PHARM

PREVENTING HATE AGAINST REFUGEES AND MIGRANTS

Set of scripts of algorithms

This project is funded by the European Union's Rights, Equality and Citizenship Programme (2014-2020). REC-RRAC-RACI-AG-2019. Grant Agreement number 875217

D8 – WP2

September 2020

http://pharmproject.usal.es

This report constitutes Deliverable 8, for Work Package 2 of the PHARM project.

September 2020

General contact: pharm.project.eu@gmail.com

For more information type the e-mail address of the corresponding author

Please refer to this publication as follows:
Type the author(s) (year). *Title & subtitle* (Deliverable n°). Pharm project 875217 – H2020.

Information may be quoted provided the source is stated accurately and clearly.
This publication is also available via https://pharmproject.usal.es/

This project is funded by the European Union's Rights, Equality and Citizenship Programme (2014-2020). REC-RRAC-RACI-AG-2019. Grant Agreement number 875217

PHARM
PREVENTING HATE AGAINST REFUGEES AND MIGRANTS

VERSION CONTROL SHEET

Deliverable number + title	D8. Set of scripts of algorithms
Prepared by	Greece Team
Work package number	WP2
Work package leader	Andreas Veglis (AUTH)
Dissemination level (PU, CO)	PU
Delivery date	September 30, 2020.
Submission date to EC	Click or tap to enter a date.
Reviewers	Carlos Arcila Calderón and Patricia Sánchez Holgado (USAL)

REVISION HISTORY

Version	Date	Summary of changes	Initials	Changes marked
1				

Contents

1. Project Description

PHARM is a European project funded by the European Union, within Rights, Equality and Citizenship programme REC-RRAC-RACI-AG-2019 (GA n. 875217). The main goal of Preventing Hate Against Refugees and Migrants (PHARM) is to monitor and model hate speech against refugees and migrants in Greece, Italy and Spain in order to predict and combat hate crime and also counter its effects using cutting-edge techniques, such as data journalism and narrative persuasion. The main result will be the identification and reduction of online hate speech, and the prediction of potential hate crimes.

This GitHub repository includes a full set of scripts of the algorithms that (will) have been developed, according to the requirements of the project.

https://github.com/thepharmproject/set_of_scripts

This way, they can be shared with the rest of the team, but also with external stakeholders for serving evaluation purposes by testing if all scripts can be run with the corresponding expected result. Of course, all these methods will further elaborated and improved as the project evolves, while new functionality will be added, according to the project's timeline (i.e. sentiment analysis etc.).

1.1 Sources Selection

Several sources have been selected for the collection of content related to hate speech. The sources include articles and comments from a list of specific Spanish, Italian and Greek websites, as well as twitter, youtube and facebook comments. The websites list includes 22 Spanish, 12 Italian and 16 Greek websites that are prone to publishing hate speech content in the articles or in the comments section. The list of websites and Facebook pages was initialized and updated by the media experts of the three participating universities. Site-specific scraping scripts have been developed for the collection of semi-structured content (including accompanying metadata) from the proposed websites. Websites that are not included in the list can be supported using the site-agnostic scraping script. Tweets are gathered using a list of hashtags and filters containing terms relevant to anti-immigration rhetoric. Youtube comments are collected using search queries relevant to immigration.

A spreadsheet in Google Docs (https://bit.ly/3jaWzEq) for adding sources of interest has been deployed. Each sheet refers to the country of interest (i.e.

Greek, Italian, Spanish). The selected sources (websites, keywords, search terms, facebook pages etc.) for each platform are included in the current version of the spreadsheet.

1.2 Scraping & Data Collection

A multi-source platform for analysis of unstructured news and social media messages has been developed. It is important that chosen sources are valid and reliable to detect hate speech against refugees and migrants. In addition to this, hate speech data should include data (texts of the news or social media messages) and meta-data (location, language, date, etc.). For this reason, different types of sources (websites and social platforms) have been chosen and the necessary technical implementations have been made to collect the necessary data from these sources.

JSON (JavaScript Object Notation) is the preferred format for storing the data. JSON is an open standard, containing labels and contents as an attribute-value pair. One of the advantages of JSON is the flexibility to store different fields depending on the features and information available for each text. In order to store, query, analyze and share news and social media messages, PHARM adopts a semi-structured format based on JSON, adapted to media features.

1.2.1 Approach

The existence of any public or private application program interface (API) to collect media data is in high priority. Hence, Twitter API and Google API are exploited for collecting data from the Twitter and YouTube platforms, BeautifulSoup and selenium for scraping texts from web pages and automating web browser interaction via the Python programming language. Semi-structured scrapers for over 50 websites (Greek, Italian and Spanish) have been developed for collecting data massively, whereas routines for gathering data from Facebook groups and pages, YouTube videos and any other website have also been implemented.

A prototype JSON-like data format with a simplified description scheme (source, title, text, metadata) has been formed, towards a unified approach for all sources (with an NoSQL approach). This format is used for storing content from websites (articles and comments), tweets and YouTube comments (for the case of twitter and YouTube the description scheme is more complex). As for the websites, much effort was put on scraping targeted websites (i.e. when DISQUS or/and Facebook users' comments are present). All implementations have been refined to be more robust, accurate, being fully compatible with the doccano (https://github.com/doccano) platform for manual annotation.

1.2.2 Implementation

This approach required coding a couple of python files. In specific, scraper_twitter.py, scraper_web.py, scraper_youtube.py, single_facebook.py, single_web.py and single_youtube.py files implement the necessary routines for collecting the content.

1.2.2.1 Twitter tweets via streaming (Twitter API)

The project supports text collection from the twitter via the appropriate API and the stream method. Tweepy is used for accessing the Twitter API. Four dictionaries for filtering tweets have been developed including Greek, English, Spanish and Italian keywords. These can been found in the "Keywords" directory. Results are stored to the "Data/scraper_twitter_data.json" file.

1.2.2.2 Comments from YouTube videos via searching (Google API)

YouTube comment collection is supported via the Google API. A search query relevant to the topic (e.g. "migration refugees") is required from the user and the comments from the top results (videos) are collected. Results are stored to the "Data/scraper_youtube_data.json" file.

1.2.2.3 Posts and replies from open Facebook groups and pages

The selenium and BeautifulSoup packages are used to gather content from open Facebook groups and pages. A single URL has to be specified. Results are stored to the "Data/single_facebook_data.json" file.

1.2.2.4 Comments from YouTube videos

The selenium and BeautifulSoup packages are used to gather content from single YouTube videos. The corresponding URLs should be specified. Results are stored to the "Data/single_youtube_data.json" file.

1.2.2.5 Articles and comments from monitored websites

The selenium and BeautifulSoup duet is exploited in this case as well. For the monitored websites, a structured approach was followed. Articles, users' comments, and meta-data are recognized and organized based on the selected description scheme. Results are stored to the "Data/scraper_web" directory.

1.2.2.6 Unstructured text data from any website

This method can be used to collect texts from single web-pages. Results are stored to the "Data/single_web_data.json" file. Data that is collected using this method is unstructured.

1.2.3 Methods/Packages Evaluated

Package	Pros	Cons	Links
BeautifulSoup	Popular and easy to use	Cannot parse dynamic content (i.e. Java-script)	https://pypi.org/project/beautifulsoup4/ https://anaconda.org/anaconda/beautifulsoup4
selenium	Can manipulate dynamic content	More difficult to tune data parsers	https://pypi.org/project/selenium/ https://anaconda.org/anaconda/selenium
tweepy	Easy to use	Compatibility issues between versions	https://pypi.org/project/tweepi/ https://anaconda.org/conda-forge/tweepy
Google API Client	Officially supported by Google	-	https://pypi.org/project/google-api-python-client/ https://anaconda.org/conda-forge/google-api-python-client

1.3 Datetime Estimation

A method for detecting and standardizing datetime information from metadata and text has been implemented. Besides location and language, when metadata is available, PHARM might use some relevant extra information for hate speech analysis. Some of this extra information, such as date or time, might be available in most cases in several different formats. This introduces the necessity of standardization.

1.3.1 Approach

For the needs of this requirement, dateparser, datefinder and parsedatetime python packages are exploited, ranked by higher accuracy to higher probability of returning a result. If the most accurate method fails to detect a datetime object, the next service is called. Detection is based on metadata analysis, where date information is commonly present. If datetime detection fails for all services for the metadata, the same workflow is applied to the text data.

1.3.2 Implementation

The aforementioned approach is implemented as a method (detect_datetime(text, metadata, language) -> datetime) in the analysis_nlp.py file.

1.3.3 Methods/Packages Evaluated

Package	Pros	Cons	Links
dateparser	Relatively high accuracy	Limited estimations	https://pypi.org/project/dateparser/ https://anaconda.org/conda-forge/dateparser
datefinder	Lower accuracy	Returns a list of estimations	https://pypi.org/project/datefinder/ https://anaconda.org/conda-forge/datefinder
parsedatetime	Baseline method	Datetime scheme should be defined	https://pypi.org/project/parsedatetime/ https://anaconda.org/conda-forge/parsedatetime

1.4 Geolocation Estimation

1.4.1 Approach

A method for detecting geolocation from text data has been coded. The geopy library, along with the nominatim geocoder have been selected. Named entities (linguistic features) are isolated from texts, according to the following ranking: GPE (countries, cities, states), LOC (mountains, bodies of water), FAC (buildings, airports, highways etc.), ORG (companies, agencies, institutions etc.). Next, location estimation is performed for each one of the discovered entities.

1.4.2 Implementation

The aforementioned approach is implemented as a method (detect_location(text, metadata, language) -> locations[]) in the analysis_nlp.py file.

1.4.3 Methods/Packages Evaluated

Package	Pros	Cons	Links
Geopy	Easy to use with a plenty of geocoders	None (so far)	https://pypi.org/project/geopy/ https://anaconda.org/conda-forge/geopy

Geopy is a Python client for several popular geocoding web services, enabling the detection of the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources. Geopy includes geocoder classes for the OpenStreetMap Nominatim, ESRI ArcGIS, Google Geocoding API, Baidu Maps, Bing Maps API, Yahoo! PlaceFinder, Yandex, IGN France, etc.

1.5 Language Detection

PHARM will mainly process text produced in Greek, Italian, and Spanish, but many of the sources might have contents in other foreign languages or local dialects. To work with these three national languages, a procedure to detect the language of the media text when it is not properly declared should be present. There already exist many algorithms designed to automatically detect the language in different kinds of texts within a range of probability.

1.5.1 Approach

A recursive approach is adopted for improved robustness. Textblob, googletrans and langdetect services are used. If a service fails the result form the next one is requested.

1.5.2 Implementation

The aforementioned approach is implemented as a method (detect_language(text) -> language) in the analysis_nlp.py file.

1.5.3 Methods/Packages Evaluated

Package	Pros	Cons	Link
textblob	Accurate and easy to use	Limited requests	https://pypi.org/project/textblob/ https://anaconda.org/conda-forge/textblob
googletrans	Accurate and easy to use	None (so far)	https://pypi.org/project/googletrans/ https://anaconda.org/conda-forge/googletrans
pycld2	Reports estimation reliability	Only for linux python envs	https://pypi.org/project/pycld2/ https://anaconda.org/syllabs_admin/pycld2
langdetect	Accurate easy to use	None (so far)	https://pypi.org/project/langdetect/ https://anaconda.org/conda-forge/langdetect

1.6 Metadata Selection

Taking into account the requirements of the project (i.e. PHARM might use some relevant extra information for hate speech analysis), the sources that will be used for gathering the relevant content (i.e. Website articles and comments, YouTube comments and Twitter tweets), interoperability and compatibility considerations for importing/exporting data to third party applications that may/should be exploited (i.e. doccano platform for annotation), the following general specifications have been set:

Format	JSON
Identifier	doccano ID and/or PID
Data grouping	PHARM ID (PID) or source url
Annotations	labels from doccano – annotation data
Metadata	fields may vary from across sources
Text	payload

The main/base data field is the text (content), accompanied by the id, annotations and meta fields. Meta field is a container, including all additional data. A fundamental/minimum set of metadata will be used for all platforms. These will be the Pharm ID (PID), the source URL, the title, author, date, tags and categories. These fields are more probable to be found for all records across different sources. The following figure gives a hierarchical view of the proposed data scheme.

A custom identifier has been designed, serving as a compact and unique representation of each record retrieved, based on a hash function of the source URL. In the cases of web scraping, metadata depends on the available metadata provided by each site and the site-specific structure. In the case of youtube and twitter comments, where the corresponding APIs are used, specific metadata have been selected and are collected along with the comment text.

Youtube									
comment_id	reply_count	like_count	video_id	video_title	channel	video_description	author_id	author_name	date
Twitter									
tweet_id	is_retweet	is_quote	user_id	username	scr_name	location	followers	friends	date

1.7 Hate Speech Detection

A couple of methods for finding key terms for hate speech detection have been implemented. These include simple string matching, approximate string matching with the use of the appropriate metrics, such as Levenshtein Distance, Damerau-Levenshtein Distance, Jaro Distance, Jaro-Winkler Distance, Match Rating Approach Comparison, Hamming Distance. Term matching also aims at being suffix agnostic, accommodating the various suffixes that may exist in nouns for many languages (i.e. Greek language features different suffixes for gender or singular/plural versions). A word-vector approach has also been developed, taking into account the semantic meaning of the terms. A hybrid dictionary-based approach with predefined phrases, along with dynamic term combinations (i.e. adjectives combined with nouns) has been implemented and is under evaluation.

1.7.1 Approach

Two modes can be used for filtering (mode=0,1,2). The first one is based in a dictionary of terms for four different languages: English, Greek, Italian and Spanish. A language model is loaded (according to the language of the text), common practices are followed (lowercasing, lemmatization, stop-words and punctuation removal), and the targeted terms are being searched in the text. If a term (or combination of terms) is found, text segments are denoted as "hate speech". The second one is based in word vectors allowing a more semantic detection. The same workflow is followed for this method as well (lemmatization etc.). If mode is set to "2" the union of the results from both methods is returned.

1.7.2 Implementation

The aforementioned approach is implemented as a method (detect_hate(text, metadata, language, mode) -> matches[]) in the analysis_nlp.py file.

1.7.3 Methods/Packages Evaluated

Package	Pros	Cons	Links
spacy	Pretrained models (EN, ES, IT, EL), lots of linguistic features (part of speech tagging, entity recognition, tokenization, lemmatization, rule based matching, word vectors, etc.)	Models with vectors are slow	https://pypi.org/project/spacy/ https://anaconda.org/conda-forge/spacy

1.8 Topic Modeling

1.8.1 Approach

A combined method TFIDF (term frequency–inverse document frequency) with Non-negative Matrix and LDA (Latent Dirichlet Allocation) is deployed for topic modeling. Detected topics and most common terms are returned. A list of topics is created based on a corpus of text items.

1.8.2 Implementation

The aforementioned approach is implemented as a method (topic_modeling(corpus[], language) -> topics[], words[]) in the analysis_nlp.py file.

1.9 Entity Collection for Hate Speech

Identifying what an unstructured text is about is a challenging task. Most of the studies use search terms (words, lemmas, stems, combination of words with rules, etc.) to filter the texts that correspond to a certain topic, but it often happens that some topic-related contents remain excluded or that some topic-unrelated contents pass the filter and become included. This fact makes highly relevant the chosen mechanism to detect the topic in a large collection of news or social media messages.

1.9.1 Approach

A similar to topic modeling approach has been implemented, including named entity detection for the identified topics.

2. How to Execute the Code

Review the first lines of code (marked as "PYTHON SETUP") in the various Python files of the project to install the necessary libraries for executing the project. Use the "main.py" file as the starting script to run the code. The supported workflows/analyses can be controlled via the "Config/main.txt" configuration file. Instructions for controlling the workflow are following and are also included in the main configuration file. This set of scripts can execute various text collection and processing tasks, as specified by the requirements of the PHARM project. These tasks can be grouped into two main categories: data collection and data analysis.

2.1.1 [TWITTER-STREAM]

Set this parameter to collect tweets via Twitter's stream function. Set the parameter to "en", "el", "es" or "it" to use the Greek, English, Spanish, or Italian keyword list. You can find and modify the keyword lists in the "Keywords" directory. Results are stored to "Data/scraper_twitter_data.json". Comment out the following line to skip this type of data collection.

[TWITTER-STREAM]="el"

2.1.2 [YOUTUBE-SEARCH] & [YOUTUBE-SEARCH-NRESULTS]

Use these parameters for collecting YouTube comments via the Google API. Set the [YOUTUBE-SEARCH] parameter to define the keywords for searching content, e.g. "migration refugees" and the [YOUTUBE-SEARCH-NRESULTS] parameter to regulate the number of returned results. Results are stored to "Data/scraper_youtube_data.json". Comment the following lines for ignoring this method.

[YOUTUBE-SEARCH]="μετανάστες"
[YOUTUBE-SEARCH-NRESULTS]="200"

2.1.3 [WEBSITE-SINGLE]

Use this parameter for collecting texts from a single webpage. Simply set the URL of the webpage. The URL can point to an open Facebook Group or Page, a YouTube video or any other website. Results are stored to "Data/single_facebook_data.json", "Data/single_youtube_data.json" and "Data/single_web_data.json" respectively. Data management is semi-structured for the Facebook and YouTube platforms, whereas is unstructured for the rest of the webpages. Comment out the following lines to skip this type of data collection.

[WEBSITE-SINGLE]="https://www.facebook.com/groups/8080169598"
[WEBSITE-SINGLE]="https://www.facebook.com/playcompass"
[WEBSITE-SINGLE]="https://www.youtube.com/watch?v=fDWFVI8PQOI"
[WEBSITE-SINGLE]="https://bit.ly/33b7jLZ"

2.1.4 [WEBSITE-MASS] & [WEBSITE-MASS-CYCLES]

Set these two parameters for collecting articles and comments from the supported websites for structured scraping. The list of the supported websites is stored in the "Config/websites.txt" file. Make use of the [WEBSITE-MASS] and [WEBSITE-MASS-CYCLES] to set the URL and define the number of scans cycles the spider will perform to the site. The algorithm automatically discovers the various pages/articles within the site, keeps a list of the already visited hyperlinks, stores content and moves on to new unvisited destinations. Results are stored to separate, according the name of the site, files in the "Data/scraper_web/" directory. Comment out the following lines for skipping this type of data collection.

[WEBSITE-MASS]="http://www.voxespana.es"
[WEBSITE-MASS-CYCLES]="100"

2.1.5 [ANALYZE-DATA]

Set this parameter for executing the data analyses methods, acoording to the project's requirements, such as topic modeling, language detection, geolocation estimation, datetime parsing, hate speech detection etc. These methods will be refined and updated with new functions, according to the project's timeline. The analyses can be applied to the data that is collected and stored in the "Data" directory. Use the following parameter to point to a specific file/s. Once code execution is finished, a new file with the same name as the original plus the "_processed" suffix will be generated. Derived data can be used for manual annotation, or, generally, for developing a "hate speech" corpus. Comment out the following line for skipping the data analysis procedure.

[ANALYZE-DATA]="Data\scraper_web*vimaorthodoxias*_data.json"

PHARM

PREVENTING HATE AGAINST REFUGEES AND MIGRANTS

ABOUT PHARM (2020-2022)

Preventing Hate against Refugees and Migrants (PHARM)

Migration to Europe has grown in the last years in scale and complexity. The so called 'refugee crisis' and the migratory pressure is particularly acute in southern EU countries as the main entrance to the EU.
The main goal of PHARM project is to monitor and model hate speech against refugees and migrants in Greece, Italy and Spain in order to predict and combat hate crime and also counter its effects using cutting-edge techniques, such as data journalism and narrative persuasion. The activities distributed in 5 coordinated work packages include:
(i) Implementation of a conceptual and methodological common framework for large-scale analysis and detection of hate speech; (ii) Implementation and evaluation of machine learning approaches to model and predict hate crimes against refugees and migrants based on hate speech features; (iii) Survey journalists to understand how they inform and raise awareness about hate speech and how they can help building and disseminating counter-narratives based in data-driven news pieces; (iv) Creation, evaluation and dissemination of counter-narrative fictional stories adapted to different characteristics of citizens using large-scale narrative persuasion.

COORDINATOR

University of Salamanca, Faculty of Social Sciences (ES)

PARTNERS

University of Milan, (IT)
Aristotle University of Thessaloniki (GR)

VISIT: http://pharmproject.usal.es

CONTACT US: pharm.project.eu@gmail.com

FOLLOW US:

@Pharm_project

Pharm_project

Pharm Project

Pharm Project