UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF A CLASSIFIER OF RADICAL TWEETS USING MACHINE LEARNING ALGORITHMS

MARIO HERNANDO SEGOVIA

2018

TRABAJO FIN DE GRADO

Título:	Desarrollo de un clasificador de tweets radicales utilizando algoritmos de Machine Learning
Título (inglés):	Development of a Classifier of Radical Tweets using Machine Learning Algorithms
Autor:	Mario Hernando Segovia
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:

Vocal:

Secretario:

Suplente:

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF A CLASSIFIER OF RADICAL TWEETS USING MACHINE LEARNING ALGORITHMS

Mario Hernando Segovia

Enero de 2018

Resumen

En los últimos años, las organizaciones terroristas han utilizado las redes sociales, especialmente Twitter, con la intención de provocar miedo, reclutar nuevos miembros y adoctrinar.

Este proyecto tiene como objetivos el desarrollo de un clasificador que nos permita determinar si un tweet es radical o no mediante técnicas de machine learning, la implementación de un servicio que nos permita analizar la radicalidad de un texto con nuestro clasificador y la realización de un dashboard para visualizar potenciales usuarios radicales.

Para conseguir nuestro clasificador, hemos desarrollado un programa software utilizando el lenguaje de programación python con las herramientas que nos proporciona para el procesado de lenguaje natural (NLTK). Para la extracción de las características que hemos considerado oportunas (NER, POS, hashtags y sentimientos) de los tweets de nuestra base de datos y para la utilización de algoritmos de aprendizaje automático por medio de datos etiquetados hemos utilizado la librería scikit-learn. Como resultado, hemos obtenido un modelo con una precisión mayor al 90% clasificando la radicalidad de los tweets de las bases de datos utilizadas durante el proyecto.

La implementación del servicio de análisis de radicalidad de un texto se ha realizado mediante la API Senpy. Para la realización de un dashboard en el que visulizar posibles usuarios radicales se ha utilizado el entorno Sefarad.

Palabras clave: NLTK, Scikit-learn, NER, POS, Hashtag, Sentiment, Feature, Pipeline, Classifier, Twitter, Dashboard

Abstract

In the last few years, terrorist organizations have been using social networks, specially Twitter, with the purpose of creating fear, recruiting new members and indoctrinating.

This project has as its objectives the development of a classifier that allows us to determine whether a tweet is radical or not using machine learning techniques, the implementation of a service for analyzing the radicalization of a text with our classifier and the execution of a dashboard for visualizing potential radical users.

For creating our classifier, we have developed a software program using the python programming language and the tools that it provides for the processing of natural language (NLTK). For the feature extraction that we have considered (NER, POS, hashtags and sentiments) of the tweet of our database and for the use of machine learning algorithms for labeled data, we have used the library scikit-learn. As a result, we have obtained a model with an accuracy greater than 90% classifying the radicalization of the tweets contained in the database used in the project.

The implementation of the service for analyzing the radicalization of a text has been carried out using the Senpy API. The development of a dashboard for visualizing possible radical users has been done using the environment Sefarad.

Keywords: NLTK, Scikit-learn, NER, POS, Hashtag, Sentiment, Feature, Pipeline, Classifier, Twitter, Dashboard

Agradecimientos

Gracias a todas las personas que me han acompañado durante este camino, en especial a mi familia y amigos.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

In the last decades, many terrorist organizations were created with the purpose of creating terror in our society. A great part of these new organizations have jihadist ideology.

First of all, the members of this organizations spread chaos on their own country with the aim for establishing salafist governments like in Egypt, where president Anwar Sadat was killed by a jihadist in 1981, but after, their objective was also attacking in occident because they think that the defeat of occident is a prerequisite for establishing their caliphate due to the support that occident governments give to Arabian regimes [11].

11-S showed the world that these groups have the capacity to attack in every part of the world, since then, the mentality of countries changed in order to avoid risks to people [22] but terrorist organizations have also changed their mentality and strategies. These organizations prepared their attacks with centralized cells and now this strategy has changed to lone wolfs that attack individually [23].

Jihadists want to recruit new members to their ranks in order to make more attacks and with more efficacy and nowadays, social networks are their more effective and useful tools for making that possible. Trying to cause fear and also spread their propaganda on the population for having more supporters are the main reasons why they use social media and the internet [2].

Unfortunately, this strategy has worked and the last terrorist attacks have been perpetrated by people indoctrinated online and that has never been in conflict zones.

In order to avoid their spread in social media, it is necessary to find and delete their accounts but it is a laborious work and it requires a lot of time for a human being.

This is why, it is necessary to make a program that detects and predicts radical content faster than humans and in order to do that, we have to use machine learning techniques.

In this project, we have focused on the Twitter social network and we are going to develop a classifier of radical tweets using machine learning algorithms.

1.2 Project goals

The purpose of this project is to create a classifier of radical tweets and creating a visual environment for analyzing tweets visually using this classifier.

The main goals of our work are:

- Preprocessing radicalism related data.
- Building a pipeline for extracting features.
- Evaluation of different machine learning algorithms and comparing their average in classification tasks.
- Creating a visual environment for analyzing our classifier results with the Twitter API.

1.3 Structure of this document

Our paper has the following structure:

Chapter 1 explains the situation that has made necessary to use machine learning technology with social media data.

Chapter 2 introduces the tools used in the project.

Chapter 3 describes the model which we have built and the classifiers used.

Chapter 4 explains the steps taken in order to give a graphical visualization of our model results.

Chapter 5 gives a conclusion for this work and provides ideas for future projects.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter, we explain the technology machine learning, which we have used in our project, with its different techniques, the documentation related with our case which has been documented before this project, in order to get notions that can help for creating a our classifier. We are going to talk also about the python libraries with which we have developed our program and about other technologies that have been used.

2.2 Machine learning

Machine Learning (ML) [10] is a science discipline from the artificial intelligence field which creates systems that employs data, experience and training in order to identify patterns with the objective of predicting future behaviors automatically without human intervention.

It tries to automatize tasks through mathematician methods, in order to give possible solutions to problems with a big complexity.

We have two types of machine learning techniques depending on whether the data has continuous values (regression) or a discrete label (classification).

In classification models, there are three types of machine learning techniques depending on the input data given. We are going to describe briefly the three of them.

- ***Supervised learning:*** it is a technique which deduces a function from trained data which have variables that can be useful for creating an accurate predictive model (features) and data which has been tagged with one or more labels (labels).

  The process of supervised learning follows the steps shown in Fig. 2.1.

  First of all, our software receives the data and extracts the features that we want (the most informative attributes).

  After, it transforms the feature data and the labels to a compatible format (matrices) for classifiers which are the algorithms that are going to train and test with the data in order to map the input to a label.

  Once the model is trained, it can assign labels to new data. First of all, we have to insert the new text, extract the same features that we took when the model was trained and finally, using the model to predict the label.

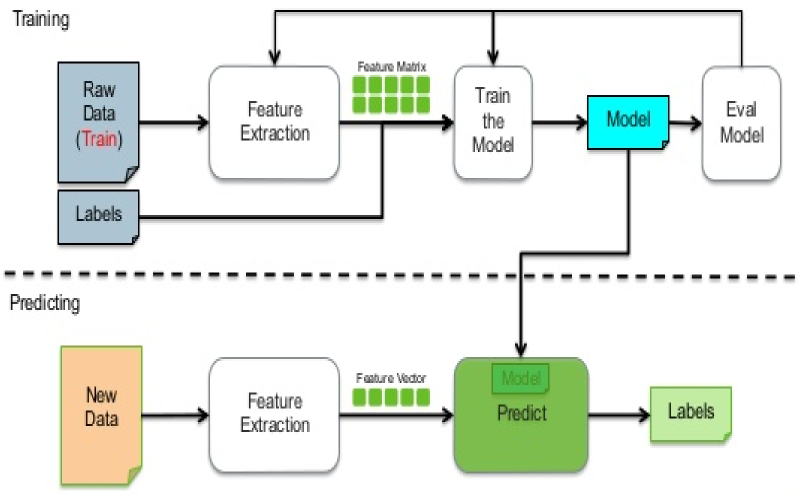


Figure 2.1: Machine learning supervised learning model [9]

- ***Unsupervised learning:*** This technique uses input data without labels, their final objective is not a classification, the purpose of this technique is finding conclusions, interesting inferences, intrinsic structures through the data.

  The main unsupervised learning method is clustering. It consists on dividing the input into groups, called clusters, whose objects are alike between them and that have differences with other groups objects.

The unsupervised learning work flow can be seen in Fig. 2.2. First of all, we receive input data and take the features that we want from it. After, we have to use a machine learning algorithm that is going to find patterns or characteristics that will allow to differ the data of the different clusters discovered.

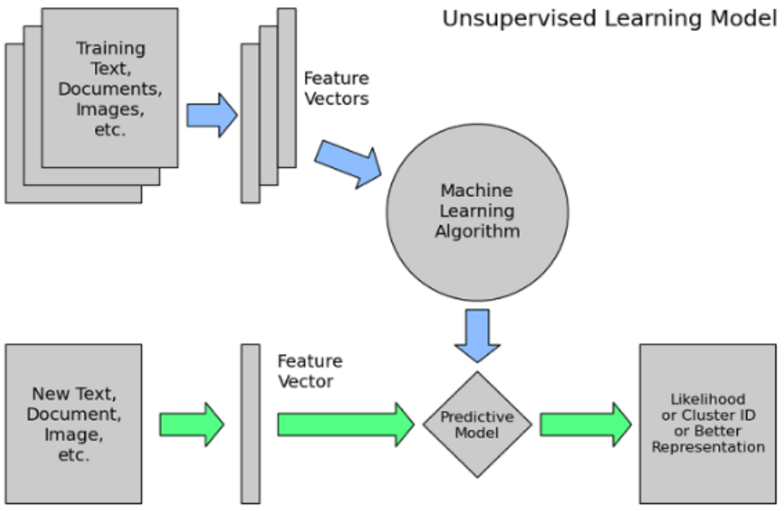

Unsupervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Likelihood or Cluster ID or Better Representation

Figure 2.2: Machine learning unsupervised learning model [12]

- ***Semi-supervised learning:*** This technique emerged for resolving the problem of classifying the output when we have a small amount of labeled data combined with unlabeled data. It uses transductive learning for deducing the correct labels in the unlabeled data and inductive learning for assigning labels to the features.

  Its work-flow can be seen in Fig. 2.3.

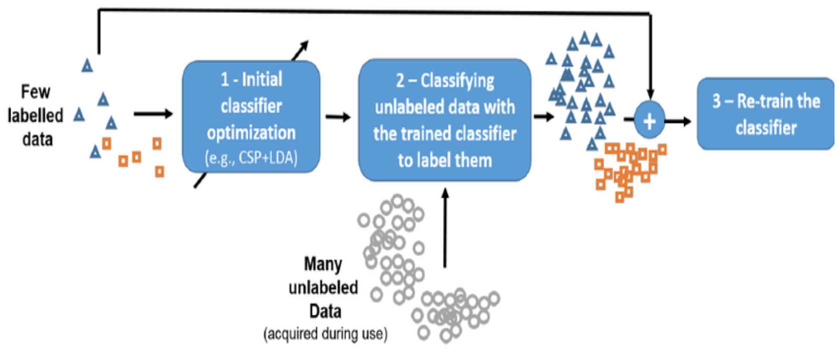

Few labelled data
1 - Initial classifier optimization (e.g., CSP+LDA)
2 – Classifying unlabeled data with the trained classifier to label them
3 – Re-train the classifier
Many unlabeled Data (acquired during use)

Figure 2.3: Machine learning semi-supervised learning model [14]

2.3 Related work

There is not an extensive amount of work related to detect extremism in social networks because terrorist organizations did not use this media to spread fear until recently.

In this section, some of the research of radicalism classifiers which are related with our job are described.

- ***Automated classification of extremist twitter accounts using content-based and network-based features [25]:*** the authors design a classifier for identifying Twitter accounts from specific geographic areas as ISIS related accounts basing on records collected by Anonymous.

  In order to fulfill that, they use the following content-based and network-based features:

  - ***Hashtags:*** first of all, they extract the most common hashtags used in ISIS related accounts. If a particular user shared a publication including one of the top 50 most common ISIS accounts hashtags, this user is classified as radical.
  - ***Hashtags tokens:*** it consists on extracting the tokens included in the top 25 ISIS related accounts hashtags and after, they check if a particular user has used one of those tokens in his tweet text, classifying him as radical in this case.
  - ***Harmonic closeness:*** it is based on the closeness of a particular user with a set of pro-ISIS accounts.

- ***Identification of extremism on Twitter [26]:*** in this paper, the authors make a classifier after a feature extraction based on sentiment analysis whose features are:

  - ***Sentiment tendency feature:*** it consists on determining whether a particular user shows positive sentiment with radicalism groups or ideology or neutral/negative sentiment. First of all, they extract tweets associated to terrorist accounts and after, for the studied user, they assign the sentiment value 1 if the user shows affection for these groups or 0 if not. They define the value S, which is the summation of sentiment values divided with the number of tweets. If this value is greater than 0.5, they classify this user as radicalist.
  - ***Ego-network extremism support feature:*** first of all, they look for a user's followers, if the user has one or more followers with S greater than 0.5 and the studied user has also S greater than 0.5, then they give the value 1 to this user. If not, they give him value 0.

  - ***Mention-network feature:*** this characteristic is based on the study of the mentions of a particular user. If the user has been mentioned by a radical account (S greater than 0.5) or if he mentions an ISIS profile and the user has S greater than 0.5 then they give the value 1 to this user. If not, they give him value 0.

- ***Robust sentiment detection on Twitter from Biased and Noisy Data [3]:*** the authors use two features before using a classifier:

  - ***Meta-features:*** this characteristic consists on mapping the part-of-speech of sentences with a part-of-speech dictionary. This feature can show the language employed on tweets, for example, tweets containing opinions will have more adjectives or interjections. After the pos-tag, they make a polarity classification (with three categories: positive, negative and neutral) and they watch the subjectivity (weak or strong).
  - ***Tweet Syntax Features:*** it is used for studying the frequency of hashtags, retweets, replies, links, the use of exclamation and question marks, emoticons and upper cases words.

  In both extractors, the frequency of the feature is divided by the number of words contained in the tweet.

- ***A semantic graph-based approach for radicalization detection on social media [20]:*** In this paper, the feature extraction is based on name entities. They use three selectors:

  - ***Conceptual semantics extraction:*** in this feature, they extract named-entities of anti-ISIS and pro-ISIS accounts tweets and then, they expand these entities with their concepts.
  - ***Semantic graph representation:*** here, they represent as graphs the semantic relation of the name-entities that appears together in a tweet.
  - ***Frequent patterns mining:*** this feature is based on applying pattern mining techniques for learning similar semantics of pro and anti ISIS accounts.

- ***Using machine learning to identify jihadist messages on Twitter [18]:*** The authors start from three datasets, one with tweets pro-ISIS, other with tweets randomly collected and the last one, with tweets from accounts which are against ISIS. In terms of feature extraction, this project has focused on:

  - ***Stylometric features:*** it consists on studying the variations of literary style between users. They have made feature extraction basing on frequency of function words, words, punctuation characters, hashtags, letter bigrams and words bigrams.
  - ***Time-based features:*** they have elaborated an extraction studying when the tweet was shared basing on the hour, day of week, period of week and period of day
  - ***Sentiment features:*** as in previous projects analyzed in this section, they determine whether the tweet as a positive sentiment or not.

2.4 NLTK

Natural Language Toolkit (NLTK) [4] is a suite of Python modules providing many natural language processing data types as tokens, tags, chunks, trees, feature structures and corpus which allows to work with human language data. It has text processing libraries for classification, tokenization, stemming, tagging, parsing and semantic reasoning. NLTK provides the following modules:

- ***Tokenizer:*** it divides a text into a sequence of tokens
- ***Stemmer:*** it has the function of reducing words to their root.
- ***Part of speech (POS):*** the function of the part of speech tagging of NLTK, *pos tag* is providing the grammatical category of a given word.
- ***NER Taggers:*** tools that are going to provide names, organizations and locations contained in the tweets.
- ***Chunking:*** it consists on selecting a subset of tokens grouped by their part of speech tags. It is useful for finding nouns and extensive words used around them with their POS category.
- ***Collocations:*** tool for finding group of words that commonly appears together.
- ***Trees:*** tool that allows to analyze graphically the structure of the sentences of a text giving to it a tree structure.
- ***Corpus:*** NLTK has a collection of writings in its corpus. We have used the stopword corpus when preprocessing the data for deleting words without relevant meaning.

2.5 Pandas

Pandas [16] is a python software library developed for data analysis, which provides easy-to-use data structures that makes flexible and intuitive working with data.

It has two elemental data structures called series, which are unidimensional arrays with indexing that can be created from dictionaries or lists and dataframes, which are two-dimensional arrays whose columns are series.

It is built on NumPy, which is the python library for fast array calculations.

Some utilities furnished by pandas [15] are:

- NaN value when it finds squares without data.
- Aggregating and transforming data.
- Inserting and deleting DataFrame's columns.
- Converting from different python data structures and NumPy to DataFrame.
- Group by functionality for adding and changing data.
- Loading data from files of different extensions.
- Aligning the data that belongs to a set of labels.
- Concatenating, merging and joining data sets.
- Time series functionalities.

2.6 Scikit-learn

Scikit-learn [8] is a Python free software machine learning library built on Numpy, SciPy and matplotlib which has tools for data mining and data analysis that will be useful in our supervised machine learning model.

Scikit-learn provides tools for joining features, converting features and labels into matrices and classifiers. As it can be seen in its documentation, it provides the following functionalities:

Functionality	**Description**
Classification	Assigning the correct label category to input data
Regression	Predicting the continuous-value assigned to input data
Clustering	Grouping into sets similar objects
Dimensionality reduction	Methods for reducing the number of variables
Model selection	Comparing, validating and choosing parameters and algorithms in order to improve the accuracy
Preprocessing	Normalization and feature extraction

2.7 Meaning Cloud

Meaning Cloud [13] is an enterprise specialized in software which offers APIs for semantic analysis with the following functions:

- Topics extraction
- Text Classification based on taxonomy or hierarchical categorization.
- Sentiment polarity analysis.
- Language identification.
- Lemmatization, Part of speech (POS) and parsing.
- Corporate reputation.
- Text clustering.
- Summarization.
- Document Structure Analysis.

2.8 Sefarad

Sefarad [5] is an application developed by the ETSIT group GSI, it allows to explore data by making SPARQL (Protocol and RDF Query Language) queries to the endpoint chose by the user, it also lets analyze and visualize data.

The representation of data is based in dashboards, which are web pages oriented to display information collected. Sefarad provides predefined dashboards that can be modified by the user. Dashboards are divided in Polymer Web Components that globally form the dashboard. The persistence layer necessary for visualization is based on the technology ElasticSearch [6], which is a RESTful and analytics engine that can store data.

2.9 Senpy

Senpy [21] is an API developed by ETSIT-UPM's group GSI. It provides tools for making a detailed multilingual sentiment analysis of texts and plugins for identifying the gender or the age of the writer of a text.

Senpy also supplies services for developers, giving a web UI where users can explore, interact and test their service with the final objective of creating their own plugin.

CHAPTER 3

Machine learning model building and evaluation

3.1 Introduction

In this chapter, the steps followed for creating the classifier are explained.

First of all, the data from which the project has been carried out is described. This data has been downloaded from kaggle, which is an online platform that makes competitions for creating the best predictive model from a dataset provided. After, the concept of pipeline is explained with all its phases that served us to build our model.

The first phase of our pipeline is the preprocessing; an introduction about the concept is going to be described with the explanation of the preprocessor developed in our project. The second phase of the pipeline is the feature extraction. This concept is going to be explained with the description of each feature extracted for our case. Finally, the last phase of our pipeline is the climax of what was done before, the algorithms of classification used, which are provided by python, are remarked.

3.2 Data

First of all, the datasets used in the project have been downloaded from the platform kaggle. The competition where the data was taken is called *Tweets targeting Isis* [7] and here, two datasets can be found, one called *isisfanboy* which contains pro-ISIS tweets and the other one, called *aboutisis* which contains neutral tweets in topics related to ISIS.

In both of them, the next columns are contained:

- ***Username:*** it is the identity of the account
- ***Tweets:*** the text posted in Twitter.

Aboutisis contains over 122.000 tweets which were collected on two different days, 7/4/2016 and 7/11/2016 starting from terms related to Isis.

Isisfanboy has over 17.000 tweets collected from more than 100 pro-ISIS accounts since the November 2015 Paris Attacks. This dataset was provided in the *How Isis uses twitter* [24] competition with more fields than the explained before, such as name, description, location, number of followers, number of statuses and the time were the tweet was published.

We have taken this dataset from the *Tweets targeting Isis* competition because here this dataset has the same columns than the *AboutIsis* one.

Inserting both datasets has been done using Pandas' functionality *read csv*, which is going to return a DataFrame object containing the columns of the datasets.

In order to make a realistic classification, the same quantity of tweets has been taken from both datasets (the first 17.392 tweets).

For both datasets, a new column named radical is added, the value of the column is "yes" in the *IsisFanboy* dataset and "no" in the *AboutIsis* data. The objective of adding this column is having labeled data for making easier to learn patterns by applying supervised machine learning algorithms.

After, these datasets have been concatenated using Pandas and with the method *values* of the object DataFrame, data is converted to a numpy object in order to make the feature extraction and applying a classifier. This data is going to enter in the pipeline, which is going to be explained in the next section.

3.3 Pipeline

We use the pipeline module provided by scikit-learn to automate the workflow of a classifier.

In this phase, we receive raw data as a input, this data has to be prepared for being ready for a machine learning algorithm and because of that, we have to make some transformations to the data in the preprocessing phase, that is going to return the data prepared for being used by the algorithm.

After preprocessing the raw data, we can make the operations that we want with this data in order to improve our classifier. We are going to make a feature extraction, which receives as an input the preprocessed data, make some operations in it and returns a new column with a new feature that can be used by the machine learning algorithm.

The pipeline has to finish with a machine learning algorithm, the last step that we are going to do is obtaining the accuracy of the classifier.

In Fig. 3.1, we can see what we have explained before. Each phase is going to be described in detail in next sections.

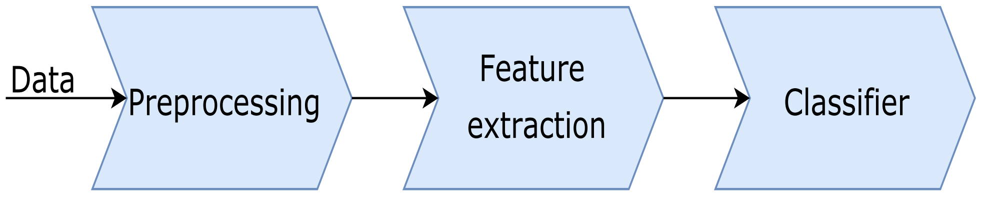


Figure 3.1: Model schema

3.4 Preprocessing

It is the process of data cleaning, in this phase, characters that don't contain content and can't change the meaning of a tweet are removed.

In order to remove the not important parts of a tweet, it is significant the concept of tokenization. It is the process of chopping up the sentence into pieces, called tokens that can be individual characters (punctuation marks), words, numbers... In conclusion, each token is the character or group of characters which are separated with spaces.

In this project, the tokenizer used is the *TweetTokenizer*, provided by NLTK, which returns a list of tokens contained in a tweet. With this tokenizer, we have made the process of tokenization and once we had the list of tokens, we have removed the followings ones:

- ***Punctuation marks:*** the tokens with this characters using the list of punctuation marks provided by the string module of Python have been removed.

- ***Stop words:*** they are words which don't give a change of meaning to the sentence, they are irrelevant and don't have a value for classifying the tweets, an example of stop words are "a", "the" and "other". In fact, search engines ignore these words. We have used the stop words provided by the NLTK corpus.

- ***Digits:*** numbers have been removed because they aren't useful in text classifications. In order to delete digits, the *isdigit* method provided by the python String module has been used.

- ***Emoticons:*** emoticons have been deleted because we don't have a dictionary relating all the emoticons of the different devices with their corresponding term and because of that, it won't help in a language classifier.

- ***URLs:*** tokens containing URLs have also been deleted because thay aren't useful in our language classifier.

Speaking about Twitter is speaking about hashtags. A hashtag is a string consisting in one or more concatenated words that are preceded by the hash symbol (#). Tweets can be grouped with hashtags, if we search for a hashtag in the Twitter bar, all the tweets containing these hashtags will be seen and consequently, this tweets will have more diffusion, which is one of the objectives of the ISIS.

The *TweetTokenizer* is the specific tokenizer for tweets, with it, hashtags can be removed; if its *strip handles* option has a boolean value of true, we will remove them, the value false will leave them. Depending on the feature desired, hashtags will be removed or not. In this project, hashtags have been removed in all the extractors, except in the hashtag one.

After obtaining the tokens, there are words which are formed from the same root that have the same meaning. In classification tasks, these words have to be unified in order to let the machine learning algorithm analyze just features containing all the variations of words; this function is made by a stemmer.

The objective of using a stemmer is normalizing the data, transforming into a common word all the words with the same root that carries the same meaning. For example, if we have as a token value the word *killer* and as another token value *killing*, the stemmer groups these words in their root, which is *kill*.

In this project, the SnowballStemmer is the one used, it transforms the words into a common root that don't have to be necessarily an existing word and WordNetLemmatizer has also been used, it transforms the words with a common root to an existing word.

3.5 Feature extraction

It consists on building new sets of features by performing operations for taking derived data of the initial dataset with the objective of helping in differentiating between the categories to the subsequent machine learning algorithms. We have to differentiate this method of giving features to our model with the feature selection method, which consists on choosing a subset of the original data input.

All the extractors are going to receive the data as an input and after preprocessing the data, they are going to make operations in order to get what they want from it.

For each feature extracted, a new column is added to our dataset that after, we are going to analyze if it improves our accuracy or not.

In order to concatenate the feature extractors (which are transformers), the class FeatureUnion provided by scikit-learn has to be used, which has as constructor a list of the feature extractors that FeatureUnion is going to join into a single one. To do so, each extractor must have the methods *get params* (for getting the feature names of the transformer) and *set params* (for setting the parameters of the transformer), these methods are inherited if we make BaseEstimator as our transformer subclass. We should also have TransformerMixin as a subclass of our extractor because this class allows us to fit and transform the data for getting the feature wanted.

After the extraction, wordcloud [1] is going to be used for watching the most common tokens extracted in each transformer. Wordcloud enables seeing a graphical representation of word frequency. It represents the words of a text with different sizes for each word depending on its frequency in the text. We are going to use this tool for representing the words, hashtags and NERs off our two data frames.

In the following subsections, the features extracted in our project are explained.

3.5.1 Word extraction

This extractor has been made using the TfidfVectorizer tool provided by scikit-learn.

TfidfVectorizer creates a matrix of term frequency–inverse document frequency (tf-idf) features, which means that the weight of tokens that appears frequently will be smaller than the weight of tokens which appear less times because these are more informative and can determine the label when it comes to make a classification in this feature.

Applying CountVectorizer followed by TfidfTransformer would have the same result. First of all, TfidfVectorizer creates a matrix of token counts from our data to a matrix of token counts (like CountVectorizer) and after, it transforms the matrix of counts to a normalized tf-idf representation (like TfidfTransformer).

TfidfVectorizer has some parameters that can be edited for adapting it to our project. We have used as the encoding format utf-8 and as the analyzer, which is the parameter used for extracting the sequence of tokens, we have chosen our preprocessing method, which was explained in the previous section.

Finally, each sentence is seen by the classifier as a vector with the tf-idf weight of each token that the sentence has.

In Fig. 3.2 and Fig. 3.3 the most frequent words contained in each dataset can be seen.

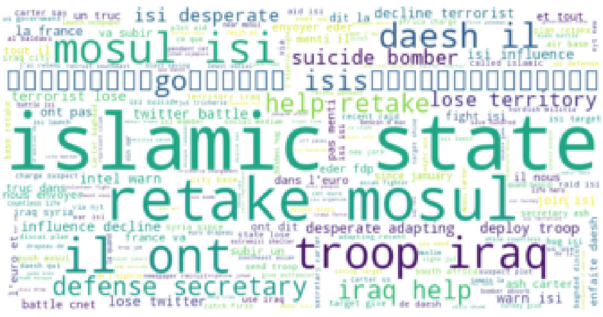


Figure 3.2: Most common words in *aboutisis* dataset

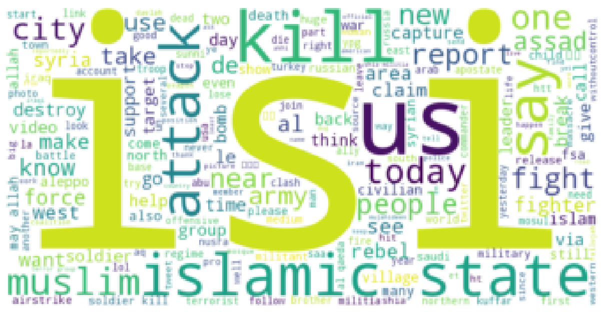


Figure 3.3: Most common words in *isisfanboy* dataset

3.5.2 N-gram extraction

An n-gram is a sub-sequence composed by n elements of a given sequence. When making a twitter classifier, it is as important analyzing n-grams as analyzing words because with n-grams expressions or groups of words that are used in *isisfanboys* or in *aboutisis* can be seen and it can help to make a more accurate classification to the machine learning algorithm. For example, with this feature, the algorithm will see in which dataset the pair of words islamic state or Donald Trump appears more frequently.

Implementing this feature has been similar to the word extraction, but in this case, it has been used the CountVectorizer followed by TfidfTransformer because CountVectorizer has a parameter called *ngram range* that allows us to choose the minimum and the maximum range of n values for different n-grams to take.

3.5.3 POS extractor

The objective of this feature extractor is analyzing the grammatical categories used in the different datasets. We have created an extractor that obtains the number of tokens with POS categories.

First of all, tokens of each sentence have been extracted from the data with the *Tweet-Tokenizer* by putting in its strip handles option the value “True” because in this extractor usernames and hashtags are not relevant.

After, the *pos tag* tool provided by NLTK has been used, it gives the POS category of each token. The objective is getting the weight of each category in all the tweets and in order to do that, a variable called *count* has been created whose value is the number of objects that *pos tag* has returned for the tweet. The weight of each POS category in the tweet is the result of dividing the number of times that each category appears in the tweet between *count* .

For each sentence, we are going to have a dictionary that has as a key the name of the POS category and as the key’s value, it has the weight of the category. The categories selected are noun, adjective, verb, adverb, conjunction, adpositions (it includes prepositions and postpositions), pronouns and numerical tokens.

Finally, the extractor returns a list containing the dictionaries of all the tweets contained in the training data, which is transformed to a format that the machine learning algorithm can understand with the DictVectorizer provided by scikit-learn, which transforms lists of feature-value mappings to vectors.

3.5.4 NER extraction

The concept of NER refers to the process of finding and storing tokens which represent people, organizations, locations, dates, times... The Standford NER Tagger that has been used finds just names, organizations and locations.

The purpose of the NER extractor is to analyze the named-entities contained in each tweet.

First of all, the data has been preprocessed using the TweetTokenizer in the mode without hashtags. After, in order to find and store the entities contained in our data, for each sentence of this data the Stanford NER tagger's method *tag sents* has been applied, which returns the corresponding token followed by its entity name or by "O" it it isn't an entity of the three wanted. We have inserted into a list the tokens that represents an organization, a location or a person followed by its entity. With this list, a dictionary containing each list item followed by the value 0 has been initialized. Consequently, the keys of the dictionary are going to be the list items.

For each tweet, we are going to modify the dictionary. If an entity of the dictionary is contained in the tweet, we are going to put the value 1 to this entity value.

Finally, a list containing all the dictionaries of the tweets contained in the training data is returned.

In Fig. 3.4 and Fig. 3.5 the most frequent NER of each dataset can be seen.

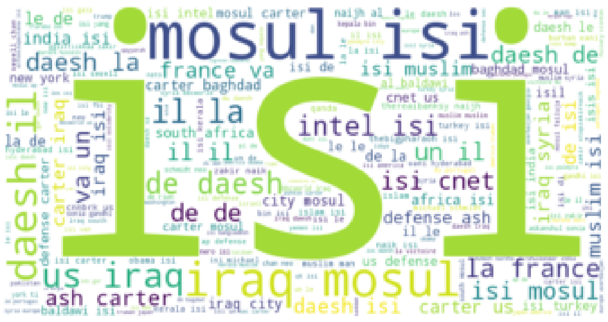


Figure 3.4: Most common NER in *aboutisis* dataset

3.5.5 Hashtag extraction

The objective of the hashtag extractor is analyzing the hashtags included in each tweet.

First of all, all the data has been preprocessed using the *TweetTokenizer* with the

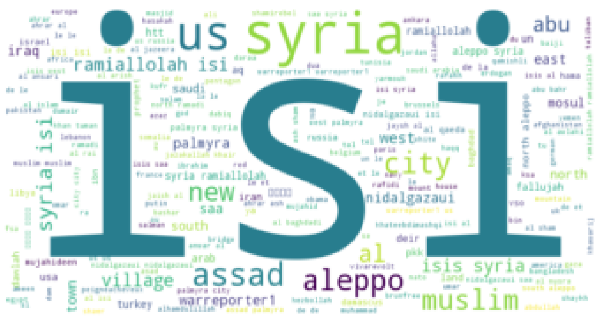


Figure 3.5: Most common NER in *isisfanboy* dataset

parameter *strip handles* with a value of false because, in this case, hashtags don't have to be deleted in the preprocessing phase. After, the tokens of each tweet have been taken in order to create a list containing the tokens starting with (#) of all the tweets. These tokens are added as keys to a dictionary, which have as values the integer 1 if the corresponding hashtag is in the tweet analyzed or a 0 if not.

Finally, as a result, a list containing the hashtag dictionary of each sentence of the training data is returned.

In Fig. 3.6 and Fig. 3.7 the most frequent hashtags of both datasets can be seen without the (#) symbol.

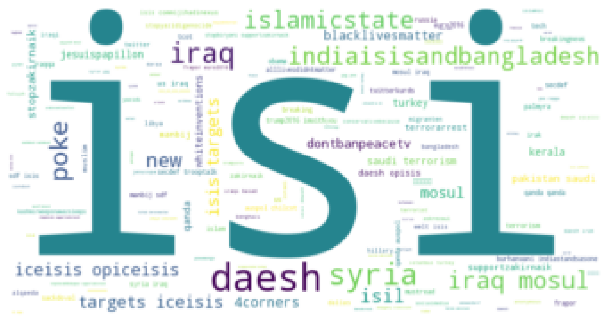


Figure 3.6: Most common hashtags in *aboutisis* dataset

3.5.6 Sentiment extraction

This extractor has been developed using the API meaning cloud for sentiment analysis. This API returns a JSON with the result of the sentiment analysis of a tweet.

In the first place, a request for each tweet to the meaning cloud sentiment-2.1 API has been made using our api key followed by the text contained in the tweet. The result of the request is a JSON including fields analyzing polarity, irony and subjectivity.

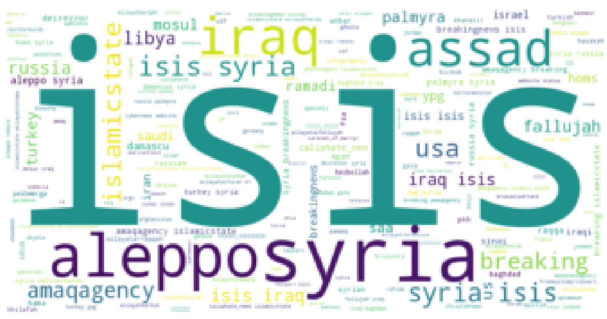


Figure 3.7: Most common hashtags in *isisfanboy* dataset

For each tweet, the value of the score tag field of each tweet has been added to the JSON, which indicates the polarity found in the text analyzed, there can be six possible values:

- ***P+:*** strong positive polarity.
- ***P:*** positive sentiment polarity.
- ***N+:*** strong negative polarity.
- ***N:*** negative sentiment polarity.
- ***NEU:*** neither positive nor negative polarity.
- ***NONE:*** sentiment not found.

Finally, the list containing all the sentiments of our training dataset is returned.

In Fig. 3.8 and Fig. 3.9, the distribution of the sentiment classification of our datasets can be seen, in these figures, we have painted with red the negative polarity (N and N+), with blue the positive polarity (P and P+) and with green the neutral tweets and tweets without polarity.

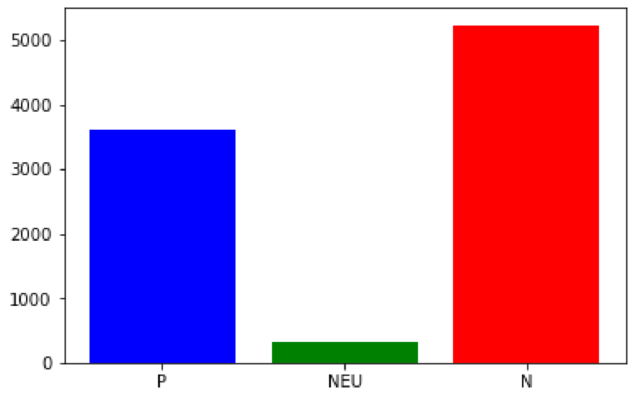


Figure 3.8: Sentiment analysis in *aboutisis* dataset

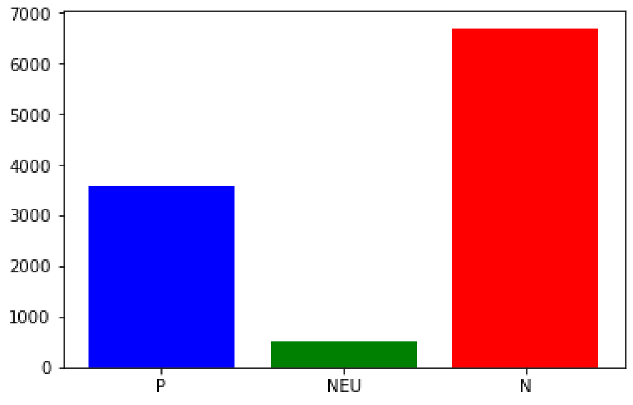


Figure 3.9: Sentiment analysis *isisfanboy* dataset

3.6 Classifiers

The last part of our pipeline is the machine learning algorithm, which learns patterns in order to assign correctly the label given to our input data. In this case, they are going to receive as input data the features extracted from the input of the pipeline, that were explained in the previous section.

The algorithms need training data, which are the data that is fitted and testing data, which is used for checking if the algorithm has assigned the correct label for the given input.

We have trained our model using cross validation, which consists on evaluating the results of an statistical analysis guaranteeing that the training and the testing data are independent. The technique of cross validation used is called *k-fold*, it consists on dividing the given data in k parts of the same size, each part is used once as testing data and (k - 1) times as training data as it can be seen in Fig. 3.10

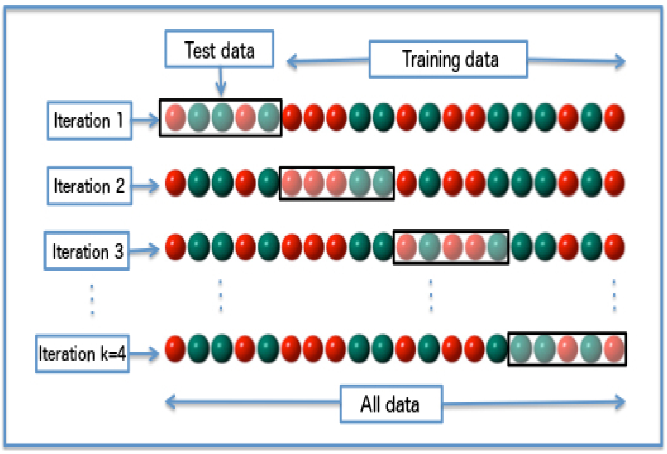


Figure 3.10: K-Fold cross-validator [27]

Machine learning algorithms constructor parameters are called hyper-parameters, they have values defined that are passed to the constructor of the algorithm. The hyper-parameters have a big impact in the accuracy of the model and because of that, all the possible value combinations of these parameters have to be analyzed; this is done with GridSearchCV, which is a tool provided by scikit-learn that looks for the best value combinations of the parameters.

We have analyzed the performance of different machine learning algorithms on our data; the tables included in this section represent the accuracy of the correspondent algorithm, which is a weighted average of the precision and has a maximum value of 1.

Each algorithm is going to be explained with detail in next subsections.

3.6.1 Multinomial Naive Bayes

Scikit-learn supplies the multinomial naive bayes classifier, which is based on Bayes' theorem with the "naive" assumption of independence between every pair of features for multinomial models.

We have modified the default alpha hyper-parameter value, this is an additive smoothing parameter. The value used in this project is 0.01, which was the one with best results with GridSearchCV.

We have used this classifier in conjunction with the Adaptive Boosting (AdaBoost) Classifier, which is a machine learning algorithm that can be used with other learning algorithms to improve their perfomance. It works by fitting firstly a classifier on the original dataset and later more copies of the classifier on this dataset but with weights of incorrectly classified instances adjusted; in consequence, the classifier focuses more on difficult cases and makes a more accurate classification.

The hyper-parameters used with the AdaBoostClassifier have been:

- ***Base estimator:*** this parameter has to have the constructor of the classifier that is used in conjunction with the AdaBoostClassifier. In this case, the value given is MultinomialNB(alpha=.01).
- ***Number of estimators:*** the highest number of estimators at which boosting is finished.
- ***Learning rate:*** parameter that shrinks the contribution of each classifier.

In Table 3.1, we can see the accuracy of the MultinomialNB classifier with each feature extractor individually. This value has been calculated using k fold with a k=10; for each iteration, the accuracy is calculated. When the cross validation finishes, the mean and the deviation of the accuracy values are calculated, these values are shown in the table.

Feature extractor	**Accuracy and deviation**
Words	0.94 (+/- 0.01)
N-grams	0.94 (+/- 0.01)
POS	0.57 (+/- 0.01)
NER	0.84 (+/- 0.02)
Hashtag	0.66 (+/- 0.02)
Sentiment	0.55 (+/- 0.02)

Table 3.1: MultinomialNB accuracy and deviation with each feature extractor

3.6.2 Support Vector Machine (SVM)

The support vector machine is a supervised machine learning algorithm that draws the examples as points in space and performs classification by searching the hyper-plane which differentiate between the label categories better. When the model has found the best hyper-plane, if it has to predict the label of new data, it will just see to which part of the hyper-plane the new data belongs.

In Fig. 3.11 it can seen graphically what has been described above.

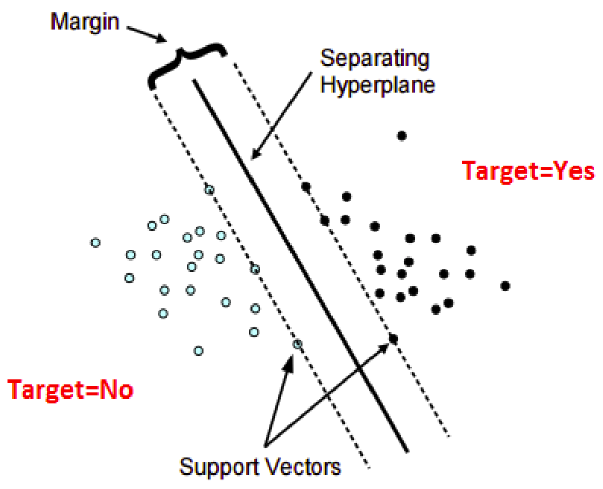


Figure 3.11: SVM [17]

SVM provides two types of algorithms. SVR is the one used for regression problems and SVC, which is the one used in this project, is typically used for classification.

The SVC hyper-parameters whose values have been defined using GridSearchCV are:

- ***C:*** It is the penalty parameter of the error term. We have given to this hyper-parameter the value 10.
- ***Kernel:*** there are five types of kernels, which are linear, polynomial (poly), radial basis function (rbf), sigmoid and precomputed kernel. The kernel selected has been the rbf because it can be seen with GridSearchCV that this kernel had the best results with our classifier.
- ***Gamma:*** it represents the spread of the kernel and, consequently, the decision region. We have given to this parameter the value 1.
- ***Probability:*** parameter whose values are true or false; it indicates whether to enable probability estimates. We have considered the value true.

The mean and deviation of the SVM model accuracy using 10-folds with each feature extractor individually can be seen in Table 3.2:

Feature extractor	**Accuracy and deviation**
Words	0.96 (+/- 0.01)
N-grams	0.96 (+/- 0.01)
POS	0.59 (+/- 0.01)
NER	0.86 (+/- 0.01)
Hashtag	0.67 (+/- 0.01)
Sentiment	0.55 (+/- 0.02)

Table 3.2: SVM accuracy and deviation with each feature extractor

3.6.3 K-nearest neighbors (KNN)

Algorithm based on nearest neighbors learning method. It consists on representing the items contained in the data in the vector space and for a new sample, the algorithm searches the k nearest data items and assigns to the new sample the label more repeated in its k nearest neighbors.

In Fig. 3.12, the process of k-nearest neighbors can be seen. There are crosses and squares, if the star item has to be assigned to one of these two categories, for a k=3, the algorithm will assign to the new sample the crosses category because it sill look for the 3 nearest neighbors of the star and there, crosses are majority. For k=5, the star will have the square category and with k=8 the crosses one.

Scikit-learn provides a classifier based k-nearest neighbors algorithm, which is called KNeighborsClassifier. For this algorithm, the following hyper-parameter values have been used:

- ***P:*** it has been assigned to P hyper-parameter the value 2, which means that the algorithm looks for the nearest neighbors using the eucledian distance.
- ***Number of neighbors:*** this is the value of k, it has been assigned to this hyper-parameter the value 13 using GridSearchCV.

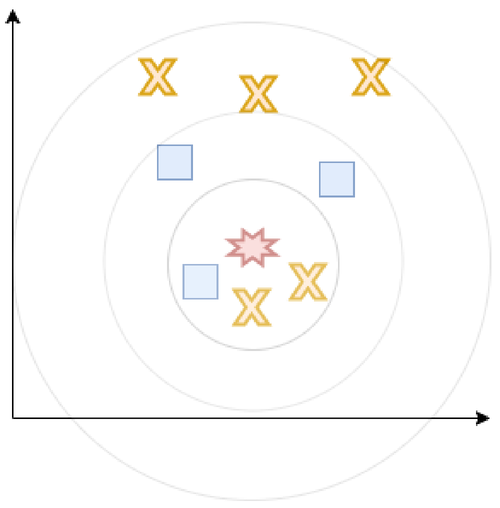


Figure 3.12: K-nearest neighbors

Finally, it can be seen in Table 3.3 the mean and deviation of the accuracy of the KNN algorithm with 10-folds for each feature extractor:

Feature extractor	**Accuracy and deviation**
Words	0.81 (+/- 0.01)
N-grams	0.81 (+/- 0.01)
POS	0.65 (+/- 0.01)
NER	0.85 (+/- 0.01)
Hashtag	0.67 (+/- 0.04)
Sentiment	0.55 (+/- 0.04)

Table 3.3: KNN accuracy and deviation with each feature extractor

3.7 Conclusion

To conclude this chapter, in Fig. 3.13 the process of our classifier that has been explained can be seen.

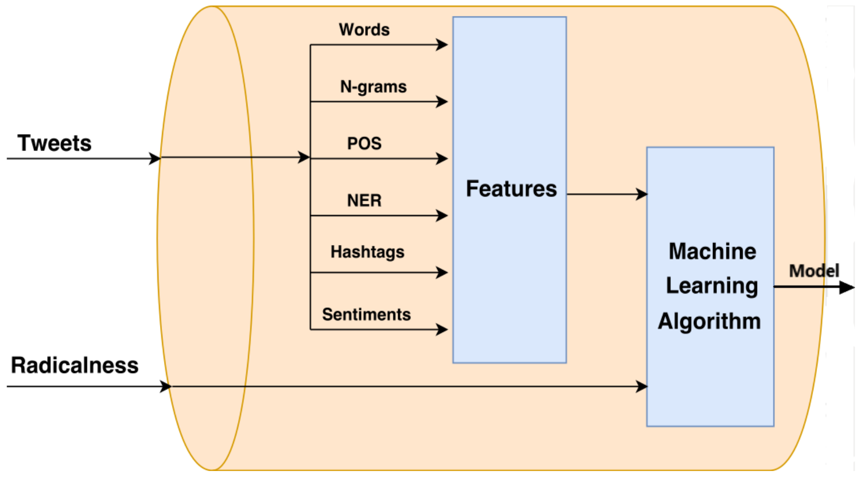


Figure 3.13: Pipelines chema

The accuracy and deviation of each algorithm explained in this chapter (MultinomialNB, SVM and KNN) using all the feature extractors (words, n-grams, POS, NER, hashtag and sentiment) that were joined in a pipeline together can be seen in the Table 3.4:

Algorithm	**Score**
MultinomialNB	0.75 (+/- 0.02)
SVM	0.92 (+/- 0.01)
KNN	0.91 (+/- 0.01)

Table 3.4: Accuracy and deviation of each algorithm using all the features extractors

As it can be seen in 3.6.1, the MultinomialNB algorithm best accuracy and deviation is with the word or the n-gram extractor and using all the extractors this accuracy hasn't been improved.

With SVM, the best result is also using the word or the n-gram extractor and this accuracy hasn't been improved.

With KNN, as we can see in 3.6.3, the best result is using the NER extractor; combining all the extractors has improved its accuracy.

The objective of this project is creating the best accurate model and because of that, we have tried all the possible feature extractors combinations.

The combinations that gave the best results were:

- ***MultinomialNB:*** The best results for this algorithm were obtained using the model with the word, n-gram and hashtag extractors together.
- ***SVM:*** the more accurate model was achieved using the word and n-gram extractor.
- ***KNN:*** the best results for this classifier were obtained using all the extractors together.

In the Table 3.5, the results obtained with these combinations (which are our best accuracy) can be seen:

Algorithm	**Score**
MultinomialNB	0.95 (+/- 0.00)
SVM	0.96 (+/- 0.01)
KNN	0.91 (+/- 0.01)

Table 3.5: Best accuracy obtained for each algorithm

CHAPTER 4

Radicalization Monitoring Service

4.1 Introduction

In this chapter, we present a service for monitoring radicalization in Twitter, which consists on analyzing Twitter data with the classifier developed in this project and showing a dashboard with the discovered potential radicalized accounts.

We have encapsulated our classifier in a Senpy plug-in with the purpose of analyzing tweets and, with the objective of creating a dashboard, we have defined a data processing pipeline using the framework Sefarad, because it provides tools for reusing Senpy plug-ins that allows us to ingest data from Twitter, carrying out the radicalization analysis with Senpy and storing the analyzed data in ElasticSearch so that other tools can exploit this data as it is described below.

The schema of our service is shown in Fig. 4.1 where it can be seen that Sefarad uses Luigi [19] module as orchestrator; with a pipeline developed with this module, Sefarad loads the data extracted from the Twitter API, passes this data to our Senpy plug-in, analyzes the data using the plug-in and loads it on ElasticSearch for creating our dashboard.

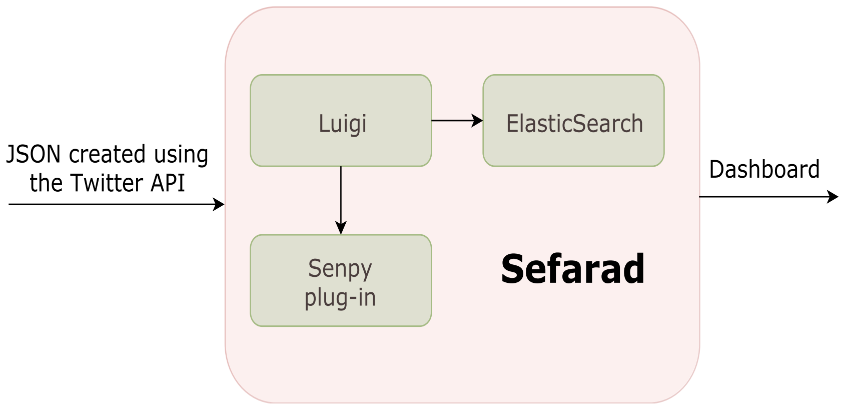


Figure 4.1: Service for monitoring radicalization in Twitter schema

4.2 Plug-in

The service created for analyzing the radicalization of an inserted text has been developed using the Senpy API which allows the user to create its own plug-in easily and deploying it in a Senpy server. A plug-in is a software program which adds the desired feature to an existing application.

In Fig. 4.2, the Senpy architecture can be seen.

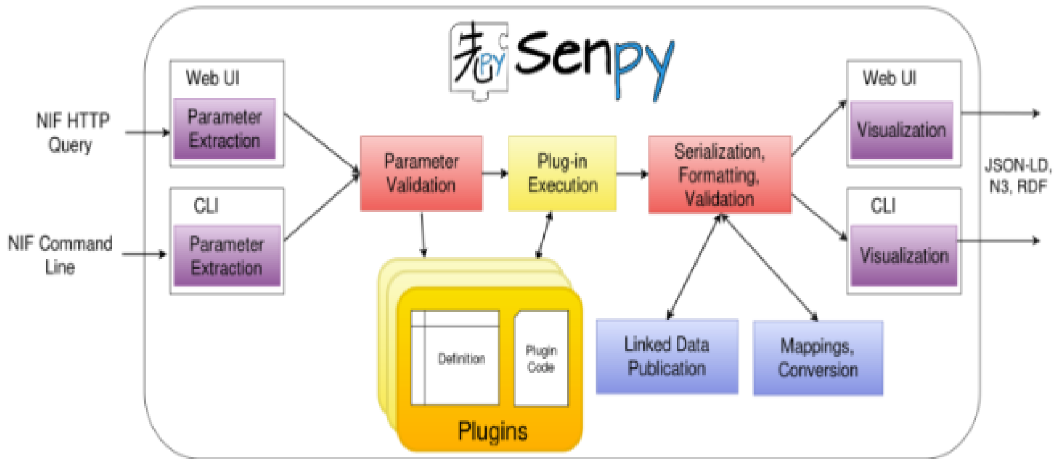


Figure 4.2: Senpy architecture

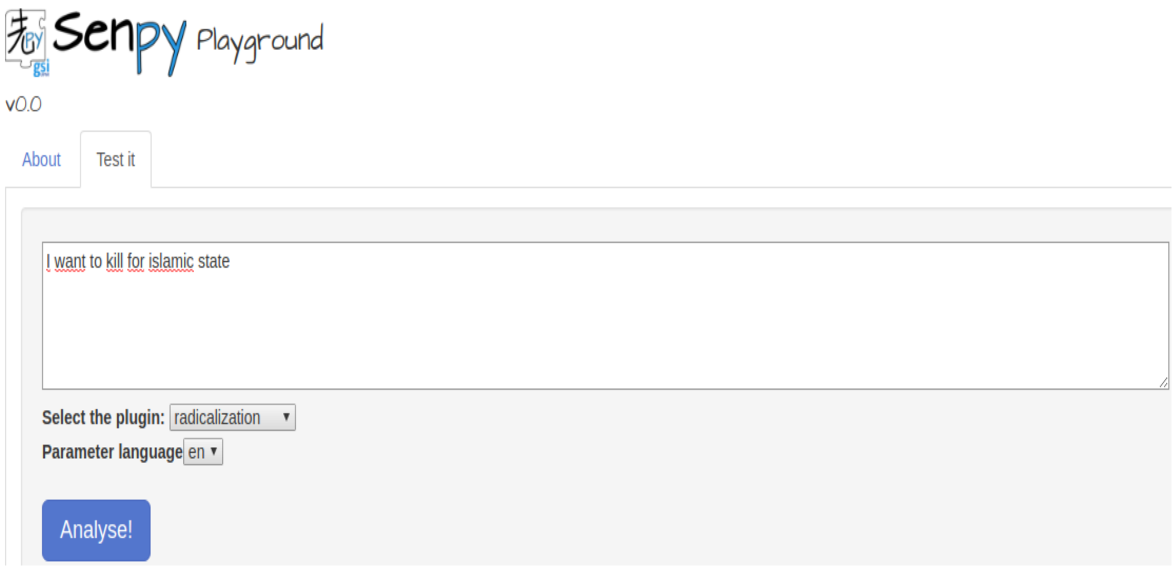
Senpy's architecture is composed by two main modules: the Senpy core, which is the building block of the service and the Senpy plugins (in yellow in the above figure), that is the part that we have to implement; it consists on the analysis algorithm and that we have to implement for using our classifier for analyzing texts.

In order to create our own plug-in, a file with senpy extension has to be created containing the name and the module name of our plug-in, a description, the author, extra parameters if required and libraries required.

After creating the senpy file, it is necessary to create a file with extension py (python file) containing the development explained in chapter 3, it means that our preprocessing methods, feature extractors and our pipeline have to be added to this file in order to use the model created with our pipeline in the plug-in.

The last part of our extension py file is adding a method where we have to take the text inserted in the Senpy playground, using our classifier and returning as result a JSON format text containing in the field (jihad:is_jihadist) a true if the classifier determines that the text inserted in the playground is radical or false if not.

After including our senpy and python file, we had our Senpy plug-in implemented. Deploying it on the Senpy server shows the Senpy playground as seen in Fig. 4.3.



I want to kill for islamic state

Select the plugin: radicalization

Parameter language en

Figure 4.3: Senpy playground

After inserting a text and pushing the *Analyse!* button, we have as a result the JSON with the field jihad:is_jihadist indicating if our classifier has determined that the text is radical or not as seen in Fig. 4.4.

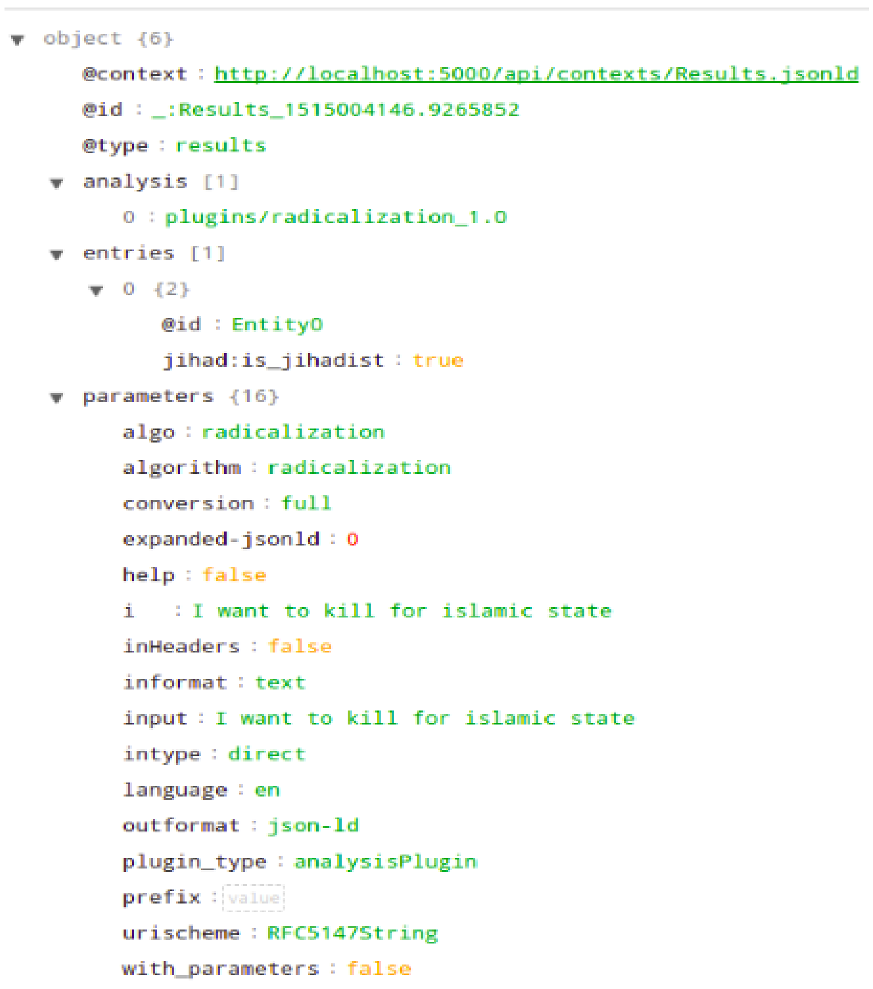

```
▼ object {6}
    @context : http://localhost:5000/api/contexts/Results.jsonld
    @id : _:Results_1515004146.9265852
    @type : results
  ▼ analysis [1]
      0 : plugins/radicalization_1.0
  ▼ entries [1]
    ▼ 0 {2}
        @id : Entity0
        jihad:is_jihadist : true
  ▼ parameters {16}
      algo : radicalization
      algorithm : radicalization
      conversion : full
      expanded-jsonld : 0
      help : false
      i   : I want to kill for islamic state
      inHeaders : false
      informat : text
      input : I want to kill for islamic state
      intype : direct
      language : en
      outformat : json-ld
      plugin_type : analysisPlugin
      prefix : value
      urischeme : RFC5147String
      with_parameters : false
```

Figure 4.4: Radicalism Senpy plug-in response

4.3 Dashboard

The dashboard, which is going to use our Senpy plug-in, has been created using the Sefarad API. It already provides dashboards ready to use. In order to employ its dashboards in our project, some modifications must be done.

In this section, the Sefarad processing pipeline seen in (Fig. 4.5) and the modifications made are explained.

4.3.1 Ingestion

The first step for creating our dashboard is downloading the data that is being represented, to do so, first of all, we have to introduce what our dashboard includes.

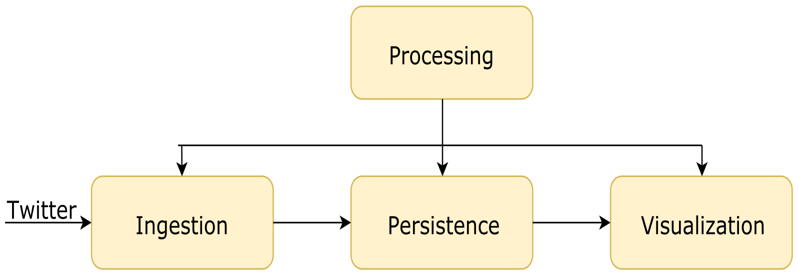


Figure 4.5: Sefarad Processing Pipeline

For our dashboard, an object called *myuser* containing the above fields is created in a JSON format.

Tweets have been downloaded using the Twitter API. We have to create our Twitter credentials for getting the consumer key, token key, consumer secret and token secret, which are fields required in order to make requests for downloading tweets.

In the Twitter API, we have to distinguish between the tweet object, which contains the information about a tweet and the user object, which contains information about an user.

In our case study, we have downloaded tweets including the queries isis, islamic state and syria, that appear frequently in our datasets as seen in chapter 3 wordclouds, these tweets downloaded are extracted with a JSON format.

A JSON object is structured as a dictionary, it has the pair key-value and we can access to its values using the keys contained in the JSON. Each tweet includes a user object inside, which can be accessed using the key *user* of the tweet.

The user object is also in JSON format. As we said before, the user object name and its image are needed. These fields can be extracted using the following user object keys:

Key name	**Description**
screen name	String representing the account name defined by the user.
profile image url	String with the URL of the profile image.

For each user object, we have to show its last 10 tweets and analyzing the radicalization of each of them, this is why last 10 tweets included in their timeline are downloaded. From each tweet, the following tweet object values are shown:

Key name	**Description**
id	Integer which represents the tweet. This number is unique, there can't be two tweets with the same id.
text	String representatoin of the tweet with UTF-8 format.
user	Profile name that has published the tweet.
created at	String which contains the date when the tweet was published.

4.3.2 Processing and Persistence

The Sefarad API uses Luigi for creating pipelines in order to facilitate analysis. The Sefarad pipeline developed with Luigi allows us to read the JSON file containing the data downloaded from Twitter and send this data to our Senpy plug-in with the objective of analyzing the radicalization of the given tweets.

For representing our *myuser* object, we must dump the result of the analysis on ElasticSearch, which is the Sefarad persistence layer and stores the data needed for visualization.

To do so, the Sefarad Luigi pipeline uses the index method provided by ElasticSearch is used. It adds or updates a JSON document in a specific index, making it searchable.

After, Sefarad takes the data dumped out on ElasticSearch, showing the data is carried out by modifying the web components for displaying just the ones wanted (wordcloud, sentiments, radicalization and *myuser* objects).

4.3.3 Visualization

A dashboard is composed by web components, which is the name given by W3C to the technology standard that allows us to create our own web elements, that after can be encapsulated between HTML5 tags in our principal webpage. Sefarad uses Polymer, which is a JavaScript library in which web components can be created.

In our dashboard, the following components are represented:

- ***Wordcloud:*** most frequent words included in the tweets downloaded with different sizes depending on their frequency.
- ***Sentiments:*** percentage of tweets with positive, negative or neutral polarity; this analysis is made with meaning cloud.
- ***Potencial radicalist users:*** in this component, users are ordered by the number of tweets classified as radical in their timeline, each user can be seen with the following fields:
  - ***User name:*** name of the twitter account.
  - ***Image:*** user profile image if the API was able to download it.
  - ***Count:*** number of tweets classified as radical by the model explained in chapter 3 and used in the Senpy plug-in.
  - ***Radicalization:*** if the user count is greater than 3, the user is labeled as radical, if the count is smaller than 3, the user is labeled as not radical.
  - ***Tweets:*** last 10 tweet objects extracted of the user timeline.
- ***Radicalization:*** percentage of users classified as radical and not radical.

Sefarad includes dashboards ready to be used, in this case, the demo-dashboard that this API provides is used and the web components wanted, with the object *myuser*, are added to this dashboard, having as a result the dashboard shown in Fig. 4.6.

We can select the radicalism percentages and choose seeing just the potential radical users or the not radical ones. If we select a user, we can see its last tweets marked each tweet with the color of its sentiment polarity as seen in Fig. 4.7.

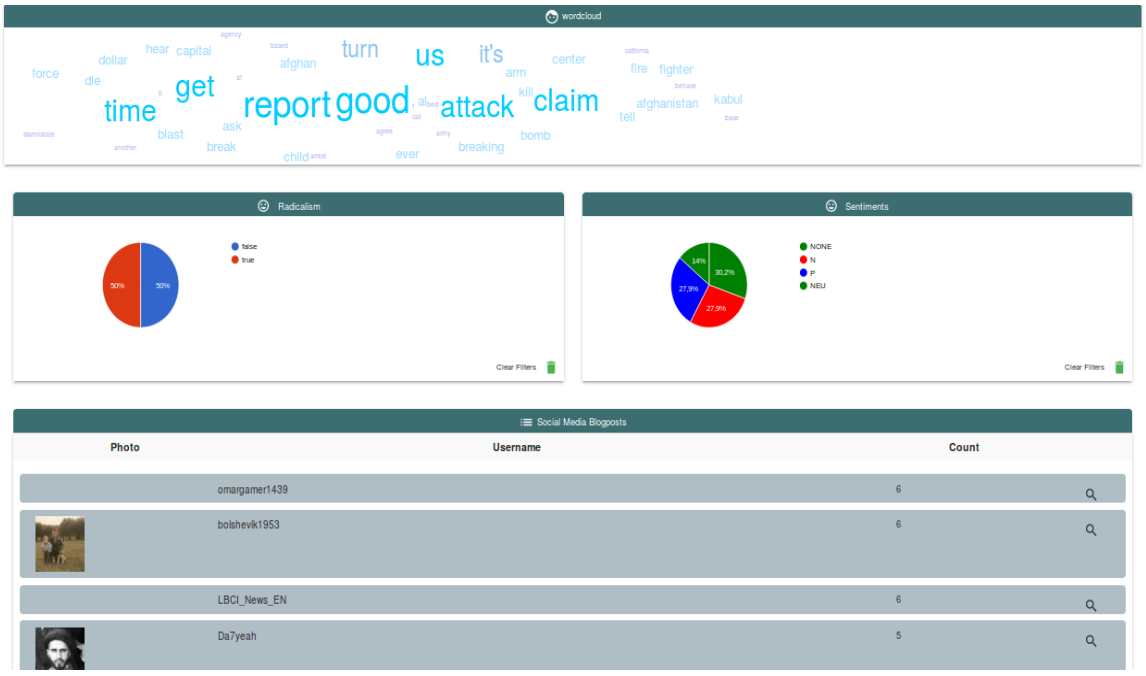
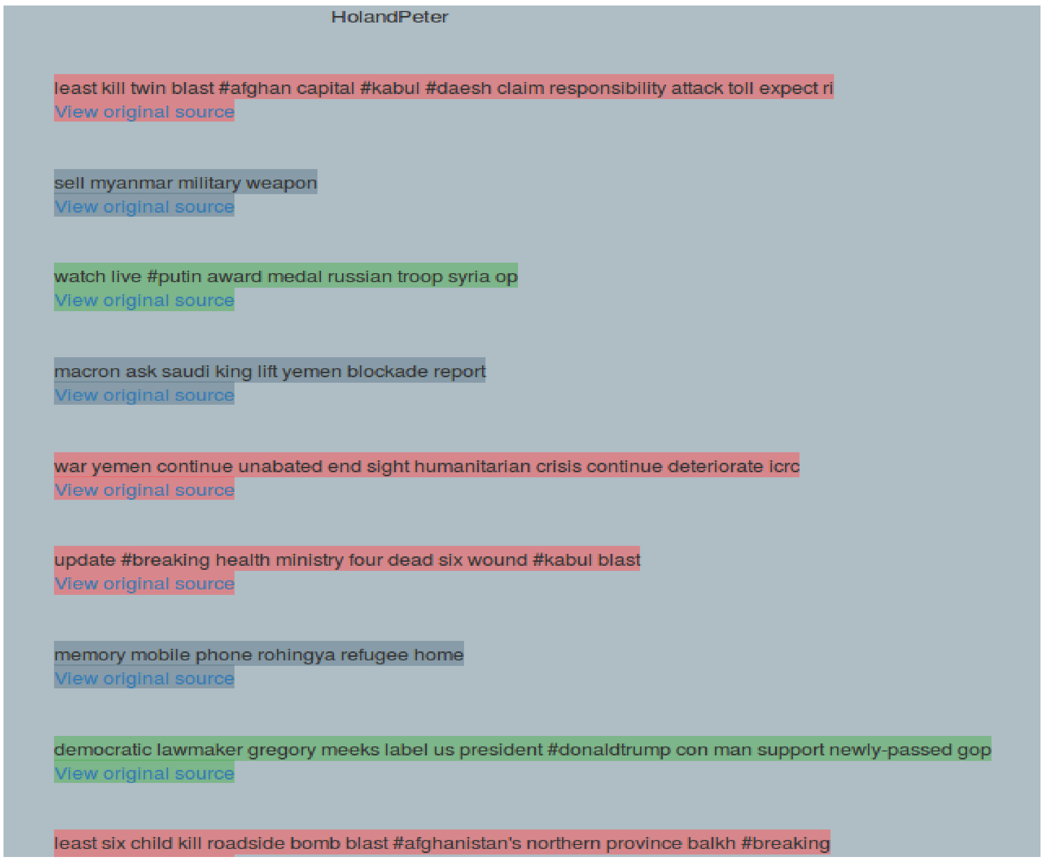


Photo	Username	Count
	omargamer1439	6
	bolshevik1953	6
	LBCI_News_EN	6
	Da7yeah	5

Figure 4.6: Radicalization dashboard


HolandPeter

least kill twin blast #afghan capital #kabul #daesh claim responsibility attack toll expect ri
View original source

sell myanmar military weapon
View original source

watch live #putin award medal russian troop syria op
View original source

macron ask saudi king lift yemen blockade report
View original source

war yemen continue unabated end sight humanitarian crisis continue deteriorate icrc
View original source

update #breaking health ministry four dead six wound #kabul blast
View original source

memory mobile phone rohingya refugee home
View original source

democratic lawmaker gregory meeks label us president #donaldtrump con man support newly-passed gop
View original source

least six child kill roadside bomb blast #afghanistan's northern province balkh #breaking

Figure 4.7: Radicalization dashboard

CHAPTER 5

Conclusions and future work

5.1 Introduction

This chapter explains the conclusions obtained during the implementation of this project, the goals achieved, problems faced and there are also some suggestions for future developments.

5.2 Conclusions

In this project, we have developed a radicalization classifier and a dashboard for analyzing if particular users who published tweets containing terrorism related words are radical or not.

The objective was obtaining the best possible accuracy in the tweets downloaded from the platform Kaggle. First of all, the accuracy with individual features was extracted but in order to get better models, we had to combine different features in order to analyze if the classifier improved its accuracy or not.

To do so, we developed six feature extractors (words, n-grams, POS, NER, Hashtags and Sentiment), the objective was having stylometric features that could help the classifier algorithm in order to have better results.

We employed the most used machine learning algorithms for classification: Multinomial Naive Bayes, Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

As it can be seen in the results, the features that gave best accuracies were the word and n-gram extractors, both transformers had the same accuracy, it can be because the n-gram extractor (its features include from one token to three) includes the word extractor feature names (features composed just by one token) followed by 2-grams and 3-grams, which don't improve the word extractor accuracy.

The second best extractor was the NER, this is because locations, people and organizations named in both datasets can be differentiated easily.

The third best accuracy has been got with the hashtag extractor, it's accuracy is greater than sixty percent with the three algorithms, so they can distinguish between hashtags of each dataset.

The POS extractor had a accuracy of about sixty percent, which mean that the grammar categories can be used for differing datasets.

The transformer that gave worse results was the sentiment extractor with accuracies smaller than sixty percent. It happens because there is almost no difference in the polarity texts of isis fans and its counterpoise dataset as we saw in the sentiment analysis results; both datasets have a big amount of tweets with negative polarity because they talk about terrorist attacks either informing or proclaiming and positive tweets talking about military victories.

Finally, combining the extractors, with MultinomialNB and with KNN, we were able to improve its accuracy; in the other hand, with SVM there wasn't a better classifier than the one using just the words or the n-gram extractor.

The other fulfilled goal was implementing an interface for using our classifier and watching its classification results, we could make it possible thanks to the Sefarad API by just using the provided demo-dashboard, adding the fields that we wanted to have and inserting the web components desired.

5.3 Future work

In this section, some topics for future developments are commented.

- **Adding new tweets to our database:** in our project we have used a database including terrorism related tweets, with fifty percent of tweets labeled as radical and other fifty as not radical but this is not realistic because this type of content doesn't appear in a very extensive amount of tweets.

- **Adding a topic classifier:** another possible future development consists on implementing a topic classifier before the classifier developed in this project. If the topic classifier determines that the tweet topic is terrorism, then our classifier has to be used in this tweet; if the topic is different, the tweet has to be automatically labeled as not radical.

- **Improving Sefarad:** during this project, we have used our classifier with the Sefarad API for creating a dashboard. During the development, we have found the problem that Sefarad hasn't the option of inserting scikit-learn's pipelines in order to have a field that includes a classification result for a dashboard created. It is proposed to create a mechanism for allowing Sefarad users to insert their pipeline for visualizing its results in a dashboard.

- **Creating a translator API:** during this project, we have just analyzed tweets written in English but a big amount of tweets that talk about ISIS is posted in Arabic. We propose as future work to create an API that translates texts offline.

Bibliography

[1] Amueller. Word cloud generator in python. https://github.com/amueller/word_cloud, 2017.

[2] Akil N Awan. Radicalization on the internet? the virtual propagation of jihadist media and its effects. *RUSI*, 152(3):76–81, 2007.

[3] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[5] Enrique Conde-Sánchez. Development of a Social Media Monitoring System based on Elasticsearch and Web Components Technologies. Master's thesis, Universidad Politécnica de Madrid, 2016.

[6] Bharvi Dixit. *Elasticsearch Essentials*. Packt Publishing Ltd, 2016.

[7] Active Galaxy. Tweets targeting isis. https://www.kaggle.com/activegalaxy/isis-related-tweets/data, 2016.

[8] Raúl Garreta and Guillermo Moncecchi. *Learning scikit-learn: machine learning in python*. Packt Publishing Ltd, 2013.

[9] Narong Intiruk. Begin with machine learning. https://www.slideshare.net/NarongIntiruk/begin-with-machine-learning-71507363, 2017.

[10] Krasimira Kapitanova and Sang H Son. Machine learning basics. *Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning*, 13, 2012.

[11] Farhad Khosrokhavar. *Inside jihadism: understanding jihadi movements worldwide*. Routledge, 2015.

[12] Amit Kumar. Supervised and unsupervised learning. http://www.allprogrammingtutorials.com/tutorials/introduction-to-machine-learning.php, 2015.

[13] MeaningCloud LLC. Meaning cloud api. https://www.meaningcloud.com/es/, 2017.

[14] Fabien Lotte. Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain–computer interfaces. 103:871–890, 06 2015.

[15] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9, 2011.

[16] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[17] David Meyer and FH Technikum Wien. Support vector machines. *R News*, 1(3):23–26, 2001.

[18] Enghin Omer. Using machine learning to identify jihadist messages on twitter, 2015.

[19] Tina Ranic and Marjan Gusev. Overview of workflow management systems. 2017.

[20] Hassan Saif, Thomas Dickinson, Leon Kastler, Miriam Fernandez, and Harith Alani. *A Semantic Graph-Based Approach for Radicalisation Detection on Social Media*, pages 571–587. Springer International Publishing, Cham, 2017.

[21] J Fernando Sánchez-Rada, Carlos A Iglesias, Ignacio Corcuera, and Óscar Araque. Senpy: A pragmatic linked sentiment analysis framework. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 735–742. IEEE, 2016.

[22] Javier Solana and Lluís Basset. *Reivindicación de la política: veinte años de relaciones internacionales.* Debate, 2010.

[23] Ramón Spaaij. The enigma of lone wolf terrorism: An assessment. *Studies in Conflict & Terrorism*, 33(9):854–870, 2010.

[24] Fifth Tribe. How isis uses twitter. `https://www.kaggle.com/fifthtribe/how-isis-uses-twitter/data`, 2015.

[25] u. Xie, J. Xu, and T. C. Lu. Automated classification of extremist twitter accounts using content-based and network-based features. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2545–2549, Dec 2016.

[26] Y. Wei, L. Singh, and S. Martin. Identification of extremism on twitter. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1251–1255, Aug 2016.

[27] Wikipedia. K-fold cross-validator. `https://en.wikipedia.org/wiki/Cross-validation_(statistics)`, 2017.