The generalization performance of hate speech detection using machine learning

Alexandra Coroiu
University of Twente
PO Box 217, 7500 AE Enschede
the Netherlands

a.coroiu@student.utwente.nl

ABSTRACT

The current need for automatic hate speech detection is supported by existing research and current implementations of natural language processing. The ability to generalize is an important characteristic of classification models used in natural language processing. In the case of hate speech detection, it assures accurate identification of abusive messages aimed at various groups, even if the model has not yet been trained on messages targeting those specific groups. This research measures the generalization performance of a machine learning implementation trained on sexist messages and tested on racist ones. The word count and term frequency - inverse document frequency features are extracted from text messages and used in a support vector machine with three different kernels: linear, radial basis function and polynomial. There is a substantial difference between the training F1 score benchmark of 0.8 and the testing F1 score result of hardly 0.3. The results show an overall low generalization performance for this classical machine learning method.

Keywords

Hate speech detection, text classification, natural language processing, machine learning

1. INTRODUCTION

The online medium is an environment that allows people to easily communicate and freely express themselves. The rise of online social networks creates an increase in user-generated content on the internet. Even though most of the generated content is respectful, social platforms also constitute a place where people can openly publish and share offensive, discriminatory messages in the form of hate speech [2]. *Hate speech* is defined as speech that attacks a person or a group based on attributes such as race, religion, ethnic origin, national origin, sex, disability, sexual orientation, or gender identity [15]. From the mentioned categories, online discrimination (on Twitter and Whisper) is most prevalent for race, sexual orientation and ethnicity. However, other groups are targeted based on behavior, physical aspects, class and disabilities [16]. The dynamics of online hate speech is influenced by real life events which can represent triggers for discrimination against a specific group [8,19]. Occasionally, hate speech on popular social platforms leads to cyberbullying, harassment and the creation of hate sites [14]. Lately, there has been an increasing interest in regulating harmful user-generated content on social platforms and therefore, suitable hate speech detection tools are needed [2].

In the past decade, the automation of hate speech detection has been researched in the field of natural language processing. This resulted in a series of different machine learning implementations based on a variety of datasets. The data used in research is collected from popular social media platforms like Twitter, Instagram, Yahoo! and YouTube. Because data collection and labelling for supervised learning is a tedious process, there are no large, varied datasets that can be used. The existing datasets used for training and testing the current classification methods contain hate speech targeting only one or two specific groups [15]. Therefore, the performance of researched methods is unknown when faced with more diverse hate speech, aimed at different populations.

The ability to generalize hate speech detection from training sets that do not cover all possible types of discrimination assures that hate speech towards any targeted group will be identified and possibly countered. Currently, there is no research on the generalization of hate speech detection in this sense. Therefore, the following research question is proposed: *What is the generalization performance of hate speech detection using machine learning?* By answering the research question, it can be determined how well hate speech concepts, learned by a machine learning model, apply to new, unforeseen discriminatory messages. This will help to better assess the quality of general hate speech detection and determine its real applicability on social platforms, where content in the form of hate speech is constantly changing because of socio-political events.

2. RELATED WORK

The state of the art has been summarized in detail in Schmidt and Wiegand's survey focused on hate speech text features; and Zhang, Robinson and Tepper's paper which provides an extensive literature review on the existing classification methods [15,23]. The most popular classical learning model for hate speech detection is Support Vector Machines (SVM). This machine learning classifier uses a vector function to define the separation between entries of different classes (e.g. *Figure 1*).

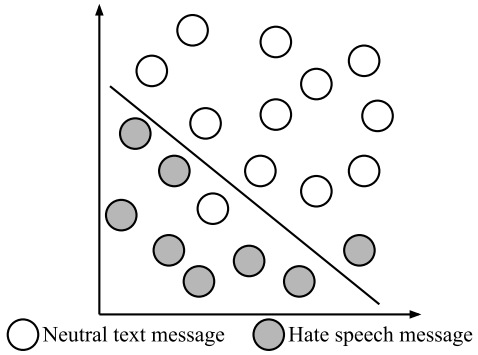

Figure 1. A class separation line created by SVM

31th Twente Student Conference on IT, July. 5th, 2019, Enschede, The Netherlands.

Classical methods used in hate speech detection research (Support Vector Machines, Naïve Bayes, Logical Regression) require the extraction of text features from data before applying a learning model. Features are a set of attributes that represent the relevant information about a data entry. Support vector machines can reach good performances with different combinations of features. Surface features, like bag-of-words and n-grams, are simple text attributes, that encode the words and other characters from text messages in a vector. These features yield good performances on their own [7]. Advanced features are used in addition to surface features to create more complex representations of the data. Word generalization is used to discover similar words (e.g. "people" and "person", "cat" and "dog") [17,20]. Sentiment analysis [4,5] and lexical resources [20] are used to derive more about the meaning and associated sentiment of words. (e.g. "stupid" has more negative connotations and "beautiful" is more positive). The extraction of these two features is usually dependent on external preconstructed word datasets. Linguistic features capture syntactic information about the text [4,5]. There is no comparative study that can prove which complex feature yields better results.

Recently, deep learning methods based on neural networks are also emerging to solve the problem of hate speech detection. [15,23] These methods do not require feature extraction; they derive abstract features from raw data themselves. Deep learning methods classify text messages based on the patterns identified in the abstract representation of features. Two of the most common deep learning approaches are convolutional neural networks (CNN) and recurrent neural networks (RNN). The former is usually used for extracting features similar to bag-of-words or n-grams [12,22], while the latter is used to capture dependencies between words [1,6]. Support vector machines are often used as a comparison benchmark for deep learning methods. The F1 score is the most commonly used performance measurement metric [23]. Support vector machines reach good performances of 0.8, while newly emerged deep learning methods can even exceed 0.9. [1,6,12,22].

3. METHODOLOGY

The chosen approach to assess the generalization performance of hate speech detection is to train a machine learning classification model on a set containing discrimination towards one group and then test on a set containing discrimination towards a different group. A support vector machine with surface features is implemented using python [10]. The generalization performance of the model is determined by comparing the measured performance on the testing set against the measured performance on the training set. The model is tuned such that the training performance benchmark is equal to the state-of-the-art value of 0.8.

3.1 Data

The selected dataset was initially developed for another research and contains 16,907 Twitter messages labeled under "sexism", "racism" or "neither" [18]. The total number of entries containing hate speech (1,970 "racism" + 3,378 "sexism") is 5,348 and makes up around 32% of the dataset, while the rest of 10,556 non-hate entries ("neither") make up the remaining 68%. The unbalanced distribution of hate and non-hate text in the dataset is representative of a realistic online sample. Messages that do not contain hate speech constitute most of the content on social platforms. For this experiment, the dataset is split into a training and a testing set, based on the two different types of labeled hate speech. The training set contains all the 3,378 sexist messages, with 7,178 non-hate messages and the testing set contains all the 1,970 racist messages with the remaining 4,381 non-hate messages. The newly created training and testing sets, of 10,556 respectively 6,351 entries, preserve the unbalanced distribution of the initial dataset (~32% hate speech, ~68% non-hate speech). For both the training and the testing set, only text data and binary labels are used. All the other Twitter data (e.g. date, user, favorite count) has been excluded. The new binary label 1 represents the hate text and replaces the initial labels for "racism" and "sexism", while the label 0 represents the non-hate text, previously labeled as "neither" (shown in *Figure 2*).

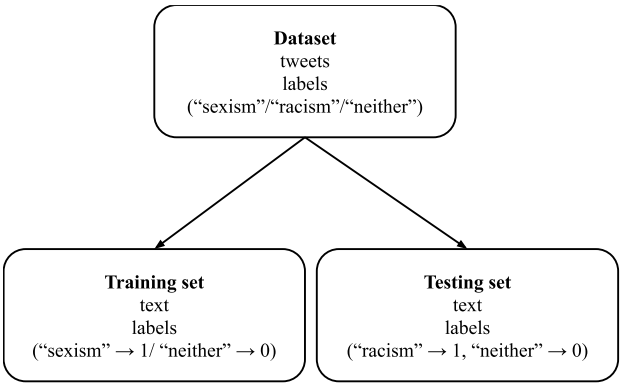

Figure 2. Dataset split

3.2 Features

The text messages from the training and the testing sets are processed into tokens using the NLTK python library [9]. The tokenizer package of this tool contains the TweetTokenizer() function which allows for the removal of unnecessary words or characters that are specific for messages encountered on social media platforms. The function is used to discard usernames, shorten elongated words and set all letters to lower case, before, splitting each message tokens. The function yields a unigram representation with each token representing one distinct word, punctuation mark, sign or emoticon (e.g. *Figure 3*).

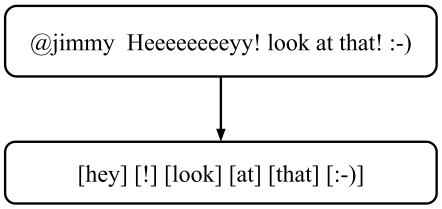

Figure 3. Tokenization of a text message

The total of 13,756 unique tokens generated from the text messages in the training set represents the vocabulary of the model. The Scikit-learn python library is used to build the vocabulary and extract text features. Each dataset entry is transformed into a feature vector with the length of the vocabulary. Two different vector representations are used.

- *Count:* each text is transformed into a vector of token counts with the CountVectorizer() function from the feature_extraction.text package.
- *Term frequency – inverse document frequency* (TFIDF): each text is transformed into a vector of token frequencies with the TfidfVectorizer() function from the same package. The values for the highest term frequencies, specific for common words that hold low significance (e.g. "the", "a") are inverted in order to minimize their influence.

The representation of a whole dataset is a matrix with one row for each entry and one column for each token in the vocabulary. Therefore, the dimensions of the training and testing matrices are 10,556x13,756, respectively 6,351x13,756. Each matrix is

mapped one-on-one with a vector of hate speech binary labels. The order of elements in the vector is the same as the order of messages represented in the matrix, so each entry can be correlated with its associated label.

3.3 Classifier

The generated matrix-vector representation is used with a support vector classifier, implemented with the Scikit-learn python library. [13] The SVC() function from the svm package has a series of parameters for the customization of this machine learning model.

- *Kernel:* Three different types of classifiers are created based on the kernel parameter that defines the basic function of the support vector: linear, radial basis function (RBF) and polynomial (of degree 2 and 3).
- *Class weight:* Balancing the class weights accounts for the uneven distribution of the both the training and the testing set, with 30% hate messages and 70% non-hate messages. This assures that the classifier is not biased towards labeling text as non-hate due to the larger size of that class.
- *C, gamma:* The values of these two parameters influence the creation of the support vector. *C* is the cost of misclassifying an entry and *gamma* is the influence of the distance between an entry and the possible vector function that is being defined. The gamma value affects only the RBF and polynomial function, while the C value affects the linear one as well.

For each classifier, the GridSearchCV() function from the model_seelction package is used to select the best values for C and gamma. The grid search yields the combination of values for both parameters that leads to the best measured classification performance. The grid search selection is based on a 10-fold cross validation process that splits the training set in ten subsets. For each subset, it trains the model on the remaining nine and measures the performance on the tenth. This results in 10 performance measurements which are then averaged in order to obtain the overall performance of the model on the training set. Cross validation assures that the measured performance of the model is not obtained by training and testing on the same data, which would result in an incorrectly high value.

Due to the high computational time needed to test several combinations through cross validation, the grid search is restricted at five medium values for C and gamma: [0.01, 0.1, 1, 10, 100]. This results in 5 trials for the linear classifier and 25 (5x5) for each of the RBF and polynomial classifiers. For each kernel option the parameter value selection is performed for both the Count and TFIDF feature vector approach.

3.4 Metrics

The performance of the classification model is measured in metrics extracted from the confusion matrix of a binary classifier (shown in *Table 1*).

Table 1. Confusion matrix

		Predicted Class	
		Non-hate	**Hate**
Observed Class	**Non-hate**	True Negative (TN)	False Positive (FP)
	Hate	False Negative (FN)	True Positive (TP)

The precision is used to measure how much was predicted correctly out of both classes and the recall is used to measure how much was predicted correctly out of the hate class. These two metrics are most suitable for class imbalanced datasets, where the results for the smaller class are more relevant to the overall performance of the model. A high number of correctly identified non-hate messages can make the performance erroneously seem better; therefore, the "true negatives" are avoided.

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

The F1 score combines the two values. This is the final metric used to assess the generalization performance of hate speech detection by comparing the obtained values for the training and the testing set.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4. RESULTS

Table 2 briefly displays the results of the classification models for comparison. The best C and gamma values resulted from the grid search trials for each model can be found in the Appendix, together with the extensive results of the classification performances for the testing set (containing the confusion matrix and values for precision and recall).

Table 2. Results

Model		**Training F1 score**	**Testing F1 score**
Linear	Count	0.803	0.329
	TFIDF	0.795	0.389
RBF	Count	0.801	0.311
	TFIDF	0.805	0.265
Polynomial degree 2	Count	0.759	0.245
	TFIDF	0.775	0.183
Polynomial degree 3	Count	0.707	0.297
	TFIDF	0.581	0.031

The models based on the linear and radial basis function reach the state-of-the-art value of 0.8 for the training set. The best training F1 score is obtained with the RBF TFIDF model, while the best testing F1 score is obtained with the linear TFIDF model. The polynomial function performs worse overall. The higher the degree of the polynomial the more the model overfits, resulting in lower scores even for the training set. Overfitting happens when the model is tailored too closely for the training set. The separation line between hate and non-hate is not suitable anymore for the data entries in the testing set. especially low values of the polynomial degree 3 TFIDF are excluded from further analysis of the results.

The average training F1 score is 0.778 and the average testing F1 score is 0.288. The difference between the two averages is 0.489, but this value is not consistent between models. A higher training F1 score does not always lead to a higher testing F1 score. The smallest difference of 0.406 is obtained between the training and the testing score of the linear TFIDF and it represents the best measured generalization performance. Nevertheless, this value is still too high, showing that the models generally perform bad

at correctly classifying hate speech from the testing set, even if F1 score values were high for the training set.

Count has a better F1 training score for the linear implementation, while TFIDF has a better score for RBF and polynomial of degree 2. For the testing set, it is the other way around. The average Count and RBF F1 scores are 0.768, respectively 0.792, for the training set and 0.296, respectively 0.279 for the testing set. The difference between these averages is 0.472 for Count and 0.513 for TFIDF. These values suggest that, overall, Count has a slightly better generalization performance than TFIDF, even though Linear TFIDF has the smallest difference among all models. For the special case of the polynomial of degree 3, Count outperforms TFIDF by far, with a difference of 0.126 on the training set, and an even greater performance difference for the testing set. Due to the removal of common, high-frequency words, TFIDF gives more importance to the specific discriminatory words used in the training set, which are not as relevant for the testing set.

The best F1 scores for the testing set are obtained when the True Positive value of the confusion matrix is the highest. The linear TFIDF correctly classifies 858 hate text messages. However, the False Positive value is also the highest. 1574 messages are classified as hate when they do not actually contain hate speech. The very low F1 testing score of the polynomial of degree 3 TFIDF model is caused by the fact that barely any messages are classified as hate: 34 True Positives and 213 False Positives. There is a tradeoff between better generalization performance and censoring messages that are not actually hate speech.

The inability to correctly classify hate and non-hate text is partially caused by the models not being able to recognize a lot of the words present in the testing set. There are 11266 tokens extracted from the testing dataset, of which 6238 cannot be found in the created vocabulary. When creating the feature vector for an entry, these out-of-vocabulary words are ignored, while they may actually hold significant meaning for the detection of hate speech. The created vocabulary is small due to the size of the initial dataset; however, the same problem might occur even if the dataset was larger. The specific slang used to discriminate the group in the testing set might not be present in the training set. Among the out-of-vocabulary words there are tokens representing terms that are almost exclusively associated with racist hate speech (e.g. ching, muslima, turkmen, nazis, soviets, #stopislam, all-muslim, islamolunatic, arabs, russ, kalishnikovs, #islamicstate, islam.that, islamist, infidels, islamofanatic, cleansing, jews, saudis, islamization). This problem would persist over time as new slang continuously develops to discriminate existing or new target groups.

5. DISCUSSION

Even though the measured generalization performance is bad, the implemented method is not representative of the large variety of hate speech detection methods. There are a lot of different classification methods described in existing literature that might lead to a better generalization performance [15,23].

Firstly, only surface features were extracted from the data used by the classification model. The use of additional advanced features might improve the generalization performance of support vector machines. A big limitation of this research was the small size of the dataset. The small training set resulted in a lot of out-of-vocabulary words. Lexical resources and sentiment analysis could be used for identifying the negative slang words specific for the testing set. Using word generalization features might improve performance by making it easier to generally recognize words from the testing set. Word embeddings is a word generalization method that creates a vector for each word in a text message. The vectors are used to represent the meaning of a word. Similar words can be identified by comparing their word vectors. External word datasets can be used to create these word vectors. Therefore, the identification of words in the testing set is no longer dependent on the small sample present in the training set. However, because all these features use external data, hate speech detection implementations would be dependent on the state of the external resources. The word datasets would need to be constantly updated according to linguistic changes to assure the recognition of any new slang that appears. The problem of out-of-vocabulary words is a current challenge in natural language processing research. New methods to deal with this issue might emerge in the future.

Secondly, different classification approaches can be experimented with. The generalization performance of other classical methods like Naïve Bayes and Logical regression, in combination with different features, can be tested. Deep learning implementations already show better results in existing literature and they might lead to better generalization performance values as well, due to the more abstract and complex representation of features. A combination between deep learning and classical methods can be considered as well, where neural networks are used only for the feature extraction stage [21]. Finally, transfer learning could be employed to train a model on a very large dataset to firstly distinguish between positive and negative messages [11]. The model can then be tailored for the problem of hate speech detection and updated accordingly when new slang or target groups appear.

Furthermore, the observed tradeoff between generalization performance and the incorrect classification of non-hate text as hate might lead to the restriction of free speech. The libertarian/egalitarian dilemma debates the issue of hate speech censoring [3]. The egalitarian view promotes censoring because hate speech is viewed as an obstacle for the people that are part of a discriminated group. On the other hand, the libertarian view promotes free speech as liberty of expression even if the speech is offensive. The behavior of hate speech classification methods can lean towards one of these views. Dealing with the observed trade off can be an ethical concern for the future when hate speech detection will be actively used in a real context.

6. CONCLUSION

The implemented support vector classifier with word count/ term frequency – inverse document frequency features, trained on a dataset containing sexist Twitter messages and tested on one containing racist messages, reveals a low ~0.3 F1 score value on the testing set, relative to the high ~0.8 value on the training set. The high difference between the two values proves a weak generalization performance for this classical machine learning method. The classifier is unable to recognize hate speech that is different from what it has been trained on. Therefore, it cannot recognize new types of discrimination that it has not encountered before. The constantly changing environment of the real world can lead to new hate speech targets, while machine learning is not able to identify discriminatory messages aimed at these new groups. This classification method is therefore not suitable for use as a hate speech detection tool on a social media platform, as it might lead to unfairly allowing discriminatory messages towards only specific targets. This research has been limited by the available resources, current state of technology and the reduced size of the dataset. The research question "*What is the generalization performance of hate speech detection using machine learning?"* can be further explored in the future. Different approaches from existing literature can be experimented with and new methods can be developed to try to solve the problem of hate speech generalization.

Finally, the generalization performance can represent a new dimension for the measurement of hate speech detection. This and possibly other new measurements can help guide the research in natural language processing in the future. Suitable performance measurements are important for the creation of hate speech classification methods that can be used to reliably and fairly censor content on social media platforms.

7. ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Lorenzo Gatti for the constructive feedback and suggestions provided throughout the development of this research work.

8. REFERENCES

[1] Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep learning for hate speech detection in tweets. *Proceedings of the 26th International Conference on World Wide Web Companion*, 759–760.

[2] Banks, J. (2010). Regulating hate speech online. *International Review of Law, Computers & Technology*, *24*(3), 233–239. https://doi.org/10.1080/13600869.2010.522323

[3] Brink, D. O. (2001). Millian principles, freedom of expression, and hate speech. *Legal Theory*, *7*(2), 119–157. https://doi.org/10.1017/S1352325201072019

[4] Burnap, P., & Williams, M. L. (2015). Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, *7*(2), 223–242.

[5] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). *Automated Hate Speech Detection and the Problem of Offensive Language*. Retrieved from https://www.aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665/14843

[6] Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M., & Tesconi, M. (2017). *Hate me, hate me not: Hate speech detection on Facebook*.

[7] Greevy, E., & Smeaton, A. F. (2004). Classifying Racist Texts Using a Support Vector Machine. *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 468–469. https://doi.org/10.1145/1008992.1009074

[8] Hanzelka, J., & Schmidt, I. (2017). Dynamics of cyber hate in social media: A comparative analysis of anti-muslim movements in the Czech Republic and Germany. *International Journal of Cyber Criminology*, *11*(1), 143–160.

[9] Loper, E., & Bird, S. (2002). NLTK: the natural language toolkit. *ArXiv Preprint Cs/0205028*.

[10] Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science & Engineering*, *9*(3), 10–20.

[11] Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

[12] Park, J. H., & Fung, P. (2017). One-step and two-step classification for abusive language detection on twitter. *ArXiv Preprint ArXiv:1706.01206*.

[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

[14] Perry, B., & Olsson, P. (2009). Cyberhate: the globalization of hate. *Information & Communications Technology Law*, *18*(2), 185–199. https://doi.org/10.1080/13600830902814984

[15] Schmidt, A., & Wiegand, M. (2017). A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, 1–10. https://doi.org/10.18653/v1/W17-1101

[16] Silva, L., Mondal, M., Correa, D., Benevenuto, F., & Weber, I. (2016). *Analyzing the Targets of Hate in Online Social Media*. Retrieved from http://arxiv.org/abs/1603.07709

[17] Warner, W., & Hirschberg, J. (2012). Detecting Hate Speech on the World Wide Web. *Proceedings of the Second Workshop on Language in Social Media*, 19–26. Retrieved from http://dl.acm.org/citation.cfm?id=2390374.2390377

[18] Waseem, Z., & Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. *Proceedings of the NAACL Student Research Workshop*, 88–93.

[19] Williams, M. L., & Burnap, P. (2015). Cyberhate on social media in the aftermath of Woolwich: A case study in computational criminology and big data. *British Journal of Criminology*, *56*(2), 211–238.

[20] Xiang, G., Fan, B., Wang, L., Hong, J., & Rose, C. (2012). Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 1980–1984.

[21] Yuan, S., Wu, X., & Xiang, Y. (2016). A Two Phase Deep Learning Model for Identifying Discrimination from Tweets. *EDBT*, 696–697.

[22] Zhang, Z., & Luo, L. (2018). Hate Speech Detection: {A} Solved Problem? The Challenging Case of Long Tail on Twitter. *CoRR*, *abs/1803.0*. Retrieved from http://arxiv.org/abs/1803.03662

[23] Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. *European Semantic Web Conference*, 745–760.

APPENDIX

Linear Count (C: 0.1)

training f1 score = 0.795

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3201	1121
	Hate	1367	609

testing recall = 0.308

testing precision = 0.352

testing f1 score = 0.329

Linear TFIDF (C: 1)

training f1 score = 0.803

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	2748	1574
	Hate	1118	858

testing recall = 0.434

testing precision = 0.353

testing f1 score = 0.389

RBF Count (C: 10, gamma: 0.01)

training f1 score = 0.806

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3212	1110
	Hate	1407	569

testing recall = 0.288

testing precision = 0.339

testing f1 score = 0.311

RBF TFIDF (C: 1, gamma: 1)

training f1 score = 0.805

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3244	1078
	Hate	1510	466

testing recall = 0.236

testing precision = 0.302

testing f1 score = 0.265

Polynomial of degree 2 Count (C: 0.01, gamma: 1)

training f1 score = 0.760

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3423	899
	Hate	1575	401

testing recall = 0.203

testing precision = 0.309

testing f1 score = 0.245

Polynomial of degree 2 TFIDF (C: 0.01, gamma: 10)

training f1 score = 0.775

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3524	798
	Hate	1697	279

testing recall = 0.141

testing precision = 0.259

testing f1 score = 0.182

Polynomial of degree 3 Count (C: 10, gamma: 0.1)

training f1 score = 0.707

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	3300	1022
	Hate	1454	522

testing recall = 0.264

testing precision = 0.338

testing f1 score = 0.297

Polynomial of degree 3 TFIDF (C: 1, gamma: 1)

training f1 score = 0.581

		Predicted Class	
		Non-hate	Hate
Observed Class	Non-hate	4109	213
	Hate	1942	34

testing recall = 0.0172

testing precision = 0.138

testing f1 score = 0.031