Co-inform

Context Matters,
Your Sources Too

Generic Co-Inform architecture – Version 4

D4.4

#ThinkCheckShare

Document Summary Information

Project Title:	Co-Inform: Co-Creating Misinformation-Resilient Societies		
Project Acronym:	Co-Inform	**Proposal Number:**	770302
Type of Action:	RIA (Research and Innovation Action)		
Start Date:	01/04/2018	**Duration:**	36 months
Project URL:	http://coinform.eu/		
Deliverable:	D4.4: Generic Co-Inform architecture – Version 4		
Version:	3.0		
Work Package:	WP4		
Submission date:	31/01/2020		
Nature:	P	**Dissemination Level:**	P
Lead Beneficiary:	Scytl Secure Electronic Voting, SA (SCYTL)		
Author(s):	David Salvador, Technical Manager (SCYTL)		
Contributions from:	Ronald Denaux (ESI) Adrià Rodríguez-Pérez, Project Manager (SCYTL) Andreas Berg, Developer (SU) Daniel Forsberg, Developer (SU) Gregoire Burel, Researcher (OU) Ipek Baris, Researcher (UKOB) Akram Sadat Hosseini, Researcher (UKOB)		

The Co-inform project is co-funded by Horizon 2020 – the Framework Programme for Research and Innovation (2014-2020) H2020-SC6-CO-CREATION-2016-2017 (CO-CREATION FOR GROWTH AND INCLUSION).

Revision History

Version	Date	Change editor	Description
0.1	07/01/2020	SCYTL	Initial version
1.0	20/01/2020	SCYTL	Merged contributions from all partners
1.1	24/01/2020	SCYTL	Different updates during hackathon
2.0	27/01/2020	SCYTL	First version ready for review
2.1	28/01/2020	OU	First review
3.0	29/01/2020	SCYTL	Final version

Disclaimer

Copyright Message

Executive Summary

This report provides a fourth and final version of the architecture of the Co-Inform platform. It focuses on interoperability at business and information level, to be developed in an agile manner.

The system architecture of the Co-Inform project is component-based and layered. Robustness, scalability and multi-user access support are important system characteristics. The architecture is designed based on general requirements from WP1 and its output is a specification of the components of the system (coming from the research project partners in WP1 and WP3), of how they interact (e.g., flows of control and data), and specification of programmable interfaces so that different partners can build their components independently. In addition to the components from the technical partners, the architecture provides infrastructural components required in the system's design, some examples being a data store, security (access control), and scheduling of automated tasks.

This deliverable upgrades the details provided for the platform architecture in D4.3. It feeds from the latest technological developments from partners involved in WP2, WP3 and WP4.

Table of Contents

1. Introduction 7
2. Platform architecture design 8
2.1. High-level diagram (data flow) 8
2.2. Back end components' description 9
i. Data collector (DC) 9
ii. Misinformation Detection (MD) 14
iii. Misinformation Flow Analysis and Prediction (MFAP) 15
iv. Plugin Gateway and Tweet Store 17
v. Rule Engine 17
vi. Perception Flow and Behaviour Mining (PBM) 19
2.3. Front end components' description 20
vii. Browser Plugin 20
viii. Dashboard 21
2.4. Hosting Architecture 21
3. API documentation 23
3.1 Data collector (DC) 23
3.2 Misinformation Detection (MD) 23
3.3 Plugin Gateway 24
3.4 Tweet Store 24
4. Conclusions 25

Table of Figures

Figure 1 High Level Architecture 8
Figure 2 Crawling and Text Analysis 11
Figure 3 Data Collector 13
Figure 4 Misinformation Flow Analysis and Prediction 16
Figure 5 API Gateway and Tweet Store 17
Figure 6 Example of rule for credible tweets 18
Figure 7 Example of rule triggering an action 19
Figure 8 Perception and Behaviour Mining 20
Figure 6 Browser Plugin 21

1. Introduction

Misinformation online generates misperceptions. The speed and ease in which false news spread on social media have a massive impact on current affairs and policies. By bringing together a multidisciplinary team of researchers and experts in computer science, behavioural science, and sociology, Co-Inform aims at engaging all stakeholders in fighting misinformation by providing them with the tools to identify 'fake news' online, understand how they spread, and provide them with verified information.

To this end, Co-Inform will integrate its ICT tools and services (WP3) and policy encodings (WP2) to deliver a co-created misinformation resilience platform in the form of:

- A browser plugin to raise citizens' awareness of fully or partially misinforming content, of related fact-checking articles and corrective information, of average citizens' perceptions towards this content, and of key pro and against comments from fellow citizens.
- A dashboard for fact-checking journalists and policymakers, showing that misinformation was detected, where it originated from, how and where it has spread and will spread in the near future, what's the current and predicted public perception, and what are the key comments about it from the public. The dashboard will also show the news articles or information that users requested to be fact-checked.

This deliverable describes the high-level architecture of the Co-Inform platform, including the interaction between its expected components.

While work within WP2, WP3 and WP4 has not yet ended, this is the final version of the architecture. It build on the contents of D4.3 and feeds from the developments in the framework of WP2, WP3 and WP4. Since most of the development work has not yet ended, this architecture aims to be flexible enough to accommodate any further requirements elicited within WP1 throughout the project, while providing a useful interface to the data through their API and user interfaces.

While the specific details will evolve with the project's needs, this general architecture will serve as a good foundation to build around.

2. Platform architecture design

2.1. High-level diagram (data flow)

The following diagram illustrates the high-level architecture proposed for the Co-Inform platform:

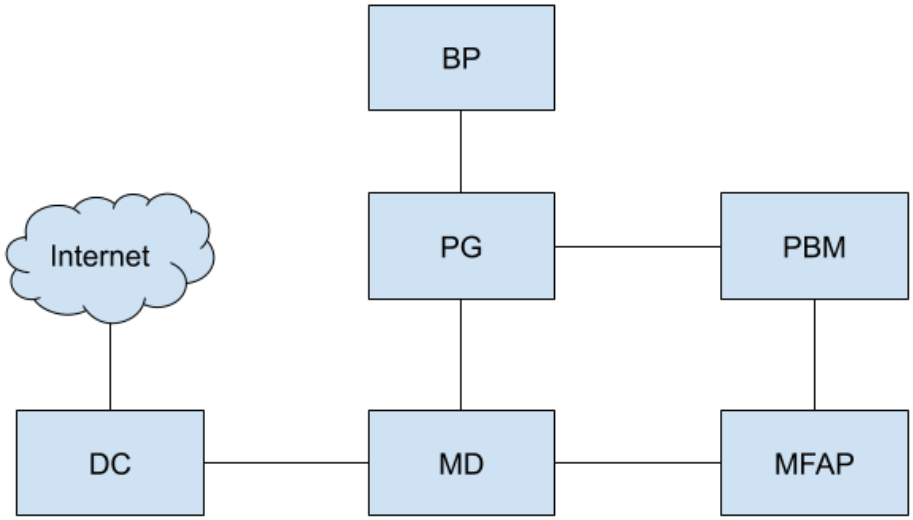

Figure 1 High Level Architecture

BP: Browser Plugin
PG: Plugin Gateway
MD: Misinformation Detection
DC: Data Collector
MFAP: Misinformation Flow Analysis and Prediction
PBM: Perception Behaviour Mining

The following components will be needed to implement the Co-Inform architecture, based on the technical proposal submitted to the European Commission and the latest developments within WP2, WP3 and WP4.

While this is the last version of the architecture document, at this stage the co-creation for the Co-Inform tools within WP1 is not final and further workshops are scheduled after this deliverable is submitted. It means that the final implementation of these components may be subject to changes. Depending on the specific requirements gathered within the activities carried out in WP1, some of these components may be integrated into a single component or some of them could be split in other modules.

For this reason, this architecture aims to be flexible enough to accommodate any further requirements elicited within WP1 throughout the project, while providing a useful interface to the data through their API and user interfaces for partners working within WP2, WP3 and WP4 (i.e., any new modules will follow the general Gateway/OpenAPI of the architecture).

2.2. Back end components' description

i. Data collector (DC)

The **Data Collector** (DC) is the module in charge of collecting the information that is processed using ESI's industry-leading text analytics technology. Broadly speaking, this module accepts as an input a list of **sources** and **configurations** that it uses to start a text analysis process. After a short time, the found documents, along with the *analysis results,* are placed in a **database**; the contents of which are searchable and accessible to other modules via an **API**.

Next, we explain the various terms, data structures and subcomponents in more detail.

The **Data Collector Front End** (DC FE) provides a way to specify and manage (i.e. add, remove, edit) sources and configurations to the DC. Although the types of supported sources have to be formally defined as part of pilot requirements in WP1, we provide support for the following types of sources: websites to crawl, RSS feeds, social media accounts/pages, search keywords and off-line documents (e.g., PDFs, Word documents, etc.). Besides specifying the type of the source, the administrator has to specify additional configuration parameters such as:

- Locator: this will typically be a URL to describe the site (e.g. https://snopes.com), rss feed (e.g. https://www.snopes.com/feed/), search engine (e.g. google, bing + keywords), location of the off-line documents. Additionally, we also support structured data sources to ingest ClaimReviews[1] published by third parties like ClaimsKG[2] and datacommons[3].
- Collection: Since we have multiple pilots in Co-Inform, we keep documents from each pilot in a separate 'section' of the database. This means that sources need to be bound to a specific collection/pilot. We assume that most sources will be specific to each pilot, although in some rare cases it may be possible that the same source will be used in multiple collections. The system may process these sources multiple times, although this could be avoided as part of the implementation of the data collector. Another reason why specifying a collection is useful is because we may want to perform special types of analyses on documents for a particular use case in the future (e.g., if we know that the use-case is migration, we may want to use a custom document classifier on these documents. Likewise, if we know that the use case is about a specific area in

[1] Available at: **http://schema.org/ClaimReview**
[2] Available at: **https://data.gesis.org/claimskg/**
[3] Available at: **https://datacommons.org/factcheck**

Sweden, we may want to provide a text analysis service that has a more fine-grained knowledge of places, people and political issues in Sweden, rather than using a generic text analysis service).

- Frequency: how often should the source be re-crawled. Some types of sources are dynamic (i.e., they change every few hours, such as news feeds), while others are static (e.g. news articles).
- Crawl depth: in the case of websites to crawl, but also social media accounts and search keywords, it can be useful to not only collect the initial website and social media posts, but also to collect further pages/posts referenced by them.

The configuration options presented above are indicative. We specify the list of configuration parameters based on requirements, implementation and usage. At the moment, we have implemented the DC FE as a graphical interface (GUI) that is only accessible to administrators from ESI. In the future we may consider providing an API, since it can be more flexible. However, it is still unclear who will be the stakeholders who should have access to this API. In general, it makes sense to restrict access to this interface as it can have an impact on the number of documents that will be in the system, thus affecting possible system performance. An overly broad selection of sources can result in overly large collections of documents, which will make the overall system harder to use as there may be too many documents to explore and analyse.

The **Data Collector** submodule uses the sources and configurations specified using the DC FE to crawl the web, find documents and analyse them. As described above, some sources may be re-crawled at defined intervals. Figure 2 describes the working of this submodule in more detail and shows that it has two main functionalities: crawling and datamining, .

- The crawler (Data Processing Engine in Fig. 2) processes the input document (e.g. an html page or PDF) and extracts its textual content and metadata (e.g. published date, author). A recent development that is relevant to Co-Inform is that many fact-checking articles are now including metadata (**https://schema.org/ClaimReview**)[4] that makes it easier for machines to gather the main claim being evaluated and provide a basic result for the fact-checking exercise. If this kind of sites will be included in the pilots, it may be necessary for the Data Collector component to include this kind of metadata as part of the extracted document (and eventually include this information as part of the database).
- The datamining step enriches the extracted document by automatically adding annotations such as categories (according to various generic taxonomies such as Intelligence, Crime and Terrorism), entities (places, people, organisations, domain specific), sentiment/emotion and relations (tuples or triples between entities and/or categories). Depending on the languages supported by the text analysers, the original document text may need to be translated. In particular, it may be necessary to translate all non-English documents into English (using third party machine translation services), to make it easier for researchers to evaluate and inspect the gathered documents. Another important step as part of datamining relates to the **extraction of relevant sentences and claims from**

[4] Fact-checking sites do this because it helps search engines such as Google or Bing to index these sites and highlight them as part of search results.

documents. This step makes it possible to assign credibility and accuracy ratings at the sentence level instead of at the document level.

The **database** provides an API for accepting and updating analysed documents and claims and stores them in representations optimised for storage and retrieval. It also provides a low-level API for retrieving analysed documents. Since this is an internal API, we do not specify it in further detail in this document and leave it as an implementation issue. The main constraints are that it must be able to store textual data (the content of the documents), metadata and extracted fields. Ideally, the internal API will allow for exploratory search of documents based on facets, since this allows exploring the dataset in an intuitive way (e.g. each search response lists the number of results which have a specific category or entity).

Finally, the database must be deployed in such a way that it safeguards in both a physical and legal manner any personal data. Therefore, the database will be compliant with requirements of information security as well as associated legal requirements laid down in GDPR and the project's Data Management Plan (D 7.1). As the project develops, data controllers and/or data processors (all project partners) will ensure to take appropriate measures in a case-by-case manner when faced with direct or indirect personal data following the provisions of GDPR article 4 et al. Further distinction will be made whenever data can be anonymised and therefore falling outside the scope of GDPR. Naturally, this needs to be continuously analysed in the consortium, since we need to balance:

- The fact that most sources are publicly available: i.e. does it make sense to anonymise mentions of Donald Trump in news articles? Similarly, mentions of users in Facebook or Twitter are already searchable.
- Informational requirements by downstream components. For example, the Misinformation Detection component may only work (or work much better) if the identity of authors of documents are known; hence in this case, anonymising this information will be counterproductive to the goals of the project.

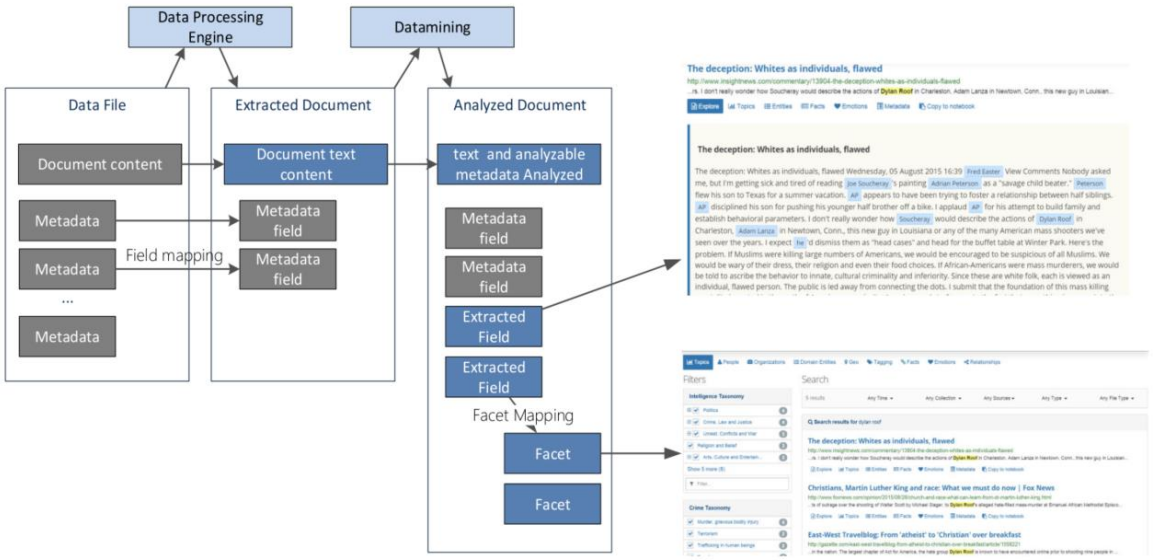

Figure 2 Crawling and Text Analysis

The **content collector API** provides an interface available to other modules in the system for exploring and retrieving analysed documents. The API is available as a web service (e.g. using REST), over a secure connection (e.g. requiring authentication and using secure HTTP). The API provides two main services:

- **(document) search**: can be used to search and explore documents in a particular collection. This service has a long list of input parameters, described in D3.1, but they can be conceptually conceived as four types of input parameters:

  - Collection: a unique name or identifier
  - Keywords: a list of keywords that need to be matched against the text in the document. If deemed necessary, we may provide a simple query language similar to that used by well-known search engines in order to allow users to make some keywords mandatory, to allow for conjunction or disjunctions of keywords, etc.
  - Filters: a list of filters to narrow the set of results. These filters will typically be based on facets returned by previous search results. Filters will include categories, entities, dates (published, acquired), authors, sources, etc.
  - Pagination: since search results may be in the order of thousands of documents, users may need to request the results in "pages", where each page has a certain number of results.

  The output of this service is also described in detail in D3.1, but they include fields like:

  - url: used to identify the document
  - metadata: a selection of the most relevant metadata (e.g., title, author, publisher, source, publishing date, published metadata).
  - Search result metadata: total number of results, pagination information, available facets.

- **Collection**: returns a list of collections (or sections of the database) that are available to search.
- **Claim/search**: can be used to find claims extracted from collected documents or structured datasets of ClaimReviews.
- **Url/Collect**: allows on-the-fly semantic analysis of documents on-line. Often, a user will want to assess the credibility of e.g. a tweet that links to an on-line document. If that document has not been previously crawled by the content collector, modules are not able to assess their credibility. To address this issue, this service receives a new URL, retrieves and analyses the document. The document can optionally be included in one of the collections.

To make integration with other components as easy as possible, we provide a specification of this API in a format that allows for easy client generation and testing, like OpenAPI (**https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0. md**) and

Swagger (https://swagger.io/solutions/api-design/). The specification can be found at https://co-inform.github.io/claim-cred-api/

In terms of **performance**, the requirements still need to be specified as part of WP1. However, in technical terms, we assume that the Data Collector should be able to process documents in both batch mode and monitoring mode. The batch mode is useful at the beginning of a pilot, as many sources need to be ingested (depending on the quantity of sources and documents this may take minutes or hours). Once the initial set of sources has been processed, the monitoring mode keeps track of new documents at regular intervals. Analysis of these results should become available after a few minutes. Searching services should provide search results in mere seconds and quicker than that if they are to be used for interactive user-facing applications (which should be possible by using appropriate paging and search query parameters).

In terms of volume of documents, we expect that the conjunction of all pilots will not require storage of more than about ten million documents. If more documents are expected or required by downstream components or specific pilots, this will need to be specified in advance, since this may require special deployment plans to ensure good performance for such a volume of documents.

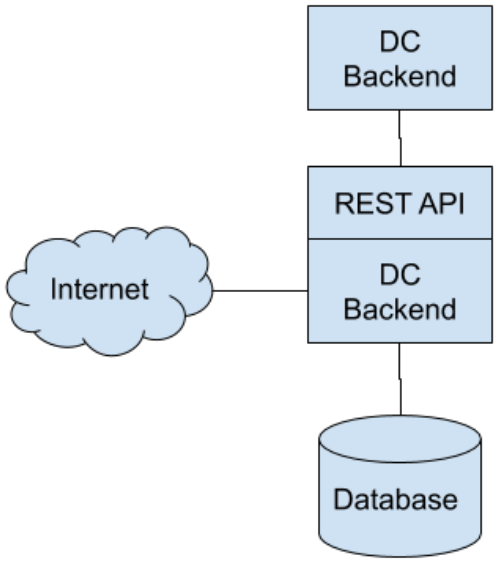

Figure 3 Data Collector

For more information about the implementation of this component within Co-inform, we refer to D3.1.

ii. Misinformation Detection (MD)

The **Misinformation Detection** (MD) module is developed as part of T3.2 in WP3 and aims at detecting misinformation using different methods. This will help managing misinforming content in large amounts of documents on social media or other information sources. As part of the work in T3.2, different models and services will be considered, such as an automated misinformation detection tool, and a dubious URL identifier service.

In general, the models developed by WP3 require **annotated data** and rely on **Machine Learning** (ML) models. Such data is obtained from existing third-party data sources and from datasets shared with the Co-Inform consortium such as the data collected through the **Data Collector** (DC) module (section 2.2.i above).

The input data is either given as URLs that need to be analysed or as textual content as well as annotation labels if available that are used for training the MD services (e.g. *information/misinformation* labels). This data is collected from different information sources such as social media websites (e.g. Twitter, Reddit) or news websites. In general, this information is useful both for detecting the claims on social platforms (new ones and sharing known ones) and for analysing their spread and evolution.

Besides a particular content or claim, the following external information is used as input:

- *Historical data about claims*. Fact-checking datasets are used both for spotting and for predictive models. This type of data needs to contain information about particular claims (e.g. *who*, *what*, *where* and *when* a claim has been done). Also, some information from the fact-checking process of known fact-checker sources: e.g., a label indicating the type of claim (e.g., factual, fake, satire, ...), the author of the reviewing process;
- *Information network and/or social network*. Another important type of input that is used is the network associated with the author of a claim or the claim information network (i.e., the interconnection between different claims). This data will be particularly relevant for bot discovery and the **Misinformation Flow Analysis and Prediction** (MFAP) module.

In general, the models return probabilities of a given information (or URL or account) to be related to misinformative or informative content (or another relevant label).

The trained MD services follow standard API development practices by providing **RESTful web service** where the claim or other information that needs to be check is submitted to a service and a **JSON object** containing probabilities and labels is returned.

Currently, three different services or components are being developed as part of the MD module: 1) ***MisinfoMe***, a service for identifying misinformation based on information sources and historical annotated data; 2) ***Claim Credibility***, a claim detection service that extracts important text snippets from documents; and 3) ***Stance Detection***, a service that assigns a veracity score to a tweet by analysing its textual features and the stances towards it.

- **MisinfoMe**: the first service takes inputs from the DC component and other data sources and provides an API that can be used to see the relation of **sources** and **twitter accounts** with misinformation. For a certain source, it is possible to see how experts (fact-checkers and other evaluations) reviewed or debunked it.

And for a certain twitter account it is possible to see the interaction it had with misinformation, fact-checking, verified news and debunked ones. These API provide the results of analysis using REST.

- **Claim Credibility:** as part of the MD module, the claim detection service is designed for extracting sentence claims from textual content so they can be verified against an historical claim database or additional components of the MD module. The model is developed using ML and is trained on multiple datasets from various domains, so it is possible to extract claims in multiple situations. Although the claim API is still evolving, it follows the Open API v3 specification (**http://spec.openapis.org/oas/v3.0.2**) and provides an interactive documentation using Swagger (**https://swagger.io**). The API accepts both textual input or URLs and returns a list of sentence claims. Another service that has been implemented as part of the MD is the claim credibility submodule, that receives a sentence and predicts its credibility based on semantic similarity with sentences in crawled documents

- **Stance Detection:** finally, a third service is designed for content analysis, which assigns the credibility of tweet by analysing the source tweet and its replies (max 8 replies due to the restriction). The service has been developed using a two-phase pipeline based on ML models, and the pipeline is trained with the RumourEval2019 dataset[5]. The pipeline assumes that the source tweet contains rumour. It first detects the stances of the tweet as supporting, denying, questioning or commenting towards the rumour. Then, a second ML-based model evaluates the stances along with textual features and predicts the probability of the veracity for true, false and unverified cases. Finally, the credibility of the tweet is assigned by aggregating the for each veracity values.

The MD module is also being extended so that parameters can be used for affecting its computation such as which Fact-checker or what content to trust. The new API is built on top of a computational graph and will return additional information besides the information credibility score such as the confidence in a result. The parametrisation aims to be used by the rule engine.

The MD module and APIs will be connected and work together with the **Misinformation Flow Analysis and Prediction** (MFAP) module by providing the primary information necessary to analyse the flow of misinformation in social networks.

iii. Misinformation Flow Analysis and Prediction (MFAP)

This **Misinformation Flow Analysis and Prediction** (MFAP) module will work together with the **Misinformation Detection** (MD) module using data similar to the one used by the MD module. The aim of this module is to determine patterns of different types of misinformation across the social networking platforms and to provide different metrics and measures associated with particular information flows in social networks in order to better understand how misinformation spread and evolves.

[5] Available at: **https://www.aclweb.org/anthology/S19-2147/**

Similar to the previous module, the different services provided for the MFPA module will require historical data from different data sources including data collected by the **Data Collector** (DC) module. This module will require different types of input such as:

- *News articles* and *historical data.* Data collected over time such as the information collected by the DC module can be used for making misinformation predictions.
- *Information network and/or social network data.* This data can be used to understand the *topology* and *typology* of the network that connects the different actors (publishers, accounts influenced by the content) identifiable in the historical data.
- *Misinformation analysis tools*. The MD module can be used for accessing important information about information nodes and actors and can be used to label some specific nodes in the social graph.

With this information available, the MFAP module should be able to track over time the diffusion of the claim instances, their evolution, reception, and amplification by different actors. The flow analysis will apply ML models that can be used to capture temporal features (such as the impact of a certain claim over time) of the flow of misinformation.

This module will have several outputs, that will be made available to other modules. The results provided will be of different nature, such as the ability to predict the effect of particular type on misinformation on particular communities and demographics.

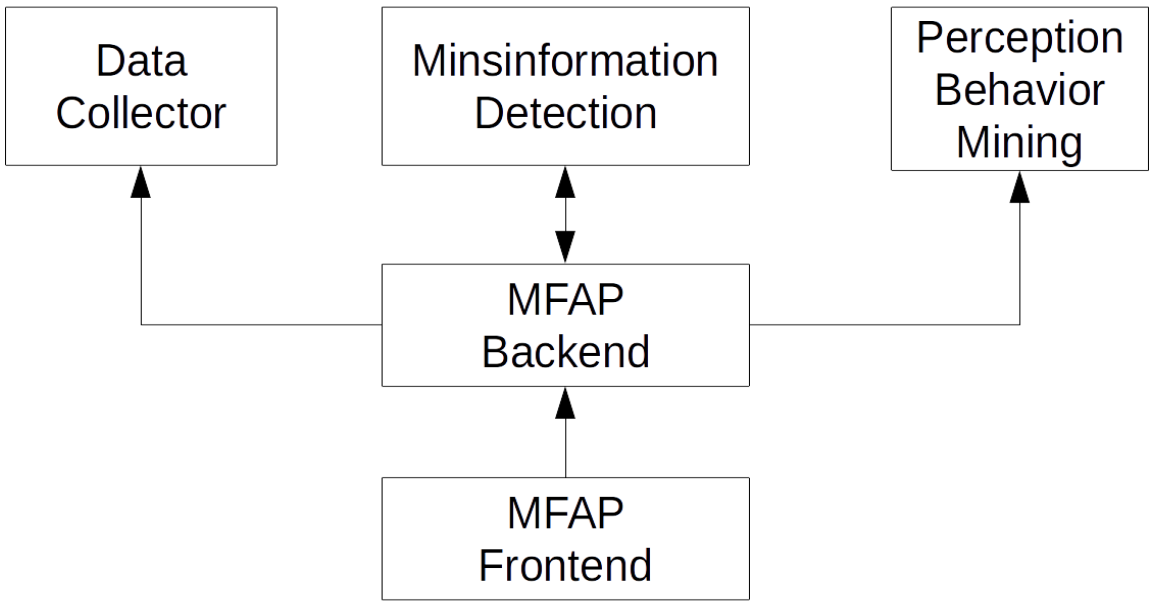

Figure 4 Misinformation Flow Analysis and Prediction

iv. Plugin Gateway and Tweet Store

Following modern web application best practices, this module acts as a single-entry point for all the requests generated by the browser to the Co-Inform platform. Once a request is received, the **Plugin Gateway** (PG) extracts certain data that will be sent to the **Misinformation Detection** (MD) module for later processing. At the same time, anonymised statistical data is sent to the **Perception Flow and Behaviour Mining** (PBM) module.

Given that the MD module can take up some considerable time solving a request, a unique identification number will be returned to the **Browser Plugin** (BP) on every request made. It is the **BP's** responsibility to query the **PG** for a resolution on the initiated request by providing the unique identification number. With this behaviour a more flexible, fault-tolerant, and scalable system is achieved.

The PG is complemented with a database acting as a **MD** cache to improve requests resolution time, so the same request is not sent twice to the **MD** module and to reduce traffic in the system. To further reduce time spent collecting information from the internet due to restrictions in public and free APIs, a Twitter cache named **Tweet Store** has been implemented. This cache saves all the Tweets that the **MD** module collects for its processing, allowing other modules to query that cache and save a call to Twitter's rate-limited API.

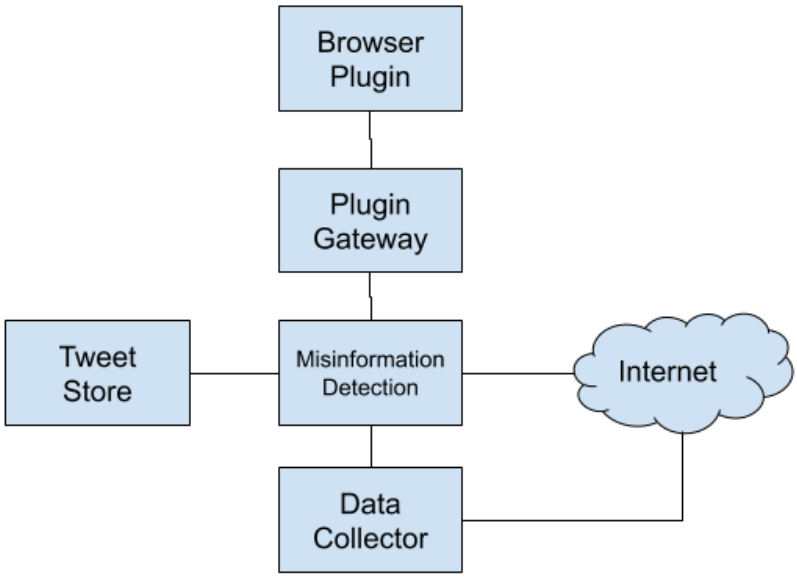

Figure 5 API Gateway and Tweet Store

v. Rule Engine

The **Rule Engine** (RE) manages the responses from the **Misinformation Detection** (MD) components and provides an appropriate response or action to the **Browser Plugin** (BP) in

real time. It will be triggered by the **Plugin Gateway** (PG) whenever a request is received and evaluate a set of previously defined in the system.

The **RE** extends the JEasy[6] rule engine, and it is integrated into the **PG** as library. JEasy supports the user defined rules in *json* format. The rules are defined as Figure 5 where "*name"* is the identifier of the rule, "*description"* describes of the rule, "*priority"* is the order of the rule execution, "*condition"* defines the parameters that activate the proper action or response of the rule, and "*actions*" define the list of actions based on the "*condition"*, which activates the rule execution.

The details of the Co-inform plugin rules are described in D2.2. Currently, the rules[7] that assign a credibility scores to tweets, such as the rule in Figure 6, and the rules that trigger the of social translucence actions such rule in Figure 7 are integrated into the platform.

```
{
            "name": "credible",
            "description": "checks the post and flags as credible",
            "priority": 2,
            "condition": "claimcredibility_tweet_claim_credibility_0_credibility > 0.5
        && claimcredibility_tweet_claim_credibility_0_confidence > 0.7",
            "actions": [
              "callback.getModuleCredibility().put(\"claim_similarity\",
        Credibility.not_credible_post);"
            ]
          }
```

Figure 6 Example of rule for credible tweets

[6] Available at: **https://jeasyrules.github.io/jeasyrules-core/**

[7] Rules for credibility can be found at: **https://github.com/co-inform/policy_manager/tree/master/src/main/resources/rules/deployment/credibility_mapping**

Rules for interventions can be found at: **https://github.com/co-inform/policy_manager/blob/master/src/main/resources/rules/deployment/interventions/**

```
{

            "name": "credible",

            "description": "checks the post and flags as credible",

            "priority": 2,

            "condition": "claimcredibility_tweet_claim_credibility_0_credibility > 0.5
        && claimcredibility_tweet_claim_credibility_0_confidence > 0.7",

            "actions": [

              "callback.getModuleCredibility().put(\"claim_similarity\",
        Credibility.not_credible_post);"

            ]

          }
```

Figure 7 Example of rule triggering an action

For more information about the implementation of this component within Co-inform, we refer to D2.1 and D2.2.

vi. Perception Flow and Behaviour Mining (PBM)

Using the analysis and classification conducted by the **Misinformation Detection** (MD) module and the **Misinformation Flow Analysis and Prediction** (MFAP) module and using the statics and reactions collected by the **Browser Plugin** (BP) made available by the **Plugin Gateway** (PG), this module will provide statistics on misinformation behaviours.

This module will provide dashboards and statistics about the misinformation managed within the whole system. The specific parameters of each end-user will be inputted into the PBM module through a **web front end**.

User characteristics that could influence their engagement with misinformation *might* include age, culture, prior opinions, interests, exposure, etc. The **PBM** module will attempt to collect such information (e.g. from user's profiles, timeline analysis) to support the prediction analysis of misinformation flow. Spreading and acceptance or rejection of misinformation can be analysed (e.g. using **opinion mining**) to gauge the user's behaviour towards a particular piece of misinformation, and how this behaviour changes (or does not change) after an intervention (which will be provided by WP5) is executed. To this end, services for extracting user characteristics and opinion will be required, which would take input data such as: user profile, user posts (timeline), and off-the-shelf opinion mining methods.

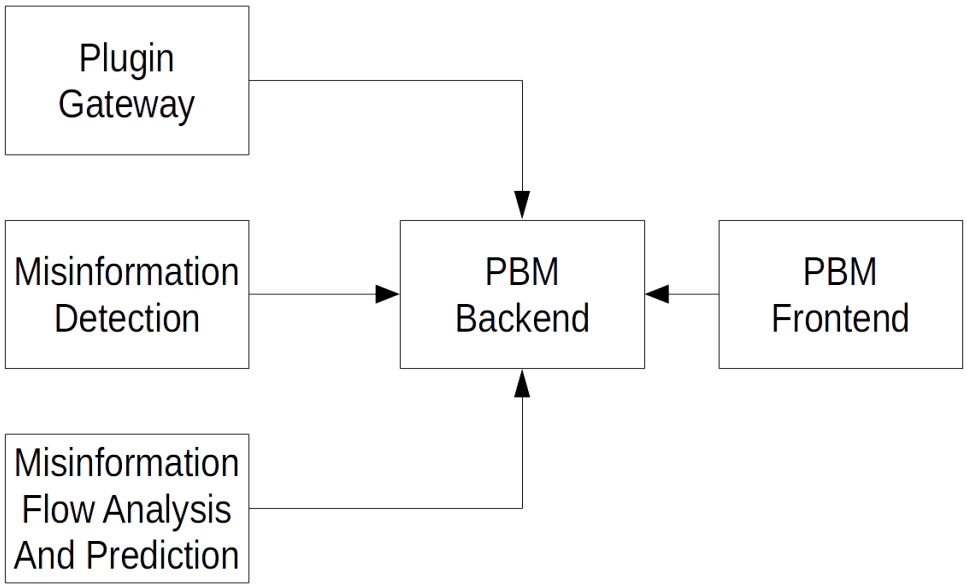

Figure 8 Perception and Behaviour Mining

2.3. Front end components' description

vii. Browser Plugin

The **Browser Plugin** will be the main end user module (i.e. for social media users). It will analyse the end user navigation data and together with both the **Misinformation Flow Analysis and Prediction** (MFAP) module and the **Misinformation Detection** (MD) module it will inform the user about the credibility of the information being accessed. The communication between the plugin and both the **MD** and **MFAP** is done through the **Plugin Gateway** (PG) module that acts as an API gateway[8].

The plugin's architecture is based on a *centralised architecture*. For every relevant website that the user loads, the plugin will send data to the **PG**. This data will be the content of the website (mainly the text) and some metadata (website's URL, timestamp). Before the data is received by the **MD** or the **MFAP**, some anonymisation methods, such as IP removal, assignation of a transaction ID, etc. will be applied. Then, the data will be processed in order to diagnose whether it corresponds to misinformation or not. After the diagnostic, a response will be sent to the plugin which will show the user a **confidence level** on the website or post that is currently visiting.

[8] **https://microservices.io/patterns/apigateway.html**

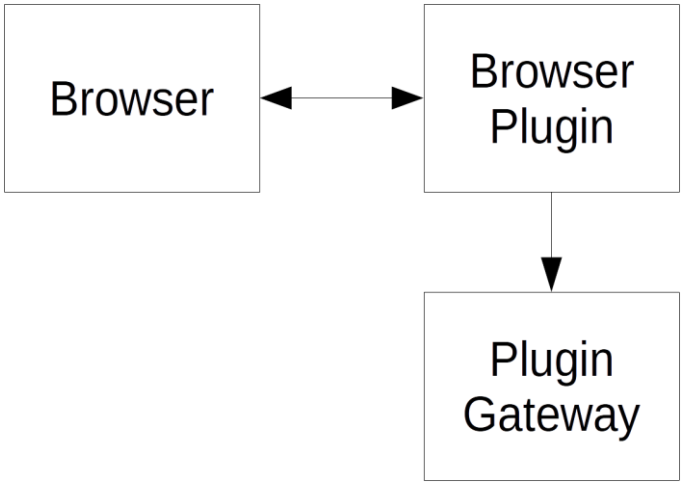

Figure 9 Browser Plugin

viii. Dashboard

The **Dashboard** will be the second end user interface, and it is being designed having the needs of policymakers and journalists in mind. The dashboard will show what misinformation has been detected, where it originated from, how and where it has spread and will spread in the near future, what's the current and predicted public perception, and what are the key comments about it from the public.

Based on the activities conducted within WP1, the following features have been considered for their implementation as part of the dashboard:

- Show more information on why a claim has been flagged, who flagged it and when.
- Provide aggregate data on misinformative content.
- Provide access to statistics for validation and information to assess validity of sources.
- Allow policymakers and journalists to give an accuracy rating.
- Show how information has spread online (information trail) over time, and at specified locations.

Furthermore, the dashboard will provide central access to other fact-checking and validation tools developed by third parties. To the extent possible, mechanisms for collaboration between and within professional groups will be also studied for their integration into it.

The dashboard will be developed as a web application, with an extendable architecture, and will be made **open source** and **freely available**.

2.4. Hosting Architecture

Following, we provide an overview of each module's hosting architecture, as well as for each of their components:

For the **Misinformation Detection** (MD) module and its components:

- MisinfoMe is hosted on Docker containers running FastAPI.
- Claim Credibility is hosted using docker containers and is an ML model developed in PyTorch
- Stance Detection is hosted on Ubuntu servers running as docker container.

The **Plugin Gateway** (PG) is hosted on a Linux server running nginx. It uses a redis cache for storage of querys and responses for quick response times. The **TweetStore** is hosted on a Linux server running nginx. It uses a MariaDB SQL database for storing the data.

3. API documentation

This section sets forth Co-Inform's application programming interface (API): a set of subroutine definitions, protocols, and tools for building application software. This documentation will allow developers to build new applications that connect to the tools.

3.1 Data collector (DC)

Documented in **https://co-inform.github.io/claim-cred-api/**, in particular endpoints

- /search
- /collections
- /claim/search
- /url/collect

3.2 Misinformation Detection (MD)

The main misinformation APIs are documented in the MisinfoMe APIs, **https://github.com/co-inform/misinfome-api**. There are three different types of APIs: 1) the legacy credibility models; 2) the new parametrised credibility model, and; 3) the claim identification API. Following, we describe the most relevant APIs.

1) For the legacy credibility model:

- /credibility/sources/
- /credibility/tweets/{tweet_id}
- /credibility/users

2) For the parametrised credibility model:

- /credibility/assessor
- /credibility/publisher
- /credibility/agent
- /credibility/document
- /credibility/rulesets

3) For the claim identification:

- /claimid//embedding
- /claimid/identify
- /claimid/extract

For Claim Credibility, **https://co-inform.github.io/claim-cred-api/**, endpoints:

- /claim/predict/credibility
- /tweet/claim/credibility

For Stance Detection, the API is described in **http://coinform-content-analysis.west.uni-koblenz.de/docs**, in particular endpoints:

- /post/veracity
- /post/veracity_test

3.3 Plugin Gateway

Documented in **https://co-inform.github.io/gateway-api**, in particular endpoints:

- /twitter/tweet
- /twitter/evaluate

3.4 Tweet Store

Documented in **https://co-inform.github.io/tweetstore-api**, in particular endpoints:

- /tweet/{tweet_id}
- /tweet/{tweet_id}/replies
- /tweet

4. Conclusions

This deliverable describes the high-level architecture of the Co-Inform platform, including the interaction between its expected components. Specifically, it provides a high-level specification for the platform components and their expected interaction, namely:

- Data Collector (DC)
- Misinformation Detection (MD)
- Misinformation Flow Analysis and Prediction (MFAP)
- API Gateway and rule engine
- Perceptions and Behavior Mining (PBM)
- Browser Plugin and the dashboards

This version of the deliverable also specifies the hosting architecture and the API documentation.

This is the final version of the architecture. It should be flexible enough to accommodate any further requirements elicited within WP1 throughout the project, while providing a useful interface to the data through the API and user interface. While the specific details will evolve with the project's needs, the general architecture will serve as a good foundation to build around.