Manual for Software Prototype of Social Media Analysis Tool to Monitor Public Attitudes towards Migration

Mehwish Alam & Yiyi Chen
FIZ-Karlsruhe
Haithem Afli
Munster Technological University

February/2022

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 882986".

Deliverable Factsheet	
Title and number	*Manual for Software Prototype of Social Media Analysis Tool to Monitor Public Attitudes towards Migration (D.5.3)*
Work Package	WP5
Submission date	February 2, 2021
Authors	Alam, Mehwish (FIZ-Karlsruhe); Chen, Yiyi (FIZ-Karlsruhe); Afli, Haithem (MTU)
Contributors	Shammary, Fouad (MTU); Kardkovacs, Zsolt (MTU)
Reviewers	Krueger, Finja (IfW); Pichierri, Francesca (FIZ-Karlsruhe)
Dissemination level	PU (Public)
Deliverable type	DEM (Demonstrator)

Version Log

Issue Date	Version	Author	Change
03/02/2022	v0.1	Mehwish Alam, Yiyi Chen, Haithem Afli	First version sent for review.
21/02/2022	v0.2	Mehwish Alam, Yiyi Chen, Haithem Afli	Revised version submitted.
22/02/2022	V0.3	Cristina Blasi	Coordinator final review and layout adjustments. Version ready to be submitted.

Executive Summary

The present deliverable describes the architecture and main functionalities of a sentiment analysis prototype for the ITFLOWS EUMigraTool (EMT) policy-making and sentiment monitoring module, which is described in Deliverable 9.4 - Key Exploitable Result No. 3. In this document, we discuss the technical details of how to use ITFLOWS Social Media Analysis Tool to monitor public attitudes towards migration. Scientific observations, conclusions, and further implications based on this work are widely discussed in Deliverable 5.4 (Alam et al. 2022).

The ITFLOWS project aims to deliver a comprehensive tool that covers the entire lifecycle of migrations management by providing an early warning system, a migration flow prediction engine, an integration process monitoring tool, and a policy-making and validation decision support system. Deliverable 5.3 is a deep learning backed sentiment analysis tool, and an evidence generator to monitor integration processes, and to develop or validate policy decision making.

Public opinion about migration is a key driving factor in migration-related policy making in every single EU Member State (Maruma 2016, NRC 2007). An analysis of the development of sentiment over time and across countries, complementing traditional data sources, is essential in several levels including but not limited to

- understanding how people think and affect others by expressing their opinions about migration related issues;
- observing how people's behaviour change or shift over time;
- predicting certain phenomena like intra-social tensions, conflicts or problems, which need public attention ahead of escalation;
- measuring effects of certain policies, regulations, strategies, communications, or integration techniques; and
- monitoring notions, common connotations, misconceptions or spread of misinformation, including hate speech.

In this document, Chapter 1 discusses the basic motivations, the foundations, used terms, notions, and considered alternatives to build ITFLOWS Social Media Analysis Tool. Scientific considerations are discussed in Deliverable 5.1 and in (Chen et al. 2021). Section 2 describes inputs and pre-requirements to use our model. Section 3 details how to use and apply the pre-trained deep learning model to deal with large amounts of

social media data. Section 4 explains how the software can be used in the case of multilingualism. Section 5 helps build a proper maintenance environment by introducing error handling methods for exceptions and errors, which may arise during the process. Concluding remarks regarding our model and our software architecture are summarised in Section 6.

Table of Contents

Abbreviations

AI: Artificial Intelligence

AMMI: Associazione Multietnica del Mediatori Interculturali

API: Application Programming Interface

BiGRU: Bidirectional Gated Recurrent Unit

BiLTSM: Bidirectional Long Short-Term Memory

BUL: Brunel University London

CA: Consortium Agreement

CEPS: Centre for European Policy Studies

CERTH: Ethniko Kentro Erevnas kai Technologikis Anaptyxis

CNN: Convolutional Neural Networks

CRI: Associazione della Croce Rossa Italiana

CSD: Centre for the Study of Democracy

CSO: Civil Society Organization

D<N>.<M>: Deliverable number #M of the ITFLOWS Project Work Package number #N

EASO: European Asylum Support Office

EC: European Commission

EMT: EUMigraTool

ESS: Exploitation Strategy Seminar

EU: European Union

FIZ: Fiz Karlsruhe–Leibniz-Institute fur Informationsinfrastruktur GMBH

FRONTEX: European Border and Coast Guard Agency

GA: Grant Agreement

GRU: Gated Recurrent Unit

ETM: Embedded Topic Model

FAIR: Findable, Accessible, Interoperable, Reusable

HRB: Horizon Results Boosters

HTML: HyperText Markup Language

IAI: Istituto Affari Internazionali

ICT: Infocommunication Technology (Information and Computer Technologies)

IFW: Institut für Weltwirtschaft

IMF: International Monetary Fund

IOM: International Organization for Migration

IPR:	Intellectual Property Rights
JRC:	Joint Research Centre
KB:	Knowledge Base
KER:	Key Exploitable Result
LTSM:	Long / Short-Term Memory
MGKB:	Migrations Knowledge Base
MTU:	Munster Technological University
NGO:	Non-Governmental Organization
OCC:	Open Cultural Center
ORE:	Open Research Europe
PWG:	Policy Working Group
RDF:	Resource Description Framework
RDFS:	Resource Description Framework Schema
TFEU:	Treaty on the Functioning of the European Union
TRC:	Terracom AE
TRL:	Technological Readiness Level
UAB:	Universitat Autònoma de Barcelona
UB:	Users Board
UN:	United Nations
UNHCR:	United Nations High Commissioner for Refugees
URL:	Universal Resource Locator
WP:	Work Package

1. Introduction

ITFLOWS generates novel insights on migration. The purpose of ITFLOWS is to provide accurate predictions and adequate management solutions of migration flow in the European Union in the phases of reception, relocation, settlement, and integration of migration, according to a wide range of human factors. These insights are provided by an evidence-based ICT enabled solution (the EUMigraTool) and precise models. All solutions have fitness for purpose continually validated by policy-makers and practitioners in cooperation with civil society organisations in a dynamic and iterative process.

The EUMigraTool is aimed to include analysis of media content from TV news (video content), web-news, and social media (text content) using deep learning and proposing novel deep architectures in generative modelling and forecasting using sequential data. Predictions incorporate algorithms that consider the two key challenges associated with the prediction of migration:

a) adequate selection of relevant data sources, and

b) correct selection of the potential drivers to be monitored and to the warning thresholds to be set.

ITFLOWS gathers, integrates, analyses, and presents data from various, mainly open data sources like Eurostat, FRONTEX, IMF, Google Trends Index, or Twitter, among others. Some of them provide hard facts about the past, some others are representing soft information about the now, e.g., how people, in general, perceive some phenomena by analysing anonym corpus of digital footprints: search patterns, and opinion statements. At this stage, we are analysing the text corpus using different points of view in order to determine linguistic expressions and contexts which have the most significant predictive power about future migrations. We also extract data that are identified as relevant indicators. The most important of them is the public opinion on migration-related fields or the "vox populi".

Thus, in this document, we fundamentally present our approach to how to extract relevant information on migration-related topics. We are using Twitter tweets (short messages), however, it can and it will be applied to other free format textual data sources (e.g., GDELT: Global Database of Events, Language, and Tone), with the exception of the entity linking module (as seen later). Hence, it decodes only Twitter-specific references.

1.1 Objective and Scope

The objective of this deliverable is to present the architecture and main functionalities of a sentiment analysis software prototype for the ITFLOWS EUMigraTool (EMT). It provides an overview of the state-of-the-art deep learning techniques used for the social media analysis required to monitor public attitudes towards migration. The deliverable also introduces MigrationsKB (MGKB), a Knowledge Base (KB) that contains the outcomes of this deliverable. This deliverable contains only the technical part to run our tools, scientific implications and background are discussed in Deliverable 5.4 (Alam et al. 2022).

ITFLOWS sentiment analysis architecture presented in this document extracts both semantical and contextual information. The latter is important because we are focusing on a very specific topic (migration). Yet, subject could be changed, and our implementation could be eventually used for other fields as well. In such case, only slight changes, fine-tuning and previously optimised settings would be required.

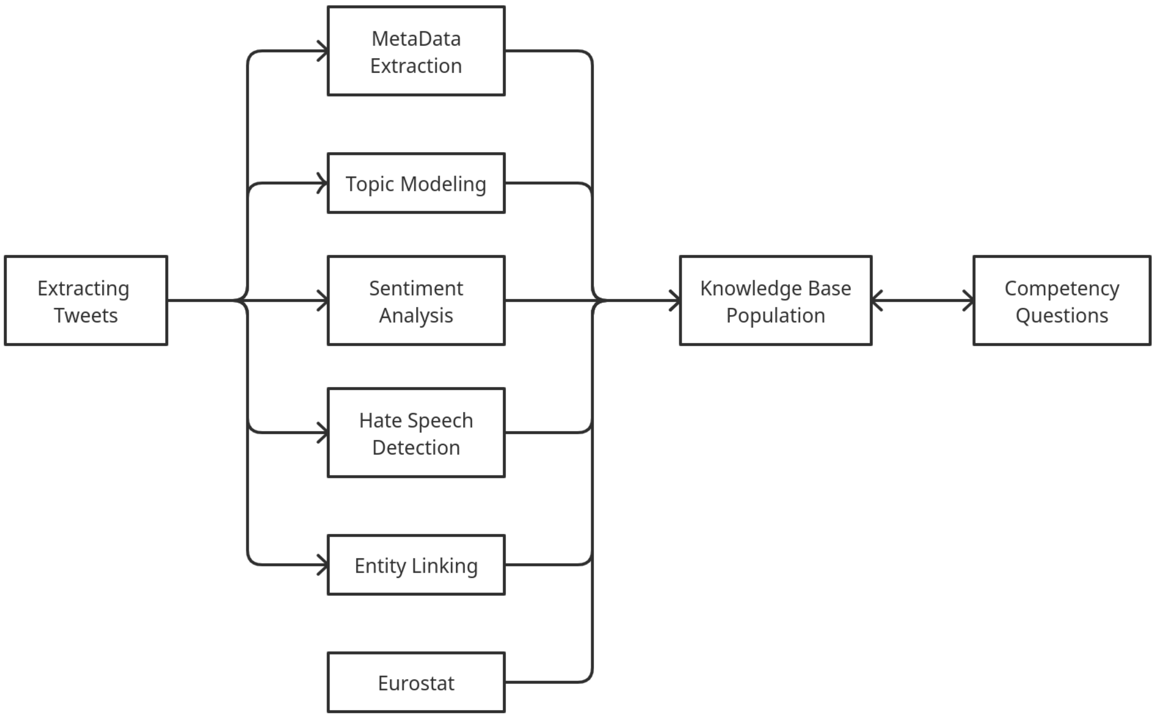

Figure 1 General Overview of the ITFLOWS Sentiment Analysis Toolkit

We extract the following elements of the Twitter corpus:

- **Factual information**
  - *metadata*: generic data on the text message itself, like tweet id, language, public metrics, geoinformation, date, user reference (Twitter user ids are

anonymised with UUID version 4 objects as specified in RCF 4122[1] from python library);[2]

  - *entity linking*: in-text references to other individuals, locations or companies.
- **Contextual information**
  - *topics*: what kind of migration-related keywords, events, the topic is discussed in tweets;
  - *sentiments*: high-level attitudes (positive, negative, neutral) perceived in tweets based on their use of, and order of the words;
  - *hate speech*: detection of extreme sentiment based on the use of certain keywords, (latent, mimicked or otherwise) offensive language.

Using these elements, we build a simple knowledge base called MigrationKB, which is applied as an input to a predictive module integrated into EUMigraTool. Key findings on our model can be found in Deliverable 5.4 (Alam et al. 2022).

Source code is organised as follows:

- **`crawler`:** contains the Tweet extraction utility;
- **`preprocessing`**: it has multiple purposes: It contains metadata extraction component, data formatters, converters, and text preprocessing routines;
- **`topic_modeling`**: implementation of the Topic Modelling component, which contains underlying training and model methods;
- **`sentiment_analysis`**: model and implementation of the sentiment analysis component;
- **`hate_speech_detection`**: Hate Speech Detection component's implementation;
- **`entity_linking`**: implementation of the Entity Linking component;
- **`data`**: contains the 3rd party data sources that we integrate into MigrationsKB.

1.2 Organisation of the Manual

The overview of our architecture is shown in Figure 1. Section 2 provides all the necessary details required to set up and start to use our solution. Section 3 discusses how to use the key components, namely:

[1] https://www.cryptosys.net/pki/uuid-rfc4122.html
[2] https://docs.python.org/3.8/library/uuid.html

- Sub-section 3.1 presents the necessary step to start the process and extract tweets if and when adequate academic credentials to access the Twitter corpus are setup.
- Sub-section 3.3 details how to use the topic modelling.
- Sub-section 3.4 presents entity linkage (dereferencing components) and its potential usage options.
- Sub-section 3.5 shows how general sentiments are presented in tweets from this corpus.
- Sub-section 3.6 summarises hate speech detection related settings and workarounds.
- Sub-section 3.7 introduces a novel factor analysis toolkit which integrates various data sources.
- Sub-section 3.8 discusses how to create the output MigrationKB.

Section 5 makes an introduction on how to use these resources in a multilingual environment, as well as what kinds of modifications shall be considered. Error handling is discussed in Section 6. These sections are followed by concluding remarks.

2. Getting Started

2.1 Availability

The software for performing sentiment analysis and hate speech detection is made available online open-source through GitHub for promoting the open-science initiative.

2.2 Download and Installation

For downloading the repository from Github, users can go to the command line prompt and type:

```
git clone https://github.com/migrationsKB/MGKB.git
```

A more intuitive way to download the repository is to press the button as shown in the image:

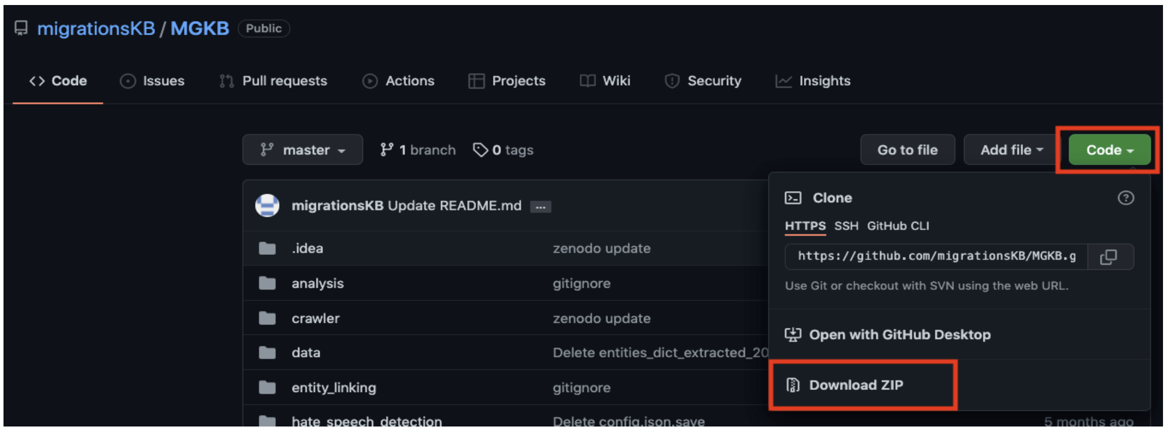

Figure 2 Easy way to download MGKB

This will create a folder “MGKB” in your current directory. In order to work with this directory, users can type the following command in the command prompt:

```
cd MGKB
```

Then users should install required libraries and packages:

Python version >=3.8.

1. Check the CUDA version, the Driver version, and find the compatible PyTorch version to install https://pytorch.org/get-started/previous-versions/

For the settings (CUDA version 10.1, Nvidia Driver Version: 418.39):

```
pip install torch==1.8.1+cu101 torchvision==0.9.1+cu101
torchaudio==0.8.1 -f
https://download.pytorch.org/whl/torch_stable.html
```

2. **Install other packages:** The requirements for installing external packages are given in the Requirement file.[3] In order to run the requirement file, users should run the following command:

```
pip install -r requirements.txt
```

2.3 File Structure

All modules described in Deliverable 5.4 (Alam et al. 2022) are contained in each of the folders in the downloaded repository. The sequence according to the pipeline are as follows and are described in detail in Section 3 for their usage:

```
crawler → preprocessing → topic_modeling → entity_linking →
sentiment_analysis → hate_speech_detection → populate_kb
```

Other folder includes `analysis, data, images, utils`. Below are the details of each of these files:

1. **analysis:** script/notebooks for analysing and generating plots for statistics of the results.

2. **data:**
   - `eurostat_stats` (files for statistics of asylum seekers and potential driving factors, which are explained in a README file);
   - `extracted` (the unique linked entities from entity linking, extracted and analysed hashtags for crawling tweets).

3. **images:** images generated analysing data and results from the models

[3] https://github.com/migrationsKB/MGKB/blob/master/requirements.txt

   - `correlations`: plots describing correlations between negative/hate speech and potential driving factors for all the countries;
   - `dist_plots`: plots describing distributions of the tweets before/after filtering, sentiments and hate speeches;
   - `etm_dist`: distribution of tweets with different numbers of topics.

4. **utils:** scripts for loading authentication file for Twitter API and other configuration files, helper functions for reading files.

3. Using the Application

This section describes in detail how to run the software application for each of the modules step by step. All commands are run in the root directory. Folder names are given at the end of the section headers in parenthesis.

3.1 Extracting Tweets and Metadata (crawler)

The first step is to crawl the tweets based on migration related keywords. In order to do so, an account of Twitter API for Academic Research is required. After obtaining the account, users should put the bearer token in the file: `crawler/config/credentials.yaml`. Create a folder data/raw.

And then run the following command:

```
python -m crawler.main_keywords
```

The outcome of this command is the crawled tweets in `gz` files as follows:

```
AT_20211220124316_2019-08-29T16:27:27.000Z.gz
```

- `AT`: country iso2 code
- `20211220124316`: crawling time
- `2019-08-29T16:27:27.000Z`: the minimal time of crawled tweets in this file, which will be used as a parameter for crawling.

3.2 Pre-processing (pre-processing)

Tweets obtained in the previous step need to be pre-processed before considering them as input to the next core modules. Perform the following steps:

3.2.1 Restructure crawled data

The input to this programme is the output from the crawler, i.e., raw data:

```
python -m preprocessing.restructure_data
```

This program gives processed data as a JSON file, as an output.

The dump from (3.1) crawling the tweets is structured as follows:

```
{'data': List(Dict),
 'includes': {'places': Dict, "media":Dict},
  'meta':Dict}
```

A dump is a dictionary represented as "data", which consists of a list of dictionaries of tweets and its metadata (represented as "includes") containing a dictionary of detailed Geoinformation for the tweets, and a list of parameters of media components (such as images) included in the tweets. The "meta" field includes the newest and oldest tweet id crawled in the dump. The detailed fields for each tweet are made available in the link https://github.com/migrationsKB/MGKB/tree/master/crawler/config/fields_expansions.[4] To prepare Geoinformation for each tweet, crawled Twitter data is restructured as follows:

```
{country_code: {
"data": Dict,
"places": Dict,
"media": Dict}
```

3.2.3 Convert Twitter data to panda DataFrames

The JSON file from the restructuring step is then converted to data frames using the following command:

```
python -m preprocessing.dict2df
```

This generates a ".csv" file. The generated file includes the following information for each tweet:

```
author_id, converstation_id, text, id, created_at, lang, long,
lat, hashtags, user_mentions, reply_count, like_count,
retweet_count, full_name, name, country, geo, country_code
```

[4] For further information about Fields in Twitter API: https://developer.twitter.com/en/docs/twitter-api/fields

- `author_id`: the unique identifier of the User who posted this Tweet. (e.g., `2244994945`);
- `conversation_id`: the tweet ID of the original Tweet of the conversation (which includes direct replies, replies of replies), which can be used to reconstruct the conversation from a tweet;
- `text`: the actual UTF-8 text of the tweet;
- `id`: the unique identifier of the requested tweet;
- `lang`: the language of the tweet, if detected by Twitter. (e.g., `de`);
- `lat`: the latitude of the geolocation tagged by the user;
- `long`: the longitude of the geolocation tagged by the user;
- `hashtags`: the string started with "#" in the text in this Tweet;
- `user_mentions`: Twitter usernames signalled with "@" in the text in the tweet;
- `reply_count`: the count of replies of the tweet, which shows the engagement metrics, tracked in an organic context, for the tweet at the time of the request;
- `created_at`: creation time of the tweet (e.g., `2019-06-04T23:12:08.000Z`);
- `like_count`: the count of likes of the tweet, which shows the engagement metrics, tracked in an organic context, for the tweet at the time of the request;
- `retweet_count`: the count of retweets of the Tweet, which is one of the engagement metrics, tracked in an organic context, for the Tweet at the time of the request;
- `full_name`: the full name of the geolocation tagged by the user (e.g., Koppl, Österreich);
- `name`: the name of the place of the geolocation tagged by the user (e.g., Koppl);
- `country`: the country name of the place of the geolocation tagged by the user (e.g., : Österreich);
- `geo`: including a bounding box of the geolocation tagged by the user (e.g., [16.310587, 48.101619, 16.366044, 48.137597]);
- `country_code`: the two-letter country codes defined in ISO 3166-1[5] of the geolocation tagged by the user.

[5] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

3.2.4 Tweet text pre-processing

Figure 1 above shows what components are using text pre-processing in our framework. The CSV file obtained from the previous step is then fed into the text pre-processing step:

```
python -m preprocessing.text_preprocessor
```

In this step, tweet text is cleaned and pre-processed by removing user mentions, reserved words (e.g., RT, FAV), emoticons, numbers, URLs, special HTML characters, punctuations, emojis and stop words, and lemmatised, and only the cleaned tweets with more than two tokens are reserved. The output of this step is a ".csv" file with an extra `preprocessed_text` following 3.2.3.

3.3 Topic Modelling (topic_modelling)

Topic Modelling serves as a way to filter out irrelevant tweets. In order to do so, an existing neural network based state-of-the-art model was used, i.e., Topic Modelling in Embedding spaces (ETM) (Dieng et al. 2020). The original source code for this algorithm is available open-source through GitHub link: https://github.com/adjidieng/ETM however, this implementation is also integrated into our repository.

The scripts and their respective parameters:

1) The script `topic_modeling.ETM.main` is used to train and evaluate ETM (the third and fourth step in this section). For training ETM, the default parameters[6] from (Dieng et al. 2020) are used, and the parameter `--num_topics` can be set accordingly, for our usage, it is either 25,50,75 or 100;

2) The script `topic_modeling.infer_topics` is used to predict topics for each tweet (the last step in this section), the following parameters should be set:

[6] https://github.com/migrationsKB/MGKB/blob/master/topic_modeling/ETM/main.py

`--model_path` the file path of the pretrained ETM model to be loaded for inferring topics;
`--data_path` the file path of the pre-processed tweets to be loaded for inferring topics;
`--batch_size` the batch size of the data to be split for fed into pretrained ETM;
`--num_words` the number of words to be shown for each topic;
`--num_topics` the number of topics of the pretrained ETM.

The **<u>first step</u>** is to convert pre-processed tweets from 3.2.4 into formats that can be fed into ETM:

```
python -m topic_modeling.ETM.data_build_tweets
```

With this step, tweets are split into train/validation/test (80%/5%/15%) datasets. The output of this step is split datasets into ".mat" files and vocabulary of the whole data in the ".pickle" file.

The **<u>second step</u>** is to pre-train skip-gram embeddings using the following command:

```
python -m topic_modeling.ETM.skipgram
```

The input to this step is the pre-processed text and the output is the word embeddings which are saved in the file: "`embeddings.txt`"

The **<u>third step</u>** is to train the ETM model using the following command, for example, train the ETM model with the number of topics 25:

```
python -m topic_modeling.ETM.main --num_topics 25
```

For our experiments, ETM was trained with the number of topics 25, 50, 75, 100. The output of this is the trained `ETM models`. With this step, for each number of topics, the best performing model is chosen based on the validation data.

The **fourth step** is to evaluate which ETM with the respective number of topics fits best for the Twitter data with the `test` dataset. In order to do so, users should run the following command with the best model for each number of topics:

```
python -m topic_modeling.ETM.main --num_topics XX --
load_from topic_modeling/ETM/models/etm_K_XX_XYZ --tc 1 -
-td 1 --mode eval
```

The output of this step is the test results for the ETM with the number of topics `XX`, which will give topic diversity, topic coherence, and the topic quality (the multiplication of the topic diversity and topic coherence). The higher topic quality is, the better. Based on these metrics, best ETM is selected for further usage.

The **fifth step** is to infer topics on all tweets with the best performing ETM. In order to do so, Twitter data needs to be converted into formats that can be fed into ETM (change the data path accordingly in the script `data_build_for_inferring_topics`) and run the following command:

```
python -m topic_modeling.ETM.data_build_for_inferring_topics
```

Then the **last step** is to infer topics for tweets. In order to do so, input the chosen best performing model path and the respective number of topics:

```
python infer_topics.py --model_path models/etm_K_XX_XYZ -
-num_topics XX
```

The output of this step is ".csv" file with extra prim_topic (the assigned topic of the tweet) from 3.2.4, and `topic2words.json`,[7] which contains the dictionary of the topic with the corresponding top topic words.

[7] https://github.com/migrationsKB/MGKB/tree/master/topic_modeling/topic_words, the relevant topics for migrations are [0,1,2,3,5,8,12,13,15,16,18,22,24,25, 29,32,34,36,41,42,45,47,48].

3.4. Entity Linking (entity_linking)

Entity linking (EL) (Wu et al. 2020) is the process of linking entity mentions appearing in web text with their corresponding entities in a knowledge base. For example, in the tweet “When young Polish immigrants arrived in the UK, they found the religious pluralism unsettling”, the mentions “Polish” and “UK” are linked to the Wikipedia entities https://en.wikipedia.org/wiki?curid=275297 https://en.wikipedia.org/wiki?curid=31717 respectively. This functionality allows searching tweets with the help of entity mentions (for further details, see Alam et al. 2022, Chen et al. 2021).

In order to perform entity linking, a neural network based state-of-the-art approach was employed, called BLINK. The software is open source and is available through the GitHub repository: https://github.com/facebookresearch/BLINK. The input to this model is the pre-processed tweets:

3.4.1 Set up Blink, and initialise the directory path `XXX` of the setup Blink in `entity_linking/main_linking.py` file, at line 33:

```
    models_path = XXX
```

3.4.2 Perform the entity linking for tweets, run the following command, given the `input_file` from the “.csv” output file of 3.3 and the `output_dir`:

```
    python -m entity_lining.tweet_entity_linking_all --
input_file XX --output_dir YY
```

The output is a dictionary of linked entities of tweets and is saved in the path “`data/extracted/entities_dict.json`”.

3.4.3 Merge the results from entity linking and results from sentiment analysis (3.5) and hate speech detection (3.6).

```
    python -m entitiy_linking.post_processing
```

3.5. Sentiment Analysis (sentiment_analysis)

In order to classify tweets according to their sentiment (e.g., positive, negative, and neutral), a neural network based approach takes into account the contextual information of the sentence, i.e., BERT was used for the classification task. In order to perform sentiment analysis of the collected tweets using transfer learning, the pre-trained BERT is fine-tuned on training dataset SemEval2017 (Rosenthal et al. 2017), Airline,[8] and Combined dataset from both, and evaluated using SemEval 2017 test dataset. Datasets are split by 80%/10%/10% for train/validation/test. The best performing BERT is chosen based on the evaluation and used for transfer learning to predict sentiment of the tweets.

3.5.1 The **<u>first step</u>** is to change the model path accordingly in the transfer_learning script line 42, for example,

```
    model_path = "bert_models/BERT_6.pth"
```

where the model "`BERT_6.pth`" is the best performing fine-tuned BERT.

3.5.2 The **<u>next step</u>** is to run the following command:

```
    python -m sentiment_analysis.transfer_learning
```

The output is the ".csv" file from 3.3 with an addition "`pred_sentiment`", which contains labels for predicted sentiments of each tweet.

3.6. Hate Speech Detection (hate_speech_detection)

To measure the negative attitude of the public towards migration in terms of hate speech, hate speech detection is performed. Tweets are classified into one of the following three classes: hate, offensive, and normal. In order to perform transfer learning in this section, all hate speech detection models are trained on recently published manually annotated data for hate speech detection, called as HateXplain (Mathew et al. 2021). The dataset is split by 80%/10%/10% into train/validation/test datasets. The model CNN+BiLSTM+Attn (Convolutional Neural Networks, Bi-directional

[8] https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Long Short Term Memory and Attention layer), as the best performing model, is selected to do transfer learning for predicting hate speeches of the crawled tweets:

3.6.1 The **first step** is to change the model path accordingly in the `hate_speech_detection.transfer_learning` script line 37, for example,

```
model_path = "hsd_models/model_epoch_1.pth"
```

where the model "`model_epoch_1.pth`" was trained and validated as the best performing hate speech detection model from CNN+BiLSTM+Attn, which is used for predicting hate speeches of the crawled tweets.

3.6.2 The **next step** is to run the following command:

```
python -m hate_speech_detection.transfer_learning
```

The output is the ".csv" file from 3.3 with an addition "`hatespeech_pred`", which contains the labels for predicted hate speeches of each tweet.

3.7. Analysis of the Factors Affecting Public Attitude towards Migrations

The dataset used for this purpose was Eurostat. All files used for this purpose are in CSV format and are available in the folder: "`data/eurostat_stats/csv`". The following statistics were used for this purpose:

1. Statistics of first-time asylum applications in EU and in the UK in 2020;[9]
2. **Real Gross Domestic Product Growth Rate** in EU.[10] These statistics were last updated on 12.05.21. UK data for the year 2020 is not available in Eurostat. The Real GDP Growth Rate for the UK is obtained from Statista,[11] which was last updated on 31.3.2021.

[9] https://asylumineurope.org/reports/country/united-kingdom/statistics/
[10] https://ec.europa.eu/eurostat/web/products-datasets/-/tec00115&lang=en
[11] https://bit.ly/3rcOPIe

3. **Total Unemployment Rate** in EU,[12] which was last updated on 13.04.21. The UK data for the year 2020 is not available in Eurostat so it was obtained from another resource[13] which was last updated on 26.01.2021.
4. **Youth Unemployment Rate** in the EU was taken from Eurostat,[14] which was last updated on 13.04.2021. UK data for the year 2020 is not available in Eurostat. The data was alternatively used from a different resource,[15] which was last updated on 20.04.2021.

The notebooks for analysis and generating plots of the correlation between the factors and the public attitudes towards migrations are in analysis folder (see section 2.2).

3.8. KB population (populate_kb)

In order to populate the KB for making it available for querying and analysis (see Alam et al. 2022, Chen et al. 2021), users should run the following command:

```
python 01_populate_kb.py
```

For generating the documentation of the RDF/S model, users should run the following command:

```
python 02_document_schema.py
```

[12] https://ec.europa.eu/eurostat/web/products-datasets/-/tps00203
[13] https://bit.ly/3GjlWOG
[14] https://ec.europa.eu/eurostat/web/products-datasets/-/tesem140
[15] https://bit.ly/340tI38

4. Multilingual MGKB

4.1 Availability

For downloading the repository from GitHub, users can go to the command line prompt and type:

```
git clone https://github.com/migrationsKB/MRL.git
```

Then they should download and installed as described in section 2.2 above.

4.2 File Structure

The sequence according to the pipeline is as follows and is described in detail in the subsequent sections in detail:

```
crawler → preprocessing → models → populate_kb
```

The util folder contains functions for loading and reading files. User should create a (1) folder output, which will contain output from the following steps, and (2) another folder datasets, which will contain all the external datasets used to fine-tune and evaluate models for language models in sentiment analysis and hate speech detection.

4.3 Using the Application

4.3.1 Crawling Tweets

This is similar to the description in section 3.1.[16] Crawled tweets stored in ".gz" files are stored in the folder `output/crawled`.

4.3.2 Pre-processing the tweets

1. Restructure the tweets and get statistics of curated data (similar to 3.2.4) by implementing the following commands:

```
python -m preprocessor.restructure_data
```

The input is the output from 4.3.1 and the output of this step is stored as ".json" files in `output/preprocessed/restructured`.

[16] The manually verified multilingual keywords for crawling is made available: https://github.com/migrationsKB/MRL/tree/main/crawler/keywords_generation/annotations/final

2. Pre-processing the crawled tweets for applying topic modelling, sentiment analysis, and hate speech detection with the following commands:

```
python -m preprocessor.preprocessing
```

The output of this step will be stored as ".csv" files in the folder `output/preprocessed/forTP`, with `preprocessed_text` for topic modeling and `preprocessed_cl` for sentiment analysis and hate speech detection.

The notebooks in the folder `pre-processing` are for pre-processing datasets from other sources (details are in D5.4, sections 4 and 5), which are used for fine-tuning and evaluating models for sentiment analysis and hate speech detection. After pre-processing the external datasets, the structure of the folder `datasets` is as follows:

```
datasets |- sentiment_analysis
          |- el/fi/hu/nl/pl/sv
               |- preprocessed
         |- hate_speech_detection
            |- dutch/finnish/french/german/greek/hungarian/
             italian/polish/spanish/swedish
               |-preprocessed
```

"/" separate subfolders, and each subfolder with language file name has a subfolder `preprocessed`, in which the pre-processed datasets split into train/validation/test are stored as ".csv" files. For fine-tuning the English hate speech detection model, the HateXplain dataset is used (Mathew et al. 2021).

4.3.3 Topic Modelling (ETM)

To train, validate, and evaluate ETM for each language, steps from Sub-section 3.3 should be followed. The input to ETM is the output of 4.3.2 (2), which is stored in the folder `output/results/ETM/{lang}` where `lang` is the language code of each language.

4.3.4 Fine-tune Language Models with Adapters

The `model.scripts.xlm-r-adapter` is used for fine-tuning language models by training adapters for both sentiment analysis and hate speech detection. The following parameters should be set:

`--lang_code` the language code of the dataset used for fine-tuning, which loads the datasets in respective languages according to the `task` (see file structure of datasets in sub-section 4.3.2), for sentiment analysis the `lang_code` could be set as: `el,fi, hu, nl, pl` or `sv`, for hate speech detection, `lang_code` could be set as: `dutch, finnish, french, german, greek, hungarian, italian, polish, spanish` or `swedish`.

`--task` the task is set to either `sa` (sentiment analysis) or `hsd` (hate speech detection).

`--checkpoint` the checkpoint is set to a model from huggingface[17], for our usage, the checkpoint is either `cardiffnlp/twitter-xlm-roberta-base` or `cardiffnlp/twitter-xlm-roberta-base-sentiment` (more details in sub-section 4.3.5 below).

4.3.5 Sentiment Analysis

For multilingual sentiment analysis, the XLM-Twitter[18] and XLM-Twitter-Sentiment[19] (Barbieri et al. 2021) are used for training and evaluating adapters for sentiment analysis tasks in 6 languages, e.g., Finnish, Greek, Dutch, Hungarian, Polish, and Swedish. Since XLM-Twitter-Sentiment is already fine-tuned on datasets for sentiment analysis in English, French, German, Italian, and Spanish, this is used for predicting sentiments of the crawled tweets directly.

1. This **first step** trains an adapter for sentiment analysis for each language with a language model:

```
    python -m models.scripts.xlm-r-adapter --lang_code sv --
task sa --checkpoint cardiffnlp/twitter-xlm-roberta-base
```

[17] https://huggingface.co/
[18] https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base
[19] https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment

The task is set as `sa`, which is short for sentiment analysis, and the language code is a two-letter code, such as `sv`, which loads the sentiment analysis datasets in Swedish (refer to the datasets file structure in 4.3.2). The `checkpoint` can be chosen from either `cardiffnlp/twitter-xlm-roberta-base` (XLM-Twitter) Or `cardiffnlp/twitter-xlm-roberta-base-sentiment` (XLM-Twitter-Sentiment).

The trained adapter and evaluated results will be stored in `output/twitter_xlmr_sentiments/{lang}/`. The best performing fine-tuned language model with the respective adapter is selected for predicting the sentiments of the crawled tweets in the next step.

2. The **next step** is to predict the sentiments of the tweets using the selected language model and the respective adapter, for example,

```
python -m models.scripts.inference --lang_code sv --task sa --
checkpoint   cardiffnlp/twitter-xlm-roberta-base-sentiment   --
use_adapter True
```

The best performing fine-tuned language model for Swedish sentiment analysis is `cardiffnlp/twitter-xlm-roberta-base-sentiment`, which is set as the checkpoint for inference, and the `use_adapter` is set to True, which loads the trained adapter from the corresponding folder resulted from sub-section 4.3.4 (1).

The text input is from the results of 4.3.3. The output from this step is stored in the folder `output/results/SA/{lang}.`

4.3.6 Hate Speech Detection

For multilingual hate speech detection, the XLM-Twitter is used for training and evaluating adapters for hate speech detection tasks in all 11 languages.

1. This **first step** trains an adapter for hate speech detection for each language with a language model:

```
python -m models.scripts.xlm-r-adapter --lang_code swedish --
task hsd --checkpoint cardiffnlp/twitter-xlm-roberta-base
```

The task is set as `hsd`, which is short for hate speech detection, and the language code is a two-letter code (e.g, `Swedish)`, which loads the hate speech detection datasets in Swedish (see datasets file structure in sub-section 4.3.2). The `checkpoint` is set as `cardiffnlp/twitter-xlm-roberta-base` (XLM-Twitter).

The trained adapter and evaluated results will be stored in `output/xlmr_hatespeech/{lang}/.`

2. The next step is to predict hate speech of the tweets using the language model with the respective adapter, for example,

```
python -m models.scripts.inference --lang_code swedish --task
hsd --checkpoint cardiffnlp/twitter-xlm-roberta-base --
use_adapter True
```

The fine-tuned language model for Swedish hate speech detection is `cardiffnlp/twitter-xlm-roberta-base`, which is set as the checkpoint for inference, and the `use_adapter` is set to True, which loads the trained adapter from the corresponding folder resulted from 4.3.4 (1).

The text input is from the results of 4.3.3. The output from this step is stored in the folder `output/results/HSD/{lang}`.

The same script as Section 3.8 was used for populating the Knowledge Base (KB) **(populate_kb)**.

5. Error Handling & Bug Reports

In case of any problems in the installations or any errors, users can contact the following persons:

- Yiyi Chen, yiyi.chen@fiz-karlsruhe.de (corresponding developer)
- Mehwish Alam, mehwish.alam@fiz-kalrsruhe.de
- Fouad Shammary, fouad.shammary@mtu.ie
- Zsolt Kardkovàcs, zsolt.kardkovacs@mtu.ie
- Haithem Afli, haithem.afli@mtu.ie

6. Conclusions

This deliverable provides basic motivations, foundations, terms, and notions for ITFLOWS Social Media Analysis. It discusses technical details of building the models responsible for analysing public attitudes towards migrations on social media. The created Github repository is extensively explained from setting up the environment to the implementation of the software prototype. The best performing model is explained for Topic Modelling, Entity Linking, Sentiment Analysis, and Hate Speech Detection. More details on the statistics or the dataset crawled and analysed along with the deeper analysis are developed in D5.4. Future versions of this manual will include an evaluation of the quality of the outcome of both sentiment analysis as well as hate speech detection by human evaluators. Moreover, it will also include an analysis of what kinds of tweets are wrongly classified.

7. References

(Alam et al. 2022) Alam Mehwish, Chen Yiyi, Heidland Tobias, Krüger Finja, Afli Haithem (2022) "Detecting Negative Attitudes and Hate Speech in Twitter Data and Analysing their Contextual Factors" ITFLOWS Deliverable 5.4.

(Barbieri et al. 2021) Francesco Barbieri, Luis Espinosa Anke, Jose Camacho-Collados, (2021), XLM-T: A Mulitlingual Model Toolkit for Twitter, retrieved from https://arxiv.org/abs/2104.12250, 2021

(Chen et al. 2021) Chen Yiyi, Sack Harald, Alam Mehwish (2021) "MigrationsKB: A Knowledge Base Of Public Attitudes Towards Migrations And Their Driving Factors." https://arxiv.org/abs/2108.07593v1

(Dieng et al. 2020) Dieng Adji Bousso, J. R. Ruiz Francisco, M. Blei David (2020) "Topic Modeling in Embedding Spaces". Transactions of Association of Computational Linguistics (TACL), vol 8, PP: 439-453.

(Rosenthal et al. 2017) Rosenthal Sara; Farra Noura; Nakov Preslav (2017) "SemEval-2017 task 4: Sentiment analysis in Twitter". In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp 502–518.

(Mathew et al. 2021) Mathew Binny; Saha Punyajoy; Muhie Yimam Seid; Biemann Chris; Goyal Pawan; Mukherjee Animesh (2021) "Hatexplain: A benchmark dataset for explainable hate speech detection". In The Thirty-Fifth AAAI Conference on Artificial Intelligence. AAAI Press.

(Marume 2016) Samson Brown Muchineripi Marume. "Public Policy and Factors Influencing Public Policy", International Journal of Engineering Science Invention 5(6), June, 2016, pp. 6-14.

(NRC 2007) National Research Council. *Engaging Privacy and Information Technology in a Digital Age*. Washington, DC: The National Academies Press.https://doi.org/10.17226/11896.

(Wu et al. 2020) Wu, Ledell, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer (2020) "Scalable Zero-shot Entity Linking with Dense Entity Retrieval." Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).