Grant Agreement No.: 761488

D3.3: Technology Bricks V2

This deliverable provides the technology bricks which have been planned for the second prototype of the CPN platform.

Work package	WP 3
Task	T3.1-T3.3
Due date	30/4/2019
Submission date	6/5/2019
Deliverable lead	ATC
Version	Final
Authors	Nikos Sarris, Marina Klitsi, Stamatis Rapanakis (ATC)
Reviewers	Ferdinando Bosco (ENG)
Keywords	Technology Bricks, APIs, prototype 2

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.1	4/3/2019	Table of Contents	Nikos Sarris, Marina Klitsi (ATC)
V0.2	21/3/2019	Initial version	Nikos Sarris, Marina Klitsi, Stamatis Rapanakis (ATC), Ferdinando Bosco (ENG), Matthias Strobbe (IMEC)
V0.3	19/4/2019	First completed version	Nikos Sarris, Marina Klitsi, Stamatis Rapanakis (ATC), Ferdinando Bosco (ENG), Matthias Strobbe (IMEC), Fulvio D'Antonio (LIVETECH)
V0.4	22/4/2019	Final edits	Nikos Sarris, Marina Klitsi, Stamatis Rapanakis (ATC), Ferdinando Bosco (ENG), Matthias Strobbe (IMEC), Fulvio D'Antonio (LIVETECH)
V0.5	24/4/2019	Reviewed version	Ferdinando Bosco (ENG)
FINAL	6/5/2019	Completed version incorporating all comments received	Nikos Sarris, Marina Klitsi, Stamatis Rapanakis (ATC), Ferdinando Bosco (ENG), Matthias Strobbe (IMEC), Fulvio D'Antonio (LIVETECH)

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	

EXECUTIVE SUMMARY

This deliverable reports on the work performed in WP3, which addresses the development of the required technology bricks for the CPN Platform. Taking into account the user requirements, as described in deliverable D1.1 “User Requirements Model”, this report presents the components and services implemented for the second prototype of the platform.

The components & services which are being presented in this deliverable are classified in three main categories (Content, Users, Mapping), based on their functionality as defined by the project’s requirements. For each technology brick, a brief description of its functionality is provided, along with API, test scenarios and installation guidelines.

The second prototype of the CPN platform includes the following 5 technology bricks, as described in this deliverable.

It has to be noted that the Sentiment Analysis module has been replaced by the «Recommender AB-Testing» module, provided by LIVETECH. Taking into account the users’ requirements and the difficulty to extract data from social networks (due to GDPR), the Consortium decided to put more effort on the development of the Recommender AB-Testing.

Layers	**Name of 'technology brick'**
Content Technology Bricks	Semantic Lifting
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Recommender AB-Testing
Mapping Technology Bricks	Twitter Analytics - TRUTHNEST

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

This Deliverable contains the basic description of the technological infrastructure of the 2nd prototype of the CPN platform which is composed by what we call 'technology bricks'. The components, APIs, and services included in the second version of the platform customization infrastructure and components, and described in this deliverable have been designed and developed according to the user requirements, as described in deliverable D1.1 “User Requirements Model”.

The CPN project foresees three releases of the 'technology bricks' in order to be available for the related pilots. Each release includes specific functionalities, chosen after a process of evaluation and prioritization of the user requirements. Starting from the reference architecture document (D2.1), the second versions of the technology bricks have been implemented and reported in this document. This version of the bricks is the second one of a cycle of three iterations and will offer a series of features in order to test the related bricks in a pilot environment.

The main goal of this document is to present the technology bricks that are foreseen at this point of the project necessary to satisfy the user requirements expected for the second pilot iteration. For each technology brick, a brief description of its functionality is provided, along with API, test scenarios and installation guidelines. In addition to the new technology bricks implemented, this deliverable also describes the updates of the already deployed technology bricks, highlighting new features implemented and how these bricks satisfy the user requirements.

Furthermore, the document provides a follow up of the comments received from the Reviewers during the 1st Review of the project.

The structure of the deliverable is organized as follows: Section 2 provides an overview of the requirements for the 2nd prototype, Section 3 describes the CPN technology bricks infrastructure, Section 4 provides updates of the already available technology bricks, and Section 5 provides a follow up of the recommendations of the Reviewers. Finally, section 6 concludes this document.

2 SUMMARY OF REQUIREMENTS FOR PROTOTYPE 2

We enlist here the requirements foreseen for the second prototype along with the list of the modules that have been necessary for satisfying these requirements.

PROTOTYPE 2

Table 1: Second prototype requirements

Requirement category	Requirement ID	Requirement description
UR-UP1: Interests (Categories, Entities, Values): What topics is the user interested in?	UR- UP1.4	The system should refine the user's interests through frequent interaction with the user (talkback)
	UR-UP1.5	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-UP2.1	The system should allow for social media integration to recommend content based on what connections like, read and share
	UR-UP2.2	The system should offer a recommendation of articles based on most liked/most shared numbers from a user's network and beyond that. (Nuzzle-Feature)
	UR-UP2.3	The system should allow for social media integration to keep track of what the user has already seen elsewhere.
	UR-UP2.4	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-UP2.5	The system should allow the user to down-/upload their network connections through user account.
	UR-UP2.6	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- UP3.1	The system must allow the user to choose a preferred time frame or frames to consume content
	UR- UP3.2	The system should create/refine time frames based on the user's consumption habits
	UR- UP3.3	The system should refine the user's time frames through frequent interaction with the user (talkback)
	UR-UP3.4	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- UP3.5	The system must allow the user to postpone a time frame for a chosen amount of time.
	UR- UP3.6	The system must allow the user to ignore a time frame completely

Requirement category	Requirement ID	Requirement description
	UR-UP3.7	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-UP5.1	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- UP5.2	The system should allow the user to set a home/main interest location
	UR-UP5.3	The system should make use of the location data of the user to determine the best point in time to offer content
	UR-UP5.4	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-UP5.5	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-UP6.1	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-UP6.2	The system must keep track of how much of each item users consume, where they stop, continue and what they skip
	UR-UP6.3	The system should interact with the user in order to refine user interests in regards to why 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-UP8.2	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-UP9.3	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-UP9.4	The user must be able to change and overwrite settings in their profile
	UR-UP9.5	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-AF1.6	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-AF2.1	The system should show users who else from their network has consumed the same content item.
	UR-AF2.2	The system should show users what else their network has shown, if there are differences
	UR-AF2.3	The system should be able to show users the content item from another user (anonymously)

Requirement category	Requirement ID	Requirement description
	UR- AF2.5	Once all articles proposed have been consumed, the system should only offer more content upon request by the users
UR-AF 3: Content/Format: In which way do we have to prepare content for the user?	UR-AF3.7	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-AF5.1	The system must offer the user an easy to access and easy to understand overview of their profile
	UR-AF5.2	The system must offer users easy access to their profile in order to change settings and data
	UR- AF5.3	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-AF6.1	The system must allow users to access content again that they have already opened before
	UR-AF6.2	The system should allow users to consume content beyond their predefined timeframe after an interaction with the user (talkback)
	UR-AF6.3	The system should allow users to actively save articles for later consumption
UR-AF 7: User Feedback: Asking users to help improve the system	UR- AF7.2	The system should include guided feedback for specific elements of the system, allowing users to (help) improve it
UR-AF 8: Temporary Categories: Users can temporarily change the personalisation algorithm	UR-AF8.1	The system should allow users to search for specific topics they are temporarily interested in
	UR-AF8.2	The system should allow users to add this search as a temporary personalisation category
	UR-AF8.3	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-AF9.1	The system should allow users to define keywords and logical combinations of them to exclude content from their personalisation
	UR-AF9.2	The system should allow users to define a time frame per keyword/logical combination
	UR-AF9.3	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-PS1.1	The system should show the access to items through users by numbers (who, when, how long)
	UR-PS1.3	The system should show which topics were most interesting to users
UR-PS 2: Integration: How should CPN be connected to the production side?	UR- PS2.4	The system should allow producers to export the record of their publications through standardized and interoperable formats
	UR- PS2.5	The system should allow for an easy contribution of content from different publishers through standardised interfaces

Requirement category	Requirement ID	Requirement description
	UR- PS2.7	The system should allow editors to easily add missing attributes to articles manually

Foreseen necessary technology bricks

The technology bricks necessary for the 2nd prototype are illustrated in the shaded cells (in blue) of the table below. It has to be noted that the bricks in the shaded cells, in orange, have already been delivered as part of the 1st CPN prototype.

Table 2: Second prototype technology bricks

Layers	Name of 'technology brick'
Content Technology Bricks	Semantic Lifting
	Relation Extraction
	Topic Extractor
	Uplifting/Depressing Article Classifier
	Frame Based Slot-Filling System
	Recommender AB-Testing
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

3 MODULES DESCRIPTION

This section describes the modules needed to implement the 2nd prototype of the CPN platform.

3.1 SEMANTIC LIFTING

3.1.1 Overview

The Semantic Lifting module allows extracting knowledge graphs from existing data. The resulting knowledge graph is richer content-wise than the existing data and can be used by other modules to achieve better results when compared to using the existing data directly.

3.1.2 Role

The Semantic Lifting is used by the Recommender to provide better recommendations to the user, because of the rich knowledge that is extracted from the existing data. Additionally, this module can also be used by other systems/partners in use cases that are independent of the Recommender. When looking at the second prototype, this module does not directly address a requirement, but it indirectly contributes to the requirements that are addressed by the Recommender.

3.1.3 Internal architecture

This module extract a knowledge graph from existing data with as goal to semantically lift this data. How a knowledge graph is generated is determine by a set of rules that can be easily updated to support different types of data. Internally, the module uses RML[1] rules to define how the knowledge graphs are generated; the module that provide a Web API to RMLMapper[2] , which executes these rules; and Comunica[3] to expand the knowledge graph with additional knowledge.

3.1.4 API

/rules/{rulesId}

- HTTP GET
- accept: text/turtle
- output: Turtle representation of the RML rules with id "rulesId"

/rules/{rulesId}

- HTTP POST
- content-type: text/turtle
- input: Turtle representation of the RML rules that need to be stored with id "rulesId"

/rules/{rulesId}/extract

- HTTP POST
- content-type: application/json
- accept: text/turtle
- input
    - path

[1] http://rml.io

[2] https://github.com/RMLio/rmlmapper-java

[3] https://github.com/comunica/comunica

        - rulesId (required): the id of the RML rules that need to be used to extract the knowledge graph.
    - Body (required): JSON object with an key-value for each existing data source, where the key is the name of the data source used in the rules and the value the data in text/plain. This can be CSV, XML, and JSON.
- output: Turtle[4] representation of the knowledge graph

/rules/{rulesId}/extract/list

- HTTP POST
- content-type: application/json
- accept: text
- input
    - path
        - rulesId (required): the id of the RML rules that need to be used to extract the knowledge graph.
    - Body (required): JSON object with an key-value for each existing data source, where the key is the name of the data source used in the rules and the value the data in text/plain. This can be CSV, XML, and JSON.
- output: a comma-separated list of terms that characterize the content of the extracted knowledge graph.

MISSING: The API testing plan and report, which sets the scope of the identified test scenarios and lists the set of module functionalities being addressed with this API. (Will be added by the end of April 2019).

3.1.5 Test scenarios

/rules/{rulesId}

- Check that the correct rules are returned when doing a GET and the rules id is valid.
- Check that the correct error is returned when doing a GET and the rules id is invalid.
- Check that the correct rules are stored when doing a POST and the rules and rules id are valid.
- Check that the correct error is returned when doing a POST and the rules are invalid.
- Check that the correct error is returned when doing a POST and the rules id is invalid.

/rules/{rulesId}/extract

- Check that correct knowledge graph is returned for example data and rules id.
- Check that correct error is returned when rules id is provided.
- Check that correct error is returned when no data is provided.

/rules/{rulesId}/extract/list

- Check that correct list is returned for example data and rules id.
- Check that correct error is returned when no rules id is provided.
- Check that correct error is returned when no data is provided.

3.1.6 Installation and administration guidelines

Installation and administration guidelines and the respective usage guide will be provided by the end of April 2019).

[4] https://www.w3.org/TR/turtle

3.2 TOPIC EXTRACTOR

3.2.1 Overview

The module is used to extract topics from articles in order to have a concise representation of the content of the article to be exploited by the recommender module.

It is able to perform 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. 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[5]). 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.

3.2.2 Role

The module is operating at the backend level of the CPN platform. It is deployed as a microservice but it is not exposed through the API gateway but it is only queried by integration microservices.

The structure of the flow is the following:

- New content is ingested periodically by the platform by querying the publishers API/RSS/feeds
- The new content is posted on the CPN message broker (Apache Kafka)
- A worker module is listening for new messages on the broker and as soon as it receives them it calls the Topic Extraction module by passing the content of the article to it (in the form of a JSON message)
- The Topic Extraction Module performs a language detection in order to apply the most suitable Natural Language Processing techniques to extract terminological candidates (topics)
- Topics are filtered and finally ranked according to TF/IDF score and the article, enriched with these topics, is saved into its final storage (Apache SOLR) to be retrieved later during the recommendation phase.

The module addresses the user requirement UR-UP1 (Interests: What topics is the user interested in?)

3.2.3 Internal architecture

The module is realized in python using Flask and several Natural Language Processing libraries:

- **NLTK**: it provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization,

[5] https://en.wikipedia.org/wiki/Tesseract

stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

- **Treetagger**[6]: it is a tool for annotating text with part-of-speech and lemma information. It was developed by Helmut Schmid in the TC project at the Institute for Computational Linguistics of the University of Stuttgart. The TreeTagger has been successfully used to tag German, English, French, Italian, Danish, Dutch, Spanish, Bulgarian, Russian, Portuguese, Galician, Greek, Chinese, Swahili, Slovak, Slovenian, Latin, Estonian, Polish, Romanian, Czech, Coptic and old French texts and is adaptable to other languages if a lexicon and a manually tagged training corpus are available.

- **Polyglot**: a natural language pipeline that supports massive multilingual applications.

The module is organized in the following way:

- **Service layer:** this layer receives the user requests and performs the conversion of the parameters, JSON body and validate them. It then transforms such parameters into appropriate python objects and passes them to the business layer for elaboration. After it receives the response from the business layer it converts it back to a JSON response and gives it back to the client.

- **Business layer**: based on the user request, as converted by the service layer, it examines the text provide and extracts topics in an unsupervised way by using a noun phrase chunking to build syntactically plausible terminological noun phrases, NPs (e.g. compounds "credit card", adjective-NPs "local tourist information office", and prepositional-NPs "board of directors"). One of the most useful sources of information for NP-chunking is part-of-speech tags. This is one of the motivations for performing part-of-speech tagging in our information extraction system. In order to create an NP-chunker, we will first define a chunk grammar, consisting of rules that indicate how sentences should be chunked. In CPN case, we define several grammars by using regular-expressions rules.

The module is invoked directly by a worker (listening to messages on the Kafka broker). The results are then stored on the enriched content storage (Apache SOLR).

3.2.4 API

The Topic Extractor module provides a single API to be used only by internal microservices:

Topic Extraction API

Method	HTTP Request	Description
POST	/extract	Extract topics from body content

Input format: the service is expecting textual data in one of the currently supported languages (English, German, Greek, Dutch) enclosed into a json object with single text key:

{“text”: “Donald Trump, the president of US,....”}

[6] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

3.2.5 Test scenarios

The service has been tested on the articles provided by the consortium partners. In the following table we show an example of the JSON output of the service applied to a Dutch document:

```
{
        "tags_txt":["Politiek"],
        "tags_ss":["Politiek"],
        "other_tags_txt":["naar eigen zeggen",
          "geschrokken van",
          "zelfs kwaad",
          "het thema diversiteit",
          "hij voor",
          "dat vlak heb ik",
          "te leren"],
        "other_tags_ss":["naar eigen zeggen",
          "geschrokken van",
          "zelfs kwaad",
          "het thema diversiteit",
          "hij voor",
          "dat vlak heb ik",
          "te leren"],
        "entities_txt":["v",
          "gent",
          "gent mieke van hecke",
          "veli yüksel",
          "open vld",
          "hecke",
          "open vld"],
        "entities_ss":["v",
          "gent",
          "gent mieke van hecke",
          "veli yüksel",
          "open vld",
          "hecke",
          "open vld"],
        "id":"5c7962c68de42b1e00c107e1",
        "url_s":"https://vrtnws.be/p.L7neGe3bE",
        "origin_s":"VRT",
        "date_dt":"2019-03-01T16:41:37Z",
        "_version_":1626877960665432064}
```

3.2.6 Installation and administration guidelines

The module is deployed as a docker container into the CPN orchestration platform

3.3 UPLIFTING/DEPRESSING ARTICLE CLASIFIER

3.3.1 Overview

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 was to build a text classifier, which takes as input an article, and classifies the article as "depressing" vs "uplifting", or "neutral". The definition of what exactly is "uplifting" and "depressing" was hard to specify upfront, and thus was refined/defined by collecting annotations from users. This pragmatic approach was adopted to maximize both (a) feasibility as well as (b) relevance for the content providers.

In order to train the model, we hired two job students in order to annotate 9,533 news articles. This annotated set was split into train (80%), development (10%) and test (10%) sets in order to train and tune the models.

We trained the models to predict the probability of a particular article to be depressing, neutral and uplifting. As our main classifier we used logistic regression ([1]). We also tried and report results for linear SVM classifier ([2]). As features, we used TF-IDF weighted bag of words ([3]). The models were trained on a train set and tuned on a validation set. The unbiased evaluation of a final model fit was performed on a test set.

3.3.2 Role

The module realizes a Content Technology Brick, as identified for Prototype #2 in D3.1 ("Initial design and APIs of Technology Bricks"). It is an independent brick, taking as input an article, and outputting the probability scores of each of the classes (depressing, neutral and uplifting). This uplifting/depressing score can be used in a reader app to filter/balance a (recommended) list of articles to read.

Specifically, the module addresses the following user requirements (see Table for Prototype #2 in Sec. 2 of D3.1):

- UR-AF 1: Bursting the Filter Bubble: How can CPN avoid filter bubbles and echo chambers? The uplifting/depressing scores can be used to balance the news by, e.g., (i) balancing 'bad' news on certain topics, which may otherwise form the majority of news the user may have consumed, with more positive notes, or (ii) expand the coverage of news by including a mixture of uplifting/depressing news items, which inherently may cover a more diverse set of topics than a user's inherent interest sphere (see also next point)

- UR-UP1: Interests (Categories, Entities, Values): What topics is the user interested in? The user may be (possibly inadvertently) mostly following given topics, which can be dominated by mostly neutral or even 'bad' news. By including positive, i.e., uplifting, news items, the user interest may be expanded, and retention of users in the news app/site may thus be positively affected.

3.3.3 Internal architecture

Features and Functionalities:

This module is composed by the following components:

1. Parser and annotations loader: the news articles used to train the model are extracted from the dumps of Deutsche Well news and merged with the annotations. The final data set contains the title, body and the url of the article as well as the annotation (depressing, neutral or uplifting). This is done inside the collect_data.py python module.

2. Model builder: the model is built using the training set and choosing the model with the best results on development set. This is done in the baseline_predictor.py python module.

3. Predictor API: this is the API component that loads the model created by Model builder component and makes it available to use through API calls and also through Front-end demo interface (see next point) component. The code for this component is located in the wse_server.server.py python module.

4. Front-end demo interface: the demo of the functionality is available through a web front-end interface that is implemented in the wse_server.templates.wse_frontend.html and the wse_server.server.py Flask html modules.

Note that for the actual classification functionality, only the trained model (i.e., item 3) is relevant; it is demonstrated through item 4. Items 1-2 are used internally to build the model. These are not exposed through an external API towards other components in the CPN pipeline.

Dependencies on External Components:

This module depends on the following external technologies/components:

- Google translation API: this API is used to translate the input article to English in order to perform classification (for non-English content)

- Machine Learning: the python sklearn machine learning library was used to train and tune the baseline models (in the Model Builder, item 2)

- Flask: this is a python micro web framework that allows to integrate the front-end demo (see above, item 4)

3.3.4 API

The following are the specification of the REST API provided operation in order to classify a particular article in depressing, neutral or uplifting:

- URL: /cpn_uplifting
- HTTP Method: POST
- Input JSON content: the content of the news article needs to be passed via JSON in a field called article. The title of the article has to be separated with a “<br>” tag from the body. The following is a very short illustrative example:

{“article”: “Title of the article <br> Content of the article”}

- Output JSON content: the output is a JSON file with the following fields:
    - confidence: the confidence of the model for different news types (depressing, neutral or uplifting) for the passed article.
    - language: the detected language of the passed news article.
    - predicted: the most likely news article classification.
    - translation: if the detected language of the article is not English, then the English translation of the article will be returned in this field.

The following is an example of the returned JSON:

```
{
  "confidence": {
    "depressing": 0.21412511135910173,
    "neutral": 0.18024488452392834,
    "uplifting": 0.6056300041169699
  },
  "language": "en",
  "predicted": "uplifting",
  "translation": null
}
```

3.3.5 Test scenarios

We have built a dataset based on user annotations of a selection of news articles. This dataset consists of a total of 9,533 annotated news articles. The data is split into:

- **Train set:** this is the biggest set containing 80% of the data. It is used to train the model.
- **Development set**: this set contains 10% of the data and is used to tune the model’s hyper-parameters in order to make it more performant.
- **Test set:** this set contains 10% of the data and is used to get an unbiased evaluation of a final model fit on the training dataset.

The following graph illustrates the train/development/test set split:

We observe an uneven distribution of categories, being the uplifting one with fewest number of data points. This makes the classification problem more challenging.

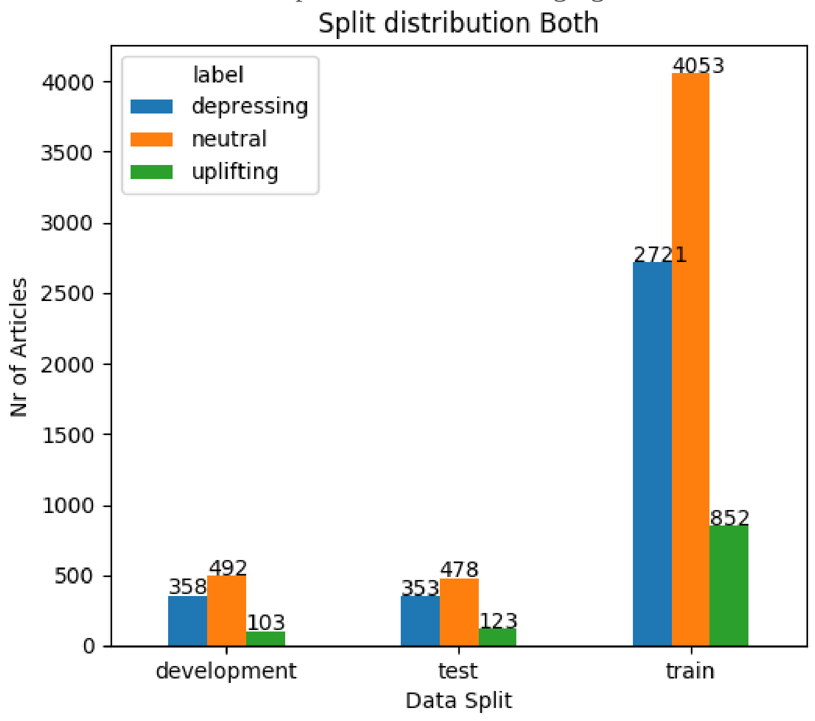


Figure 1: UPLIFTING/DEPRESSING ARTICLE CLASIFIER –Dataset Splitting

The current model achieves an accuracy of 0.72. The following table illustrates the performance of the different models:

Model	Accuracy	Precision UP	Recall UP	F1 UP
Logistic Regression	0.72	0.51	0.29	0.37
SVM	0.72	0.43	0.29	0.35
Binary SVM	0.86	0.45	0.35	0.40

Table 3: Performance of different models

3.3.6 Installation and administration guidelines

At the moment, the model is executed locally and on an internal imec server. A docker installation package in order to be deployed on any machine will be provided.

3.4 RECOMMENDER AB-TESTING

3.4.1 Overview

The module is used to be able to test different versions of the recommender at the same time on different user groups in order to compare the behaviour of different approaches to content personalization. Every publisher is able to create an unlimited number of groups and recommenders and to associate a specific recommender to a group. A/B testing can produce concrete evidence of what actually works in personalization. The module can be used for continuously testing new techniques and approaches in order to optimize conversion rates and gain a better understanding of the customers.

3.4.2 Role

The module is tightly integrated with the Recommender module and exposes an API that allow every publisher to create recommenders and user groups. It is a "configuration" module and recommenders and groups can be created and modified in any moment by publisher/administrators of the platform. **Recommenders** can be created by instantiating the ones currently offered by the platform i.e.:

- **Content-based recommendation**: based on unsupervised keyword extraction and named entity extraction, semantic uplifting techniques etc.
- **Collaborative filtering techniques:** assessing users consumption history similarity
- **Most Popular Recommendations**: recommendations based on trending items
- **Random recommendations**: to add variety to the recommendations
- **Composite recommender:** quota based combinations of the above techniques

User groups are created by specifying a name for the group and then, by calling a specific API, users are added to groups by specifying their user ids. The AB-Testing interacts with two modules of the CPN platform: the already mentioned Recommender module and the API Gateway that is needed to publicly expose this internal API to publishers.

The module covers a special case of the requirement UR-PS 1 (Detailed Analytics: Giving Newsrooms a more detailed feedback on their audience) since it allows the newsrooms to experiment with different recommendation approaches, collect different feedbacks and formulate insights.

3.4.3 Internal architecture

The module is realized in python using Flask and PyMongo libraries. The module is organized in the following way:

- **Service layer**: this layer receives the user requests and performs the conversion of the parameters, JSON body and validate them. It then transforms such parameters into appropriate python objects and passes them to the business layer for elaboration. After it receives the response from the business layer it converts it back to a JSON response and gives it back to the client.

- **Business layer:** based on the user request, as converted by the service layer, it prepares the queries or the data to be written in the database (by invoking Recommender Factories). The implementation of the Recommender Factory is as follows:

```
def factory(request):
    if isinstance(request, RandomRecommenderRequest):
        return
MostRecentRecommender(None,relative=relativedelta(days=request.days),storage=AppConfig.ITEM
_DAO,users=AppConfig.USERS_DAO)

    if isinstance(request, ContentBasedRecommenderRequest):
        return
CBRecommender(None,relative=relativedelta(days=request.days),solr_dao=AppConfig.ITEM_DAO,
user_dao=AppConfig.USERS_DAO)

    if isinstance(request, CollaborativeFilteringRecommenderRequest):
        return
CFRecommender(None,relative=relativedelta(days=request.days),solr_dao=AppConfig.ITEM_DAO,u
ser_dao=AppConfig.USERS_DAO)

    if isinstance(request, CompositeRecommenderRequest):
        if not request.sorting:
            request.sorting="score"
        return CompositeRecommender(None, quotas=[RecommenderQuota(factory(recc),v) for (recc,v)
in request.quotas],sorting=request.sorting)

    raise Exception("%s not supported"%request)
```

- **Data layer:** it performs the actual queries on the MongoDB storage by means of a DAO class. The interface of the DAO class is described in the following table

```
class ABTestingDAO:

    def add_to_group(self,group,user_id):
        raise NotImplementedError()

    def map_group_recommender(self,group,recommender_id):
        raise NotImplementedError()

    def recommender_for_user(self,user_id):
        raise NotImplementedError()
```

The module is interacting with the Recommender module and uses as a storage engine MongoDB

3.4.4 API

CPN is a multi-tenant platform for content personalization: each tenant (for example the consortium partners DW, VRT and DIAS) has full control on the type of recommendations and user groups. This API is not available to the end-users but only to platform tenants (publishers/media companies)

Recommender management API

Method	HTTP Request	Description
POST	/admin/<tenant>/recommenders	Create a new recommender for the tenant.

Method	HTTP Request	Description
GET	/admin/<tenant>/recommenders	List all the recommenders defined by a specific tenant.

Method	HTTP Request	Description
GET	/admin/<tenant>/recommenders/<id>	Retrieve and show a recommender given its id.

Method	HTTP Request	Description
DELETE	/admin/<tenant>/recommenders/<id>	Delete a recommender given its id

Method	HTTP Request	Description
GET	/admin/<tenant>/users/<userid>	Retrieve the recommender used to personalize content of a specific user.

Groups management API

Method	HTTP Request	Description
PUT	/admin/<tenant>/users/<userid>/<group>	Adding a user to a specific user group.

Method	HTTP Request	Description
GET	/admin/<tenant>/groups	List all the groups defined by a specific tenant

Method	HTTP Request	Description
GET	/admin/<tenant>/map/<group>	Retrieve the recommender used to personalize content of a specific group.

Method	HTTP Request	Description

DELETE	/admin/<tenant>/groups/<group>	Delete a user group.

Recommender/groups mapping API

Method	HTTP Request	Description
PUT	/admin/<tenant>/map/<group>/<recommender_id>	Associate a recommender to a specific group

3.4.5 Test scenarios

In the following table we show a typical json request that can be used to create a composite recommender combining three techniques (Content-based, Collaborative Filtering and Random) with 33%/33%/34% quotas.
POST /admin/<tenant>/recommenders

```
{/admin/<tenant>/recommenders
 "__class__": "CompositeRecommenderRequest",
 "quotas": [
  [
   {
    "__class__": "ContentBasedRecommenderRequest",
    "days": -0.041666666666666664
   },
   0.33
  ],
  [
   {
    "__class__": "CollaborativeFilteringRecommenderRequest",
    "days": 0.041666666666666664
   },
   0.34
  ],
  [
   {
    "__class__": "RandomRecommenderRequest",
    "days": 0.041666666666666664
   },
   0.33
  ]
 ],
 "sorting": "date"
}
```

The output of this call will be a string representing a new recommender id (for example id=*5cb903a7c72eacb74cc93b42*).
If we have defined a group *test* we can associate this new recommender to the group by executing the following call:

PUT /admin/VRT/map/test/5cb903a7c72eacb74cc93b42
that can be read in this way: the tenant *VRT* is giving to the group *test* the recommendations produced by the recommender *5cb903a7c72eacb74cc93b42.*

3.4.6 Installation and administration guidelines

The module is deployed as part of the recommender module. So the installation guidelines follow the one already described for that module (deployment as a docker container into an orchestration platform).

3.5 TWITTER ANALYTICS - TRUTHNEST

3.5.1 Overview

The module provides information about the entities of the most important Twitter accounts a user follows. It also provides information about the user's network activity. The output also includes the user's subscribed lists and profile information. With this information, the social network profile of the user can be determined and used to improve the recommendation

3.5.2 Role

The module takes as input the Twitter handle of the user (screen name) and produces a json response that is submitted to a Kafka topic that the Recommender listens to.

The requirements addressed by this module are UR-UP 2.1 - UR-UP 2.6 (UR-UP2: Network – Making use of connections the user already has through social media).

3.5.3 Internal architecture

The module is a based on a Spring Boot web application that contains an embedded Tomcat. Like all the other project modules, it is delivered as a Docker image. The input of the module is a json object containing the user's Twitter username, the user's id on the CPN platform, as well as a pair of Twitter access tokens. There are used to call the Twitter REST API and fetch data from Twitter. The tweets are analysed with the use of the Stanford NLP library to extract the named entities. The most important 5 accounts that the user follows are included in the results, to reveal what the users is interested to.

The Stanford NLP library has been included in the project as a maven dependency.

3.5.4 API

The module offers a REST API to accept new requests. The method uses json web tokens to validate the input. In case of an error (for example a Twitter rate limit exception), an exception is thrown with an appropriate explanation message. Both input and output of the module are json objects. No user data are stored and no results caching exists.

Example output:

```
{

  "cpnUserId": "5b8fe613e7cffb000a11bafa",

  "profile": {

    "status": {
      "code": 1,
      "message": "Computed result."
    },
    "user": {
      "userId": 18117445,
      "screenName": "stamrapanakis",
      "fullName": "Stamatis Rapanakis",
      "profileImage":
"https://pbs.twimg.com/profile_images/874620915849539585/WrqEOzMI_normal.jpg",
      "location": "Athens, Greece",
      "verified": false,
      "description": "Software Engineer, Athens, Greece.",
      "url": "",
      "followersCount": 26,
      "listedCount": 0,
      "utcOffset": "",
      "createdAt": 1229268015000,
      "language": "en",
      "friendsCount": 78,
      "postsCount": 115,
      "entities": [
      ],
      "subscribedLists": [
      ],
      "topFriends": [
        {
          "userId": 6253282,
        "screenName": "TwitterAPI",

        "fullName": "Twitter API",

        "profileImage":
"https://pbs.twimg.com/profile_images/942858479592554497/BbazLO9L_normal.jpg",

        "location": "San Francisco, CA",

        "verified": true,

        "description": "The Real Twitter API. Tweets about API changes, service issues and our
Developer Platform. Don\u0027t get an answer? It\u0027s on my website.",

        "url": "https://developer.twitter.com",

        "followersCount": 6140062,

        "listedCount": 12993,

        "utcOffset": "",

        "createdAt": 1179900073000,

        "language": "en",

        "friendsCount": 12,

        "postsCount": 3654,

        "subscribedLists": [

          "@TwitterAPI/meetup-20100301",

          "@TwitterAPI/team"

        ],

        "friendsEntities": [

          "Ghergich \u0026 Co.",

          "Bradley"

        ],

      }

        {…}, // second top friend

        {…}, // third top friend

        {…}, // fourth top friend
```

```
    {…} // fifth top friend

  }
```

The module’s API is described on a relevant Swagger documentation.

3.5.5 Test scenarios

Testing is focused on the Twitter rate limit cases. In that case the response contains an error code (0) and an explanation message:

```
{
  "cpnUserId": "5b8fe613e7cffb000a11bafa",
  "profile": {
    "status": {
      "code": 0,
      "message": "Unable to compute social profile due to Twitter exception .."
    }
}
```

3.5.6 Installation and administration guidelines

The module uses a Tomcat server to deploy the application and implement the REST API’s. No results are stored in a database. Therefore the administration tasks are limited to that of the embedded Tomcat.

The module is deployed as Docker container into the CPN platform and exposes its own APIs through the API gateway. Internal port 8070 is exposed and accessible to other components.

4 UPDATES ON THE AVAILABLE BRICKS

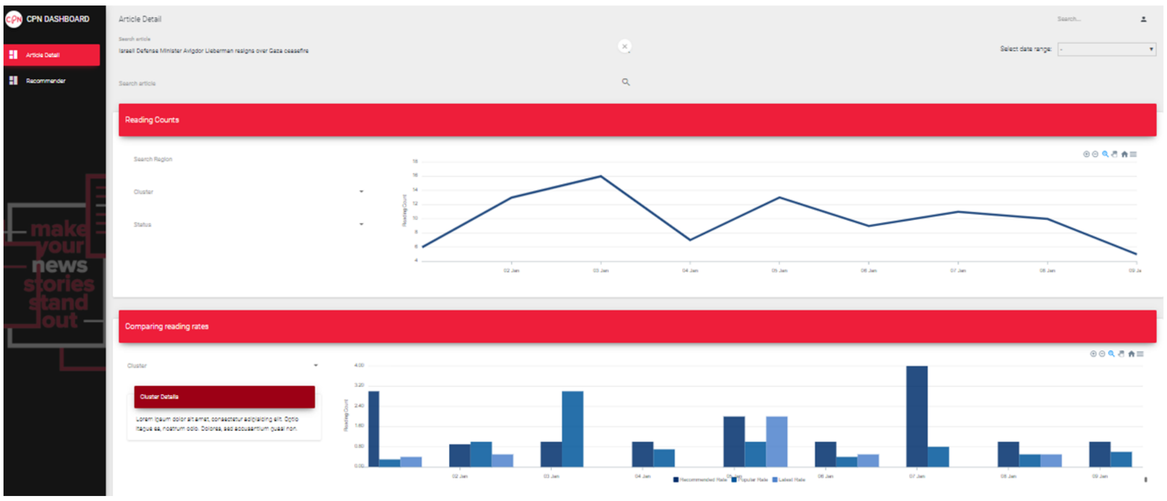
This section describes the updates on the technology bricks that were delivered, as part of the 1st prototype of the CPN platform.

4.1.1 PRODUCER'S APP – V2

The second version of the Producer's app, as well as including some minor fix respect on the first version, principally offer the new web dashboard for producers.

The Producer's App dashboard allows to editors to improve their news production exploiting a series of analytics and functionalities. In particular, the dashboard collects the data from the CPN platform and provides these data in aggregated form through a web UI, proposing effective tools of data visualization.

Below a screen of the web dashboard:



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio itaque ea, nostrum odio. Dolores, sed accusantium quasi non.

Figure 2: Producer's App – Web dashboard Articles analytics section

4.1.2 Role

In the context of the CPN project, the second version of the producer's app addresses the following user requirements:

- PS1.1 - The system should show the access to items through users by numbers (who, when, how long)
- PS1.3 - The system should show which topics were most interesting to users
- PS2.4 - The system should allow producers to export the record of their publications through standardized and interoperable formats
- PS2.5 - The system should allow for an easy contribution of content from different publishers through standardised interfaces
- PS2.7 - The system should allow editors to easily add missing attributes to articles manually

4.1.3 Internal architecture

The second version of producer's app consists of four Docker containers. In addition of the previous two containers, now including:

- One Docker container for dashboard server-side, based on Node.js and Express.js that expose business logic and REST APIs for both the client application and CPN platform

- One Docker container for dashboard client-side, based on Vue.js framework. It is the Web dashboard UI

The dashboard server container exploit the database (MongoDB) already included on the producer's app first release.

4.1.4 API

The second version of Producer's app exposes a series of new APIs trough the Dashboard server container. These APIs are divided into two groups:

- APIs for client application (Dashboard Web App)
- APIs for CPN platform

The first group of APIs is only accessible from the dashboard client and it is not accessible to other CPN components neither exposed through the CPN API Gateway. This group of APIs includes:

- Authentication/Authorization APIs
- Data Aggregation and Analytics APIs for Data Visualisation

The second group of APIs is made available to other CPN modules and exposed through the CPN API gateway. In particular, these APIs offer an access to anonymized statistics data, as for example article visualisations and readings, as well as the list of articles enriched with analytics.

These APIS are documented with OpenAPI specification[7] (Swagger v2.0 and are testable via CPN API gateway interface.

4.1.5 Test scenarios

The Producer's app collects in background all the data becoming from the CPN platform, from both Apache Kafka and the Orchestrator, related to user behaviour respect on the articles.

An editor access to Web Dashboard UI with its credentials and can view all the analytics related to its contents. It selects an article from the list, then chooses a date range and views a series of charts that allows him to evaluate the performance of the article. It navigates to another section of the dashboard and visualise the performances of the recommender system, analysing statistics on the articles suggested by the recommender.

4.1.6 Installation and administration guidelines

The module consists in four docker containers. A docker-compose.yml is provided to build and push the service on CPN private registry.

[7] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md

The internal port are:

- Producer's app -> 8080
- Dashboard server -> 8081
- Dashboard client -> 3001
- MongoDB -> 27017

The containers stack needs Apache Kafka and Orchestrator integration, in order to collect data from the CPN platform. Below a graph, that represents all the stack dependencies (internal and external):

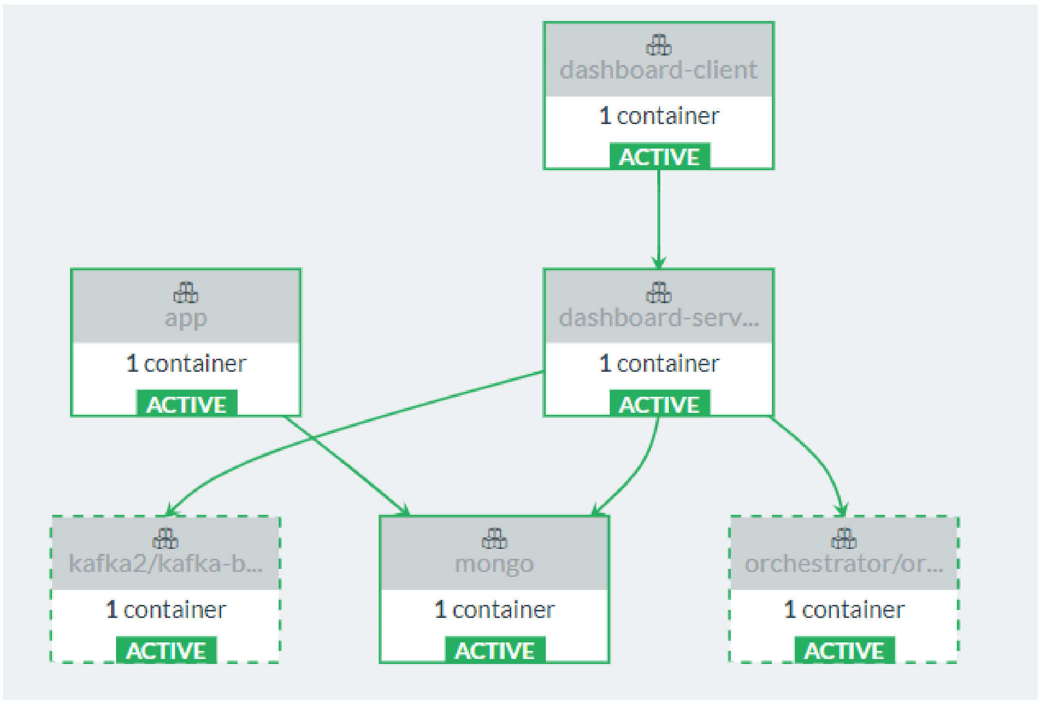
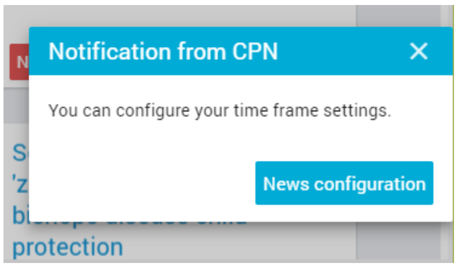


Figure 3: Producer's app stack dependencies

4.1.7 READER'S APP – V2

Notifications: The Recommender can send notifications to the Reader's App to display information or request direct feedback. This is achieved with the use of Firebase messaging and is supported also on the mobile application. The Recommender only needs to specify (to a respective web service) the recipient, the message type and the actual message.



You can configure your time frame settings.

Figure 4: Notification message (when the browser has the user's focus)

If the Reader's app application is not open or has not the browser focus, a notification message is presented on the bottom right corner of the browser. This happens only if the user has allowed notifications from the CPN domain.

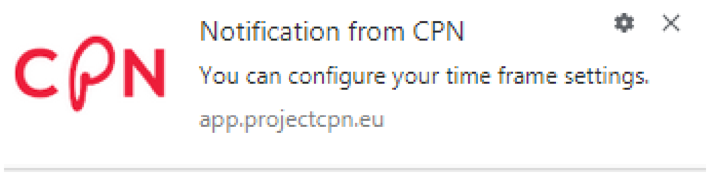
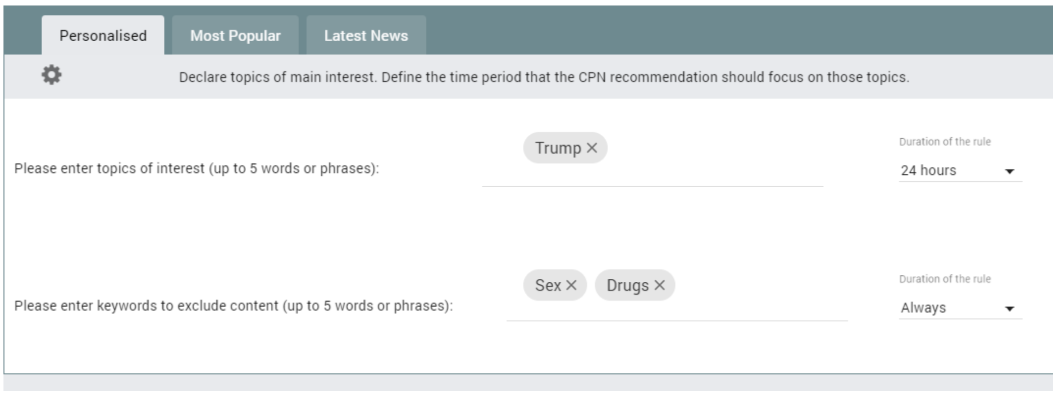

Notification from CPN
You can configure your time frame settings.
app.projectcpn.eu

Figure 5: Browser notification message

The notification message can contain a button/ link or an input form to request user feedback. It is up to the Recommender to decide when to use this functionality.

Topics declarations: The user can personalize CPN's recommendation by defining a list of keywords ("topics") that interest him during a specific time period. The topics are a list of phrases and the duration options are "1 hour", "24 hours", "1 week", "Always". Similar, he can define a list of "topics" that he is not interested in. This setting is enabled on the "Personalized" tab and is currently not used by the Recommender (e.g. the values are not propagated to the CPN API Gateway).



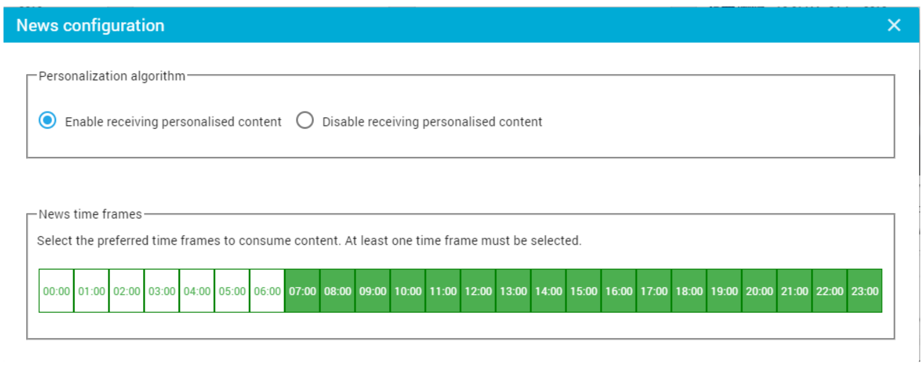
Declare topics of main interest. Define the time period that the CPN recommendation should focus on those topics.

Please enter topics of interest (up to 5 words or phrases): Trump ✕ — Duration of the rule: 24 hours

Please enter keywords to exclude content (up to 5 words or phrases): Sex ✕ Drugs ✕ — Duration of the rule: Always

Figure 6: Defined and exclude topics

News configuration: The user has the ability to disable the Personalization algorithm via the "News configuration" setting. He can also select the preferred time frames to consume content. Upon timeframe start, he will receive a respective notification message.


News configuration

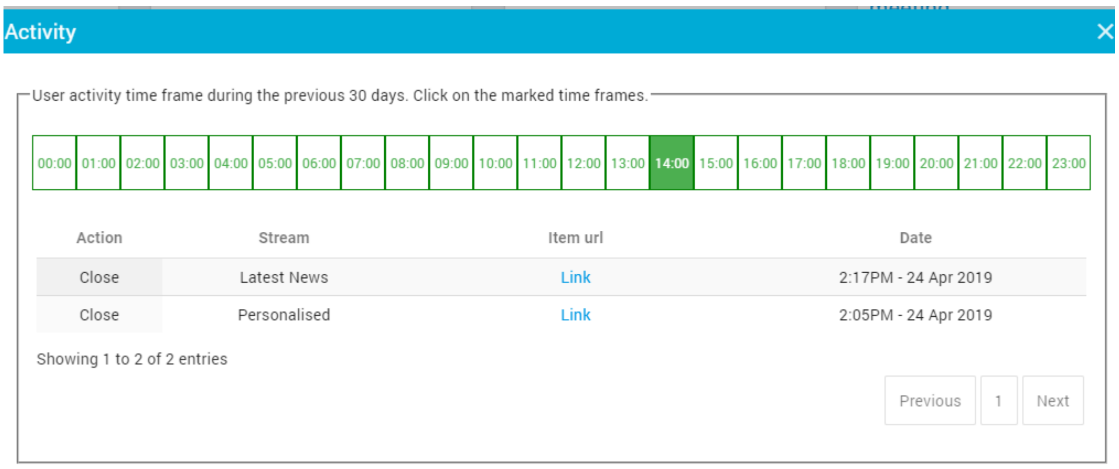
Personalization algorithm
[✓ Enable receiving personalised content] Disable receiving personalised content

News time frames
Select the preferred time frames to consume content. At least one time frame must be selected.

00:00 01:00 02:00 03:00 04:00 05:00 06:00 [✓ 07:00] [✓ 08:00] [✓ 09:00] [✓ 10:00] [✓ 11:00] [✓ 12:00] [✓ 13:00] [✓ 14:00] [✓ 15:00] [✓ 16:00] [✓ 17:00] [✓ 18:00] [✓ 19:00] [✓ 20:00] [✓ 21:00] [✓ 22:00] [✓ 23:00]

Figure 7: News configuration

User activity: The user activity time frames are being recorded and displayed to the user for the last 30 days. The time frames with activity are indicated and the respective actions are being displayed in a table view.


Activity

User activity time frame during the previous 30 days. Click on the marked time frames.

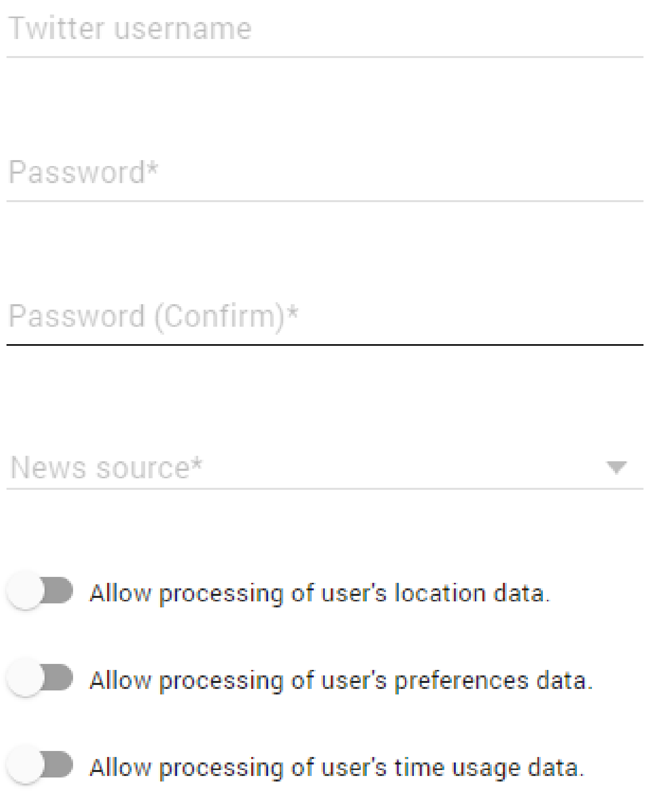
00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00

Action	Stream	Item url	Date
Close	Latest News	Link	2:17PM - 24 Apr 2019
Close	Personalised	Link	2:05PM - 24 Apr 2019

Showing 1 to 2 of 2 entries

Figure 8: User Activity

Social media connection (Twitter) and permissions: The recommendation can be improved by either filling a Twitter user name during the registration or by setting it on the user profile page. The Twitter analytics module (TruthNest) will examine the user's networks posts and send the result to the recommender.


Twitter username

Password*

Password (Confirm)*

News source*

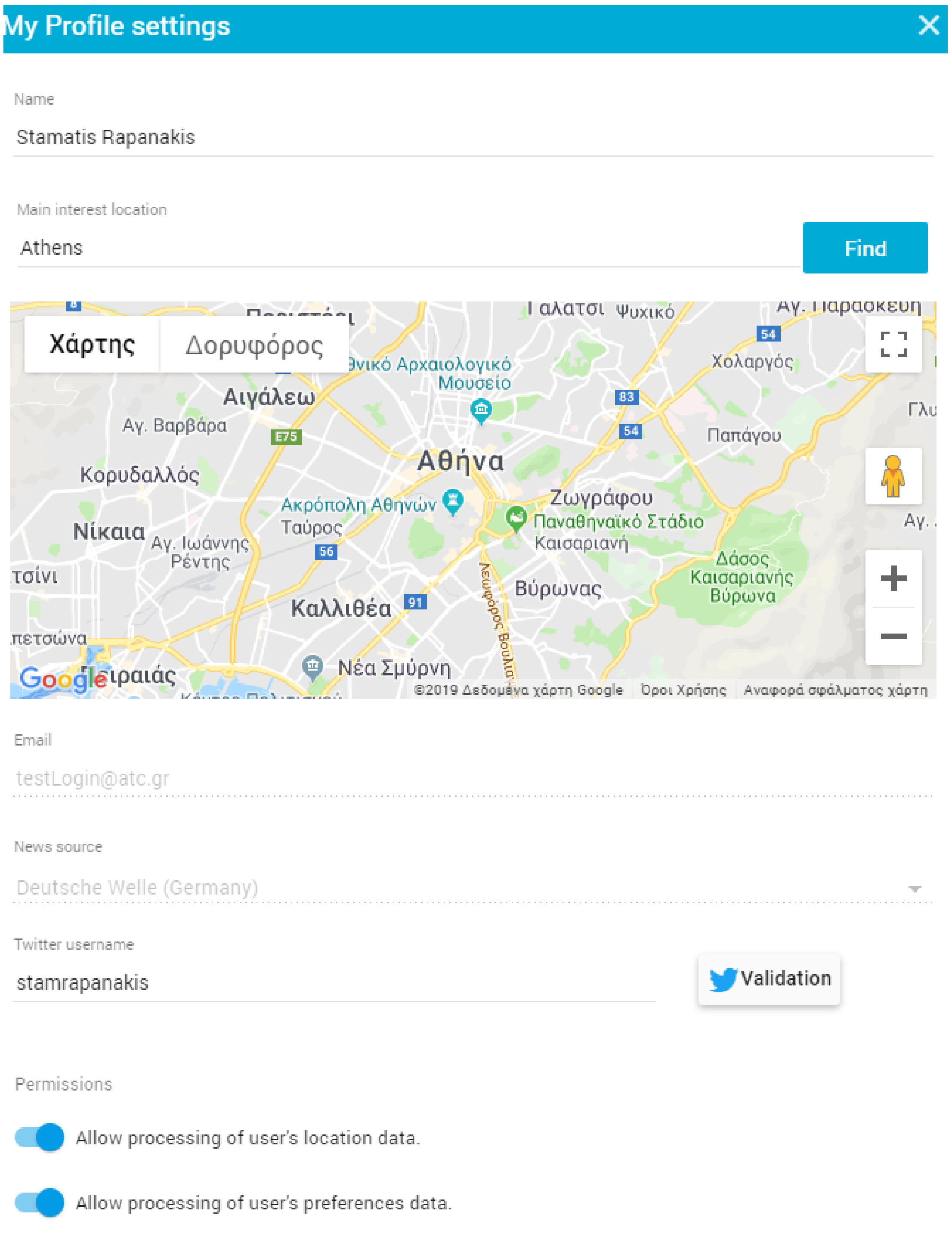
Allow processing of user's location data.

Allow processing of user's preferences data.

Allow processing of user's time usage data.

Figure 9: Registration fields

The permissions (location usage, preferences and temporal usage information) are propagated to the Personal Data Receipts module.


My Profile settings

Name
Stamatis Rapanakis

Main interest location
Athens

Email
testLogin@atc.gr

News source
Deutsche Welle (Germany)

Twitter username
stamrapanakis

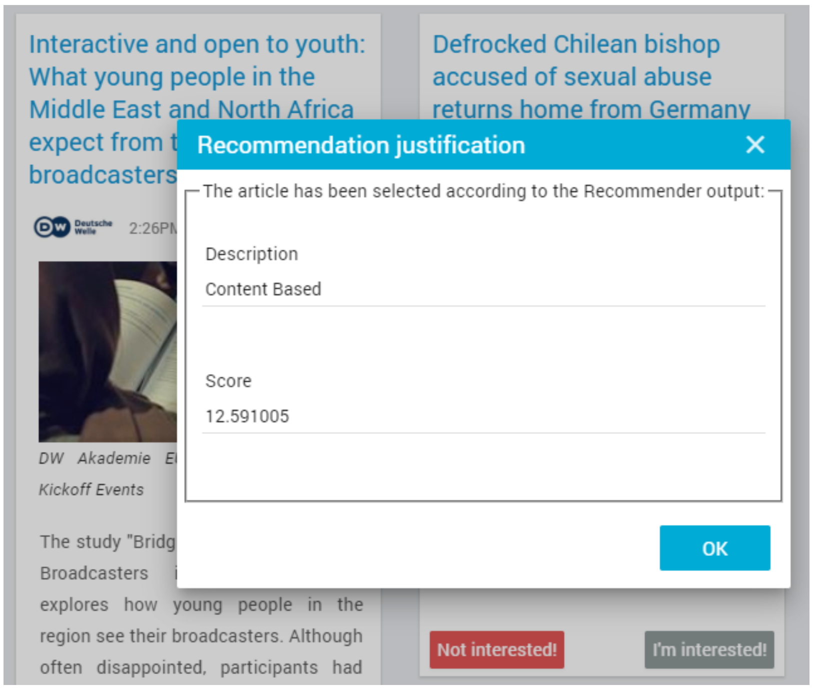
Permissions

Allow processing of user's location data.

Allow processing of preferences data.

Figure 10: Edit user profile settings

Justification of recommendation: Reader's app is able to show information (provided directly by the Recommender) explaining why a specific article has been proposed to the user. Currently a description and a score field value are being shown.


Interactive and open to youth: What young people in the Middle East and North Africa expect from t
broadcasters
2:26PM
DW Akademie E
Kickoff Events
Defrocked Chilean bishop accused of sexual abuse returns home from Germany
Recommendation justification
OK
Not interested!
I'm interested!

The article has been selected according to the Recommender output:

Description
Content Based

Score
12.591005

The study "Bridg
Broadcasters i
explores how young people in the region see their broadcasters. Although often disappointed, participants had

Figure 11: Justification fields

5. FOLLOW-UP OF RECOMMENDATIONS AND COMMENTS FROM PREVIOUS REVIEW(S)

Comments on D3.1 Initial design and APIs of technology Bricks

- It is not clear how the conceptual architecture of section 3 uses D2.1. Please explain.

The CPN Reference architecture document (D2.1) introduces the main actors to whom the platform is addressed: end users and media professionals. It justifies the selection of microservices architecture and describes patterns associated with it. At a logical layer, the platform services can be conceptually divided into three layers: content layer, mapping layer and user layer.

The conceptual architecture of section 3 of D3.1 document describes the content creation, readers and user profile workflow. These workflows are associated respectively with the content, mapping and user logical layers. The content layer is composed of the content procurement and knowledge extraction types of services. The content creation workflow covers both activities. Similar, the user layer offers specific services to deal with the user data itself, typically described by the user profile workflow. The mapping layer (maps content onto targeted users) is partially described by the user profile and reader's workflows.

- Please mention which technical requirements (from D1.4) are taken into account.

The conceptual architecture has been created taking into account the D2.1: Reference Architecture as well as the requirements derived from WP1 tasks. The technical requirements that were considered are mentioned on the 3.1.2 and 3.2.2 sections of the D1.4 - exceptions are the technical requirements related to the user requirements under question (table 4).

- Please mention which modules are you going to develop 1) from scratch, 2)customise/adapt, 3)just use. What is the TRL level of these modules? (Please include the table as it was presented during the review). There are some basic conceptual issues.

All the modules have been adapted to the CPN needs. The TRL level of each module is presented in the following table:

Semantic Lifting	TRL 2 → 5
Relation Extraction	TRL 4 → 5
Topic Extractor	TRL 7 → 8
Uplifting/Depressing Article Classifier	TRL 1 → 4
Frame Based Slot-Filling System	TRL 1 → 4
Recommender AB-Testing	TRL 6 → 7
User Modelling	TRL 4 → 5
Reader's App	TRL 4 → 5
Personal Data Receipts	TRL 4 → 5
Producer's App.	TRL 4 → 5
Distribution Framework	TRL 4 → 5
Twitter Analytics	TRL 4 → 5
Recommender	TRL 4 → 5

- Page 14: UR-AF 1.5 (avoid filter bubbles): "the system should allow users to choose favourite sources" But that is exactly what filter bubble is! Has the Consortium renounced its goal of avoiding filter bubbles (Note: of course users should be able to choose favourite sources but that cannot be put forward as the solution to filter bubbles).

We've gone different ways to address this requirement:

- This particular requirement was not followed-up on, since we don't have a multi-source app, but rather one to a few sources only. (With VRT and DW being public broadcasters that are per setup "un-biased").
- We've taken a closer look into the filter bubble definition and issue and we decided that we can best address it in CPN by making it clear to the user where their bias is in terms of news consumption. This is the developing feature with the topic-overview that's not yet implemented.
- In the recommendation mix there is a random suggestion part (controlled dithering) that offers you articles outside your spectrum, plus you might get recommendations from the newsroom as well, helping you to read beyond your own interest bubble

- Page 15: UR-UP2 (Making use of connections user already has through social media) – it is not clear this doesn't also make filter bubbles worse.

This requirement wasn't addressed further as we decided against it. We don't use user's connections from social media.

- Page 16: UR-AF2 (Avoiding FOMO: how to ensure people think they know everything there is to know). It is not clear how what the recommend will not increase FOMO.

By offering the users a view of the recommendation stream next to the most popular and the most recent tab, the user can always check whether he/she has not seen some news. In addition the system is offering the user control over how much time they want to spend reading the news and when - not limiting it to those specific times, but helping them to overcome the feeling, by regulating themselves. This is based on the assumption that an endless stream makes people think there is always more to read, without finding an end. By building up trust in the algorithm, we allow people to move more and more to the point where they can accept that the algorithm has shown them everything they need/want to know at that point in time and with the amount of time they have available at that point. A message "you're all caught up" at the end of the current list also helps to give people that feeling.

Comments on D3.2 Technology Bricks v1

- As in previous document, the idea of being able to choose favourite sources as a method to avoid filter bubbles is given (AF1.5).

Please see above -> UR-AF 1.5

- Page 17: User Modelling – English, Dutch, Greek – will other languages be incorporated during the project? How onerous will it be to add others?

A detailed table regarding the corporation of additional languages, is included in Section 5.1 of this deliverable.

Comments on D1.4 Technical Requirements

- Bursting the filter bubble – most of these requirements don't seem to come from the previous documents. How were they derived?
- "Highlight differences between perspectives" – laudable but difficult.
- Content formats – will there be automated versioning?
- Transparency about recommendations – laudable but difficult.
- Many of the user requirements from D1.2 are not dealt with.

The D1.4: Technical Requirements (platform and service requirements) provides the implementation aspects for the delivery of the CPN platform taking into account the full range of requirements for such service. The design of the CPN platform is driven by the usage scenarios and user requirements defined earlier in WP1 (D1.1 User requirements). However, a list of requirements has been identified and reported in D3.1 with the aim to be further analysed by the Consortium.

Regarding the “Bursting the filter bubbles” requirement (UR-AF1) the vision has been changed, as the CPN platform has been oriented as a single source provider. We address this requirement focussing mainly on the transparency of the recommendation algorithm and by giving to the users a clear information on what they read over a certain amount of time and by helping the user to make decisions and to understand if they are "reading into a bubble".

The content format requirements (UR-AF3) and Production Side requirements (UR-PS1.2 and UR-PS2.6) are not easily addressable with the available technology bricks, but we foresee to take into account them by exploiting the hackathon results - additional components will be integrated into the CPN platform. More details on this will be available at the end of the hackathon in June 2019 and reported in the D2.4 (v3).

- (AF 1.2, AF1.4) – This is understandable and expected but a list of requirements that have been put off or won’t be dealt with need to be provided.

The following table provides the list of requirements that will not be implemented, as decided by the Consortium. A short explanation leading to this decision is also provided.

Table 4: List of requirements under question

Requirement category	Requirement ID	Requirement description	Explanation
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	It was decided that the cold start problem will be resolved by quickly learning form the first user actions
	UR- UP1.7	The system should allow users to assign and change preferences (1-5) to categories themselves	This is incompatible with the requirements of automatic user profiling and the fact that the categories are automatically defined in a dynamic fashion
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	Stance detection was abandoned as its benefits were deemed questionable in content originating in mainstream sources
	UR-AF 1.4	The system should make it easy for the user to see a bias of a content item or source	Stance detection had been abandoned as its benefits were deemed questionable in content originating in mainstream sources
	UR- AF1.5	The system should allow users to choose favourite sources	This was deemed unnecessary as every user is attached to a single 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	Was implemented by tracking the percentage of the article that was displayed by the reader

Requirement category	Requirement ID	Requirement description	Explanation
	UR-AF 3.3	The system should allow users to choose whether they prefer an overview or all content at once	It was deemed unnecessary as a short summary is usually contained in the first paragraph of a news item
	UR-AF 3.6	The system should be able to put global news in a local relevance context for users	It was deemed impossible to satisfy as there was no relevant brick or technology expertise on the consortium to tackle this adequately
	UR- AF3.8	The system should allow users to filter content by language	This was deemed unnecessary as content in CPN is in particular languages for particular users
	UR-AF 3.9	The system should allow users to filter content by complexity within a language	It was deemed impossible to satisfy as there was no relevant brick or technology expertise on the consortium to tackle this adequately
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	This was dealt with by monitoring what percentage of the article was consumed
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	A section in the dashboard monitors performance of articles but it was found impossible to link this to specific attributes
UR-AF4: Sources: Where does the necessary content come from?	UR- AF4.2	The system should allow for additional content sources, outside the consortium	In the decided implementation the platform allows a single source for every user, so this is not possible

5.1 LANGUAGE DEPENDENCY IN CPN BRICKS

CPN Bricks	Partner	Is it language dependent	If yes, which languages will it cover by the end of the project	If yes, how much effort approximately is needed to add a new language	Comments/Notes
Personal Data Receipts	DCAT	Yes	English, Greek, French, Dutch and German can be covered, by the end of the project	-	-
Distribution Framework	DCAT	Yes	English will be used by the end of the project	An additional language can be added in a period of 2 weeks per language	-
Producer's app	ENG	No	-	-	The only thing that could require a translation could be the web interface that will be prepared for pilot 2 - this will be later discussed and agreed among the Consortium.
Relation extraction component	IMEC	Yes	English	2 person months, needed for - Building a large corpus of annotated sentences with the target relations. Around 30 full days of annotations by native speakers - Training the model and potential feature annotations (using the 'semantic label propagation paradigm'), which we believe would require a similar workload as for the annotations (1 PM)	

CPN Bricks	Partner	Is it language dependent	If yes, which languages will it cover by the end of the project	If yes, how much effort approximately is needed to add a new language	Comments/Notes
Uplifting vs depressing annotations	IMEC	Yes	English	It is too early to provide concrete effort estimates for other languages.	-
Frame-based text enrichment	IMEC	Yes	English	It is too early to provide concrete effort estimates for other languages.	-
Topic Extractor	LIVETECH	Yes	English, German, Dutch and Greek	NLP world is highly dishomogeneous as it depends strongly on available linguistic resources and language characteristics. Among the characteristics that must be analysed in order to estimate the effort needed to include a new language we can cite: ▪ Morpho-syntactic complexity: some languages are naturally richer in inflections than others (e.g. different word endings for linguistic cases vs plural/singular). This can make the topics clustering and terminological candidates' selection simpler and more effective in some languages and more difficult on others. ▪ different alphabets: some content needs to be transliterated to be processed by some tools (e.g. translating cyrillic to latin alphabet) ▪ Word segmentation: in some languages the boundaries between words are explicit, typically marked by a blank, while in others segmentation is more problematic. Languages which do not have a trivial word segmentation process include Chinese, Japanese, where sentences but not words are delimited, Thai and	Other languages that could be easily integrated are: French, Spanish, Romanian, Croatian, and Serbian. Depending also on some business opportunities we will evaluate the possibility of including some of these languages to the capabilities of the module.

CPN Bricks	Partner	Is it language dependent	If yes, which languages will it cover by the end of the project	If yes, how much effort approximately is needed to add a new language	Comments/Notes
				Lao, where phrases and sentences but not words are delimited, and Vietnamese, where syllables but not words are delimited. As a result, the effort of integrating new languages in the module can be roughly estimated as follows: ▪ A minimum of 2 weeks of effort for languages in the same family (or somehow close) to the ones already supported by the platform ▪ A maximum of 3/4 months effort for integrating more exotic languages. This process could, in some extreme cases, include buying, when not freely available, linguistic resources such as electronic dictionaries, thesauri, linguistic corpora, etc. On average, the maximum expected of the resources to be bought for treating a very specific language, could be estimate in around 4000 euros.	
Recommender AB-Testing	LIVETECH	No	-	-	-
Twitter Analytics	ATC	Yes	English	Approximately 2 – 3 person weeks per language and the setup of named entity recognition software for each language.	-
Reader's App	ATC	No	-	-	Only the UI is language dependent

6. CONCLUSIONS

This Deliverable reports on the implementation of the technological infrastructure of the 2nd prototype of the CPN platform. 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.

7. REFERENCES

[1] CPN: D1.1 User Requirements Model

[2] CPN: D2.1 CPN Reference Architecture

[3] CPN: D3.1 Initial Design & APIs of Technology Bricks

[4] Kleinbaum, David G., et al. Logistic regression. New York: Springer-Verlag, 2002.

[5] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[6] Robertson, Stephen. "Understanding inverse document frequency: on theoretical arguments for IDF." Journal of documentation 60.5 (2004): 503-520.