SOCIAL MEDIA MINING FOR HEALTH MONITORING

~ TEAM STELLAR

Prudhveer Reddy Kankar, Sai Shanthan Venreddy, Sankeerth Tella

Abstract

The data collected for social media if used efficiently can predict and analyze various situations and create wonderful Machine Learning models. One among them is to use the social media mining for health monitoring purposes. In this project we use Twitter posts, to retrieve the adverse effects of various drugs used by the tweeters. In order to do this the data has to be processed because the language used by the people while tweeting is raw and unstructured. After pre-processing we have to classify between the tweets containing adverse effects from those which don't have any such mentions. The next step is feature extraction where we use named entity recognition to retrieve the span of the tweet which contains the mention of ADR. The third step is knowledge extraction using machine learning techniques for text classification which includes model training and testing. A tweet is then classified into either related or unrelated. Then we link this tweet containing the related and proper ADR to standard concept IDs in the MedDRA vocabulary (lower level terms). Finally, we build a website for visualization.

1 Introduction

Social media is a popular medium for the public to voice their opinions and thoughts on various health related topics. A recent Pew Research Center study says that nearly half of adults worldwide and two-thirds of all American adults use social networking on a regular basis. Due to the wealth of data available, researchers have been analyzing social media data for health monitoring and surveillance. However, social media mining for health issues is fraught with many linguistic variations and semantic complexities in terms of the various ways people express medication-related concepts and outcomes. This project requires processing imbalanced, noisy, real-world, and substantially creative language expressions from social media to extract and classify mentions of adverse drug reactions (ADRs) in tweets.

Twitter is one of the most popular resources, with 289 million monthly active users, 58 million tweets per day, and 2.3 billion search queries per day. Social media is currently being used as a resource for various tasks ranging from customized advertising and sentiment analysis to tasks specific to the public health domain (e.g., monitoring influenza epidemics, sexual health, pharmacovigilance, and drug abuse. The project focuses on performing pharmacovigilance from social media data. It is now well understood that social media data may contain reports of adverse drug reactions (ADRs) and these reports may complement traditional adverse event reporting systems, such as the FDA adverse event reporting system (FAERS). However, automatically curating reports from adverse reactions from Twitter requires the application of a series of NLP methods in an end-to-end pipeline. The three tasks of this project represent three key NLP problems in a social media based pharmacovigilance pipeline — (i) automatic classification of ADRs, (ii) extraction of spans of ADRs and (iii) normalization of the extracted ADRs to standardized IDs.

2 Literature Review

One of the earliest works similar to our project we are doing now was done in 2016. The goal of this paper [1] is to attract researchers that have explored automatic methods for the collection, extraction, representation, analysis, and validation of social media data for public health surveillance and monitoring, including epidemiological and behavioral studies. It discusses novel approaches to text and data mining methods that respond to the specific requirements of social media and that can prove valuable for public health surveillance. This paper also surveys recent research on using social media data to study public health.

Another paper [2] which deals with the same issue was also published in 2016. In this paper the writer has taken three popular medications which are proven to be abusive in nature and determined their relative abusiveness percentages. The writer took a positive drug which is proven to have no abusiveness factor and used it as a metric to make those determinations. This entire process was done in three stages. In the first stage the data is collected from twitter, only those tweets which contain the names of any of the four drugs were retrieved. In the second stage, Cohens Kappa was used to generate testing and training data which will be later used in the third stage. In the third stage, supervised learning techniques were used to categorize the data into Abusive, No-Abusive and Unclear categories. The annotated data generated previously was used to measure accuracy and other evaluation metrics.

Another paper[3] discusses the project which is almost similar to our project. As in previous years, the tasks focused on mining health information from Twitter. This paper challenged the community with new problem. The problem focuses on performing pharmacovigilance from social media data. It is now well understood that social media data may contain reports of adverse drug reactions (ADRs) and these reports may complement traditional adverse event reporting systems, such as the FDA adverse event reporting system (FAERS). However, automatically curating reports from adverse reactions from Twitter requires the application of a series of NLP methods in an end-to-end pipeline.

3 Dataset

In each task the data set is modified as per the requirement of the task working towards the final goal. The data is taken from Social Media Mining for Health Applications (#SMM4H) workshop.

Task 1: From given data, we have to classify the tweets containing an Adverse Effect (AE) from those that do not. The data provides is 55,420 Tweets. 146 Positive, 55274 Negative.

Format of the data: TWEETID <-tab-> USERID <-tab-> TWEET<-tab->CREATED_AT <-tab-> CLASS (1=ADR; 0=NON-ADR)

Task 2: From the given tweets, we identify the text span of the reported ADRs in the tweets. Then we check if its relevant ADR or non-relevant ADR and then we try to extract the ADR. 2247 Tweets.

Format of the data: TWEETID <-tab-> BEGINNING OF ADR<-tab-> ENDING OF ADR IN TWEET <-tab-> ADR (OR NOT) <-tab-> ADR EXTRACTION <-tab-> DRUG <-tab-> TWEET <-tab-> MEDDRA CODE<-tab-> MEDDRA TERM

Task 3: Tweets are linked to standard concept IDs in the MedDRA vocabulary using extracted ADRs (lower level terms).

Format of the data: TWEETID <-tab-> BEGINNING OF ADR<-tab-> ENDING OF ADR IN TWEET <-tab-> ADR (OR NOT) <-tab-> ADR EXTRACTION <-tab-> DRUG <-tab-> TWEET <-tab-> MEDDRA CODE<-tab-> MEDDRA TERM

4 Architecture

Task 1: Automatic classification of tweets mentioning an ADR. This is a binary classification task for which systems are required to predict if a tweet mentions an ADR or not.

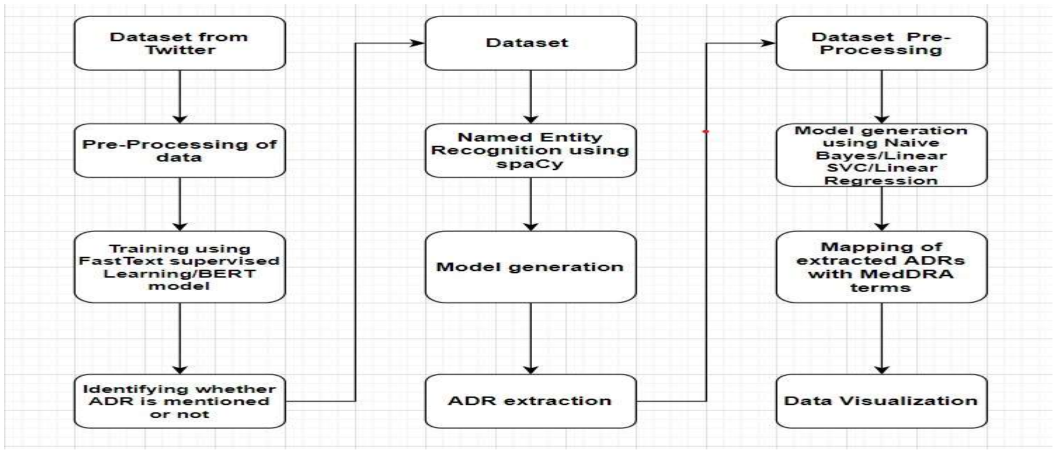

Dataset from Twitter
Pre-Processing of data
Training using FastText supervised Learning/BERT model
Identifying whether ADR is mentioned or not
Dataset
Named Entity Recognition using spaCy
Model generation
ADR extraction
Dataset Pre-Processing
Model generation using Naive Bayes/Linear SVC/Linear Regression
Mapping of extracted ADRs with MedDRA terms
Data Visualization

Figure 1: Architecture

Task 2: Automatic extraction of ADR mentions from tweets. This is a named entity recognition (NER) task that typically follows the ADR classification step (Task 1) in an ADR extraction pipeline.

Task 3: Automatic extraction of ADR mentions and normalization of extracted ADRs to MedDRA preferred term identifiers. This is an extension of Task 2 consisting of the combination of NER and entity normalization tasks: a named entity resolution task.

After completion of these tasks, data visualization is done using Tableau.

5 Methodology

We have applied various techniques to meet the required objectives of all tasks and they are elaborated below:

Task 1: Automatic classification of adverse effects mentions in tweets

We have implemented two models for task 1. They are BERT and FastText.

Fasttext:

FastText is a model developed by Facebook especially for word embeddings and text classification. It is a significant improvement over Word2Vec model because of it ability to deal with words which are out of corpus using word-N-grams. There are two models in FastText: Skip Gram and CBOW. We mainly replied on CBOW model beaus of its unique ability to extract the context of the sentence which is very significant for our task.

Preprocessing:

Data cleaning is a very important task. To obtain the best results, we have to make sure that the data is something close to proper English, and because we work on tweets, this is no easy task. We have done preprocessed the tweets by removing emojis, HTML characters, hashtags, punctuations etc. We internally did not remove stop words because removing them might change the context of the tweet.

FastText Prerequisites:

Fasttext needs labeled data to train the classifier. For example positive tweets need to be labeled as _label_positvive and negative tweets need to be labeled as _label_negative.

Under Sampling:

Because the dataset provided was highly skewed with very low amount of positive tweets compared to that of negative tweets, we had to perform under sampling of data by experimenting choosing 400,500, 600 negative tweets. The best score was observed for 600 tweets.

Training the model:

With our data prepared we have to use the function fastText.train_supervised to train the model. We have used the following parameters:

- lr: Learning rate. We set it at 0.0001.
- epoch: Number times we go through the entire dataset. We use 100.
- wordNgrams: An n-gram is a contiguous sequence of max n words from a given sample of text, tweet here. We set it at 2.
- dim: Dimension of word vector. We use 10.

Once trained, we need to assess how good our model is. For this, we can use the two measures Precision and Recall which are the output of fastText functionmodel.test. However, due to the nature of our problem, precision and recall give similar figures and we can focus on f1 score only.

BERT:

BERT stands for "Bidirectional" Encoder Representations from Transformer. We have to focus on the term Bidirectional because traditional models either get trained form left to right or from right to left or both. But BERT model is trained by using all the words at once. Hence it is called so.

Tokenization:

For this task we use the standard BERT tokenizer. To feed the text to BERT, it must be split into tokens and these tokens must be mapped to their index in tokenizer vocabulary. We add special tokens to each input by prepending CLS token to the beginning of every sentence and SEP to the end of the sentence. This token has special significance. BERT consists of 12 Transformer layers. Each transformer takes in a list of token embeddings, and produces the same number of embeddings on the output. On the output of the final (12th) transformer, only the first embedding (corresponding to the [CLS] token) is used by the classifier.

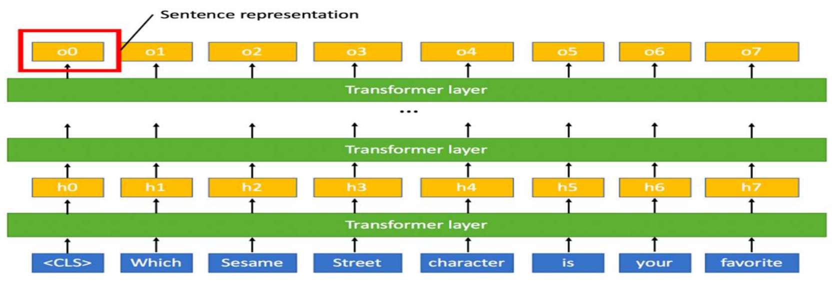


Figure 2: BERT Model

It is important in BERT to fix the length of the text input. We have a maximum of 81 tokens per input and therefore we maintained a standard length of 128 tokes by padding 0's to the end of each input.

Training the model:

We'll be using BertForSequenceClassification. This is the normal BERT model with an added single linear layer on top for classification that we will use as a sentence classifier. As we feed input data, the entire pre-trained BERT model and the additional untrained classification layer is trained on our specific task. AdamW is a class from the huggingface library. For the purposes of fine-tuning, the authors recommend choosing from the following values:

- Learning rate (Adam): 2e-5
- Number of epochs: 4
- eps: 1e-5.

Next step is to train the loop. fundamentally for each pass in our loop we have a trianing phase and a validation phase. At each pass we need to:

Training loop:

- Forward pass (feed input data through the network)
- Backward pass (backpropagation)
- Tell the network to update parameters with optimizer.step()
- Track variables for monitoring progress

Evaluation loop:

- Forward pass (feed input data through the network)
- Compute loss on our validation data and track variables for monitoring progress.

For evaluating this model we have specifically used Mathews Correlation Coefficient as performance measure (-1 to +1). Because it takes into account True Negative score which is not used in calculating precision, recall and f1. The MCC score is 0.771.

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

Task 2: Extraction of Adverse Effect mentions

This task deals with extracting the ADRs from the given tweet, if exists. We identify the text span of the reported ADRs and distinguish ADRs from similar non-ADR expressions. We used advanced named entity recognition (NER) approach using spaCy. spaCy is regarded as the fastest NLP framework in Python, with single optimized functions for each of the NLP tasks it implements.

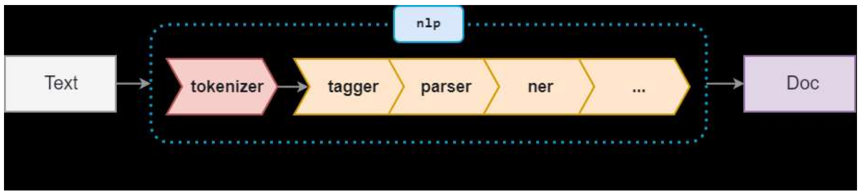


Figure 3: SpaCy NER

To obtain the best results, we have to make sure that the data is something close to proper English, so we do some data cleaning. We have preprocessed the tweets by removing emojis, HTML characters, hashtags, punctuations etc. We internally did not remove stop words because removing them might change the context of the tweet. Further, we also converted all the text to lowercase.

Next, we do ADR tagging in which data consists of the starting and end points of the ADR and ADR tag. We now send this data for NER processing using SpaCy. SpaCy provides an exceptionally efficient statistical system for NER in python, which can assign labels to groups of tokens which are contiguous. Apart from these default entities, spaCy also gives us the liberty to add arbitrary classes to the NER model, by training the model to update it with newer trained examples.

We have used en_core_web_sm ,which is a core model. Assigns context-specific token vectors, POS tags, dependency parse and named entities. Using this core model, we train our model. Then, we use GoldParse object, which collects the annotated training examples, also called the **gold standard**. It's initialized with the Doc object it refers to, and keyword arguments specifying the annotations, like tags or entities. Its job is to encode the annotations, keep them aligned and create the C-level data structures required for efficient access.

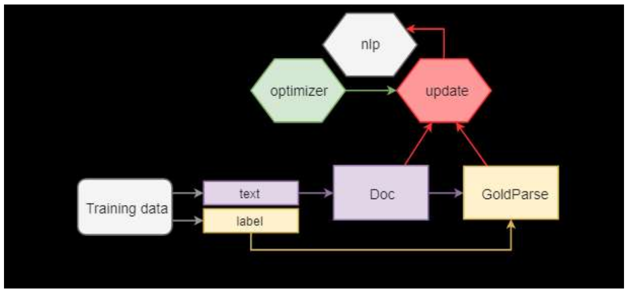


Figure 4: Gold Parse

Training the model:

- Load the model, set up the pipeline and train the entity recognizer
- create the built-in pipeline components and add them to the pipeline.nlp.create_pipe works for built-ins that are registered with spaCy
- add labels
- get names of other pipes to disable them during training
- reset and initialize the weights randomly – but only if we're training a new model
- nlp.begin_training() [begin training]
- batch up the examples using spaCy's minibatch

Especially if you only have few examples, you'll want to train for a number of iterations. At each iteration, the training data is shuffled to ensure the model doesn't make any generalizations based on the order of examples. Another technique to improve the learning results is to set a dropout rate, a rate at which to randomly "drop" individual features and representations. This makes it harder for the model to memorize the training data. So, we also set dropout and update the model. Then, test the saved model.

Task 3: **Mapping of adverse drug reaction mentions (ADR)**

In this task, we try detect tweets mentioning an ADR and then we map the extracted colloquial mentions of ADRs in the tweets to standard concept IDs in the MedDRA vocabulary.

In the pre-processing phase, for this particular data set, our text cleaning step includes HTML decoding, remove stop words, change text to lower case, remove punctuation, remove bad characters, and so on from the given dataset. It also means we can eliminate data which has missing values. For this sub-task,we use dropna function which is used to eliminate unknown values. Data frame is read and all rows with any Null values are dropped.

After text cleaning and removing stop words, the next steps includes feature engineering. We converted our text documents to a matrix of token counts (CountVectorizer), then transform count.matrix to a normalized tf-idf representation (tf-idf transformer). Basically TF-IDF is a smart way of rating importance of a word inside of the document. In our case we will be able to identify what are the relative differences between recipes. We will use scikit-learn built-in TfidfVectorizer. After that, we trained several classifiers from Scikit-Learn library. To make the vectorizer => transformer => classifier easier to work with, we will use Pipeline class in Scilkit-Learn that behaves like a compound classifier. Pipeline of transforms with a final estimator. Sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be 'transforms', that is, they must implement fit and transform methods. The final estimator only needs to implement fit. The transformers in the pipeline can be cached using memory argument.

Naive Bayes Classifier for Multinomial Models: (Baseline Model)

After we have our features, we can train a classifier to try to predict the tag of a post. We started with a Naive Bayes classifier, which provided a nice baseline for this task. scikit-learn includes several variants of this classifier; the one most suitable for text is the multinomial variant. MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). Then, we fit the model after applying pipeline to get the required results and we achieved an F1 score of 0.14.

Linear Support Vector Machine:

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. LinearSVC is another implementation of Support Vector Classification for the case of a linear kernel. Note that LinearSVC does not accept keyword kernel, as this is assumed to be linearSimilar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme. Hence, it is widely regarded as one of the best text classification algorithms.

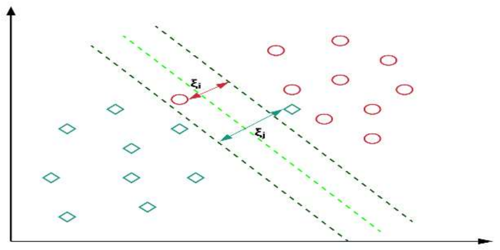


Figure 5:Linear SVM

scikit-learn includes several variants of this classifier; the one most suitable for text is the linearSVM variant. After transforming through the pipeline of countVectorizer, tf-idfVectorizer and linearSVM ,hyper-parameters are tuned and fit to the model to achieve results, which has 0.34 F-1 score and is an improvement over baseline model.

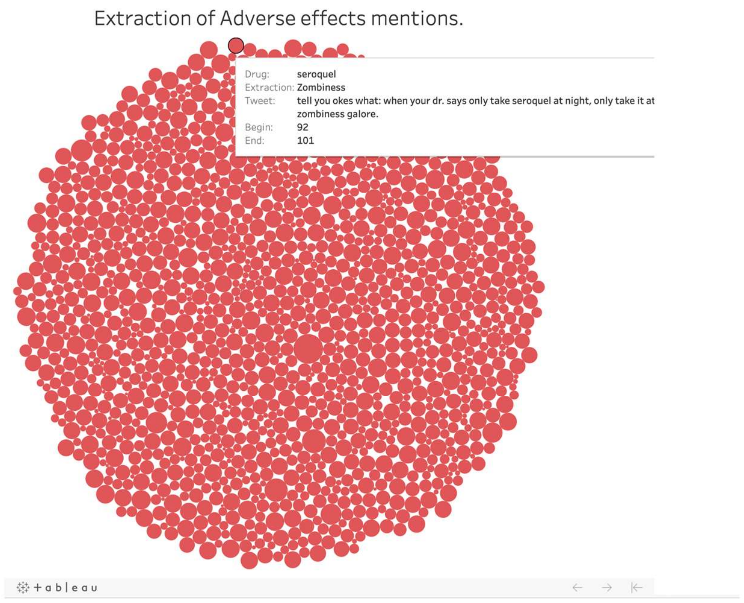
Multinomial Logistic regression(Improved model):

Logistic regression is a simple and easy to understand classification algorithm, and Logistic regression can be easily generalized to multiple classes. Multinomial logistic regression is a form of logistic regression used to predict a target variable have more than 2 classes. scikit-learn includes this classifier; the one most suitable for text is the multinomial variant. After transforming through the pipeline of countVectorizer, tf-idfVectorizer and Multinomial Logistic regression classifier, hyper-parameters are tuned. When training the model, we'll call the fit() method, pass it our training data and labels, batch size and epochs. The results has an F-1 score of 0.35, which is a slight improvement over LinearSVM model.

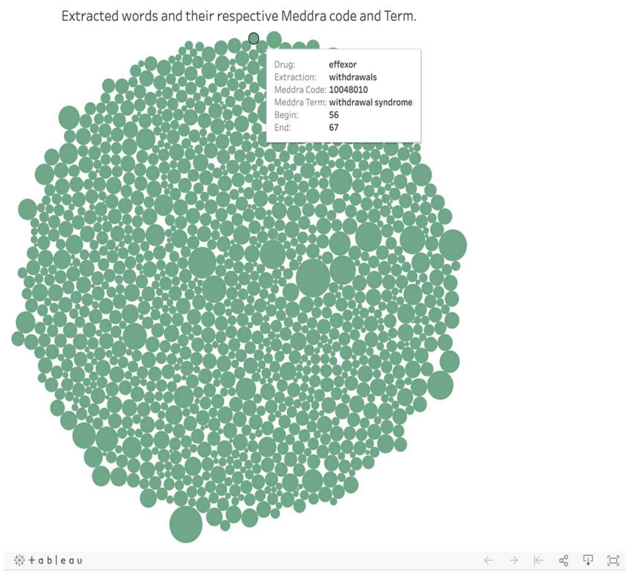
6 Data Visualization

We built a website and hosted it using AWS. The data visualization is done using Tableau and you can find them in following website:

http://ec2-18-216-171-204.us-east-2.compute.amazonaws.com/



Drug: seroquel
Extraction: Zombiness
Tweet: tell you okes what: when your dr. says only take seroquel at night, only take it at
zombiness galore.
Begin: 92
End: 101



Drug: effexor
Extraction: withdrawals
Meddra Code: 10048010
Meddra Term: withdrawal syndrome
Begin: 56
End: 67

Figure 6: Tableau Data Visualization

7 Results

Task 1:

Model	Results
FastText supervised model	F-measure - 0.83
BERT	F-measure - 0.88 Mathews Correlation Coefficient -0.771

Task 2:

Model	Precision	Recall	F-measure
SpaCy-en_core_web_sm	49.54	47.14	48.31

Task 3:

Model	Micro F1-score	Macro F1-score	Weighted F1-score
Multinomial Naïve Bayes	0.16	0.07	0.14
Linear SVM	0.33	0.20	0.34
Multinomial Logistic Regression	0.36	0.21	0.35

8 Conclusion

For Task 1,2,3 depending on the given dataset and the analysis performed we concluded that BERT, SpaCy, Linear SVM/Logistic Regression are the best models respectively.

9 Contribution

Task 1 - Prudhveer Reddy Kankar , Sankeerth Tella

Task 2 – Prudhveer Reddy , Sai Shanthan Venreddy

Task 3 – Sai Shanthan Venreddy, Sankeerth Tella

Presentation, Visualization & Documentation of project – Team

10 References

https://www.researchgate.net/publication303127692_SOCIAL_MEDIA_MINING_FOR_PUBLIC_HEALTH_MONITORING_AND_SURVEILLANCE
https://link.springer.com/content/pdf/10.1007/s40264-015-0379-4.pdf
https://www.aclweb.org/anthology/W19-3203.pdf
https://spacy.io/usage/linguistic-features
https://www.nltk.org/genindex.html
https://cloud.google.com/natural-language
PEW Research Center. Demographics of Social Media Users and Adoption in the United States. 2017. http://www.pewinternet.org/fact-sheet/social-media/. Accessed March 3, 2018.
Kennedy B, Funk C. Public Interest in Science and Health Linked to Gender, Age and Personality. PEW Research Center; 2015. http://www. pewinternet.org/2015/12/11/public-interest-in-science-and-health-linkedto-gender-age-and-personality/. Accessed July 1, 2018.
Social Media Mining for Toxicovigilance: Automatic Monitoring of Prescription Medication Abuse from Twitter Abeed Sarker, Karen O'Connor, Rachel Ginn, Matthew Scotch, Karen Smith, Dan Malone, Graciela Gonzalez
Souvignet J, Declerck G, Asfari H, Jaulent MC, Bousquet C. OntoADR a semantic resource describing adverse drug reactions to support searching, coding, and information retrieval. J Biomed Inform. 2016; 63: 100–107
https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568