Social Semantic Emotion Analysis for Innovative Multilingual Big Data Analytics Markets

D3.3 MixedEmotions Big Data Platform Architecture, final version

<table>
<thead>
<tr>
<th>Project ref. no</th>
<th>H2020 644632</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>MixedEmotions</td>
</tr>
<tr>
<td>Start date of project (dur.)</td>
<td>01 April 2015 (24 Months)</td>
</tr>
<tr>
<td>Document due Date</td>
<td>31 March 2017 (Month 24)</td>
</tr>
<tr>
<td>Responsible for deliverable</td>
<td>Paradigma Tecnológico</td>
</tr>
<tr>
<td>Reply to</td>
<td><a href="mailto:cnavarro@paradigmatecnologico.com">cnavarro@paradigmatecnologico.com</a></td>
</tr>
<tr>
<td>Document status</td>
<td>Final</td>
</tr>
<tr>
<td>Project reference no.</td>
<td>H2020 644632</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Project working name</td>
<td>MixedEmotions</td>
</tr>
<tr>
<td>Project full name</td>
<td>Social Semantic Emotion Analysis for Innovative Multilingual Big Data Analytics Markets</td>
</tr>
<tr>
<td>Document name</td>
<td>MixedEmotions_D3.3_31_03_2017_Final_Version_of_the_MixedEmotions_Platform_PT</td>
</tr>
<tr>
<td>Security (distribution level)</td>
<td>PU</td>
</tr>
<tr>
<td>Contractual delivery date</td>
<td>31 March 2017 (M24)</td>
</tr>
<tr>
<td>Deliverable number</td>
<td>D3.3</td>
</tr>
<tr>
<td>Deliverable name</td>
<td>Final Version of the MixedEmotions platform</td>
</tr>
<tr>
<td>Type</td>
<td>Other</td>
</tr>
<tr>
<td>Version</td>
<td>Final</td>
</tr>
<tr>
<td>WP / Task responsible</td>
<td>WP3 / Paradigma Tecnológico</td>
</tr>
<tr>
<td>Contributors</td>
<td>PT (Carlos Navarro De Martino, Hugo Viejo), NUIG (Housam Ziad, Paul Buitelaar, Cécile Robin, Mihael Arcan, Ian Wood), UPM (Carlos Ángel Iglesias, Fernando Sánchez), ST (Giovanni Tummarello), PX (Pavel Matějka, Marek Klimes), BUT (Lubomír Otrusina), UP (Hesam Sagha), DW (Andy Giefer), ES (Francesco Adolfo Danza, Vincenzo Masucci)</td>
</tr>
<tr>
<td>Project Officer</td>
<td>Martina Eydner</td>
</tr>
</tbody>
</table>
Index

Executive summary

1. MixedEmotions Overview
   1.1. Platform Introduction
   1.2. Distribution of the MixedEmotions platform

2. MixedEmotions Capabilities
   2.1. Emotion recognition from Audio, Video and Text
   2.2. Sentiment Recognition
   2.3. Entity Extraction and Entity Linking
   2.4. Knowledge Graph
   2.5. Suggestion Mining
   2.6. Fusion
   2.7. Social Network Analysis
   2.8. Topic Extraction
   2.9. Twitter Crawler
   2.10. Youtube Crawler
   2.11. Analytics Visualization Module Kibi
   2.12. Machine Translation
   2.13. Speech to Text
   2.14. Gender Recognition from Audio
   2.15. Age Estimation from Audio

3. MixedEmotions platform architecture
   3.1. Platform architecture overview
      3.1.1. Processing modules
      3.1.2. Ingestion
         3.1.2.1. Twitter Crawler
         3.1.2.2. YouTube Crawler
      3.1.3. Graph Analysis: Social Semantic knowledge graph and network analysis
      3.1.4. Analytics and Visualization
      3.1.5. Pipeline orchestrator
   3.2. MixedEmotions Platform as a Toolbox
   3.3. MixedEmotions as a Big Data platform
      3.3.1. Microservices architecture
      3.3.2. MixedEmotions Services as Docker containers
      3.3.3. Mesos as MixedEmotions resource manager
      3.3.4. Marathon as MixedEmotions containers manager
3.3.5. Mesos-DNS as MixedEmotions discovery service
3.3.6. MixedEmotions pipeline orchestrator

4. MixedEmotions Modules
4.1 MixedEmotions Platform Open Source modules
m1. English Sentiment Extraction
m2. Czech and English Sentiment Extraction
m4. Spanish and English Sentiment Extraction
m5. English Emotion recognition
m6. Audio Emotion recognition
m7. Spanish and English Emotion recognition
m8. Spanish Entity Extraction
m10. English Entity Extraction
m13. Spanish Topic Extraction
m16. Suggestion mining
m20. Twitter media crawler
m21. Fusion
m22. Social Network Analysis
m25. Social semantic Knowledge Graph
m27. Video Emotion recognition
m31. Analytics module "Kibi"
m32. YouTube Crawler

5. MixedEmotions Platform proprietary modules
m3. Italian and English Sentiment Extraction
m9. Italian and English Entity Extraction
m11. External Italian and English Topic Extraction
m12. External English Topic Extraction
m15. External Italian and English Entity Linking
m17. Speech to text
m18. Machine translation
m23. Audio Emotion extraction
m24. Recommendation engine
m28. Age estimation from audio
m29. Gender identification from audio

7. MixedEmotions Use cases
7.1 MixedEmotions Capabilities: Example use cases
7.2 Pilot 1: Social TV
1.2.2 Pilot 2: Brand Reputation Management
1.2.3 Pilot 3: Call Center
7.1. Pilot 1: Social TV
7.2. Pilot 2: Brand Reputation Management
7.3. Pilot 3: Call Center

8. Conclusions

APPENDIX A: JSON-LD, THE COMMON FORMAT FOR INPUT AND OUTPUT FOR THE PLATFORM MODULES

- Introduction
- MixedEmotions Schema
- Examples
  - Entry
  - Sentiment Analysis
  - Suggestion Mining
- Emotions
  - Named Entity Recognition
  - Complete example

APPENDIX B: DOCKERIZATION

- Docker in a nutshell
- Installation
- How to create a Docker image
  - Creating a Dockerfile
  - Building images
  - Run containers
- Publishing images
  - MixedEmotions platform regarding Docker Hub

References
Executive summary

This document presents the final version of the MixedEmotions platform. This document gives a general overview of the whole platform, including the platform capabilities, pilot and use cases. It also includes information on the architecture.

The main challenge of the platform was the integration of modules. The platform should be able to handle the use of numerous modules, each of them created individually by different partners and in different programming languages. Moreover, there was the need for a lighter version of the platform, one that could be run in a single machine, for demo and testing purposes.

In order to solve these issues, the MixedEmotions platform has adopted a microservice based architecture. This kind of architecture relies on smaller components, independent for the most part from each other, and which communicate via a REST interface. Moreover, those microservices are deployed using containers, which make them more independent and scalable.

Taking advantage of the characteristics of this type of architecture, the MixedEmotions platform has now “toolbox” capabilities, which means that the users will be able to use only the components they need and even add their own. The microservice architecture also provides the platform with Big Data capabilities, as this type of architecture allows the platform to increase horizontally with very little cost. It also allows a single machine mode.

With this architecture, the focus has been put solely into the new modules. The support components such as databases are left for the user to choose and install, although they are still in the platform as recommended components. The idea is that a new user could test some modules independently, even in his own machine. Then, once he has tried the modules which could be relevant for him, he can shift to using them in a production environment, even in a scalable cluster of cloud machines.
1. MixedEmotions Overview

1.1. Platform Introduction

The MixedEmotions platform is a Big Data Toolbox for multilingual and multimodal emotion analysis. It can extract emotions from text, audio and video. However, it also has many other capabilities, such as sentiment analysis, social network analysis and knowledge graphs visualization among others.

The platform has been conceived as a toolbox which anyone can download and easily start using, with the processing of natural language data from text, audio and video, but it also includes the ability of being used in a Big Data production environment. With its microservice architecture, the platform can be deployed in a cluster of machines and be scaled horizontally as needed.

The platform also offers a set of optional tools including data crawlers, linked data analyzers and visualization tools, not directly related to Natural Language Processing, but that can be very interesting for potential projects using the platform.

The platform is mostly composed of open source modules. However, some additional modules that have been developed and tested as part of the project will be made available with a proprietary license. They will be listed as part of the platform's components, as they have been developed for the use cases' purpose. Users interested in those modules will be able to purchase their licenses by contacting the appropriate partner.

1.2 Distribution of the MixedEmotions platform.

All the information regarding the MixedEmotions platform is centralized in MixedEmotions' home page at http://mixedemotions.insight-centre.org/. This site contains all the information pertinent to the MixedEmotions platform, including examples, live demos, instructions and links to the repositories.

The open source modules of the MixedEmotions platform will all be available in the MixedEmotions repository in GitHub at https://github.com/MixedEmotions. The platform documentation, detailing the installation and operation instructions will be also made available in that platform at https://github.com/MixedEmotions/MixedEmotions.

Docker images for open source processing modules of the MixedEmotions platform will be available in the Docker Hub repository of images, along with licensing and operation instructions at https://hub.docker.com/u/mixedemotions/.
Most open source modules will be present both in GitHub and Docker Hub. A few do not have a Docker Image readily available, usually because they require an application token, although they provide a Dockerfile for generating Docker images.

2. MixedEmotions Capabilities
Here is a succinct description of the main MixedEmotions capabilities.

2.1 Emotion recognition from Audio, Video and Text

MixedEmotions has many modules focused on extracting emotions from text, video and audio. There are two main emotion representations. First is the category representation, where the media is classified with emotions such as joy, fear, sadness, disgust or anger. The other one is the dimensional representation, where each piece of media is given a vector in a multidimensional space. The most common is the VAD space, where emotions are represented in a tridimensional space of Valence, Arousal and Dominance.

2.2 Sentiment Recognition

MixedEmotions also has the capability of recognizing sentiment from media. Sentiment is usually characterized as positive, negative and neutral.
2.3 Entity Extraction and Entity Linking

The Entity Extraction modules detect concepts such as people, locations or organizations. The modules with Entity Linking capabilities will additionally assign DBpedia ids to those entities.

2.4 Knowledge Graph

This MixedEmotions module will use DBpedia information to create a Knowledge Graph. This can be used in conjunction with the Entity Linking module.

2.5 Suggestion Mining

The suggestion mining module will detect recommendations and ideas for improvement inside text comments.
2.6 Fusion

The fusion module will add the emotion data from all the three media a video is composed of: audio, image and text transcription, to produce a more accurate emotion characterization.

2.7 Social Network Analysis

The Social Network Analysis module will retrieve user data from Twitter and compose the relationship between Twitter actors. This is mainly used to characterize emotion and information propagation across the network, but also has other uses, for example identifying influencers among a subnet.

2.8 Topic Extraction

Topic extraction modules characterize media by their general subject. Topic examples could be sports, finance or technology.

2.9 Twitter Crawler
This module connects to the Twitter Streaming API and collects tweets that contain certain keywords. This module needs Twitter API keys to function.

### 2.10 Youtube Crawler

This module retrieves youtube videos and their metadata regarding certain keywords. This module needs the corresponding API keys.

### 2.11 Analytics Visualization Module Kibi

This module is an application to visualize data persisted in Elasticsearch. It is a Kibana fork with added graph capabilities.

### 2.12 Machine Translation
This module translates a text from several languages into English.

2.13 Speech to Text

These modules create a text transcription from an audio file.

2.14 Gender Recognition from Audio

This module identifies whether the speaker is male or female.

2.15 Age Estimation from Audio
This module returns an estimated age of the speaker.
3. MixedEmotions platform architecture

3.1. Platform architecture overview

The schema in the next Figure provides a high-level specification of the architecture of the MixedEmotions platform, including the core toolbox for natural language processing, visualization tools, linked data analysis tools and recommended persistence engines.

Figure 1. Full architecture of the MixedEmotions platform.

We distinguish the following elements in the MixedEmotions Big Data platform:
1. **MixedEmotions platform Natural Language Processing modules.** These are the main components of the platform, and the ones that will be used in any of the modes the MixedEmotions platform can be used. Those Natural Language Processing modules are deployed as REST services inside a Docker Image. Docker containers from those images will be deployed in a Marathon and Mesos framework in the Big Data mode.

2. **Ingestion.** This part gathers modules that collect data from Social Network APIs, such as Twitter or YouTube. These data should be put into some persistence system. Depending on data volume, it could simply be a regular filesystem, a distributed filesystem such as HDFS, or some kind of database.

3. **Graph analysis.** The tools in this part are about social network analysis from Social Media data and knowledge graph. They need certain graph persistence systems to work, such as OrientDB or the Unipop plugin for ElasticSearch.

4. **Visualization and data discovery.** A visualization and data discovery module can be used to obtain graphical representations of the data. Kibi is an extension of Kibana that allows graph representation and queries using linked data. Alternatively, if the user does not need this capabilities, regular Kibana is a simpler alternative. Both of these tools work through the reading of an Elasticsearch index.

5. **Orchestrator.** This is a software that manages the execution of various platform modules in a pipeline. It monitors the travel of the data along the platform, coordinating the execution of the modules specified by the user. As the modules work with a REST interface, it will be straightforward to integrate those modules with some external REST services. This orchestrator should be able to learn ip addresses of the Docker containers which contain the Natural Language Processing modules, by querying the Mesos DNS service.

### 3.1.1 Processing modules

These are the Natural Language Processing modules in the MixedEmotions Platform. As the platform comprises multiple modules, from different partners and different technologies, it is important to define a way to integrate them together.

The integration paradigm selected for the platform is a Microservice architecture approach, as introduced in the previous section. Further explanation will be found in section 3. In broader terms, a microservice architecture is one in which every functionality is provided by an independent service with a single functionality. Those services are usually deployed using containerization. In this project the microservices are deployed using Docker containers. Also, those services use a generic standard of communication, usually HTTP with a REST interface.

In general, NLP functions are expected to accept some text, audio or video, which will be analyzed for the computation of a result related to a given context or problem. The functions addressed in MixedEmotions are the following ones:

- Emotion recognition: the emotions expressed in the data.
- Topic extraction: the topic areas that the data corresponds to.
- Entity/concept extraction: the entities identified in the data.
- Sentiment extraction: the sentiments expressed in the data.
- Suggestion mining: reviewers’ recommendations for the brand expressing ways to improve.
- Speech transcription: converts pieces of speech (audio) to text.
- Machine translation: translates texts from Czech, Spain, German, Italian into English.

To integrate all the different modules in the platform, and let the platform be also a toolbox in which the services can be used independently, the modules in the platform will be composed of modular services (hence modules) that are also independently usable. To further facilitate the integration, all of those services will have a REST interface and will return a JSON as output.

A common format for data integration has been defined. This format is JSON-LD (JSON for Linked Data) which is an implementation of NIF for JSON. More details can be found on Appendix A.

Regarding the integration and deployment technique used for the services, there will be two kind of services:

- Docker services: these services will be more closely integrated in the Big Data Platform. More details can be found on section 3.4.
- External REST Services. Another approach to provide the functionality of a module is by means of a REST service. In this case, the MixedEmotions platform must provide the mechanism to call this service and receive its response to the requested analysis of the data.

### 3.1.2 Ingestion

These modules are responsible for collecting data from the internet. In the MixedEmotions platform, ingestion modules are the following ones:

- Twitter crawler
- YouTube crawler

#### 3.1.2.1 Twitter Crawler

The Twitter crawler collects tweets according to the selected keywords. In MixedEmotions, this component is developed in Python and can be deployed as a web service. As such, and because the limitation for retrieving tweets lies in the limited quota of the Twitter API and not in the processing capabilities, this element is to be deployed as a single instance.

#### 3.1.2.2 YouTube Crawler

The YouTube crawler downloads YouTube videos along with comments and other information attached to videos. In MixedEmotions, this component is developed in Python. Because of the limitation of the Google API, this module is to be deployed as a single instance.
3.1.3. Graph Analysis: Social Semantic knowledge graph and network analysis

The functionality that provides the social semantic knowledge graph in the MixedEmotions platform will be given through a set of processes which perform the following operations:

- On the knowledge graph side, they extract useful pieces of data from major datasets (e.g. Wikidata, Dbpedia). Then, those are transformed using various approaches like interlinking or ontology matching into a format that can then be used for enhancing the queries and functionalities.

- On the social network side, a social graph will be constructed based on the information extracted from social networks. In addition to the plain data extracted from the network, the graph will be enriched with user and content “metrics”, which give further insight on social context. These metrics include calculated parameters such as “centrality”, “betweenness” and “influence”. Figure 5 depicts the architecture of the modules for social network analysis in the platform.

![Social Network Analysis Architecture](image)

Figure 5. Social Network Analysis architecture

3.1.4. Analytics and Visualization

In order to be able to appreciate and optimize the results based on the use of the advanced linguistic technologies it is fundamental to represent and visualize the processed data. It was thus very important to have a visualization tool included in the platform. To this end, the platform includes a state of the art streaming relational analytics system, Kibi, developed as a fork of Kibana, a visualization tool for Elasticsearch. Kibi improves Kibana by including the capabilities of representing and using Knowledge graphs for data filtering.
3.1.5. Pipeline orchestrator

The pipeline orchestrator will be the element responsible for getting the configuration of the MixedEmotions platform from the user and running the software according to the specified requirements. It must take into account the order of the modules in the platform.

The MixedEmotions platform includes an example pipeline orchestrator, that can be used as is or as a starting point for developing a custom orchestrator. This orchestrator reads configuration files in a specified format and can work with REST services or with Docker services deployed with Marathon in a Mesos cluster. In order to do so, it will ask Mesos-DNS about where those services are and will perform some balancing.

This orchestrator can be used also in Standalone mode, where Docker services are deployed by themselves. When doing so, they can be accessed as a regular REST service, the only requirement is to know their ip and port.

3.2. MixedEmotions Platform as a Toolbox

In order for the MixedEmotions platform to catch the interest of potential users, it is important to reach a wide audience. For that purpose, it was a main objective of the architecture to be in the form of a flexible “toolbox” instead of a rigid platform. This way, the final user will be able to select only the software modules that are suited for its project needs, omitting the rest. For example some users might be really interested in processing tweets, thus needing the Twitter Crawler, the text processing modules and maybe the Kibi visualization tool, whereas others might want to use the Emotion Extraction from audio module, depending on their objectives.

Taking this approach further, and even though the Big Data aspect of the platform is fundamental to the project, there is also the option to offer the functionalities of MixedEmotions without the Big Data capabilities, by running them in a single machine. In this scenario, that we named “stand alone mode”, the user will only need to have a Docker server so that he can download and use the Docker images of the desired modules from the platform. It could for example be useful for them to test the suitability of some modules in a project or to test the viability of custom modules.

This demonstrates the flexibility and versatility of the platform, shaped by the constraints imposed by such a multi-partnership project. This was made possible thanks to the integration of the MixedEmotions platform with external REST services and thanks the flexibility of the container technology.

The consortium successfully tested this approach in the hackathon sessions in Galway (23-30 June 2016), as it was impossible to provide attendants with a cluster of machines each.
3.3. MixedEmotions as a Big Data platform

3.3.1 Microservices architecture

The MixedEmotions platform includes several modules for language processing. The main requirements were for the modules to be flexible enough, thus be able to be written in different coding languages and developed by different companies. Also, the platform had to be able to handle Big Data processing. A microservice architecture was thus chosen as the best approach to address those two main issues.

Microservices are independent services with a single purpose and they communicate via some network interface, usually using containers technology. However, for this kind of architecture to work, more components have to be present. First, there needs to be a pipeline orchestrator that makes the corresponding calls to the services. It should function as a manager, that launches the services, checks their status and relaunches them if they are down. This manager may rely on a resource manager, which combines the resources available from all the machines and offers them to the manager, thus reducing the complexity the manager has to deal with. It is also indispensable to have a discovery service that informs the pipeline orchestrator of where each service is deployed. This kind of architecture can also have a log aggregation service and a distributed configuration service. That will not be necessary in this platform, because the service modules do not share configurations such as common databases credentials.

Microservices are usually deployed using containerization. Containerization is also known as lightweight virtualization and basically consists in packaging every service with a very barebone operative system. Then, those packages, known as images, are deployed as containers, that are very small virtual machines. Containers scale greatly, as many of them can be deployed in a single machine or in many machines as needed. Finally, the containers are isolated from each other, which means components will never experience dependencies incompatibilities.

To implement the containers, Docker was chosen as the best technology to use. Among the container technologies is the most lightweight and the easiest to use. It is indeed the most widespread tool of its kind at the moment, which will benefit the future adoption of the platform.

The microservice architecture needs other components to integrate those microservices. The main ones are usually discovery, orchestration, configuration and logging. Many “stacks” of technologies are currently available. For this project we will be using part of the Apache Mesos stack, mainly Mesos, Marathon and Mesos-DNS, Zookeeper, as they are open source, widely used and in the knowhow of Paradigma.
3.3.2. MixedEmotions Services as Docker containers

The most important part of the MixedEmotions platform are the processing modules. There are many of them, each one processing a different task, and having its own characteristics and requirements. Therefore, a great challenge of the MixedEmotions platform was to integrate all of them together. The usual chosen option is the integration of each library. However, with a Big Data architecture such an approach would involve a high effort to adapt all the libraries to the platform. After analysing the resources needed to integrate just a small sample of services, it was evident that the classical approach was not appropriate for the integration of these services.

For the integration of the platform modules, Docker has been chosen. First, it allows to encapsulate each service in its own so-called “image”, thus removing the need of adding dependencies to the platform for each service as well as preventing potential conflicts. In addition, Docker, when combined with Mesos, allows for a more consistent level of parallelism, which will be explained in the next section. Another advantage of having Docker images is that
processing modules are easily detachable, so that the toolbox aspect of the platform is further reinforced.

In order to enable the integration of the services, the only requirement is that Docker containers have to expose a REST application. Furthermore, to facilitate the integration, the services will respond using JSON-LD as explained in appendix A.

3.3.3. Mesos as MixedEmotions resource manager

Mesos is an open source resource manager that, in a few words, enables the use of a machine cluster as a single machine, as it distributes memory and CPU load evenly among the machines of the cluster. It is the foundation of many PaaS (Platform as a Service) offerings. Mesos makes “offerings” of CPU cycles and RAM memory and automatically balances load between the servers of the cluster.

In the MixedEmotions platform, Mesos will handle the execution of the different services, using Marathon as a container manager and Mesos-DNS as discovery service. Additionally, having Mesos will let the users use Spark, a big data processing framework, if they will to do so.

3.3.4. Marathon as MixedEmotions containers manager

Marathon is another tool that is used for executing services in Mesos. With Marathon, the user just has to define the number of instances, memory and cpu allocated for a service. Marathon will use Mesos to execute those services in the cluster, using the machines that have more resources. Also, if one of the instances fails, Marathon will automatically start a new instance, making for an environment very fault-tolerant. In case of heavy load, the user can scale up the number of services using the Marathon interface.

Marathon also integrates seamlessly with Docker. The Marathon service configuration allows to use a Docker image URL in a repository as definition of a service. If that is the case, Marathon will download the image and start the corresponding number of instance. There is only one limitation of this approach, and that is that it is not possible to know the ip and port of the service instance beforehand.

3.3.5. Mesos-DNS as MixedEmotions discovery service

For overcoming that gap, the MixedEmotions Platform uses Mesos-DNS, which is a service that can be used as a DNS and a REST service that returns the ip and port of the service instances, as well as serving as a simple balancer. Mesos-DNS is also executed as a Marathon service.

Therefore, the execution of a MixedEmotions Platform pipeline will consist of an execution of the Spark Orchestrator, which is running in Spark. This will ask Mesos-DNS for the ip and port of an instance of the service to be executed whenever an item is to be processed. Then the main
orchestrator will send the item to the corresponding service, which is running in Mesos and orchestrated by Marathon.

### 3.3.6. MixedEmotions pipeline orchestrator

Finally, there is the need to have some component that uses all the services deployed in the microservices architecture. That will be the pipeline orchestrator.

In most cases, the final user will be developing their own orchestrator according to their needs. However, for demonstration purposes, a “default” pipeline orchestrator will be provided.

These orchestrators will not need to delve into the nuances of the Microservice architecture, as they only execute REST requests to the corresponding microservices and parse the results. However, in order to do that, they need the capacity to make queries to Mesos-DNS, the discovery service.
4. MixedEmotions Modules

The core of the MixedEmotions platform are the MixedEmotions modules. These are the “tools” that comprise the MixedEmotions Toolbox. Many of them are focused on extracting emotions from text, audio or video images, although there are many other capabilities, such as sentiment extraction, entity linking or linked data extractors.

Regarding their distribution, MixedEmotions Modules can be Open Source or Proprietary. Open Source modules are distributed in the MixedEmotions GitHub repository and in the majority of occasions also available as a Docker image in the Docker repository. Proprietary modules have a commercial license that can be purchased.

In these lists, a schematic representation of the modules capabilities is made for each module. A more detailed explanation for each module can be found in the MixedEmotions project repository in GitHub https://github.com/MixedEmotions/MixedEmotions or in previous deliverable D3.2.

4.1 MixedEmotions Platform Open Source modules

Table 1 includes the list, description and deployment strategy for the MixedEmotions open source modules. The modality defines the type of input the module needs in order to work. The integration column defines whether the module is available in Docker Hub or only on GitHub. Modules GitHub sources are available on https://github.com/MixedEmotions, and Docker images in Docker Hub on https://hub.docker.com/u/mixedemotions/.

<table>
<thead>
<tr>
<th>Id</th>
<th>Functionality</th>
<th>Language</th>
<th>Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>Sentiment Extraction from Text</td>
<td></td>
<td>Docker image</td>
</tr>
<tr>
<td>m2</td>
<td>Sentiment Extraction from Text</td>
<td></td>
<td>Docker image</td>
</tr>
<tr>
<td>Meme</td>
<td>Function</td>
<td>Image</td>
<td>Language</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>m4</td>
<td>Sentiment Extraction from Text</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="en" alt="English" /></td>
</tr>
<tr>
<td>m5</td>
<td>Emotion recognition from Text</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="en" alt="English" /></td>
</tr>
<tr>
<td>m6</td>
<td>Emotion recognition from Audio</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="en" alt="English" /></td>
</tr>
<tr>
<td>m7</td>
<td>Emotion recognition from Audio</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="en" alt="English" /></td>
</tr>
<tr>
<td>m8</td>
<td>Entity Extraction</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="es" alt="Spanish" /></td>
</tr>
<tr>
<td>m10</td>
<td>Entity Extraction and Linking</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="en" alt="English" /></td>
</tr>
<tr>
<td>m13</td>
<td>Topic Extraction</td>
<td><img src="#" alt="Diagram" /></td>
<td><img src="es" alt="Spanish" /></td>
</tr>
<tr>
<td>Module</td>
<td>Description</td>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>m16</td>
<td>Suggestion mining</td>
<td>Docker image</td>
<td></td>
</tr>
<tr>
<td>m20</td>
<td>Twitter media crawler</td>
<td>GitHub source</td>
<td></td>
</tr>
<tr>
<td>m21</td>
<td>Fusion</td>
<td>Docker image</td>
<td></td>
</tr>
<tr>
<td>m22</td>
<td>Social Network Analysis</td>
<td>Docker image</td>
<td></td>
</tr>
<tr>
<td>m25</td>
<td>Social semantic Knowledge graph</td>
<td>Docker Image</td>
<td></td>
</tr>
<tr>
<td>m27</td>
<td>Emotion recognition from Video</td>
<td>Service</td>
<td></td>
</tr>
<tr>
<td>m28</td>
<td>Analytics module “Kibi”</td>
<td>Standalone, GitHub source</td>
<td></td>
</tr>
</tbody>
</table>
Table 1. List of modules included in the MixedEmotions platform.

<table>
<thead>
<tr>
<th>Module</th>
<th>GitHub Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Youtube crawler</td>
<td></td>
</tr>
<tr>
<td>Orchestrator</td>
<td></td>
</tr>
</tbody>
</table>

m1. English Sentiment Extraction
This module predicts the polarity for the given sentence. The module uses Neural Network based classification architectures, Long Short Term Memory and Convolutional Neural Networks. The predicted polarity belong to the classes “positive”, “negative” and “neutral”. The module is written in Python 3.5 and uses machine learning libraries like scikit-learn and Keras.

m2. Czech and English Sentiment Extraction
The sentiment analysis module from BUT uses the Stanford CoreNLP, the LingPipe and the VADER Sentiment Analysis tools. It utilizes various classification algorithms including SVM (Support Vector Machine) and Naive Bayes classifier.

The module performs sentiment analysis of tweets as well as general texts in English and Czech language. Having a suitable dataset, models for different domains and languages can be simply trained.

Since most of the tools are implemented in Java, the module is written in Java as well.

m4. Spanish and English Sentiment Extraction
Spanish and English Sentiment Extraction written in python, using the Senpy framework¹. As any other Senpy application, the module consists of the senpy core server and a set of plugins that provide the logic for each analysis. There is an official public docker image for the senpy server², and several images for different sets of plugins. The dockerfile and image for the open source Senpy plugins are available³. The proprietary plugins have been integrated in other private images built using a similar method and cannot be released on the Docker Hub due to licensing issues.

m5. English Emotion recognition
Categorical emotion recognition module written in Java using the SVM classifier from the Weka machine learning framework. Two models are provided trained on different data sets: annotated news headline data from SemEval 2007 Task 14 (Affective Text) [2] and the Twitter Emotion Corpus [3] consisting of tweets with emotion hash tags as noisy labels [2]. The models produce

¹ [http://github.com/gsi-upm/senpy](http://github.com/gsi-upm/senpy)
² [https://hub.docker.com/r/gsiupm/senpy/](https://hub.docker.com/r/gsiupm/senpy/)
³ [https://github.com/gsi-upm/senpy-plugins-community](https://github.com/gsi-upm/senpy-plugins-community)
a single emotion label from Ekman’s six basic emotions (joy, fear, disgust, anger, surprise, sadness) for each submitted text. It is currently available via a RESTful web-service, with the intention to create a docker image in the near future.

m6. Audio Emotion recognition

Audio emotion recognition module written in C++ and Java, through RESTful web-service. The recognition system is based on Bag-of-Audio-Word models for arousal and valence [1] trained on RECOLA french database. Moreover, this module can be used for other languages since audio signals are not restricted to certain languages.

This module has the capability to integrate an automatic speech recognition module, in order to segment audio data into sentence-level chunks and recognise emotions for each sentence. To embed an external ASR you should create three .sh files inside platform_audio_er folder:

- **asr_upload.sh**: with the first parameter denoting the location of the audio file and the output should have the following fields in the form of:
  ```json
  {
  "result": {
    "info": {
      "name": "FILENAME.wav"
    }
  }
  ```

- **asr_run.sh**: with the first parameters as FILENAME.wav. The output should have the following fields:
  ```json
  {
  "result": {
    "info": {
      "id": "ID",
      "state": "waiting"
    }
  }
  ```

- **checkpending.sh**: with the parameter ID and returns:
  ```json
  {
  "result": {
    "info": {
      "id": "ID",
      "state": "waiting"
    }
  }
  ```
  OR
  ```json
  {
  "result": {
    "info": {
      "id": "ID",
      "state": "finished"
    }
  }
  ```

Once ASR completed its task by running again ‘asr_run.sh FILENAME.wav’ the final output is in the form of:

```json
{
  "result": {
    "one_best_result": {
      "segmentation": [
        {
          "start": 1600000, "end": 4200000, "word": "<s>"},
        {
          "start": 4200000, "end": 11100000, "word": "I"},
        {
          "start": 11100000, "end": 14000000, "word": "AM"},
        {
          "start": 14000000, "end": 19100000, "word": "FINE"},
        {
          "start": 19100000, "end": 22800000, "word": "</s>"
        ]
      }
    }
  }
}
```

Where `<s>` denotes ‘silence’, </s> is dot (.), and start/end are 0.0000001s → 1600000 = 0.16s

m7. Spanish and English Emotion recognition

Python module for emotion extraction from text, which exposes a REST/NIF API via the senpy framework⁴ framework. Users may choose between two different algorithms to calculate emotions. The first one uses a combination of the ANEW lexicon for recognition (using the <Valence, Arousal, Dominance> model), and a mapping to Ekman’s six emotion categories using pre-calculated centroids. The second algorithm relies on the WordNet-Affect lexicon to annotate text with one of Ekman’s categories. The module is highly modular and configurable, which allows for the addition of new languages to the module by adding WordNets in other languages. Machine translated WordNet lexicons are available for all 23 official European languages providing emotion recognition capabilities for those languages with the WordNet-

⁴ [http://github.com/gsi-upm/senpy](http://github.com/gsi-upm/senpy)
Affect based algorithm. Some algorithms have a commercial license and will be deployed in a standalone service and not included in the Docker Hub image. The open source code can be found at: https://github.com/gsi-upm/senpy.

**m8. Spanish Entity Extraction**
Entity extraction module written in Python. Exposes a REST interface which returns an array with the detected entities. The entities are based on titles from wikipedia articles in 2012. The user is expected to change or to expand those entities regarding their use case.

**m10. English Entity Extraction**
Java entity extraction module written in Java 8. It requires a trained model to identify the named entities, and an indexed over Wikipedia lexicons to perform entity disambiguation. Lexicons are extracted from Wikipedia snapshot from Oct, 2014. This module is provided as a Jar as well as in docker. It takes a text input in Json and returns a Json file which contains extracted entities, their types and Wikipedia link.

**m13. Spanish Topic Extraction**
Spanish Topic extraction implemented in Scala Spark. It uses a manual taxonomy that is to be defined by the final user. Returns a list of topics detected.

**m16. Suggestion mining**
This module checks if the given sentence contains a suggestion. The module needs a model data that has to be distributed on every node. In the background, this module uses neural network based classifiers which classifies each sentence of a given text into *suggestion* and *non-suggestion* classes.

**m20. Twitter media crawler**
The Twitter media crawler is a Python application that gets data from Twitter via streaming API. It returns public statuses that match one or more keywords specified by user. The default access level allows up to 400 track keywords. The data is stored in JSON format and can be accessed via API.

There are two types of API requests that can be used to get the data files, the interval and the timestamp. The interval service provides the data corresponding to the interval between two given timestamps and the timestamp service provides the data newer than the given timestamp.

To use this module a new Twitter account has to be registered and the credentials has to be provided to the application. See more details in the module manual.

**m21. Fusion**
Multimodal analysis which combines the results of emotion/sentiment recognizers from video, audio and transcription to yield higher recognition performance.
m22. Social Network Analysis
This module processes social network data, enriching it with contextual information (as described in D4.9 and D4.10). It includes a data crawler.
The service is deployed as a set of Docker containers linked together. In particular, there is: 1) a scanner container, 2) a celery container with scanner tasks, 3) a redis container that acts as a message broker between the previous two containers, and 4) an orientdb container.
The source code can be found at: https://github.com/gsi-upm/scaner.

m25. Social semantic Knowledge Graph
This application is meant to be used as a last step after a combination of prior modules. After running the entity extraction module on the data, the recognized entities are passed to this module. Each entity will be matched to DBpedia, and its type (Person, Organisation, Event, Location) will be extracted alongside with expanded information surrounding it. All of this will eventually be gathered and combined under the shape of a knowledge graph, given as output.

m27. Video Emotion recognition
This module extracts per-frame emotion information from video, regardless of language. Additionally it provides estimated age, gender, rough gaze direction, and unique person identities within a video.

m31. Analytics module “Kibi”
Kibi is a visualization tool based on Kibana that is a full-fledged project. Its core components are open source, although it also has an enterprise proprietary edition with more functionalities. Kibi has been developed to provide cross Structured and Unstructured data analytics and discovery. In this way, one can browse and discover patterns both in the unstructured text and in the emotional or other information which have been extracted algorithmically or provided by connected/linked data. The source for the open source components of this project can be found at: https://github.com/sirensolutions/kibi. More information is available at https://siren.solutions/kibi/.

m32. YouTube Crawler
The YouTube Crawler is a Python application that collects information about activities and comments belonging to YouTube videos. The application can download video and audio data as well. The module utilizes the YouTube Data API to download the data. The data is stored as JSON files.

The module has two modes of operation. The first one allows users to download all information from a particular YouTube channel and the second one is a keyword based search mode. The search mode can search for all videos related to a keyword given. The module is also able to update the previously downloaded data (e.g. comments or likes).
Downloading of data from YouTube is limited to 50,000,000 requests per day (operation write needs over 50 operations, read 10-20 operations). Due to the YouTube Data API restrictions, the search mode can retrieve the data for 500 videos in maximum at once.

To use this module a new Google account has to be registered and the credentials has to be provided to the application. See more details in the module manual.
5. MixedEmotions Platform proprietary modules

As stated earlier, it is important that the MixedEmotions platform is as versatile as possible. To that end, the MixedEmotions platform permits the use of external services to substitute the modules it has, for example, if the user has its own sentiment analysis tool.

The integration depends on the orchestrators. For the default orchestrator, the requirement is that the external services expose a REST. Usually, that interface that admits a POST containing a JSON-LD and returns a JSON-LD adding new processed data (as defined in Appendix A).

Taking advantage of this platform feature, some proprietary modules have been developed in conjunction with the platform and are going to be used in the pilots. They are to be included in the platform for the users willing to purchase their license, as they have been used along the open source modules. These modules will have professional support and are currently used in the platform. More information about these can be found in https://github.com/MixedEmotions/MixedEmotions.

<table>
<thead>
<tr>
<th>Id</th>
<th>Functionality</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>m3</td>
<td>Sentiment Extraction from Text</td>
<td><img src="https://flagicons.com/png/fs.png" alt="English" /> <img src="https://flagicons.com/png/it.png" alt="Italian" /></td>
</tr>
<tr>
<td>m9</td>
<td>Entity Extraction</td>
<td><img src="https://flagicons.com/png/fs.png" alt="English" /> <img src="https://flagicons.com/png/it.png" alt="Italian" /></td>
</tr>
<tr>
<td>m11</td>
<td>Topic Extraction</td>
<td><img src="https://flagicons.com/png/fs.png" alt="English" /> <img src="https://flagicons.com/png/it.png" alt="Italian" /></td>
</tr>
<tr>
<td>m12</td>
<td>Topic Extraction</td>
<td><img src="https://flagicons.com/png/fs.png" alt="English" /></td>
</tr>
</tbody>
</table>
### m3. Italian and English Sentiment Extraction

External service provided with a REST interface. More information can be found in [https://developer.cogitoapi.com/docs](https://developer.cogitoapi.com/docs).
**m9. Italian and English Entity Extraction**
External entity extraction module. Accessible via REST interface. More information can be found in [https://developer.cogitoapi.com/docs](https://developer.cogitoapi.com/docs).

**m11. External Italian and English Topic Extraction**

**m12. External English Topic Extraction**
Topic extraction named Saffron, written in Java. It is based on the use of an automatically constructed domain model to measure the coherence of a candidate term within a domain. See more in [2]

**m15. External Italian and English Entity Linking**

**m17. Speech to text**

Speech Transcription converts speech signals into plain text. After speech to text conversion, the text can be easily read, edited, searched, processed by text-based data mining tools or archived. This model is trained for English and is accessible through REST interface.

The engine is specially optimized for noisy and colloquial speech. It is based on state-of-the-art techniques for acoustic modeling, including discriminative training, neural network-based features, and speaker and channel adaptation techniques. It is compatible with a wide range of audio sources: GSM/CDMA, 3G, and VoIP-based, Live Broadcast and Recorded speech with emphasis on spontaneous telephony speech. It contains dictionary with 50 thousand words and on our internal test set reaches 72% Word Error Rate.

Phonexia REST server is written in C++ and based on open-source library Poco. It uses asynchronous processing, because the system is relatively slow (it process 100 seconds of audio in 60 seconds of CPU processing time). The system itself is scalable and right now it is set to use 4 cores.

This module can be used in conjunction with the m6 module to separate the video into sentences.
Contact [info@phonexia.com](mailto:info@phonexia.com) for more information about this.

**m18. Machine translation**
This is a module that can translate from a variety of languages to English. The target languages are German, Italian, Spanish and Czech.

The translation module takes as input a text in a source language and translates it into a target language. Currently, the module covers translations between English and several foreign languages, i.e. German, Spanish, Italian and Czech. The translation models, necessary to translate textual information, were trained on more than 10 million parallel sentences from different domains, e.g. Europarl (European commission talks), DGT (translation memories of European union law) or Subtitles (subtitles of movies).

The module, which is called through a REST service, uses the widely used phrase based statistical machine translation methodology within the MOSES\(^5\) decoder.

Contact http://nlp.insight-centre.org/contact/ for more information on this.

m23. Audio Emotion extraction
Frame level emotion extraction from bottleneck features from audio in English, German and Czech. Contact info@phonexia.com for more information.

m24. Recommendation engine
This module recommends videos, using data extracted from comments from users. Provided that both user behaviour data as well as feedback on given recommendations can be collected, the RE is expected to give quickly improving results with the help of automatic machine learning mechanisms.

**Input** (ID A/V, ID USER)
**Output** ( list of lists of ID A/V recommended)
Contact http://www.expertsyst.com/ for more info,

m28. Age estimation from audio
This module estimates the age from an audio, regardless of language. Contact info@phonexia.com for more information.

m29. Gender identification from audio
This module identifies the gender of a speaker, regardless of language. Contact info@phonexia.com for more information.

\(^{5}\) http://www.statmt.org/moses/
7. MixedEmotions Use cases

7.1 MixedEmotions Capabilities: Example use cases

Here are some example use cases to show the potential capabilities of the MixedEmotions platform. The first three are the pilots developed during the MixedEmotions project lifespan, however these are just some example applications, as the MixedEmotions platform is a very versatile platform with many possible applications.

7.1.1 Pilot 1: Social TV

Pilot 1 provides a real-time event monitoring in particular (but not only) in the context of the broadcast industry. The business case will provide a real-time monitoring support tool for journalists and anchor men which will give them an additional instrument for the management of long interviews and talk shows by regularly checking sentiments and emotions of trending topics on social media platforms.

Additionally, a second scenario for Social TV was defined that enables users to access a recommendation system based on emotion analysis provided by MixedEmotions. The Recommendation Engine (RE) is based on:
1. the assumption that TV users do not necessarily look for content with similar emotional characteristics.
2. the mood management theory: TV users are in principle hedonistic and aim for optimising their mood.
3. a dual process theory of media entertainment:
   a. Hedonistic gratification: Enjoyment (a purely pleasurable experience) → “joyful”
   b. Eudaimonic gratification: Appreciation (a meaningful/valuable experience associated with mixed emotions) → “intriguing”

Modules usage.
Pilot 1 exploits the following modules:
1. External CogitoAPI service, gathering together M3, M9, M11 and M15. Pilot 1 analyses video teasers and rss feeds for extracting entities and topics, that are used for searching tweets.
2. M4 and M7 for analysing tweets.
3. External Recommendation Engine, M24, for recommending videos.
4. Social Semantic Knowledge Graph, M25, for exploring additional information regarding extracted entities (by M9 and M15).
5. Emotion extractors M23 and M27, for analysing audio and images of a video.

7.1.2 Pilot 2: Brand Reputation Management
The objective of the Brand Reputation Management is to obtain information about the position of a certain brand. For that objective, this pilot will obtain data for that brand in some social media and newsfeeds and analyse it. The analysis will include sentiment extraction, emotion extraction, suggestion mining, concept extraction and topic extraction. The results is presented in a web dashboard.

As for the data extraction, the pilot use the Twitter media crawler (m20) and the Youtube crawler (m32). The video analysis needs then the Emotion Recognition from video (m27) and the Emotion Recognition from audio (m6) modules. In combination to the latter, the Speech to Text module (m17), a proprietary one, is accessed as a service.

Concerning the processing of texts, the pilot makes use of the Sentiment Extraction (m4) and the Emotion extraction (m7) modules based on Senpy, although it uses the service version that hosts the proprietary algorithms. Also, to get the most information off the text, it uses the following modules: Spanish Entity extraction (m8), the English Entity Linking (m10), English and Spanish Topic detection (m12, m13) and Suggestion mining (m16).

The pilot also analyzes the network of retrieved tweets using the Social Network Analysis Scanner (m22) to trace influential users and emotion propagation. The pilot uses the Social Semantic Knowledge Graph module (m25) as well, in order to extract more insight from the data.

Everything is finally analyzed and put together using the visualization tool of the platform Kibi (m31).

### 7.1.3 Pilot 3: Call Center

The objective of this Pilot is to perform acoustic emotion recognition thanks to the platform. The obtained data helps with rating of recordings in Contact Centers. Together with other parameters that were already in use (especially cross-talks, speech speed, speaker turn count, key-word spotting for emotion words), it helps to detect problematic parts in recordings and scripts or identify challenging topics.

Functionalities, which can help identify emotions in the speech (recordings) and analyze these parts are important for Contact Centers because these are the key moments of unsuccessful calls. Supervisors / quality managers could use the information to improve results and increase success (weaker agents in this area; verbal expressions and phrases that cause these emotions; also the phrases which are useful; in a real time notice for agents about incorrect process which will help them handle the situation and so on).

This pilot currently works with audio data and mainly uses Emotion Recognition from audio modules (m6, m23) and Speech to Text module (m17). Experimentation are still ongoing on using the Sentiment Extraction modules (m1, m2, m4) on top of Speech to Text module. In
addition, information from Age and Gender recognition modules (m29, m30) is also used to condition other outputs, to offer extra analysis dependent on these modalities.
8. Conclusions

This document presents the final version of the MixedEmotions platform. The document includes the open source distribution of the platform, the modules that compose the toolbox and the overall platform architecture.

The MixedEmotions platform is offered as an open source platform supported by NUIG after the end of the project. Most of the modules are open source and available in GitHub and Docker Hub.

Regarding the architecture, the document presents the microservice choice made in the platform. This kind of architecture provided the platform the flexibility to be used as a toolbox, in which users could select which modules to use, but also provides the platform Big Data processing capabilities. The microservice architecture was selected after assessing the requirements of the platform. It had to integrate several modules from several partners, written in different programming languages while being an open source big data platform. Adapting to these restrictions, the MixedEmotions platform evolved into a much more robust platform, one that can be used as a toolbox.

To draw a final conclusion, the MixedEmotions project is now coming to an end, and the platform is released. However, the platform use is not over. Partners are interested in continuing in using and developing the platform. Also, being an open source platform supported by an academic partner, and taking into account its characteristics aforementioned, the platform could evolve more and more thanks to the open source community.
APPENDIX A: JSON-LD
COMMON FORMAT FOR INPUT AND OUTPUT FOR THE PLATFORM MODULES

Introduction

As MixedEmotions is designed as a platform that enables interchangeable modules and even integrates with external modules, it is imperative to define a standard exchange format. For this purpose a MixedEmotions schema using JSON-LD has been defined. (More information about JSON-LD [here](#)). The complete MixedEmotions schema definition is available in its own description page, [here](#).

**NIF (NLP Interchange Format)** defines a vocabulary and an API for NLP services. The key concepts to understand are:

- All the text analysed is a String. To every String is given a URI (unique identifier)
- All strings belong to a Context
- Strings may have have attributes such as: entities, sentiment, lemma...

The NIF specification also defines how the URI should be computed. In a nutshell, URIs look like:

`http://example.org#char=0,40`
(In blue is the URI of the Context)
(In yellow is the index of the String within the context)

A NIF document would look similar to this example:

```html
<http://example.org#char=0,40>
    rdf:type nif:RFC5147String , nif:Context ;
    nif:beginIndex "0" ;
    nif:endIndex "40" ;
    nif:isString "My favourite actress is Natalie Portman."
```

In principle, NIF has been created with RDF in mind. The example above uses the turtle notation. A more developer-friendly alternative would be JSON. Or, rather, **JSON-LD**. JSON-LD documents are JSON documents with some conventions/constraints on the structure and fields they contain, which are used to add semantics to the document.

Using JSON-LD, the NIF example we had before would look as follow:
There are different ways to structure the same data in JSON-LD. The final scheme that we follow in our API is different, as it includes much more information about each context, as well as the sentiment/emotion analysis processes that produced the results.

**MixedEmotions Schema**

Here is the defined MixedEmotions schema.

```json
{
  "@context": {
    "nif": "http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#",
    "topics": {
      "@id": "dc:subject"
    },
    "entities": {
      "@id": "me:hasEntities"
    },
    "suggestions": {
      "@id": "me:hasSuggestions"
    },
    "emotions": {
      "@id": "onyx:hasEmotionSet"
    },
    "sentiments": {
      "@id": "marl:hasOpinion"
    },
    "entries": {
      "@id": "prov:used"
    },
    "analysis": {
      "@id": "prov:wasGeneratedBy"
    },
    "dc": "http://dublincore.org/2012/06/14/dcelements#",
    "me": "http://www.mixedemotions-project.eu/ns/model#",
    "prov": "http://www.w3.org/ns/prov#",
    "nif": "http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#"
  }
}
```

We give here a small explanation of each field.

- **Entries**: The sentence or sentences to be analyzed. Regarding this project, entries will always consist of a single text, but this schema supports multiple entries.
- **Analysis**: parameters and other data of the analysis that has been performed on the entries.
• **Topics**: topics detected in the entry.
• **Entities**: entities detected in the entry.
• **Suggestions**: whether or not this entry contains suggestions.
• **Emotions**: emotions detected in this entry.

The namespaces define the ontologies to be used when filling those fields. Most are standard, but the consortium created a new one for the cases where the existing ones did not suit the project needs.

**Examples**

These are more concrete examples of the MixedEmotions' schema, each focusing in a particular field.

**Entry**

This example covers the basic example in the NIF documentation: [http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html](http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html).

```
{
   "@context": "http://mixedemotions-project.eu/ns/context.jsonld",
   "@id": "http://example.com#NIFExample",
   "analysis": [],
   "entries": [
      {
         "@id": "http://example.org#char=0,40",
         "@type": [
            "nif:RFC5147String",
            "nif:Context"
         ],
         "nif:beginIndex": 0,
         "nif:endIndex": 40,
         "nif:isString": "My favourite actress is Natalie Portman"
      }
   ]
}
```

**Sentiment Analysis**

```
{
   "@context": "http://mixedemotions-project.eu/ns/context.jsonld",
   "@id": "me:Result1",
   "analysis": [],
   "entries": [
      {
         "@id": "me:SAnalysis1",
         "@type": "marl:SentimentAnalysis",
         "marl:maxPolarityValue": 1,
         "marl:minPolarityValue": 0
      }
   ]
}
```

```
"nif:RFC5147String", "nif:Context"
], "nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome.

"entities": [ ], "suggestions": [ ], "sentiments": [ ]

"@id": "http://micro.blog/status1#char=80,97", "nif:beginIndex": 80, "nif:endIndex": 97, "nif:anchorOf": "You'll be awesome.", "marl:hasPolarity": "marl:Positive", "marl:polarityValue": 0.9, "prov:wasGeneratedBy": "me:SAnalysis1"
]

"emotionSets": [ ]
]
]
}

Suggestion Mining

{
"@context": "http://mixedemotions-project.eu/ns/context.jsonld", "@id": "me:Result1", "analysis": [ ]

"@id": "me:SgAnalysis1", "@type": "me:SuggestionAnalysis"
], "entries": [ ]

"@id": "http://micro.blog/status1", "@type": [ ]

"prov:wasGeneratedBy": "me:SAnalysis1", "nif:RFC5147String", "nif:Context"

"nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome.

"entities": [ ], "suggestions": [ ]

"@id": "http://micro.blog/status1#char=16,77", "nif:beginIndex": 16, "nif:endIndex": 77, "nif:anchorOf": "put your Windows Phone on your newest #open technology program"
]

"sentiments": [ ], "emotionSets": [ ]
]
**Emotions**

```json
{
    "@context": "http://mixedemotions-project.eu/ns/context.jsonld",
    "@id": "me:Result1",
    "analysis": {
        "@id": "me:EmotionAnalysis1",
        "@type": "me:SuggestionAnalysis"
    }
}
```

```
"nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome."
```

```
"nif:anchorOf": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome."
```

```
"prov:wasGeneratedBy": "me:EAnalysis1",
"onyx:hasEmotion": {
    "@id": "http://micro.blog/status1#char=0,109",
    "onyx:hasEmotionCategory": "wna:liking",
    "onyx:hasEmotionCategory": "wna:excitement"
}
```

**Named Entity Recognition**

```json
{
    "@context": "http://mixedemotions-project.eu/ns/context.jsonld",
    "@id": "me:Result1",
    "analysis": {
        "@id": "me:NER1",
        "@type": "me:NER"
    }
}
```

```
"nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome."
```

```
"nif:anchorOf": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome."
```

```
"prov:wasGeneratedBy": "me:EAnalysis1",
"onyx:hasEmotion": {
    "@id": "http://micro.blog/status1#char=0,109",
    "onyx:hasEmotionCategory": "wna:liking",
    "onyx:hasEmotionCategory": "wna:excitement"
}
```

```
"prov:wasGeneratedBy": "me:EAnalysis1",
"onyx:hasEmotion": {
    "@id": "http://micro.blog/status1#char=0,109",
    "onyx:hasEmotionCategory": "wna:liking",
    "onyx:hasEmotionCategory": "wna:excitement"
}
```
```
"prov:wasGeneratedBy": "me:EAnalysis1",
"onyx:hasEmotion": {
    "@id": "http://micro.blog/status1#char=0,109",
    "onyx:hasEmotionCategory": "wna:liking",
    "onyx:hasEmotionCategory": "wna:excitement"
}
```
"nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome. #opensource",

"entities": {
"@id": "http://micro.blog/status1#char=5,13",
"nif:beginIndex": 5,
"nif:endIndex": 13,
"nif:anchorOf": "Microsoft",
"me:references": "http://dbpedia.org/page/Microsoft",
"prov:wasGeneratedBy": "me:NER1"
},

"@id": "http://micro.blog/status1#char=25,37",
"nif:beginIndex": 25,
"nif:endIndex": 37,
"nif:anchorOf": "Windows Phone",
"me:references": "http://dbpedia.org/page/Windows_Phone",
"prov:wasGeneratedBy": "me:NER1"
},

"suggestions": [],
"sentiments": [],
"emotionSets": []
}

Complete example
This example covers all of the above cases, integrating all the annotations in the same document.

{
"@context": "http://mixedemotions-project.eu/ns/context.jsonld",
"@id": "me:Result1",
"analysis": {
"@id": "me:SAnalysis1",
"@type": "marl:SentimentAnalysis",
"marl:maxPolarityValue": 1,
"marl:minPolarityValue": 0
},

"@id": "me:SgAnalysis1",
"@type": "me:SuggestionAnalysis"
},

"@id": "me:EmotionAnalysis1",
"@type": "me:SuggestionAnalysis"
},

"@id": "me:NER1",
"@type": "me:NER"
"entries": [{
  "@id": "http://micro.blog/status1",
  "@type": ["nif:RFC5147String", "nif:Context"],
  "nif:isString": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome. #opensource",
  "entities": [
    {
      "@id": "http://micro.blog/status1#char=5,13",
      "nif:beginIndex": 5,
      "nif:endIndex": 13,
      "nif:anchorOf": "Microsoft",
      "me:references": "http://dbpedia.org/page/Microsoft",
      "prov:wasGeneratedBy": "me:NER1"
    },
    {
      "@id": "http://micro.blog/status1#char=25,37",
      "nif:beginIndex": 25,
      "nif:endIndex": 37,
      "nif:anchorOf": "Windows Phone",
      "me:references": "http://dbpedia.org/page/Windows_Phone",
      "prov:wasGeneratedBy": "me:NER1"
    }
  ],
  "suggestions": [],
  "sentiments": [],
  "emotions": []
},
{
  "@id": "http://micro.blog/status1#char=0,109",
  "nif:anchorOf": "Dear Microsoft, put your Windows Phone on your newest #open technology program. You'll be awesome. #opensource",
  "prov:wasGeneratedBy": "me:EAnalysis1",
  "onyx:hasEmotion": []
}]}
{ "onyx:hasEmotionCategory": "wna:liking"
},
{
"onyx:hasEmotionCategory": "wna:excitement"
}
APPENDIX B: DOCKERIZATION

Docker in a nutshell

Docker is a framework that permits to encapsulate services within a "container", so that services can be deployed with all its dependencies and in any environment.

You can find more information about docker on their official website https://www.docker.com

Installation

Please refer to: https://docs.docker.com/engine/installation/

How to create a Docker image

There are two ways of creating a Docker image: by submitting an existing container into an image or by creating a Dockerfile.

Creating a Dockerfile

For creating images, the best way is to have a Dockerfile which is a file that defines how an image will be constructed. The Dockerfile should be named “Dockerfile” and should be located at the root of the project. A best practise would also be to include the Dockerfile in the GitHub repository. The file basically defines how to construct the desired image. It starts from another available Docker image as a base (defined in the FROM instruction), to which some commands are applied in order to form the final image.

A quick example:

```
FROM python:3-onbuild
MAINTAINER Carlos Navarro cnavarro@paradigmadigital.com
ADD src/ /usr/src/app
WORKDIR /usr/src/app
RUN "pip install tornado"
CMD ["python3", "topic_service.py", "2712"]
```
EXPOSE 2712

In this example, we suppose that there is some python app under the src subfolder that will be started with the command “python3 topic_service.py 2712” and which listens on the port 2712. It needs a dependency to be installed beforehand with “pip install tornado”. These are the most basic commands:

- FROM: base image which will be used as a starting point for building the image. More details can be found here: https://hub.docker.com/explore/. Some examples are: python:3-onbuild, java:7 and java:8
- MAINTAINER: author
- ADD: copies some local folder to the image folder
- RUN: executes a command
- WORKDIR: changes directory
- CMD: commands to execute when running an instance
- EXPOSE: opens port (in which the service should be running)

Therefore, the basic steps for creating a Dockerfile are:
1. Find a suitable base image
2. Add a MAINTAINER line
3. Add the source files with ADD
4. Install missing dependencies using RUN
5. State the process to be run in CMD
6. Declare the port to be accessed with EXPOSE

Building images

Once Docker is installed, images can be built:

```bash
docker build -t {name} {basefolder}
```

For example: `docker build -t topic_image`.

Then the list of available images can be obtained with:

```bash
docker images
```

If deleting an image is needed, that is done with:

```bash
docker rmi {imagename}
```

Another way of creating an image is to do it from an existing container.
Run containers
Once the image is built, one can start to create and run containers. There are many ways to do this. We give here two quick examples of commands.

To run a daemonized container with ports open:

```
docker run --name suggestion_container -d -P suggestion_image
```

To run a terminal inside the container (very useful when developing the future image):

```
docker run --name suggestion_container -t -i suggestion_image /bin/bash
```

Of course, multiple containers of a single image can be run, but each with its own unique name.

The running container can be shown using the command ‘`docker ps`’, and all containers with ‘`docker ps -a`’.

The port mapping can be found with this command:

```
0.0.0.0:32768->2712/tcp
```

Stopping and deleting images is done through the commands `docker stop {container_name}` and `docker rm {container_name}`

Publishing images

Once an image is created, it can be pushed into an online repository such as Docker Registry or Docker Hub (which are free and public). From there, images can be retrieved from anywhere.

To that aim, the image first has to be properly labeled, indicating the Docker repository, name and possibly the version:

```
docker tag topic_image:1.0.0 mixedemotions/topic_image:1.0.0
```

Then it is pushed into the appropriate repository (which has to be created beforehand):

```
docker push mixedemotions/topic_image
```

From there, the image can be downloaded using (if the tag “latest is created):

```
docker pull mixedemotions/topic_image
```
MixedEmotions platform regarding Docker Hub

A repository for MixedEmotions’ Docker images has been created in hub.docker.com/r/mixedemotions

Please note that the repository is public, and is aimed to, in order to allow the general public to use the platform. However it is possible to put a disclaimer with the specific license of the module in the description.

Regarding naming convention, here is the format suggested:

mixedemotions/id_modulename_partner

Thus, in order to upload certain image to the repository, the following example commands can be used:

docker tag topic_service mixedemotions/13_topic_service_pt
docker push mixedemotions/13_topic_service_pt

Regarding versioning, it is very strongly recommended to have a tag for each past stable version. It is mandatory to have a "latest" for the current version. The following lines allows to create a version tag:

docker tag topic_service mixedemotions/13_topic_service_pt:0.1.0
docker push mixedemotions/13_topic_service_pt:0.1.0

Once the repository is created we suggest the following structure:

Short description: One sentence description of the module
Full description: License, detailed description and how to use the service within the container.

References

References:


Emojis used from the Noto project: https://www.google.com/get/noto/help/emoji/smiley-people.html