for storing the sentiment metadata associated to documents)

4.13.4 3rd party dependencies & hosting environment

MongoDB, Tesseract (if OCR is needed)

4.13.5 Hardware specifications

The module is thought for processing large quantities of data in an efficient way. However for very large corpora it is recommended to run different containers with at least 8-16gb RAM.

4.13.6 Packaging

docker image

Co-funded by the Horizon 2020 Framework Programme of the European Union

5 CONCLUSIONS

This deliverable has presented the work that has been performed in the context of tasks T3.1-3.3, of work package WP3. Having as a starting point the user requirements, as depicted in deliverable D1.1, “User Requirements Model", a set of components, APIs, and services, have been planned to be delivered by the respective partners. These components, APIs, and services constitute the first version of the platform infrastructure.

Each component has been individually described through a brief presentation of the provided functionality. The description is accompanied by the API/service through which the component is accessible/embeddable. In such a case, the API/service parameters, inputs, output, and API examples are provided.

This deliverable reports the first version of the platform components, which will be integrated, by the end of M10 (June 18). Subsequent versions of the platform components are expected to provide updated versions of the currently available components, APIs, and services, along with possible new components, APIs, and services, in order to adapt to possible new requirements and functionality needed by the constantly evolving CPN platform.

Co-funded by the Horizon 2020
Framework Programme of the European Union

6 REFERENCES

[1] CPN: D1.1 User Requirements Model

[2] CPN: D2.1 CPN Reference Architecture

Co-funded by the Horizon 2020
Framework Programme of the European Union