QROWD - Because Big Data Integration is Humanly Possible

Innovation action

D4.4 – Crowdsourced multilingual data harvesting and extraction framework

Author/s	Claus Stadler, Gordian Dziwis, Patrick Westphal (InfAI), Luis-Daniel Ibáñez, and Eddy Maddalena (Soton)
Due date	30.11.2018
Version	1.0
Dissemination level	PU
Status	Final

Project co-funded under Horizon 2020 Research and Innovation EU programme, grant agreement no. 732194

Table of contents

1. INTRODUCTION **5**

2. PRELIMINARIES **7**

2.1 NATURAL LANGUAGE PROCESSING 7

2.1.1 QUALITY MEASURES 7

2.1.2 NLP INTERCHANGE FORMAT 8

2.1.3 CROWDSOURCING IN NATURAL LANGUAGE PROCESSING 9

2.2 APACHE NIFI 9

2.3 FOX 10

2.4 MINER 10

3 DESIGN **11**

3.1 SPECIFICATION 11

3.2 ARCHITECTURE 12

3.2.1 SOURCES 13

3.2.2 NLP ENGINES 14

3.2.4 CROWDSOURCING SERVICES 15

3.2.4.1 DATA PREPROCESSING AND UNIT DATA GENERATION 15

3.2.4.2 ENTITY CONFIRMATION TASK TEMPLATE 17

3.2.4.2 TASK OUTPUT 19

3.2.4.2 INTEGRATION IN APACHE NIFI 20

4 CONCLUSIONS **21**

ABSTRACT

In this deliverable, we present our approach to harvesting reputable, relevant open data, covering the transport/traffic domain, collected from international organisations (e.g. published on their blogs, as RSS/Atom feeds, via Twitter etc.). Unstructured data like this still makes up an important portion of the Web and consuming unstructured data in big data workflows is still a challenge. We approach this challenge by connecting the data harvesting stage with state-of-the-art natural language processing (NLP) tools, and crowdsourcing services developed in WP3. By leveraging Apache Nifi for data routing and transformation, we reduce manual data processing steps to a minimum and provide an automated data processing pipeline. With the NLP frameworks FOX and Miner we analyse the harvested text corpora and extract named entities and relationships between them. The crowdsourcing services developed in WP3 are utilized to gather feedback on the performance of the NLP frameworks. This feedback channel is used to guide the training of the underlying NLP algorithms in order to assess and improve their performance.

EXECUTIVE SUMMARY

This deliverable describes the overall workflow for multilingual data harvesting and extraction, and describes individual building blocks, how they were implemented and how they interact. This workflow is tailored to establish 'human in the loop' feedback channels to guide and improve information extraction from text data. Accordingly, the interfaces to the crowdsourcing components developed in WP3 are detailed as well.

This deliverable serves as a technical documentation for developers involved in building and integrating data acquisition components backed by crowd feedback. Besides this it could also serve as a reference for other researchers aiming at improving NLP tasks like named entity recognition or relation extraction by means of human validation.

The deliverable covers three focus areas touching on individual technical issues of textual data harvesting and information extraction with the human in the loop. We first explain how we harvest unstructured multilingual data and present our approach which allows a seamless integration of diverse data sources. We exemplify the harvesting and integration with Twitter data and RSS feeds. For the information extraction part we show how to include the natural language processing (NLP) tools FOX and Miner which are used for named entity recognition and the detection of relations between these named entities. Finally, we present how we include crowd feedback into this workflow for validating the extracted entities and relations. This crowd feedback is eventually used to guide the NLP engines and improve their information extraction performance.

All processes are implemented in a state of the art dataflow centric architecture and integrated into the QROWD platform. The components are independent, reusable and configurable through a graphical user interface which enables users and researchers to prototype natural language processing workflows and involve the crowd to improve the information extraction results.

The crowdsourcing components utilized to improve the multilingual data harvesting and information extraction are further detailed in deliverable D3.2. The storage and integration approach of the extracted information is explained in deliverable D7.3. The upcoming deliverable D8.3 will give an overview of the QROWD platform as an architecture and software system executing the multilingual data harvesting and information extraction.

1. INTRODUCTION

Nowadays citizens increasingly interact with the Web or other Internet services through their mobile devices. Besides browsing the Web, e.g. to read news, they also contribute content, e.g. in social networks, messaging services, or blogs. In combination with their location information citizens themselves can serve as a valuable information source for smart cities. In particular, text messaging services such as Twitter and open standards like RSS[1] can be mined for information relevant for decision makers and policies. To give an example, in cases of natural disasters, emergencies, or accidents, tweets may be a much faster source of information compared to, e.g. local news stations. However, even though these tweets usually appear in vast quantities and may even carry useful geolocation metadata, not all of them are unflawed, which also holds for other social media channels. Thus, new approaches are needed to integrate these big information streams. Here we tackle this problem by combining machine learning with the power of the crowd in an automated way.

Machine learning and crowdsourcing are diametrically opposed approaches for analysing unstructured data. Crowdsourcing tasks are often cost prohibitive, while machine learning is not yet a match for human intelligence. By combining both, the QROWD platform provides cheap and high quality data extraction capabilities. We minimize the cost in terms of higher system complexity by integrating the components into the open source data flow engine Apache Nifi[2]. Apache Nifi automates the flow of data between the components and comes with a graphical user interface for configuring data flow processes. This reduces the requirements for adopting the framework to other use cases.

On the machine learning side we use the natural language processing (NLP) tools FOX[3] and Miner. FOX is developed by the AKSW working group of the University of Leipzig, while Miner was developed by AI4BD. Both tools provide fine-grained named entity recognition and relation extraction.

Miner and FOX train their algorithms on freely available annotated corpora. Extending those annotated corpora, and thus providing more variant and less biased training data, results in a better model learned by these machine learning tools, which eventually leads to a better performance for the NLP tasks. Additionally, new concepts can be added to the model, by providing the right annotations. Producing these annotations is a work-intensive but not complex task, which makes it a good fit for crowdsourcing. In the QROWD platform we integrated these NLP tools with the crowdsourcing services developed in WP3.

The overall workflow is as follows: As a first preprocessing step we try to determine the language used in the input text data and store this as metadata when handing the data off to the NLP tools. This enables a multilingual processing, required for a European wide deployment. The language detection is implemented as an Apache Nifi processor, provided by AI4BD, which wraps their language detection engine. To establish the integration with the crowdsourcing tools developed in WP3 we defined templates for the actual crowdsourcing tasks. These templates provide a crowd

[1] http://www.rssboard.org/
[2] https://nifi.apache.org/
[3] http://aksw.org/Projects/FOX.html

worker with an intuitive interface for verification of the NLP engines' results. All the annotations performed or validated by the crowd workers are saved in a database and are the basis for retraining the NLP engines.

In the following chapter we summarize the key technologies and tools utilized. In section 3 we explain their role in the overall platform architecture. The main conclusions are drawn in section 4.

2. PRELIMINARIES

2.1 NATURAL LANGUAGE PROCESSING

Unstructured data, like e.g. free text, still makes up an important portion of the Web. Concerned with transforming this unstructured data into structured data is **Natural language processing (NLP)**, a subfield of computer science, information engineering, and artificial intelligence. NLP works on the interface between computers and human (natural) languages, in particular on how to program computers to process and analyze large amounts of natural language data. Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation. Relevant for the QROWD project are the subfields **named entity recognition** (**NER**) and **relationship extraction (RE).**
NER is the task of information extraction that seeks to locate and classify named entity mentions in unstructured text into predefined categories such as persons names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. State-of-the-art NER tools can provide fine-grained classifications of entities, where the actual types are represented by ressources from the DBpedia knowledge base, describing the (potentially ambiguous) concepts. For example “Adam Smith” would not just be classified as a *person*, but specified as the *economist* Adam Smith by linking to the corresponding DBpedia resource. RE is concerned with the detection and classification of relationships between the entities found with NER.
The machine learning algorithms applied, are trained with **annotated corpora**. Corpora are datasets of natural language, and a single set of text data annotated with the same specification is called an annotated corpus. Annotation is usually done by adding relevant metadata to a dataset. Any metadata tag used to mark up elements of the dataset is called an annotation over the input. However, in order for the algorithms to learn efficiently and effectively, the annotations must be accurate and relevant to the task the machine is being asked to perform. For this reason complete and correct language annotations are a critical for high quality NER and RE.

2.1.1 QUALITY MEASURES

Typical measures to evaluate classification systems are **precision**, **recall** and the **f-measure**. If there is a set of elements with a type, to be extracted and classified by an algorithm, then precision is the fraction of correctly classified elements over the total number of selected elements. Recall is the fraction of elements of type r that are correctly extracted, over the total number of elements of type r. The f-measure is the harmonic mean of precision and recall.

For NER precision, recall and f-measure can accordingly be defined as:

$$P = \frac{Number\ of\ correctly\ extracted\ entities}{Total\ number\ of\ extracted\ entities}$$

$$R = \frac{Number\ of\ correctly\ extracted\ entities}{Total\ number\ of\ entities\ in\ text}$$

$$F1 = \frac{2PR}{P+R}$$

And for a relationship extraction they are defined as:

$$P = \frac{Number\ of\ correctly\ extracted\ relationships}{Total\ number\ of\ extracted\ relationships}$$

$$R = \frac{Number\ of\ correctly\ extracted\ relationships}{Total\ number\ of\ relationships\ in\ text}$$

$$F1 = \frac{2PR}{P+R}$$

2.1.2 NLP INTERCHANGE FORMAT

Because the QROWD project is committed to Linked Data, the annotation will be expressed by means of the **NLP Interchange Format (NIF)**[4], which is an RDF/OWL-based format that aims to achieve interoperability between NLP tools, language resources and annotations. The core of NIF consists of a vocabulary, which can represent strings as RDF resources. A special URI Design is used to pinpoint annotations to a part of a document. These URIs can then be used to attach arbitrary annotations to the respective character sequence. Based on these URIs, annotations can be interchanged between different NLP tools.
A short example of NIF is shown in Figure 1 below. The sentence “Barack Obama is a president” was parsed by a NER engine and found an entity *Person* named “Barack Obama”.

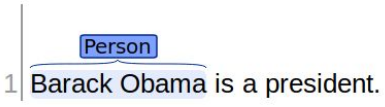

Fig 1. Annotation Example

Figure 2 shows an excerpt of the JSON-LD NIF output, the context and engine specific information was omitted.

```
{
  "@graph": [{
       "@id": "http://ns.aksw.org/fox/demo/document-1#char0,12",
```

[4] Rizzo, Giuseppe, et al. "NERD meets NIF: Lifting NLP Extraction Results to the Linked Data Cloud." LDOW 937 (2012).

```
      "@type": "nif:Phrase",
      "anchorOf": "Barack Obama",
      "beginIndex": "0",
      "endIndex": "12",
      "referenceContext": "http://ns.aksw.org/fox/demo/document-1#char0,28",
      "taClassRef": ["dbo:Person", "foxo:PERSON",
"http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Person",
"schema:Person"],
      "taIdentRef": "dbr:Barack_Obama"
  }, {
      "@id": "http://ns.aksw.org/fox/demo/document-1#char0,28",
      "@type": ["nif:CString", "nif:Context"],
      "beginIndex": "0",
      "endIndex": "28",
      "isString": "Barack Obama is a president."
  },
...
```

Fig 2. NLP Interchange Format example

2.1.3 CROWDSOURCING IN NATURAL LANGUAGE PROCESSING

Human linguistic annotation is crucial for natural language processing tasks but can be expensive and time-consuming. For NER and RE the annotations are the base on which a researcher can train a machine learning algorithm to obtain a model which is then used to extract the required information from the text. Those extracted annotations can be examined and corrected by humans and fed back to the algorithm to further improve the quality.
Crowdsourcing is one possibility to reduce the costs associated with creating an annotated corpus. It comprises practices of obtaining information or input into a task or project by enlisting the services of a large number of people, either paid or unpaid, typically via the Internet. A downside of crowdsourcing is the overhead associated with setting up a system for creating, distributing and collecting the tasks. Another approach to reduce the costs, is to streamline the annotation and relearning workflow such that data scientists can do the annotation themselves. An example for this approach is Prodigy[5].

2.2 APACHE NIFI

The QROWD Project employs **Apache Nifi** for automating the flow of data between the components. While the term 'dataflow' is used in a variety of contexts, we use it here to mean the automated and managed flow of information between systems. This problem space has been around ever since enterprises had more than one system, where some of the systems created data and some of the systems consumed data. The problems and solution patterns that emerged have been

[5] https://prodi.gy/

discussed and articulated extensively. A comprehensive and readily consumed form is found in the Enterprise Integration Patterns[6]. The dataflows are produced and consumed by processors. In Apache Nifi new processors can be added to enhance its capabilities. Apache Nifi maintains provenance information and comes with an easily manageable GUI for creating customized dataflows.

2.3 FOX

The **Federated knOwledge eXtraction (FOX)** framework developed by the AKSW working group of the University of Leipzig is a highly accurate open-source framework that implements RESTful web services for named entity recognition and relation extraction. This framework should be leveraged in the QROWD project, because it achieves a higher F-measure than other state-of-the-art named entity recognition frameworks by combining the results of several approaches through ensemble learning[7]. Moreover, it disambiguates and links named entities to DBpedia by relying on the AGDISTIS[8] framework. As a result, FOX provides users with accurately disambiguated and linked named entities in several RDF serialization formats.

2.4 MINER

Miner is a tool to extract information from unstructured data such as text. It is based on machine learning using neural networks, vector space models and dictionaries to extract named entities. In order to tune the system (which is part of the supervised learning) crowdsourcing services developed in QROWD can serve as user curated/generated training data provider, and can therefore improve the quality for the named entity extraction. The extracted information is transformed and linked within the platform and stored in a knowledge graph.

[6] Hohpe, Gregor, and Bobby Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.
[7] Speck, René, and Axel-Cyrille Ngonga Ngomo. "Ensemble learning for named entity recognition." *International semantic web conference*. Springer, Cham, 2014.
[8] Usbeck, Ricardo, et al. "AGDISTIS-graph-based disambiguation of named entities using linked data." *International Semantic Web Conference*. Springer, Cham, 2014.

3 DESIGN

This section explains the requirements and outlines the architecture of our solution. Afterwards the components are specified and their interaction in terms of the overall dataflow is visualized.

3.1 SPECIFICATION

This task is concerned with the harvesting of reputable, relevant open data covering the transport/traffic domain collected from international organisations matching the use cases.

The exact nature of the use cases will depend on the environment where the QROWD platform will be deployed. Thus this architecture must be able to ingest data from heterogeneous source. Use cases in the context of transport and traffic will be time sensitive, such that, for example, in case of a traffic accident officials could get an overview over the situation with videos uploaded by citizen before first responders arrive. Both concerns are addressed by using Apache Nifi as a dataflow framework to handle streaming data in the most common formats.

Semantic analysis of text corpora with NLP frameworks FOX and Miner in an automated way

For semantic analysis, FOX and Miner extract named entities and relations. To do this in an automated way from a text corpus, both tools must be integrated into the Apache Nifi dataflow. First the NLP engine must be provided with the text to analyse and then the resulting extraction must be collected. There are two ways to achieve this: (i) via an HTTP processor which queries an independently running instance of the NLP engine per API call, or (ii) by integrating the NLP engine directly into the overall dataflow by implementing an Apache Nifi processor which wraps the NLP engine. Using HTTP is faster to implement, but does not provide the ability to configure the NLP engine through the Apache Nifi GUI. FOX is integrated via HTTP by InfAI while Atos develops an Apache Nifi processor for Miner, which will be used here.

Integration with crowdsourcing services developed in WP3 to gather feedback on, and improve the performance of named entity recognition and relation extraction

The extraction task can produce two kinds of errors, type I and type II. Type I corresponds to a false positive. Classifying Barack Obama in the sentence: “Barack Obama is a president.” as an apple would be an example for a type I error. While a type II error is a false negative and in this example this would mean that the NLP

engine failed to extract 'Barack Obama' as an entity. Additionally there are errors which stem from an intrinsic uncertainty. Is or was Obama a president? Or is Barack Obama just a person with the same name as the former president of the U.S.?
These problems inherent in NLP are the main motivation for having humans in the loop. The deliverable D3.2 describes the crowdsourcing services deployed on the QROWD platform. These services will be connected to the NLP framework. Because of resource constraints not all annotations can be evaluated and corrected by humans. In the case of Miner we can base the decision of which annotations to verify by the crowd on the confidence score which is stored with the actual annotation results. FOX does not provide a confidence score with its output, so annotations can be selected either randomly, or following a different criterion, for crowdsourcing.

3.2 ARCHITECTURE

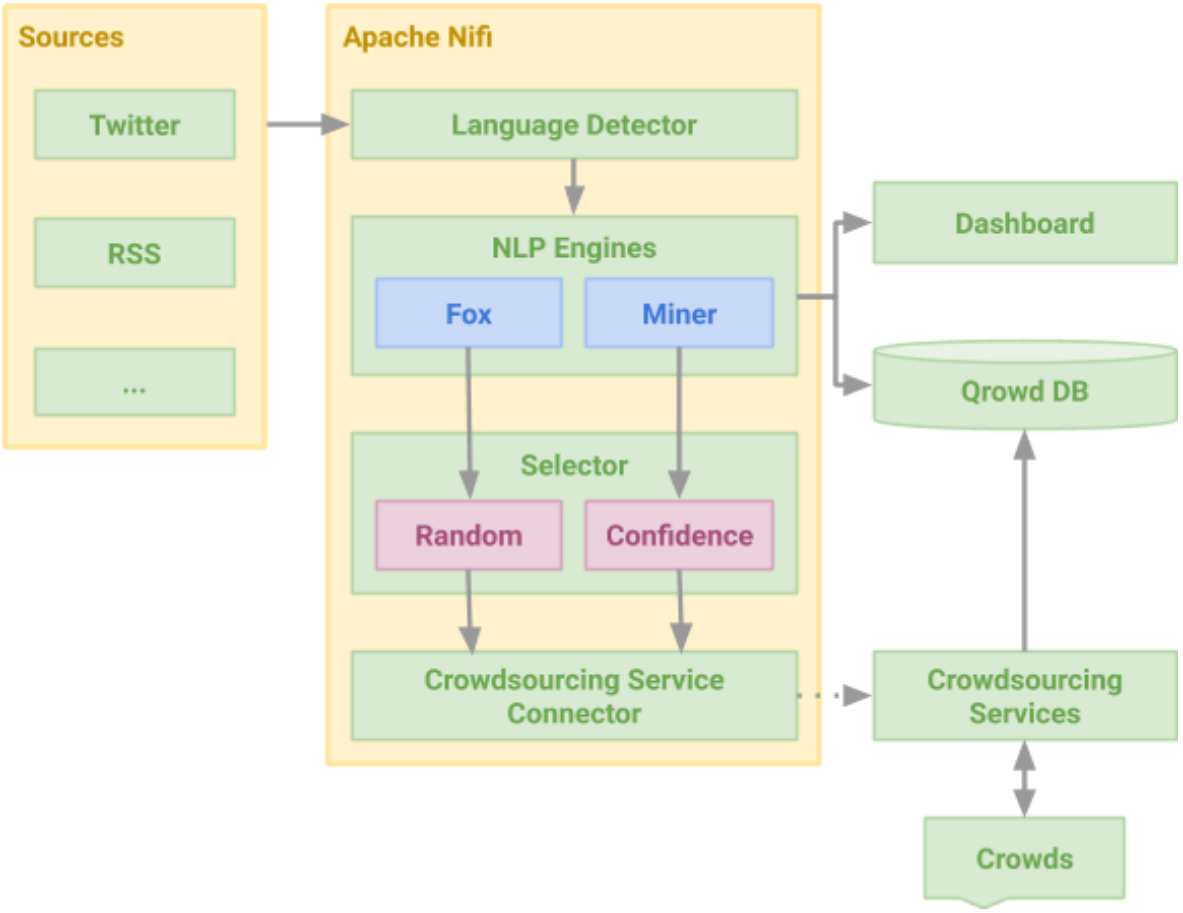

Fig 3. Framework architecture diagram

The overall architecture and data flow is shown in Figure 3. Data enters the system from the **Sources**. First implemented data sources are Twitter and RSS. **Twitter** is an online news and social networking service on which users post and interact with messages known as "tweets". Those tweets are restricted to 280 characters each. Since many tweets talk about events (with useful location information), such as

natural disasters, emergencies, and accidents, they can be valuable information sources for city officials. **Rich Site Summary (RSS)** is a type of Web feed which allows users and applications to access updates to online content in a standardized, computer-readable format. Most news outlets provide an RSS feed.
Sources are queried by the corresponding Apache Nifi processors in regular intervals. To query Twitter Apache Nifi already provides a processor[9]. Since RSS builds on World Wide Web standards it can be queried by a processor which sends HTTP requests. Based on the response data a flow file is generated with the response data as the content. This flow file is then sent to to the Language Detector.
The **Language Detector** is implemented by an Apache Nifi processor provided by AI4BD which wraps their language detection engine. This step is necessary because the language of the extracted text usually cannot be known in advance. For example a Twitter user could tweet in multiple languages. This language detection step adds a language identifier to the metadata of the flow file.
For processing the data with the NLP engines the text is extracted from the payload of the flow files. The language tag and the text data serve as input for the NLP engines. The NLP engines return the text with annotations identifying the contained entities and relations. All of those are sent to the dashboard and saved in the QROWD DB as well. The **QROWD DB** is the database which, among others, contains the annotated text corpora. The annotations are stored in the NLP Interchange Format in a triple store. The dashboard provides an aggregated view on the extracted annotations.
The classifications of the specific NLP engines are also sent to the Selector. The **Selector** component selects a subset of the classifications for validation by the crowd. FOX and Miner have different selection strategies. Miner provides a confidence score with its classifications. Based on this confidence score, classification results with a score lower than a specific threshold are sent to the Crowdsourcing Service Connector for crowd validation. Because FOX does not provide a confidence value with its results, a percentage of all classification is randomly selected and routed to the Crowdsourcing Service Connector. The **Crowdsourcing Service Connector** connects to the **Crowdsourcing Services** described in deliverable D3.2.

3.2.1 SOURCES

[9] https://github.com/apache/nifi/tree/master/nifi-nar-bundles/nifi-social-media-bundle

Property	Value
Twitter Endpoint	**Sample Endpoint**
Consumer Key	**KEY**
Consumer Secret	Sensitive value set
Access Token	**TOKEN**
Access Token Secret	Sensitive value set
Languages	en, it
Terms to Filter On	trento
IDs to Follow	No value set
Locations to Filter On	46.0700,11.1200,46.0800,11.1800

Fig 4. Configuration panel of Twitter processor

Figure 4 shows the configuration panel of the processor fetching tweets from Twitter. The bold printed properties are for connecting to the Twitter API. The following properties enable the user to filter the tweets. Using the "Locations to Filter on" specifies a bounding box by its latitude and longitude values. When conducting geographical searches, the search API will first attempt to find tweets which have a latitude and longitude value within the queried bounding box, and in case of not having success, it will attempt to find tweets created by users whose profile location can be reverse geocoded to a latitude and longitude within the queried bounding box, meaning that it is possible to receive tweets which do not include latitude and longitude information. The "Languages" property restricts tweets to the given language, expressed by an ISO 639-1[10] language code. Language detection is best-effort. With "Terms to Filter On" and "IDs to Follow" the tweets can be restricted to containing specific keywords or being tweeted from a specific ID.

3.2.2 NLP ENGINES

To integrate the NLP engines into the QROWD platform text annotation requests need to be generated, sent to the respective NLP engine, and the results need to be collected after the annotation finished.

```
{
  "input" : "The philosopher and mathematician Leibniz was born in Leipzig in
1646 and attended the University of Leipzig from 1661-1666.",
  "type": "text",
  "task": "ner",
  "output": "jsonld",
  "lang": "en"
}
```

[10] https://www.iso.org/iso-639-language-codes.html

Fig 5. JSON payload for FOX-request

For requesting a text annotation from FOX a HTTP GET request is sent to a FOX instance. Figure 5 shows a JSON body for such a request. The JSON data includes the text to annotate in the *input* field. The *type* field determines if the input field specifies the text itself or a URL indicating the location of the text. FOX does not identify the language of the text by itself, so the *lang* field is required. The *output* field tells FOX which RDF serialisation to use for the NIF output.

3.2.4 CROWDSOURCING SERVICES

The search for errors in the phase of entity extraction will be limited to type I errors. The question “Is this entity annotation correct?” is simpler and easier to crowdsource than questions for type II errors. We designed a crowdsourcing task template to confirm results from FOX and Miner with crowd workers, to be deployed in the Crowdsourcing Services architecture described in D3.2. In the following, we assume the use of Figure Eight[11] as crowdsourcing platform. The input for the task template is the FOX or Miner output containing the automatically annotated entities. As a configuration parameter one can set the number of independent judgements wanted per text annotation. Usually, a value between five and ten is enough to ensure a good level of reliability of the obtained assessments.

3.2.4.1 DATA PREPROCESSING AND UNIT DATA GENERATION

The first step to make use of our task template is to instantiate it in Figure Eight via its API. When an instance of the task is running we can start the evaluation of a text annotation result by uploading to a task file called Data Unit.
Even though we only introduced FOX and Miner we don’t want to restrict the overall workflow to those two, but also want to allow the deployment other NLP tools. To make the template approach work with all those different output formats of the individual NLP tools we added a preprocessing step to unify the annotation results to a simplified format, containing only the data necessary for our crowdsourcing purposes. Besides the transformation scripts for FOX and Miner shipped with the QROWD platform, users can plug in their own converters to be executed in this transformation step. Figure 6 below shows an example of the format required by the task template, it includes:

- *tweet_id*: For identification; can be an ID coming from the NLP tool
- *tweet_text*: Text of the tweet
- *entities*: List of detected entities within the tweet, each entity needs to have the following attributes
    - *text*: Text that was matched to an entity
    - *pos_first_char*: Starting position of the substring that was matched to an entity
    - *pos_last_char*: End position of the substring that was matched to an

[11] https://www.figure-eight.com/

entity

- *url*: Link to a page that provides information to crowd workers about the discovered entity to help them assess the correctness of the matching. In our examples, we chose Wikipedia pages for two reasons: (i) since we link to DBpedia resources, for which it is relatively easy to find the corresponding Wikipedia page, and (ii) since Wikipedia articles follow a common, proven structure and layout which should be familiar to the crowd workers it should be easier to grasp the content.


```
{
  "tweet_id": "tweet_1",
  "tweet_text": "The Italian businessman Sergio Marchionne got awarded of an
honorary degree at the University of Trento since was chairman and CEO of
Ferrari.",
  "entities": [
    {
      "url": "https://en.m.wikipedia.org/wiki/Sergio_Marchionne",
      "text": "Sergio Marchionne",
      "pos_first_char": 24,
      "pos_last_char": 40
    },
    {
      "url": "https://en.m.wikipedia.org/wiki/University_of_Trento",
      "text": "University of Trento",
      "pos_first_char": 83,
      "pos_last_char": 102
    },
    {
      "url": "https://en.m.wikipedia.org/wiki/Ferrari",
      "text": "Ferrari",
      "pos_first_char": 134,
      "pos_last_char": 140
    }
  ]
}
```


Fig 6. JSON payload for crowdsourcing task

Once we have this standardized representation of annotated tweets that need to be verified, we have to create a Data Unit file, holding each of these entries. However, we cannot simply add the nested JSON payload sketched above since Figure Eight restricts the JSON structure of a Unit Data array to only hold un-nested key-value pairs. To overcome this limitation, in each position of the Unit Data array, we place a JSON object having only two keys and two values: (1) *tweet_id*, that contains the id of the tweet (copied there to ease its access), and (2) *tweet_obj*, that holds a string containing a quoted and stringified version of the annotated tweet JSON data in our simplified format.

3.2.4.2 ENTITY CONFIRMATION TASK TEMPLATE

In this section we describe how a crowd worker sees the task. The interface, depicted in Figure 7 and 8, first shows the instructions on how to perform the task correctly, then the contributor is presented with a view comprised of three panels. (1) The top panel, “Tweet”, shows the tweet to be evaluated, having the text matched to an entity that requires confirmation in bold. (2) In the bottom-left, the "Entities" panel showing a vertical list of the matched entities. By clicking on an entity, a drop-down list appears for the contributor to confirm the annotation, invalidate it, or to express uncertainty. (3) In the bottom-right, of the panel is present a frame box that shows the Wikipedia page of the currently selected entity.

Once a worker finishes the assessment of all entities in all tweets the task ends; thus a new button who allows the worker to get rewarded appears.

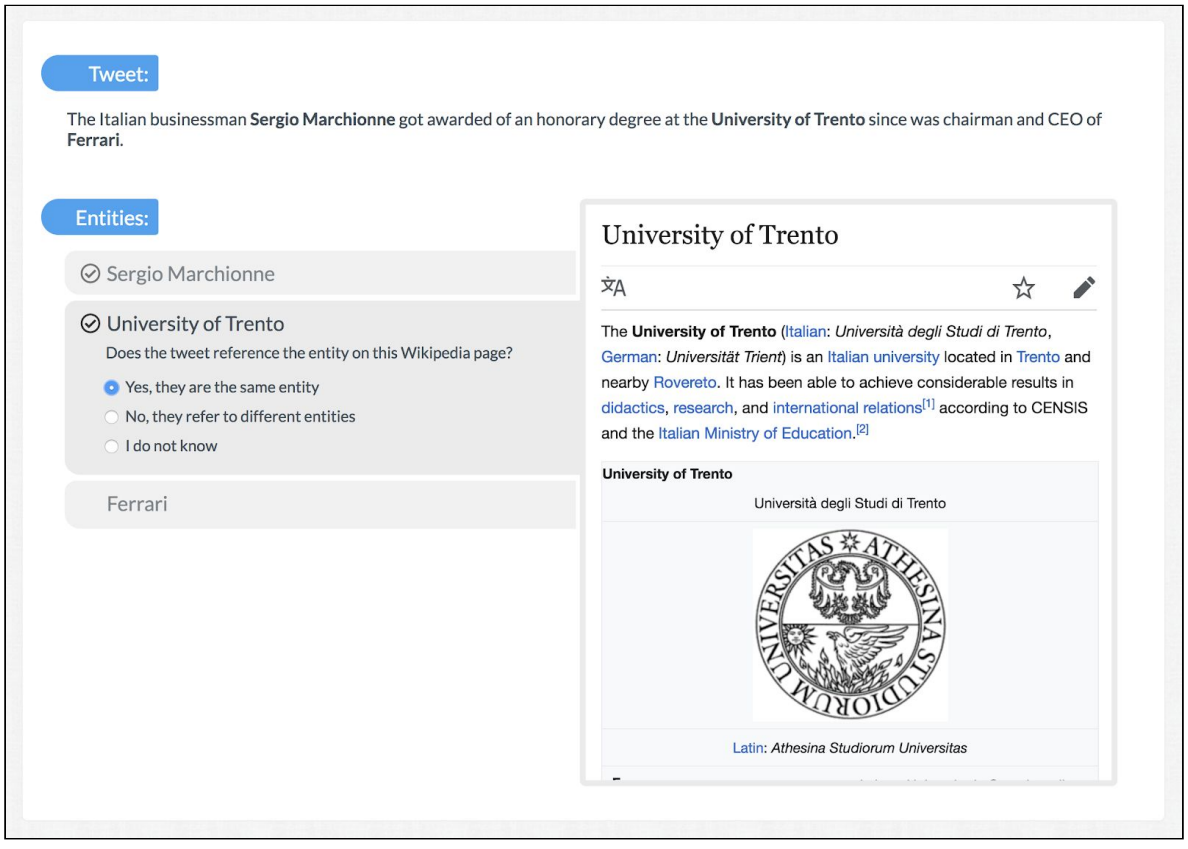

Fig 7. The interface for the entity confirmation task

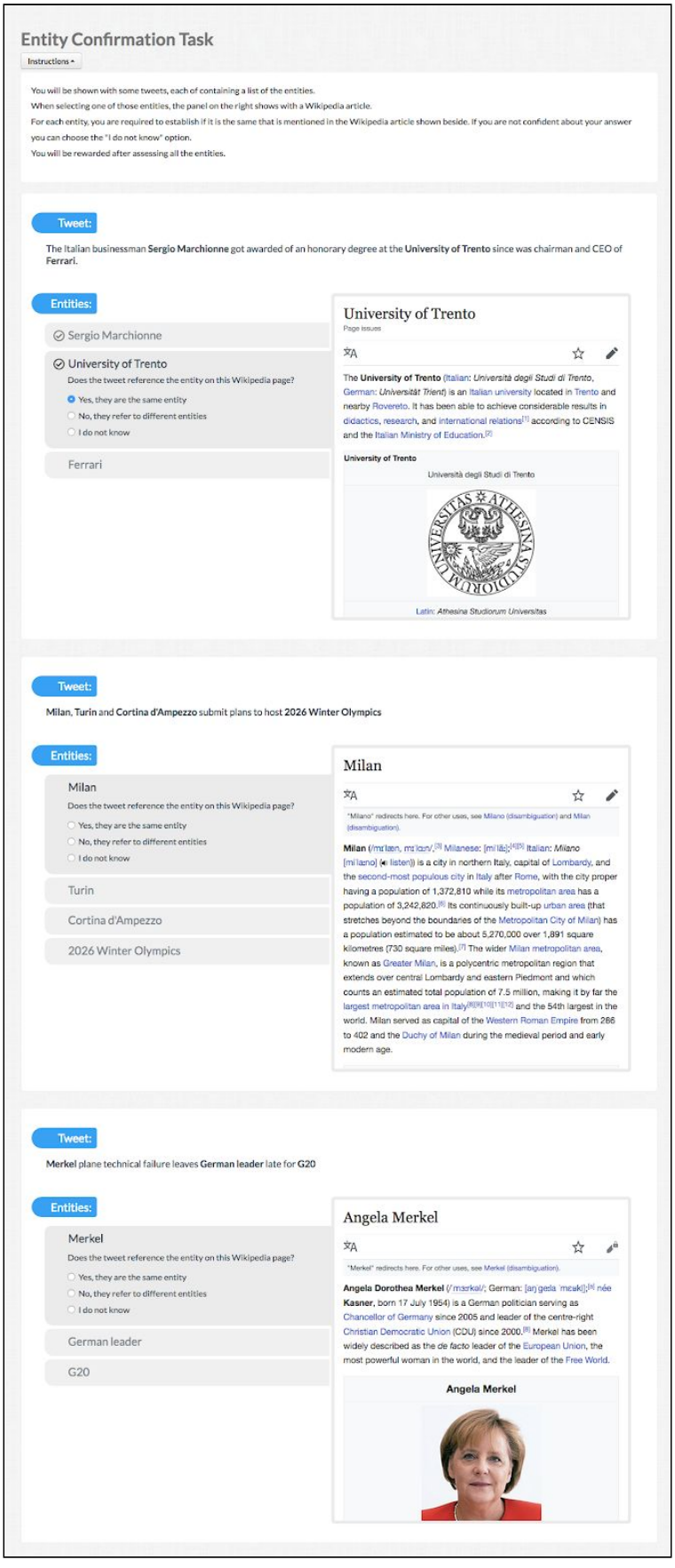

Fig 8. Entire task page, composed of task instructions and three tweets to classify

3.2.4.2 TASK OUTPUT

The output of a task run is a JSON file, having listed the tweets as well as their annotated entities, and for each of them showing the correspondent judgement assigned by the crowd worker (which is one choice among “same entity”, “different entities”, and “I do not know”). The file contains also information about the workers who generated it. The fragment of the result file that shows the crowd worker annotations is shown in Figure 9 below.


```
[
  {
    "tweet_id": "tweet_3",
    "entities": [
      {
        "entity_text": "Sergio Marchionne",
        "judgement": "same_entity"
      },
      {
        "entity_text": "University of Trento",
        "judgement": "same_entity"
      },
      {
        "entity_text": "Ferrari",
        "judgement": "same_entity"
      }
    ]
  },
  {
    "tweet_id": "tweet_id_5",
    "entities": [
      {
        "entity_text": "Milan",
        "judgement": "same_entity"
      },
      {
        "entity_text": "Turin",
        "judgement": "same_entity"
      },
      {
        "entity_text": "Cortina d'Ampezzo",
        "judgement": "i_do_not_know"
      },
      {
        "entity_text": "2026 Winter Olympics",
        "judgement": "same_entity"
      }
    ]
```

```
  },
  {
    "tweet_id": "tweet_8",
    "entities": [
      {
        "entity_text": "Merkel",
        "judgement": "same_entity"
      },
      {
        "entity_text": "German leader",
        "judgement": "different_entities"
      },
      {
        "entity_text": "G20",
        "judgement": "same_entity"
      }
    ]
  }
]
```

Fig 9. Output of a crowdsourcing task

3.2.4.2 INTEGRATION IN APACHE NIFI

The task template is integrated in Apache NiFi and can be deployed to Figure Eight when confirmation of automatically extracted entity is needed. The output of FOX or Miner cannot be used directly as input for the task but needs to be preprocessed with a Jolt[12] transform to a JSON file having the same structure as sketched in Figure 10 below.

```
{
  // Our text of choice
  text     : "Barack Obama is a president.",
  // The entities entry holds all entity annotations
  entities : [
    // Format: [${ID}, ${TYPE}, [[${START}, ${END}]], ${INTERLINK}]
    // note that range of the offsets are [${START},${END})
    ['T1', 'Person', [[0, 12]], 'https://en.wikipedia.org/wiki/Barack_Obama'],
  ],
}
```

Fig 10. Jolt JSON transformation specification

[12] https://github.com/bazaarvoice/jolt

4 CONCLUSIONS

We demonstrate a workflow for harvesting and analyzing reputable, relevant open data, covering the transport/traffic domain, collected from international organisations. For this we integrated NLP and crowdsourcing components with Apache Nifi into the QROWD platform. This enables the QROWD platform to harvest Twitter and RSS feed messages, detect their language and perform named entity recognition and relationship extraction on the harvested data. We built templates for crowdsourcing the verification of these classifications and connected the framework to the QROWD platform crowdsourcing services.

Crowdsourcing the validation of classifications will be further examined for named entity recognition. While it is relatively easy to create a crowdsourcing task for verifying entities the NLP engine classified, it is more difficult to annotate entities the NLP Engines did not catch. For this, the main features of a full blown distributed annotation tool are required. For example the ability to choose from a list of possible entities and highlighting the location of this entity in the text. Creating and managing annotations is a complex workflow, so we will evaluate, if already available open source tools like Brat[13] or Inception[14] could be a good fit.

Retraining the models needs further examination, as for now the retraining does not happen in an automated way.

Regarding the different output formats and our unification approach we will further investigate whether there are already existing, easy to use and established vocabularies we can utilize to increase the reuse of the NLP tools' annotation results and the acceptance with respect to the transformation step.

Besides this we will also examine whether we could extract so called *latent location information*[15] from text data for location dependent use-cases.

[13] http://brat.nlplab.org/index.html
[14] https://inception-project.github.io/
[15] Lee, Sunshin. *Geo-Locating Tweets with Latent Location Information*. Diss. Virginia Tech, 2017.