Hate speech and offensive language detection in tweets using Data Stream Mining

ARABI Sami, KACIMI Oualid Imad, BOUDEGZDAME Nada Ikram and DJELLAL Aniss Mohamed

Paris Descartes University, 2020

1. Introduction

With the advent of real time online applications that generates a huge amount of data streams such as video surveillance systems, internet traffic, tweets etc. Traditional data mining techniques are not adapted for the characteristics of data streams, such as high speed, continuous flow and fast changing. Therefore, it is very important to modify the traditional data mining techniques in order to handle steaming data [1] in an efficient way, this is known as Data Stream Mining.

For data stream mining, the successful establishment of innovative algorithms must consider the following specific reasons [2]:

- Data objects generated and appeared continuously from the real time applications.
- At the time of data processing, the data objects of the data streams are generated without any control.
- The volume of data is outsized potentially.
- While processing data, the data objects are discarded. In a specific given period one can store part of the data, using a forgetting mechanism to discard them later.
- The unknown data generation process is probably may change over the period.

In this paper, we will apply the concepts of data stream mining on a case study which is hate speech offensive language detection for tweets using Twitter Streaming API. The paper is structured as follows. In Section 2 we will discuss other works related to Data Stream Mining, in Section 3 we will present our problematic. Section 4 describes the solution we brought. In Section 5 we will compare and explain the different results we obtained and finally Section 6 concludes this paper.

2. Related works

There are many articles that address the subject of data stream mining, we have taken a few of these articles. In what follows we will present a summary of these articles, then we will expose the link with our work.

M. KHOLGHI *et al* [3] reviewed and analyzed data mining applications for solving data stream mining challenges. At first, they presented a comprehensive classification for data stream mining algorithms based on data mining applications. In this classification, they separate algorithms with preprocessing from those without preprocessing.
In addition, they classify preprocessing techniques in a distinct classification: Data-based solutions which include Sampling, Sketching, Aggregation… and Task-based solutions that include approximation algorithms, sliding window and algorithm output granularity. Then they proposed a classification for challenges and related research issues in Data Stream Mining and the approaches for each challenge.
First, for memory management research issue, we have fluctuated and irregular data arrival rate and variant data arrival rate over time, the approach would be using summarizing techniques. For data preprocessing, the quality of mining results could

be reduced, so we use Light-weight preprocessing techniques. Next, for compact data structure issue, we have limited memory size and large volume of data streams, the solution would be incremental maintaining of data structure, novel indexing, storage and querying techniques. And for visualization of results, we are confronted to problems in data analysis and quick decision making by user, there is not solution for this problem and this problem still is a research issue.

Shazia Nousheen M *et al* [4] inspected few algorithms for classification, clustering and association with respect to data mining, for managing the data stream.
For clustering, some algorithms have been presented such as CluStream [5]: It partitions the grouping procedure in taking after two online component and components offline. Online segment stores the synopsis of information as micro-clusters. Offline component applies the k-means grouping to group micro-clusters into bigger segments. Another algorithm is ClusTree [6], It separates the grouping procedure in taking after two online and offline parts. Online part is utilized to learn micro cluster. The smaller scale bunches are ordered in various leveled tree structure.
For classification, many algorithms exist, we will mention the Hoeffding Tree Algorithm [7], Hoeffding comes from Hoeffding bound which is used for tree induction, it gives certain level of certainty on the selection of best attribute to divide the tree, thus it is possible to develop the model in light of certain number of occurrences that we have seen. Fast Decision Tree [8] makes a few changes to the Hoeffding tree calculation to enhance both rate and precision. It parts the tree utilizing the present best trait. But it can't deal with concept drift in data streams. So VFDT algorithm was further formed into the Concept-adapting Very Fast Decision Tree calculation (CVFDT) it runs VFDT over sliding windows. The goal being to have the most upgraded classifier.
And finally, association which is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. The author of the paper listed some methods for the association rules. In [9] to extract data from frequent item sets Lin has proposed a technique which uses a sliding window for its calculations. This model finds and keeps up continuous item sets in sliding windows. Just part of the information streams inside the sliding window are saved and processed when the information streams in. In [10] an algorithm is presented by Yang for extraction of short rules of association in a database, they have utilized the calculations to produce the result of frequent item sets. The outcome sets comprise of most of the item sets that bolster estimations of which are more prominent than or equivalent to the threshold.

Albert Bifet *et al* [11] discussed the challenges that Twitter streaming data poses, focusing on sentiment analysis. Some of these challenges are in twitter own conventions such as hashtags # which are used to denote subjects or categories, mentions @ which shows that users may reply to other users by indicating usernames using this character, and RT is used at the beginning of the tweet to indicate that the message is a so-called “retweet”, a repetition or reposting of a previous tweet. For the case of sentiment analysis, which consists of classifying messages into two categories depending on whether they convey positive or negative feelings, a tweet can contain a significant amount of information in very compressed form, and simultaneously carry positive and negative feelings. Also, some tweets may contain sarcasm or irony. However, a significant advantage of Twitter data is that many tweets have author-provided sentiment indicators: changing sentiment is implicit in the use of various types of emoticons that can be used to label the training data. Another challenge is the unbalanced classes, the rate in which the Twitter Streaming API delivers positive or negative tweets may vary over time; we cannot expect it to be 50% all the time. Hence, a measure that automatically compensates for changes in the class distribution should be preferable. Then, the author of the paper experimented with three fast incremental methods that are well-suited to deal with data streams: multinomial naïve Bayes, stochastic gradient descent, and the Hoeffding tree which was explained above. Hoeffding tree is the slowest of the three methods, as the current implementation
does not use sparse instances as multinomial na¨ıve Bayes and SGD do. Accuracy is similar for the three methods, but this is not the case when one considers the Kappa statistic. Again, Kappa provides us with a better picture of relative

predictive performance. In this stream, we see that multinomial naïve Bayes and SGD perform comparably. An advantage of the SGD-based model is that changes in its weights can be inspected to gain insight into changing properties of the data stream. Considering all tests performed and ease of interpretability, the SGD-based model, used with an appropriate learning rate, can be recommended for the data that was used.

3. Problem

The Internet is one of the greatest innovations of mankind which has brought together people from every race, religion, and nationality. Social media sites such as Twitter have connected billions of people and allowed them to share their ideas and opinions instantly [12]. But this also gave a new place for hateful ideas, racism, xenophobia, and offensive language.
Subrahmanyam *et al* [13] reported
that in internet chatrooms, 3% of utterances words are curse words, this represents 1 word out of 33. Hate speech is more than just harsh words.
It can be any form of expression intended to vilify, humiliate, or incite hatred against a group or class of people. It can occur offline or online or both. It can be communicated using words, symbols, images, memes, emojis and video [14].
Over half (53%) of UK adult Internet users reported seeing hateful content online in 2018, an increase from 47% in 2017 [15]. In 2016, 34% of 12-15-year olds recalled seeing hateful content online. This increased to 45% in 2018 [16].
For this reason, we wanted to bring a solution to identify hateful speech and offensive language in tweets stream, using data stream mining techniques and frameworks. We want for each incoming tweet to classify it into two categories: Offensive/Hateful speech (1) or not Offensive/Hateful speech (0). Thus, it is a binary classification task for a data stream.

4. Solution

Our solution to the problem exposed above consists of two parts:
We will train our model using a dateset [17] containing 24783 instances. 20620 instances are positive instances (meaning hateful/offensive speech) and 4163 are negative instances. We will experiment four different models for the classification of tweets: Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Gradient Boosting and Neural Networks.
On the other hand, we will extract tweets from a continuous data stream using Apache Spark Framework and use our pretrained model to predict for each incoming tweet if it is hateful/offensive or not in real time. in what follows we will explain the different steps to reach the final result.

Tweets preprocessing
Text preprocessing is an important step for natural language processing task. It is needed for transforming textual data from human language to machine-readable format for further processing.
In our tweet preprocessing we've done the following tasks:

1. Normalization: Refers to a series of tasks that aims for putting words from our tweets into equal footing and allows processing to proceed uniformly. It also aims to remove irrelevant information. In our case, we performed the following tasks:
    - converting all text to the same case (lower case).
    - removing URLs.
    - removing punctuation.
    - removing multiple white spaces.
    - removing email addresses.
    - removing numbers.
    - removing hashtags #.
    - removing mentions @.

   It is important to mention that normalization steps can change depending on the study case. In our case, having numbers, or mentions do not provide us any information on the tweet being hateful or not.
2. Tokenization: It is the process of splitting a given text into smaller pieces called tokens. Words, numbers, punctuation marks, and others can be considered as tokens.
3. Stop words removal: Stop words are the most common words in a language like ("the", "a", "on", "is", "all") for English language. These words do not carry any important information and are usually removed from texts, to save computing time and efforts in processing large volumes of text.
4. Lemmatization: Lemmatization and Stemming both aim to reduce inflectional forms to a common base form. The difference is that lemmatization uses

lexical knowledge bases to get the correct base forms of words, meanwhile stemming could give us undefined words. So, we have chosen to work with lemmatization.

Data Transformation

Now that we have cleaned and preprocessed our tweets, we need to transform them into numerical values so we can feed them into our models.
In order to do that, we use the Doc2Vec method, which is an extension of Word2Vec but applied to a whole document (ie multiple tweets) instead of only one.
Compared to more traditional methods such as bag of words, where we just encode for each token the value “1” if it is present in a certain document or “0” if it’s not, Doc2Vec encodes the documents in a way that each word keeps its meaning, ie similarities between words will be captured, so in our case, words that are “hateful” would be closer (in terms of coordinates) to each other than words that are not.

The problem of unbalanced datasets

Our dataset contains 20620(83.21%) positive instances for 4163(16.79%), so it is unbalanced. Unbalanced datasets can lead to poor results in terms of generalization ability. Ultimately, this means that we will not end up with a good model. And the reason includes [18]:

- The algorithm receives significantly more examples from one class, prompting it to be biased towards that particular class. It does not learn what makes the other class “different” and fails to understand the underlying patterns that allow us to distinguish classes.
- The algorithm learns that a given class is more common, making it “natural” for there to be a greater tendency towards it. The algorithm is then prone to overfitting the majority class.

Many solutions exist to solve this problem: Undersampling, Oversampling, VAE etc…We have chosen to work with oversampling, more exactly with SMOTE.

“…Synthetic Minority Oversampling Technique or SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b….” [19]

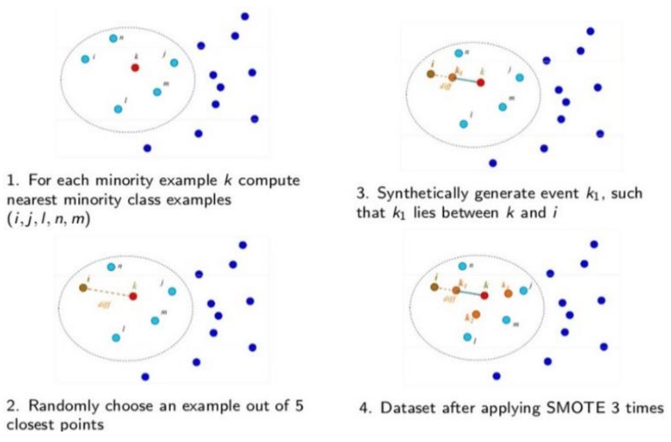
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)
3. Synthetically generate event k_1, such that k_1 lies between k and i
2. Randomly choose an example out of 5 closest points
4. Dataset after applying SMOTE 3 times

Figure 1 SMOTE Oversampling [20]

This procedure can be used to create as many synthetic examples for the minority class as are required. to oversample the minority class to balance the class distribution.

The Twitter Streaming API

In order to have access to the twitter stream, we will be using Tweepy which is a python library for accessing the Twitter Streaming API. The latter provides API products, tools, and resources that enable to harness the power of Twitter's open, global, and real-time communication network [21].
The Twitter API returns data in JSON format, we will only be looking at the text part corresponding to the tweet itself.

Application architecture

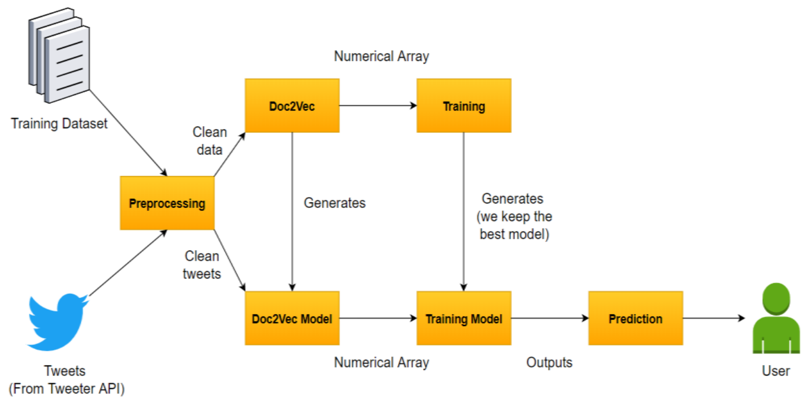

Figure 2 Application architecture

As seen in Figure 2, our application is going to be divided into two parts:

- The first part consists of training a model based on our dataset after preprocessing it and transforming it into numerical instances.
- The second part consists of using Twitter Streaming API to receive tweets and make prediction using the model trained in the first part.

We have already gone through the first part, the second part is done using PySpark framework for Data Stream Mining which we will be explaining below, but first some definitions:

- **RDD (Resilient Distributed Dataset):** Represents an immutable, partitioned collection of elements that can be operated on in parallel.
- **DStream (Discretized Stream):** is a continuous sequence of RDDs (of the same type) representing a continuous stream of data.
- **Batch Duration**: the time interval at which streaming data will be divided into batches.
- **Window size:** The duration of the sliding window. The window slides over a source DStream, the source RDDs that fall within the window are combined. (Must be a multiple of batch Duration).
- **Map:** Applies a function to each element of the RDD and returns a new RDD.
- **Reduce:** Reduces the elements of an RDD by merging them using an operator, or using a key

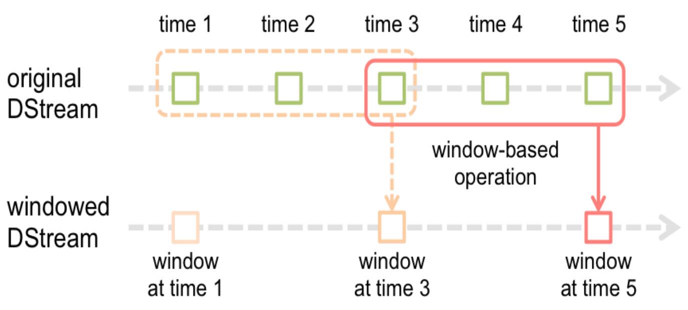

Figure 3 Spark Streaming Window Operations [22]

The figure 3 shows a sliding window, every time this window passes over a DStream source, the RDDs that are contained in the window are combined. In the example the Window size is equal to 3 units of time (must be a multiple of batch size).

Twitter Hate speech & Offensive Language detection

Now that we obtained our data stream with the Twitter Streaming API and Spark, we can apply different methods on the RDD such as map, which returns a new RDD by applying a function to each element of this RDD. In our case, we want to apply the different preprocessing methods that we mentioned earlier (Punctuation removal, tokenization etc…).

Then, we will transform our data using the Doc2vec model that we've used in the training phase to have similar inputs for our model. Finally, we also reduce the tweets by their key to merge similar tweets in order to have a smaller RDD.

After preprocessing our tweets, we're going to use the best performing model that we've obtained from the training phase (See Section 5) to predict if a tweet is hate speech/offensive or not. Then, we simply display for each element of our RDD the classification prediction for each tweet.

5. Experiments

We tested our data on different supervised learning techniques, in order to compare the results and choose the one with the best results. After a long search among the articles, we choose the techniques that gave the best results, so comparing them will give us a much closer look on the most suitable technique for detecting hate speech in tweets. These four techniques were chosen:

- Support Vector Machines (SVM).
- Boosting.
- k Nearest Neighbor (kNN).
- Neural Network.

For each classifier a grid search was performed to have the best estimator model possible.

The search parameters in SVM classifier were kernel, C and gamma.

For Boosting the search parameters were the number of estimators, max features and the learning rate.

As for the search parameters of KNN were the number of neighbors and weights.

Finally, the search parameters for Neural Network were activation function, solver and the learning rate.

There is not a consensus in literature about which evaluation metrics to use. While there are many techniques to evaluate the performance of the algorithm, we chose a large list of evaluation metrics in order to have a better look on the performance of each classifier. A list of score has been calculated for each classifier and the results are the following. Note that a comparison between the classifier will be presented in further suction.

Metric	Definition
Accuracy	Refers to the proportion of correct predictions made by the model
Balanced Accuracy	Defined as the average of recall obtained in both hate speech and non-hate speech tweets.
Precision	Indicates which proportion of hate speech tweets, are actually hate speech content
Average Precision	Computes the average precision from prediction scores
Recall	Express the percentage of hate speech tweet did the algorithm correctly classify
Negative Brier loss	Measures the mean squared difference between the predicted probability assigned to the possible outcomes for item i, and the actual outcome. Across all items in a set N of predictions. [23]
F1 score	F-score produces an average of precision and recall
F1 micro	Calculate metrics globally by counting the total true positives, false negatives and false positives [25]
F1 macro	Calculate metrics for each label, and find their unweighted mean [25]
F1 weighted	Produces a weighted average of precision and recall
Negative Logistic loss	Defined as the negative log-likelihood of the true labels given a probabilistic classifier's predictions. [24]
Jaccard	Computes the average of Jaccard similarity coefficients, also called the Jaccard index, between pairs of label sets.
Roc AUC	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores

Table 1 Metrics definitions

SVM

The results for SVM classifier are the following:

Scoring	Value
Accuracy	0.893073302
Balanced Accuracy	0.750355279
Average Precision	0.910575309
Precision	0.907871116
Recall	0.966159326
Brier score loss	0.106926698
F1 score	0.93754419
F1 micro	0.893073302
F1 macro	0.783111189
F1 wighted	0.885242604
Logistic loss	3.693180336
Jaccard	0.882431233
Roc auc score	0.750355279

Table 2 SVM Results

For a better visualization of the results the next histogram is given:

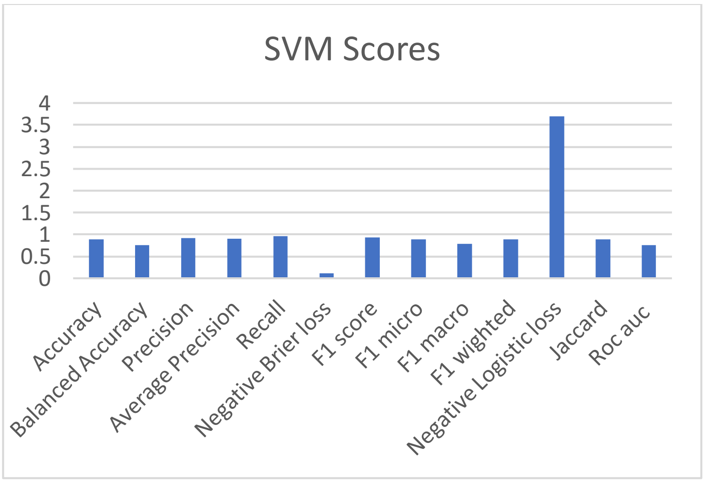

Figure 4 Results Histogram for SVM classifier

Boosting

The results for Boosting classifier were the following:

Scoring	Definition
Accuracy	0.899663753
Balanced Accuracy	0.76691875
Average Precision	0.915529664
Precision	0.912808
Recall	0.968380088

Brier score loss	0.100336247
F1 score	0.941213554
F1 micro	0.899663753
F1 macro	0.79950586
F1 wighted	0.892878556
Logistic loss	3.465550473
Jaccard	0.888955046
Roc auc score	0.76691875

Table 3 Boosting Results

Corresponding histogram:

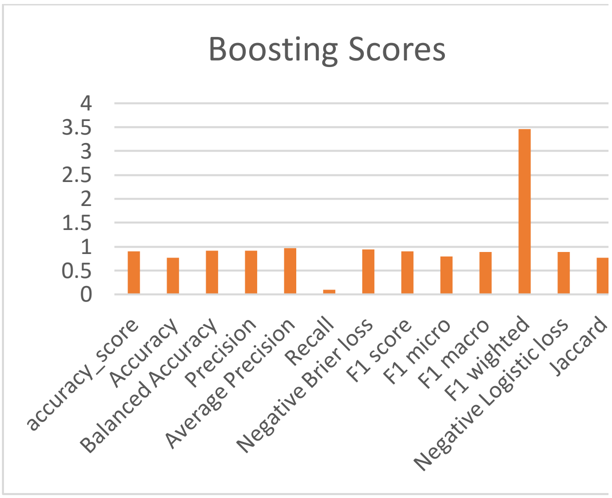

Figure 5 Results Histogram for Boosting classifier

KNN

The results for KNN classifier were the following:

Scoring	Value
Accuracy	0.967047747
Balanced Accuracy	0.927551908
Average Precision	0.974036317
Precision	0.972176582
Recall	0.986767791
Brier score loss	0.032952253
F1 score	0.980360721
F1 micro	0.967047747
F1 macro	0.939032135
F1 wighted	0.966597496
Logistic loss	1.138148023
Jaccard	0.961477987
Roc auc score	0.927551908

Tableau 4 KNN Results

We give the histogram for these results:

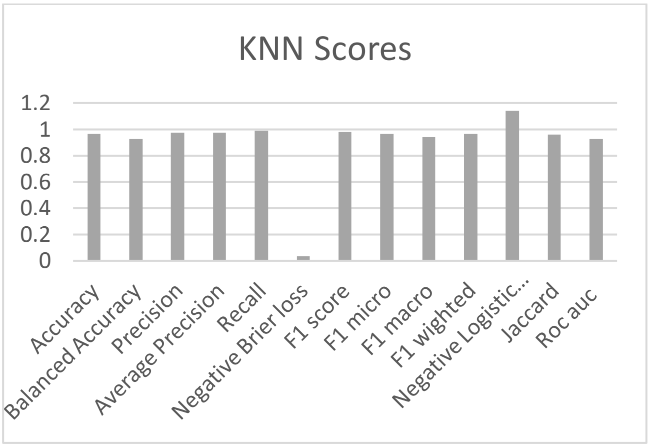

Figure 6 Results Histogram for KNN classifier

Neural Network

Finally, the results for the neural network model:

Scoring	Value
Accuracy	0.905581708
Balanced Accuracy	0.784120006
Average Precision	0.922067901
Precision	0.919133325
Recall	0.968081659
Brier score loss	0.094418292
F1 score	0.944514701
F1 micro	0.905581708
F1 macro	0.900175323
F1 wighted	0.814006674
Logistic loss	3.26114658
Jaccard	0.894862962
Roc auc score	0.784120006

Tableau 5 Neural Network Results

And the corresponding histogram:

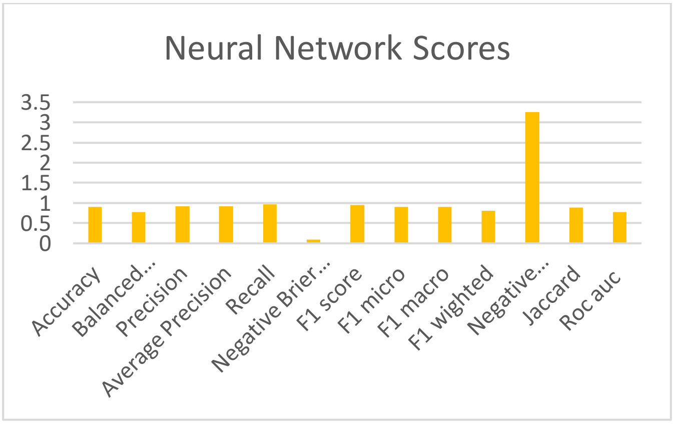

Figure 7 Results Histogram for Neural Network classifier

Comparative study of the classifiers

In order to have a clear look to the scores, the histogram in Figure 8 is presented. It shows the score results for the four classifiers:

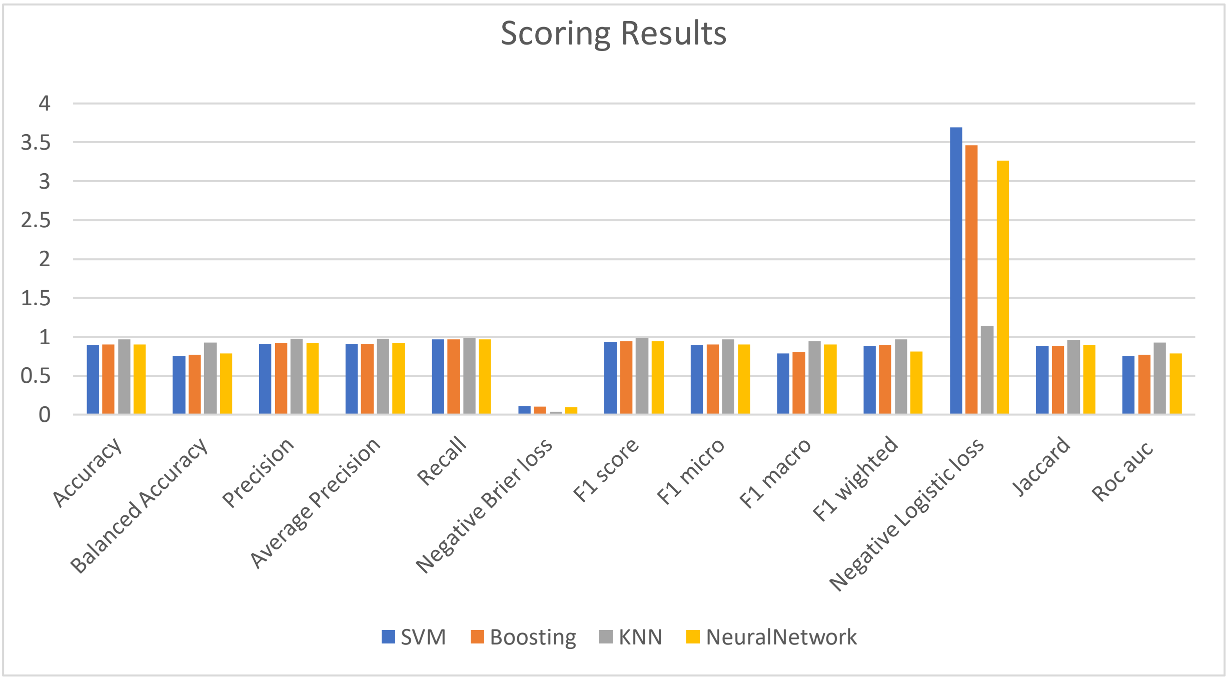

Figure 8 Evaluation of Classification Models

First, for accuracy, KNN clearly beats the other classifiers, followed by the neutral network model. SVM and Boosting having both similar scores. We observe similar result for balanced accuracy which is expected.

Next, for precision, KNN has the highest score. The other three classifier have similar score with a slightly higher precision for Neural Network. The same results are given for the average precision. For recall, the four classifiers have similar result with a slightly higher value for KNN classifier.
As for the negative brier loss, KNN has the lowest (thus the best) score, followed by Neural Network, Boosting and finally SVM.

Then for F1 score, the results for the four classifiers are almost similar with a larger score for KNN. For micro F1, a bigger difference is observed between KNN and the other classifiers. As for F1 macro, the same results are observed with bigger difference then micro F1 between KNN and both boosting and SVM. Finally, for weighted F1, different results are observed, with KNN still having the highest score, followed by SVM and boosting with same score and the neutral network having the lowest score.

For negative logistic loss, we observe a remarkable difference between KNN and the other classifiers with SVM having the highest loss and KNN the lowest.

As for Jaccard score, SVM has the lowest score, next is boosting, then neural network and finally KNN with the highest score.
Finally, for Roc AUC score, KNN has the highest value, followed by Neural network, boosting and then SVM.

In summary, we noticed that the KNN classifier outcomes the other classifier for all the evaluation metrics. Thus, the KNN classifier is the most suitable technique in our case for detecting hate speech and offensive language in tweets. The Neural Network model gave promising results as well. Boosting and SVM didn’t give satisfying results since they gave the worst scores.

6. Conclusion and future work

Twitter Streaming API enables any user to receive a large quantity of tweets in real time, data stream mining techniques can be used to extract and mine those tweets.

In this paper, we have first presented some works that are related to ours. Then, we described all the preprocessing steps to first clean, then transform the tweets into usable data that can be used to train a model that can classify incoming tweets into hateful/offensive or not in real time. We tested four different models that are: KNN, SVM, Gradient Boosting and Neural Networks. Considering all tests performed, the KNN model seems the most adequate for our study.

In future works, we would like to extend the presented results by adding new evaluation criteria, with other models in addition to the four we mentioned above. We would also like to improve the preprocessing steps by detecting more “hidden” words, such as words written in leet speak, and finally propose other data transformation methods besides Doc2Vec.

7. References

[1] V. Sidda Reddy, T.V. Rao, Govardhan A., “Data mining techniques for data streams mining”, Vol. 4, No. 1, March, 2017, pp. 31-35

[2] Dr.V.Kavitha, S.Subhasini, “A Summarization Paradigm of Open Challenges for Data Stream Mining Issues”, IJARCET, Volume 7, Issue 8, August 2018.

[3] M. Kholghi, M. Keyvanpour, “An analytical framework for data stream mining techniques based on challenged and requirements”, IJEST, May 2011

[4] Shazia Nousheen M and Dr. Prasad G R, “A Survey Paper on Data Stream Mining”, IJERT, Vol.5 Issue 08, August 2016

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams,” in Proceedings of the 29th international conference on Very large data bases - Volume 29, ser. VLDB ’03. VLDB Endowment, 2003, pp. 81–92

[6] Kranen,Assent,Baldauf,Seidl, “Self-Adaptive any time clustering”,ICMD, 2009

[7] Domingos P. and Hulten G. (2000) Mining High-Speed Data Streams. In Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining.

[8] Hulten G., Spencer L., and Domingos P. (2001) Mining TimeChanging Data Streams. ACM SIGKDD Conference.

[9] [Lin, 2005] Chih-Hsiang Lin, Ding-Ying Chiu, Yi-Hung Wu, Arbee L. P. Chen; Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window; SIAM Int'l Conf. on Data Mining; April 2005

[10] [Yang, 2004] Li Yang, Mustafa Sanver; Mining Short Association Rules with One Database Scan; Int'l Conf. on Information and Knowledge Engineering; June 2004.

[11] Bifet A., Frank E. (2010) Sentiment Knowledge Discovery in Twitter Streaming Data. In: Pfahringer B., Holmes G., Hoffmann A. (eds) Discovery Science. DS 2010. Lecture Notes in Computer Science, vol 6332. Springer, Berlin, Heidelberg

[12] Mathew, Binny & Dutt, Ritam & Goyal, Pawan & Mukherjee, Animesh. (2019). Spread of Hate Speech in Online Social Media. 173-182. 10.1145/3292522.3326034.

[13] Subrahmanyam, K., Smahel, D., Greenfield, P., et al. Connecting developmental constructions to the internet: Identity presentation and sexual exploration in online teen chat rooms. Developmental psychology 42, 3 (2006), 395

[14] Brittan Heller, J.D., and Larry Magid, Ed.D, “The parent’s and educator’s guide to combating hate speech”, connectsafly.org

[15] ofcom.org.uk , Adults: Media use and attitudes report 2019, Published 30 May 2019.

[16] ofcom.org.uk, Children and parents: Media use and attitudes report 2018, Published 29 January 2019

[17] Automated Hate Speech Detection and the Problem of Offensive Language, Proc. ICWSM

[18] German Lahera, Unbalanced Datasets & What To Do About Them, Jan 22, 2019.

[19] Haibo H, Yunqian M., Imbalanced Learning: Foundations, Algorithms, and Applications 1st Edition, July 1, 2013, Page 47.

[20] Dal Pozzo, Caelen, Waterschoot and Bontempi, Racing for unbalanced methods selection, Published on Jan 20, 2015.

[21] Twitter API Docs, https://developer.twitter.com/ , April 2020.

[22] Apache Spark Docs, https://spark.apache.org/, April 2020.

[23] Scikit-Learn Docs, https://scikit-learn.org, April 2020.

[24] Scikit-Learn Docs, https://scikit-learn.org, April 2020.

[25] Scikit-Learn Docs, https://scikit-learn.org, April 2020.