Twitter dataset reduction algorithm

1st Gabriel Ichcanziho Pérez-Landa
School of Engineering and Sciences
Tecnologico de Monterrey
Atizapán de Zaragoza, México
https://orcid.org/0000-0002-0668-8737

2nd Francisco J. Cantu-Ortiz
School of Engineering and Sciences
Tecnologico de Monterrey
Monterrey, México
https://orcid.org/0000-0002-2015-0562

3rd Hector G. Ceballos
School of Engineering and Sciences
Tecnologico de Monterrey
Monterrey, México
https://orcid.org/0000-0002-2460-3442

***Abstract*—This Twitter, more than a social network, has become one of the most used sources for creating text datasets. This allows the building of new Machine Learning models capable of classifying a text in different ways. When experts in the field manually tag datasets, they are treasured. This value is because the classification model can extract essential characteristics of a text according to the class to which it belongs. Labeling a tweet dataset is a long and tedious task that requires multiple people. However, not all tweets that are labeled contain information relevant to their classification. This work presents an algorithm that allows preserving the tweets with more information. With this, a cleaner dataset is obtained. Finally, one of the uses of this algorithm is to reduce the number of tweets to label.**

Keywords—Twitter, Machine Learning, NLP

I. INTRODUCTION

Twitter is a social network characterized by the briefness of the texts, with a maximum of 280 characters. In the first quarter of 2019, Twitter reported 330 million users and 500 million tweets per day [1].

Twitter has excellent popularity among developers and scientists due to the ease and freedom to obtain information from its social network. Through its Application Programming Interface API, it is possible to search, filter, and download tweets [2]. Its outstanding versatility has made it possible to train various Machine Learning (ML) models that use Natural Language Processing (NLP) to obtain essential characteristics from the writings.

According to Javed *et al*. [3], Twitter is an ideal platform for sentiment analysis. There are mainly two ways to use the Twitter API to create datasets for ML models. The first way is unsupervised. This way allows for obtaining large-scale datasets in which unsupervised algorithms can classify tweets [4]. It is also useful to get the linguistic characteristics of a group of people, for example, the most used words, the average number of words that a tweet has, among others.

On the other hand, the second way to use the Twitter API to create datasets is supervised. This form is usually used for manual classification, where humans with experience in the area of interest will read each tweet and label them. At work, Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter written by Waseem et al. [5]. They created a dataset with 16,907 tweets, which were manually tagged as racist, sexist, or none. The creation of new Twitter-based datasets tagged by humans is essential because it allows the training of new ML models that can extract the characteristics that make a text be considered in a certain way.

The most common way to create these new datasets is to use a programming language that connects to the Twitter API. Then start looking for tweets containing keywords that represent the topic to be classified [6]. These can be filtered by language, location, and other parameters. Once enough tweets have been downloaded, a group of experts sets about tagging the dataset. However, in this work, we ask ourselves the following research question: How many of the tweets that are downloaded to be tagged are useful for training the Machine Learning model?

Take, as an example, the dataset created by Waseem *et al.* [5]. In total, 16,907 tweets were tagged by humans. In their work, they report having obtained a result of 73.93 in the F1-Score test. The exciting thing is to wonder, 16,907 were enough tweets? What would have happened if they had tagged more or fewer tweets? Would the results of the F1-Score test have changed?

Based on the previous questions, this work seeks to create an algorithm that allows the reduction of tweets in the creation of a new dataset. The main characteristic of the algorithm proposed in this work is the decrease in the number of tweets tagged. It tries to maintain the quality of the classifier. For this, the Area Under the Curve (AUC) metric was used to compare the algorithm. What was the AUC of a dataset before and after using our algorithm, and how much did it reduce the number of tweets.

II. METHODS AND DATA

A. Obtaining datasets

To verify the Twitter Dataset Reduction TDR algorithm effectiveness, four datasets were selected that were created using Twitter as the sole source of data extraction. Another critical point is that these datasets were manually labeled. A person or group of people had to read each text and classify it one by one.

TABLE I. SIMPLE DESCRIPTION OF THE DIFFERENT DATASETS USED IN THIS WORK,

Author	Problem	Dataset origin	Dataset size	Classification
Agarwal[7]	Hate speech	Twitter	31,962	Binary
Davidson [8]	Hate speech	Twitter	24,783	Multiclass
Crowd Flower [9]	Gender Classification	Twitter	20,050	Multiclass
DataTurks [10]	Cyber-Trolls	Twitter	20,001	Binary

Table I shows a summary of the characteristics of the datasets collected for this work. It is essential to mention that the datasets presented here are public and that each of them was slightly modified so that they all had the same format of only two columns, classification, and text. Some datasets had additional information such as the tweet's id, the date of publication, or other details. However, this information was discarded to prioritize only the tweet's classification based

solely on the information contained in the text present in the tweet.

B. Tweet cleaning

An essential step in developing the TDR algorithm is to clean each tweet. After the text has been preprocessed, the number of useful words it contains is counted. When this number of words is equal to or greater than a range, the message is considered to collect enough relevant information.

When we refer to the process of cleaning a tweet, it means that we must eliminate those elements of the text that do not provide relevant information for subsequent tagging. Twitter has several of its platform-specific symbols, and removing them does not change the message's intention.

T able II shows the steps that were taken to clean up the tweet. The first step is to remove unknown characters and remove hyperlinks and re-tweet the symbol "RT" and hashtag "#." After this first cleaning, we can begin to homogenize the words so that, for example, the words: "LOVE, Luv, love" are all interpreted in the same way and not as independent words.

The first step is to convert the colloquial abbreviations into their real expressions so that "gr8" becomes "great" or "dd" into "dear," among others. After each word is made lowercase, we can also remove user tags in tweets with this step. We do this to reduce the corpus' total vocabulary, since these words, being usernames, are very specific and generally have no meaning.

The final step in tweet cleaning is to remove punctuation symbols, numbers, and short words, also known as stop words. Because these words are connector pronouns, among others, and a message can be interpreted without them [11].

TABLE II. ALGORITHM PROPOSED TO CLEAN A TWEET. RETURN A TEXT WITH THE MOST IMPORTANT WORDS OF THE TWEET.

Action	Result	Number of words
Read the Tweet	RT @Pepe î,202 my year <4 gr8!!!. I luv you more than cats. I slept 8 hours yesterday. #eatFood https://t.co/6lsJ6D4J	22
Remove unknown chars, hyperlinks, Hashtag symbol, retweet text.	@pepe I, 2020 my year <3 gr8 !!! I luv you more than cats. I slept 8 hours yesterday. Eat Food	20
Clean abbreviations, remove @users, convert to lowercase	i , 2020 my year <3 great ! ! ! . i love you more than cats . i slept 8 hours yesterday . eatfood	25
Remove string punctuations, numbers and stop words. Apply lemmatization, and keep emojis.	Year <3 great love cat sleep hour yesterday eatfood	9

For example:

- "I love my cats."
- "She loves her cat."

They both talk about loving cats, so the above sentences can easily be converted to:

- "love cats."
- "loves cat."

We observe that the previous sentences are very similar to each other. The last step is to apply lemmatization. Which is a method in which the words are taken to their base expression or their dictionary form [12] so that:

- *love cats*, it transforms into: **love cat.**
- *loves cat*, it transforms into: **love cat**.

Finally, we can conclude that the above sentences contain the same useful information, and that both sentences can be interpreted as "love cat." However, something important is that this cleaning method allows the preservation of emojis or emoticons. These expressions have been standardized in contemporary writing and must be treated as individual tokens since they can add a lot of information [13].

- "I love my cats ☺☺."
- "She loves her cat."

After cleaning the messages above, we would have:

- "love cat ☺☺."
- "love cat."

It is easy to recognize that the first message is more effusive than the second, that is the power that emojis have in the way of writing on social networks.

C. Dataset cleaning

In this section, we will quickly explain how the TDR algorithm works. The algorithm consists of four steps: 1 reading the original dataset, 2 creating an auxiliary dataset, 3 generating a mask, 4 cleaning the original dataset with the mask. Table III shows an illustrative example of how the TDR algorithm works with a fictitious dataset that contains four tweets, each one with its corresponding words, and they have already been previously tagged.

TABLE III. EXAMPLE OF THE STEPS TO CLEAN UP THE DATASET,

1: Original Dataset

Id	Class	Tweet
0	0	Word1, Word2, Word3
1	1	Word1
2	0	Word1, Word2
3	1	Word1, Word2, Word3, Word4

2: Auxiliar Dataset

Id	Class	Tweet
0	0	Word2, Word3
1	1	
2	0	Word2
3	1	Word2, Word3, Word4

3: Mask

Id	Class	Boolean
0	0	True
1	1	False
2	0	False
3	1	True

4: Clean Dataset

Id	Class	Tweet
0	0	Word1, Word2, Word3
4	1	Word1, Word2, Word3, Word4

The first step of the TDR algorithm is to have the database that you want to reduce. The most important thing about this

algorithm is that it only needs the "tweet" column that refers to the text that will later be tagged. Other columns that contain information from the tweet, such as id, location, date, or if the dataset has already been tagged with a class, that information is not necessary either.

The second step consists of passing each tweet through the filtering process described in the previous section, tweet cleaning, see table 2. For demonstration purposes, table 3 shows that the auxiliary dataset was created by removing the word "word1" from each one. Messages from the original dataset

The third step consists of applying a mask to the texts generated in the auxiliary dataset; this mask is obtained by doing the following boolean check:

$$Len(Tweet) \geq n \quad (1)$$

Where Len is a function that returns the number of words present in a tweet. And n is an editable parameter that means the minimum number of words that the tweet must contain to be considered with enough useful information. For example, in table III, n = 2.

The fourth step is to apply the mask of useful tweets to the original dataset; this allows creating a subset that contains only the tweets that are known to contain enough helpful information. Tweets that once went through the cleaning process and had a word count of less than n are simply discarded. This procedure generates a smaller dataset, in which it is known in advance that all tweets can be read by humans and tagged manually. The TDR algorithm's most significant advantage is that it reduces the number of tweets to tag and almost does not alter its performance when applying a text classification model.

III. Results

The purpose of the datasets described in section II-A is to train a Machine Learning model to distinguish and classify text. The scikit-learn Python library [14] was used to train four classification models:

1) Random Forest Classifier.
2) Logistic Regression.
3) Decision Tree Classifier.
4) Ada Boost Classifier.

Because the datasets used are text, the first thing that was done was to preprocess each dataset with the function described in section II-B. Finally, the Term frequency - Inverse document frequency (TFIDF) algorithm was used to convert the words to a numerical representation. Then we use the proposed classifiers and obtain the Area Under the Curve (AUC), a widely used performance measure to know the efficiency of the model that has been trained.

Based on the above, the way we tested the TDR algorithm's efficiency was by comparing the AUC of the original dataset against the AUC of the modified dataset using TDR. Table IV shows a summary of the AUC values that each classifier had in each partition, which is a modification to the parameter n of equation 1. The names of the datasets present in table IV correspond concerning those present in table I.

- Dataset 1: Hate speech - Agarwal.
- Dataset 2: Hate speech and offensive language Davidson.
- Dataset 3: Gender classification, Crowd Flower.
- Dataset 4: Cyber-Trolls, Data Turks.

The meaning of the partitions present in table IV are as follows:

- Partition 0: Repeated messages were removed from the original dataset, but the TDR algorithm was not used.
- Partition 1: Repeated messages were eliminated from the original dataset; the TDR algorithm was used using a value of n = 1.
- Partition 2: Repeated messages were eliminated from the original dataset; the TDR algorithm was used using a value of n = 3.
- Partition 3: Repeated messages were eliminated from the original dataset; the TDR algorithm was used using a value of n = 5.

Table V shows a summary of the results obtained in table IV. We can observe the values corresponding to the number of tagged texts, the number of different words after applying the cleaning algorithm, and the AUC results before and after applying the algorithm TDR. It should be noted that the percentage of decrease is relative to the value that was had before applying the TDR algorithm. The formula to obtain the decrement value is the following:

$$Decrease\ (\%) = 100\left(1 - \frac{ValueAfter}{ValueBefore}\right) \quad (1)$$

TABLE IV. Results of the proposed filter. The original dataset was cleaned four times, each time passed through four classifiers. The result of the best classifier was saved as well as the number of unique words before and after using the filter.

Name	# Posts	# Words Before	# Words After	Random Forest	Logistic Regression	Decision Tree	AdaBoost	Best Result
Dataset 1	31961	67168	39582	0.748506	0.646948	0.739408	0.680408	0.748506
Partition 0	29530	67168	39582	0.707623	0.601745	0.699076	0.639820	0.707623
Partition 1	28361	67079	39582	0.683002	0.599049	0.701107	0.643991	0.701107
Partition 2	27414	66581	39344	0.687027	0.579829	0.709864	0.639785	0.709864
Partition 3	23952	63610	37676	0.669884	0.591697	0.687182	0.634460	0.687182

Dataset 2	24783	59462	18445	0.759535	0.767573	0.798183	0.782934	0.798183
Partition 0	24783	59462	18445	0.755858	0.767573	0.801100	0.782934	0.801100
Partition 1	23716	58381	18445	0.755863	0.761527	0.788943	0.787418	0.788943
Partition 2	22453	56927	18298	0.759434	0.763075	0.792774	0.786046	0.792774
Partition 3	18015	51799	17488	0.743769	0.479889	0.800042	0.796075	0.800042
Dataset 3	20050	66795	24083	0.630025	0.632378	0.582315	0.579246	0.632378
Partition 0	18444	66795	24083	0.601451	0.615770	0.558830	0.571733	0.615770
Partition 1	17386	65094	24081	0.577639	0.603565	0.560178	0.567478	0.603565
Partition 2	16805	63987	23830	0.581672	0.594072	0.565192	0.572334	0.594072
Partition 3	14743	59921	23099	0.595123	0.603177	0.563199	0.579558	0.595123
Dataset 4	20001	31724	14516	0.896041	0.729325	0.802884	0.662434	0.896041
Partition 0	14125	31724	14516	0.541626	0.549643	0.574551	0.558062	0.574551
Partition 1	14125	31724	14516	0.547494	0.549643	0.572911	0.558062	0.572911
Partition 2	12393	30580	14217	0.532478	0.553678	0.571734	0.559067	0.571734
Partition 3	9346	28180	13412	0.520817	0.528799	0.544056	0.540979	0.544056

TABLE V. AVERAGE DATA DECREASE AND AVERAGE LOSS OF AREA UNDER THE CURVE AFTER FILTERING THE DATASET.

Number of labeled posts

Name	Before	After	Decrease (%)
Dataset 1	31960	23952	25.05
Dataset 2	24783	18015	27.31
Dataset 3	20050	14743	26.47
Dataset 4	20001	9346	53.28
		Average	33.03

Number of unique words

Name	Before	After	Decrease (%)
Dataset 1	39582	37676	04.81
Dataset 2	18445	17488	05.18
Dataset 3	24083	23099	04.08
Dataset 4	14516	13412	07.61
		Average	05.42

Area Under the Curve

Name	Before	After	Decrease (%)
Dataset 1	0.7076	0.6871	02.05
Dataset 2	0.8011	0.8000	00.11
Dataset 3	0.6157	0.5951	02.06
Dataset 4	0.5746	0.5440	03.05
		Average	01.82

We can see that for all the datasets, a decrease of more than 25% was achieved in each one. It is critical to know that for the AUC comparison, the before value was taken from partition 0 against Partition 3 this decision was made based on the fact that the originals datasets can contains repeated tweets. In conclusion, the most significant loss of AUC was lower than 03.10% in comparison against Partition 0 and 3.

Fig. 1 shows a graphical representation of table V. The X-axis shows the variables of Table V as follows:

- **TA:** Number of labeled text after apply TDR.
- **AA:** AUC after apply TDR.
- **WA:** Number of unique Words after apply TDR.

On the other hand, the Y-axis represents the relative percentage, where 100% is the variable's value before applying the TDR algorithm. The dark reddish areas of the bars represent the percentage of decrease.

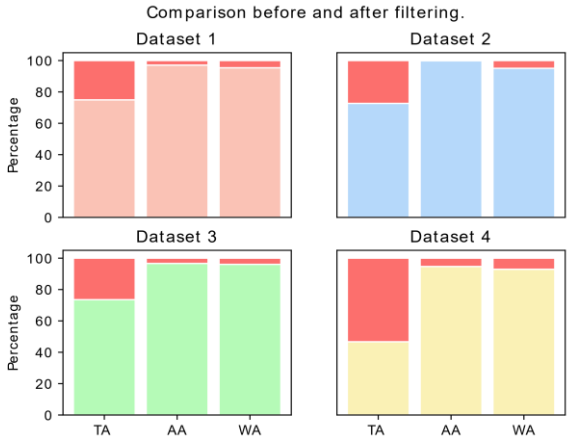

Fig. 1: Average data decrease and average loss of area under the curve after filtering the dataset. Red indicates a decrease.

Table VI shows a comparison between the ten most used words in the dataset before and after applying the TDR algorithm; this table is particularly interesting because we can observe the decrease in each dataset's number of words. The order of the words was not modified, except in dataset number 4.

TABLE VI. MOST USED WORDS IN EACH DATASET BEFORE AND AFTER BEING FILTERED.

Dataset 1 Bef.		Dataset 1 After		Dataset 2 Bef.		Dataset 2 After	
Word	Count	Word	Count	Word	Count	Word	Count
Love	2469	Love	2267	Bitch	11370	Bitch	8539
Day	2168	Day	1884	Hoe	4290	Hoe	3105
Get	1728	Get	1576	Get	3063	Get	2781
Happy	1568	Happy	1409	Like	2842	Like	2625
Go	1406	Go	1266	Fuck	2237	Fuck	1930
Dear	1177	Dear	1086	Pussy	2114	Pussy	1595
Make	1159	Maker	1039	Ass	1567	Ass	1408
Like	1086	Like	1005	Shit	1293	Shit	1194
Life	1056	Life	969	Go	1266	Go	1157
Today	974	New	891	nigga	1215	nigga	1101

Dataset 3 Bef.		Dataset 3 After		Dataset 4 Bef.		Dataset 4 After	
Word	Count	Word	Count	Word	Count	Word	Count
Get	1625	Get	1485	Hate	2776	Hate	1555
Got	1127	Go	975	Fuck	2475	damn	1343
Like	1050	Like	941	damn	2452	Get	1187
Make	828	Make	738	Get	1781	Fuck	1040
Love	805	Love	722	Ass	1724	Like	996
One	776	One	682	Suck	1619	Ass	935
Time	686	Time	637	Like	1504	Lol	879
New	678	New	601	Lol	1428	Suck	860
See	625	See	591	Go	1047	Go	731
Day	612	Say	554	know	1010	know	665

Given that the most used words preserved their hierarchy before and after using the algorithm, it was also possible to show that the distribution of the classes (number of texts classified with different labels) was slightly affected. Table VII shows the amount of data labeled by each category in each dataset as well as the percentage of the said class concerning the total data before and after applying the TDR algorithm. Finally, Fig. 2 is a graphic representation of table VII.

TABLE VII. DISTRIBUTION OF THE CLASSES AFTER AND BEFORE APPLYING THE TDR ALGORITHM.

Dataset	Class	Number of elements		Dataset (%)	
		Before	After	Before	After
1	1	27517	22318	93.18	93.17
	2	2013	1636	07.02	06.83
2	1	1431	1025	04.94	04.88
	2	19190	13799	66.29	65.06
	3	8326	6384	28.86	30.10
3	1	6585	5285	35.70	35.84
	2	6145	4986	33.32	33.82
	3	5714	4475	30.98	30.34
4	1	2690	1974	19.05	21.13
	2	11435	7372	80.95	80.87

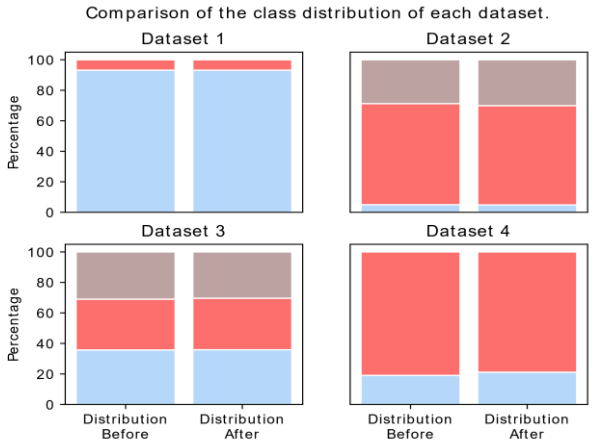

Fig. 2: Distribution of the classes in the original and cleaned dataset, the number of colors in each bar shows the number of different classes each dataset had, while the height of the same refers to the percentage it had of the complete tagged texts.

IV. DISCUSSION

The results present in this work show that the TDR algorithm's effectiveness managed to reduce the number of tweets in a dataset by more than 20%. The AUC test result showed that in the worst-case scenario, 3.05 AUC points were lost. The algorithm showed that only 6 removing irrelevant tweets. Therefore, the distribution of classes before and after using it varies very little.

The tweet cleaning process has been developed, considering several examples from the literature. The basics like removing stop words, applying lemmatization, and converting words to lowercase are fundamental parts of a cleanup process. Removing URLs, users, special Twitter symbols are specific steps to clean tweets. Finally, keeping emojis or cleaning abbreviations were proposed to clean tweets in greater depth and preserve critical values.

Using another method of cleaning tweets may result in changing the effectiveness of the TDR algorithm. Using five words as the minimum value of useful words a tweet should have is a proposition. After applying a cleaning filter to the dataset, the results of other partitions were compared. Using five words was shown to reduce the dataset quite a bit without losing much AUC. Changing this parameter will tighten the filter and, therefore, further reduce the number of tweets to save.

Among the main applications of the TDR, the algorithm is that it generates two datasets for you. The first is a clean version of the original dataset, without repeating tweets and with useful tweets. The second is the version ready to be used

in a classification model. In such a way, the tagging experts can read the dataset with the original tweets, and you already have the processed tweets ready to work with them.

The TDR algorithm can be used to ensure that a dataset has several different and useful tweets to tag. Assuming that you want to create a dataset of 10k tweets, we can see that valuable tweets' real value will be lower, approximately 25% less. Finally, the dataset would go from having 10k to 7.5k with an AUC decrease of less than 2%.

Some ideas to improve the results present in this work are the following: After having used the algorithm for the first time to clean and reduce tweets, apply a filter that counts which are the least repeated words, eliminate the words that do not exceed a threshold and repeat the process

ACKNOWLEDGMENT

The authors acknowledge the financial support from CONACYT-PNPC Grant number 1048425, Tecnologico de Monterrey Tuition scholarship A01651212, CONACYT-SNI Grant numbers SNI9804, SNI147217, Tecnologico de Monterrey, and the research groups on Machine Learning and Intelligent Systems

REFERENCES

[1] J. Clement. Twitter: number of active users 2010- 2019. https://www.statista.com/statistics/282087/ number-of-monthly-active-twitter-users/, 08 2019. Accessed: 2020-04-08.

[2] Twitter. Consuming streaming data. https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data, 2019. Accessed: 2020-04-08.

[3] Iqra Javed, Hammad Afzal, Awais Majeed, and Behram Khan. Towards creation of linguistic resources for bilingual sentiment analysis of twitter data. In International Conference on Applications of Natural Language to Data Bases/Information Systems, pages 232–236. Springer, 2014.

[4] Jimmy Lin and Alek Kolcz. Large-scale machine learning at twitter. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pages 793–804, 2012.

[5] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL Student Research Workshop, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics.

[6] Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on twitter using a convolution-gru based deep neural network. In Aldo Gangemi, Roberto Navigli, MariaEsther Vidal, Pascal Hitzler, Raphael Troncy, Laura Hollink, ¨ Anna Tordai, and Mehwish Alam, editors, The Semantic Web, pages 745–760, Cham, 2018. Springer International Publishing.

[7] Rahul Agarwal. Twitter hate speech. https://www.kaggle.com/ vkrahul/twitter-hate-speech, 07 2018.

[8] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In Proceedings of the 11th International AAAI Conference on Web and Social Media, ICWSM '17, pages 512–515, 2017.

[9] Crowd Flower. Gender classifier data. https://data.world/ crowdflower/gender-classifier-data, 04 2016.

[10] DataTurksl. Dataset for detection of cyber-trolls. https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls, 07 2018.

[11] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In Proceedings of the International Joint Conference on Neural Networks, 2003., volume 3, pages 1661–1666. IEEE, 2003.

[12] Nizar Habash, Owen Rambow, and Ryan Roth. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt, volume 41, page 62, 2009.

[13] Petra Kralj Novak, Jasmina Smailovic, Borut Sluban, and Igor ´ Mozetic. Sentiment of emojis. ˇ PloS one, 10(12), 2015.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.