MixedEmotions Technical Webinar

Ian Wood[1], Carlos Navarro[2], Hesam Sagha[3] , John McCrae[3]

26th January, 2017

[1] National University of Galway, Ireland

[2] Paradigma Digital, Madrid, Spain

[3] University of Passau, Germany

1. Preparation

These instructions are for using the MixedEmotions platform in a single machine.

1.1 Download some examples

https://drive.google.com/open?id=0B40OwoRRV6rCMGZzRVRtWFBac28

1.2. Docker Server

Docker Server has to be installed. Go to www.docker.com for instructions on installation. The Windows version is less mature and has not been tested, we advise installing in a virtual machine running linux if you will be working with a Windows computer.

1.3. Docker Images

Docker Images for MixedEmotions can be found in the MixedEmotions' dockerhub: https://hub.docker.com/u/mixedemotions/ .

To download the docker images, run the docker pull command. Note that the downloads are substantial and this may take a while! Note that, depending on how docker is installed, you may not need to use "*sudo*" with these commands. To verify that the downloads completed successfully, use the command "*docker images*".

Some examples:

```
sudo docker pull mixedemotions/08_entity_extraction_pt
sudo docker pull mixedemotions/13_topic_image_pt
sudo docker pull mixedemotions/10_entity_linking_nuig
sudo docker pull mixedemotions/tut_emotion_lexicon_nuig
sudo docker pull mixedemotions/06_audioanalysis_up
```

1.4. Optional: Elasticsearch and Kibi

For an easier assessment of the results, it can be a good idea to download Elasticsearch (https://www.elastic.co/products/elasticsearch) and Kibi, a Kibana fork developed also for the MixedEmotions toolbox (https://siren.solutions/kibi/)

2. MixedEmotions examples

The examples below showcase the MixedEmotions data analysis platform and a selection of data analysis modules. These modules exist either as docker images (these run locally) or as REST web services (these can be accessed via a browser or with command line utilities such as *curl*).

Since there are several docker modules demonstrated, it is a good idea to stop the docker containers that are not in use. The command "*docker ps"* lists the running docker containers and their docker ids (the 16 digit codes). Each container can be stopped with the command "*docker stop <id>"* where you replace *<id>* with the 16 digit id code.

2.0. The MixedEmotions Platform (Mini-Orchestrator)

There are some configuration files alongside a compiled jar here:
https://drive.google.com/open?id=0B40OwoRRV6rCMGZzRVRtWFBac28

Uncompress this folder and enter it.

This mini-orchestrator demonstration uses one docker service and two REST services. First launch the used docker containers (look sections 3.2.1 and 3.2.1 for mor references) :

```
sudo docker pull mixedemotions/08_entity_extraction_pt
sudo docker run -p 32769:2812 --name entities_pt -d mixedemotions/08_entity_extraction_pt
sudo docker pull mixedemotions/13_topic_extraction_spanish
sudo docker run -p 32770:2712 --name topic_pt -d mixedemotions/13_topic_image_pt
```

To run a demonstration of the single computer demonstration version of the platform, change to the folder *semanticsTutorial/* and run the following command:

```
java -cp  MixedEmotionsExampleOrchestrator-assembly-0.15.jar
orchestrator.ListFutureOrchestrator conf/textPipelineUPM_PT.conf
input/textPipeline_UPM_PT.txt
```

Results will be stored in *output/* directory (as indicated in the "conf/textPipelineUPM_PT.conf" file). If you are running *elasticsearch*, the results will also go to the elasticsearch index – you should see them at "*localhost:9200/_search"*. Results stored in elasticsearch can be visualized with kibana at "*localhost:5601"*. Alternatively, you can view the generated json output (the file "*output/textPipeline.txt"*) using an online json viewer such as http://jsonviewer.stack.hu/.

See the extra handout "MixedEmotions Mini Orchestrator" for details on configuration file and input file syntax.

2.1. Emotion Recognition From Text

2.1.1. Emotion Extraction

We will be recognising emotion in text using two REST web services. The first web service for emotion recognition from text uses a supervised emotion model (SVM) trained on tweets containing emotion hash tags. This model currently has not been ported to the MixedEmotions JSON-LD input and output formats. Open the following link in a browser or use *curl*:

```
http://140.203.155.226:8080/emotion-api/api/emotion/classify?text=What_happy_times!&type=tweet
```

The output is:

```
JOY
```

Another SVM model, trained on SemEval 2007 task 14 data (emotion annotated news headlines) can be accessed with "*type=text*", however it's performance is poor due to little training data (~2000 headlines) and mediocre annotator agreement.

The second REST service also has a web interface demonstrating emotion detection alongside other emotion and sentiment detection models. These emotion models are lexicon based.

The following call extracts emotions with both a continuous dimensional representation as well as categorical for the spanish text "esto es horrible". Open the following link in a browser or use *curl*:

```
http://senpy.cluster.gsi.dit.upm.es/api/?i=esto%20es%20horrible&algo=EmoTextANEW&lang=es
```

The output uses the MixedEmotions JSON-LD format, including the original text, analysis results and provenance information such as the parameters used in the analysis. Near the end you will find:

```
"entries": [
  {
    "@id": "Entry_1473434739.79",
    "emotions": [
      {
        "@id": "Emotions0",
        "onyx:hasEmotion": [
          {
            "@id": "Emotion0",
```

```
            "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#arousal": 5.75,
            "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#dominance": 3.04,
            "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#valence": 2.28,
            "onyx:hasEmotionCategory": "http://gsi.dit.upm.es/ontologies/wnaffect/ns#sadness"
          }
        ]
      }
    ],
    "language": "en",
    "nif:isString": "esto es horrible"
  }
]
```


2.1.2. Sentiment Extraction

This call utilises the same REST interface to extract sentiment from the spanish text “no es impresionante”. Open the following link in a browser or use *curl*:

http://senpy.cluster.gsi.dit.upm.es/api/?i=No%20es%20impresionante

The output here is again in the MixedEmotions JSON-LD format. Towards the end you will find:


```
"entries": [
  {
    "@id": "Entry0",
    "nif_isString": "No es impresionante",
    "sentiments": [
      {
        "@id": "Opinion0",
        "marl:hasPolarity": "marl:Negative",
        "marl:polarityValue": "-1",
        "prov:wasGeneratedBy": "sentiText"
      }
    ]
  }
]
```


2.1.3. Suggestion Mining

The suggestion mining module is currently available as a REST service. A web interface can be accessed for experimentation, or you can try the REST service directly. Open the following link in a browser or use *curl*:

http://140.203.155.226:8080/miso/rest/suggest/getSuggestions?json={"text":["This is a fabulous hotel. The breakfasts are great - fresh fruit bagels, muffins, hot eggs and sausage etc. Note, the room can only accommodate two people who are close. Do not

```
expect your family of four to be comfortable in one room. I highly recommend the fabulous little Italian restaurant just around the corner from the hotel, Bon Amici. I will stay at this hotel everytime I come to New York."],"textSource":"general","language":"en"}
```

The output is:

```
["I highly recommend the fabulous little Italian restaurant just around the corner from the hotel, Bon Amici."]
```

2.2. Linked data

2.2.1. Entity Extraction

Running the container:

```
sudo docker pull mixedemotions/08_entity_extraction_pt
sudo docker run -p 32769:2812 --name entities_pt -d mixedemotions/08_entity_extraction_pt
```

Check that the container “entities_pt” is running:

```
sudo docker ps
```

You should see something like:

```
CONTAINER ID     IMAGE                                                COMMAND   CREATED
STATUS
fb204d4de6db        mixedemotions/08_entity_extraction_pt   "python3  4 weeks ago         Up 2
minutes

PORTS                     NAMES
0.0.0.0:32769->2812/tcp   entities_pt
```

And example of concept extraction for the text “Cristiano Ronaldo seguirá cuatro años más en el real madrid”

```
http://localhost:32769/?text=Cristiano%20ronaldo%20seguir%C3%A1%20cuatro%20a%C3%B1os%20m%C3%A1s%20en%20el%20real%20madrid
```

You should be receiving something like:

```
{
        concepts:
        [
                "real madrid",
                "cristiano ronaldo"
        ]
}
```

2.2.2. Topic Extraction

Running the container:

```
sudo docker pull mixedemotions/13_topic_extraction_spanish
sudo docker run -p 32770:2712 --name topic_pt -d mixedemotions/13_topic_image_pt
```

Check that the container is running for the following example:

text = El jefe del BBVA Francisco González ha destacado que la nueva estructura de la red comercial BBVA le permitirá seguir presente en el 100% de las poblaciones de más de 8.000 habitantes

http://localhost:32770/?text=El%20jefe%20del%20BBVA%20Francisco%20Gonz%C3%A1lez%20ha%20destacado%20que%20la%20nueva%20estructura%20de%20la%20red%20comercial%20BBVA%20le%20permitir%C3%A1%20seguir%20presente%20en%20el%20100%%20de%20las%20poblaciones%20de%20m%C3%A1s%20de%208.000%20habitantes

2.2.3. Entity Linking

Running the container:

```
sudo docker pull mixedemotions/10_entity_linking_nuig
sudo docker run -p 8888:5000 -d --name nuig_el
mixedemotions/10_entity_linking_nuig
```

Sample Document:

```
{
    "@id": "http://example.com#MyExample",
    "entries": [{
        "@id": "myexample",
        "nif:isString": "Daley Blind has enjoyed a fine start to life under Jose
Mourinho"
```

```
    }]
}
```

Save as test2.json

Using the Service:

```
curl -X POST -d @test2.json http://localhost:8888
```

Expected Result:


```
{"analysis":[{"@id":"me:EntityLinking","@type":"me:NERAnalysis"}],
"entries":[{"entities":[{"score":38.0, "@id":"myexample#char=0,11",
"nif:beginIndex":0, "nif:endIndex":11, "nif:anchorOf":"Daley Blind",
"me:references":"http://dbpedia.org/page/Daley_Blind",
"prov:wasGeneratedBy":"me:EntityLinking"}, {"score":1.0,
"@id":"myexample#char=51,64", "nif:beginIndex":51, "nif:endIndex":64,
"nif:anchorOf":"Jose Mourinho",
"me:references":"http://dbpedia.org/page/José_Mourinho",
"prov:wasGeneratedBy":"me:EntityLinking"}], "suggestions":[], "sentiments":[],
"emotionSets":[], "@id":myexample, "@type":["nif:RFC5147String","nif:Context"]}],
"@context":"http://mixedemotions-project.eu/ns/context.jsonld",
"@id":"http://example.com#MyExample"}
```


2.2.4. English Concept Extraction

```
http://140.203.155.226:8080/entitylinking-api/api/nel/linking?text=Brad_Pitt_is_a_Hollywood_actor
```

The output is:


```
[
{"EntityText":"Brad
Pitt","EntityType":"PERSON","URI":"http://dbpedia.org/page/Brad_Pitt","score":1457.0},
{"EntityText":"Hollywood","EntityType":"LOCATION","URI":"http://dbpedia.org/page/Hollywood","score
":9052.0}
]
```


2.2.6. Emotion WordNet Lexicon

Running the container:

```
sudo docker pull mixedemotions/tut_emotion_lexicon_nuig
sudo docker run -p 8890:8890 -p 1111:1111 -d --name emo-virt
```

```
mixedemotions/tut_emotion_lexicon_nuig
```

Access the lexicon from http://localhost:8890/sparql/

2.3. Emotion extraction from audio

2.3.1. Paralinguistic feature extraction from audio

```
sudo docker pull mixedemotions/06_audioanalysis_up
sudo docker run -p 32768:8080 --name audioservice -d mixedemotions/06_audioanalysis_up
```

Check with “sudo docker ps” if the service “audioservice” is running. If it is something similar to this should be shown:

```
CONTAINER ID  IMAGE                                 COMMAND             CREATED
STATUS
451855f2c91f  mixedemotions/06_audioanalysis_up  "catalina.sh run"   4 weeks ago
Up 5 seconds

PORTS                           NAMES
0.0.0.0:32768->8080/tcp   audioservice
```

Test that the service is running properly with

```
curl
"http://localhost:32768/er/aer/getdims?dims=arousal,valence,gender,age&url=http://tv-download.dw.com/dwtv_video/flv/wikoe/wikoe20151114_wiruebli_sd_avc.mp4&timing=20,30" -o audioout.json
```

where:

- dims: dimensions to be extracted {arousal,valence,gender,age}
- url: the url of the video
- timing: time slots to extract the features {start1,end1;start2,end2;...}

And then check that the content is a json document like this:

```
[{
    "@context": [{"emovoc":
"http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/emotionml/ns"}],
    "entries": [{"emotions": [{
        "@id": "Entry0#time=20,30",
        "prov:wasGeneratedBy": "me:SpeechAnalysis",
        "onyx:hasEmotion": [
            {
                "emovoc:arousal": 0.122,
```

```
                "confidence": 1
            },
            {
                "emovoc:valence": 0.061,
                "confidence": 1
            },
            {"gender": [
                {
                    "CLASS_PROB": 0,
                    "CLASS_NAME": "Male",
                    "CLASS_IDX": 0
                },
                {
                    "CLASS_PROB": 1,
                    "CLASS_NAME": "Female",
                    "CLASS_IDX": 1
                }
            ]},
            {
                "age": 22,
                "confidence": 1
            }
        ]
    }]}],
    "analysis": [{
        "@type": "onyx:EmotionAnalysis",
        "onyx:usesEmotionModel": "emovoc:pad-dimensions",
        "@id": "me:SpeechAnalysis"
    }]
}]
```


2.3.2.Usage

```
java -cp DockerSparkPipeline-assembly-0.5.jar orchestrator.FutureOrchestrator
conf/audioemotionProject.conf input/audioemotion.txt
```

Approximate run time: 5 mins

In order to POST a file or Upload a file through GUI, check the following link:

http://localhost:32768/er/aer

2.4. Video Processing

This module provides frame-based face analysis in videos. The capabilities are detection of age, gender, emotion, bounding box, gaze, and facial landmarks.

You can upload a file via:

http://pchradis.fit.vutbr.cz:9000

Note:

- Maximum video upload size is 60 MB
- Faces should to be fairly large - e.g. >96x96px

Examples of calls:

http://pchradis.fit.vutbr.cz:9000/status/10cbbdc3f5be43a9bd49949f37e588aa
http://pchradis.fit.vutbr.cz:9000/video/10cbbdc3f5be43a9bd49949f37e588aa
http://pchradis.fit.vutbr.cz:9000/results/10cbbdc3f5be43a9bd49949f37e588aa

The REST methods are:

[GET] / --- exemplar form for video file upload

[POST] /upload --- for video upload
Returns:
status - OK/FAIL
uuid - Identifier of the uploaded video used in the following calls. The results can only be accessed using this <uuid>.

[GET] /status/<uuid> --- provides information about the video processing state
When ['data']['status'] is "DONE", you can access results.

[GET] /video/<uuid>
Returns video with visualization of face detections, facial landmark localization, and estimation of gender, age, and expression.

[GET] /results/<uuid>
Returns analysis results in JSON format.
[frame_id][face_id]['action_units'] -- binary presence and "strength" of facial atcion units 1,2,4,5,6,7,9,10,12,14,15,17,20,23,25,26,45,28 - see https://www.cs.cmu.edu/~face/facs.htm
[frame_id][face_id]['age'] -- estimated age in years
[frame_id][face_id]['emotions'] -- probabilities of facial expression - the order is anger, disgust, fear, smile, sad, surprised, neutral
[frame_id][face_id]['gender'] -- estimated gender [female_prob, male_prob]
[frame_id][face_id]['bounding_box'] -- facial bounding box position in pixels
[frame_id][face_id]['gaze_left'] -- 3D gaze direction vector of left eye; [0,0,-1] is directly into camera
[frame_id][face_id]['gaze_right'] -- 3D gaze direction vector of right eye
[frame_id][face_id]['head_pose'] -- Head pose represented as rotation vector [rot)x, rot_y, rot_z] -- see https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula or http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#void Rodrigues(InputArray src, OutputArray dst, OutputArray jacobian)
[frame_id][face_id]['landmarks'] -- 68 facial landmarks [x,y] -- see https://cdn-images-1.medium.com/max/800/1*AbEg31EgkbXSQehuNJBlWg.png