Classifying Racist/Sexist Tweets

Stephanie Doe

Lexa Huang

Zhijian Li

Tyler Wu

UCLA Mathematics

Math 156: Machine Learning

Dr Hanbaek Lyu

June 12, 2021

Introduction

In the Internet age, hate speech online has become an increasingly pressing issue, and it is vital to be able to efficiently automate a hate-recognition algorithm to identify such instances. Past research has seen success making use of Support Vector Machines (Greevy et al. 2004), and templates *([intensity][user intent][hate target])* (Mondal et al. 2017), or keywords (MacAvaney et al. 2019). Although existing research suggests "neural models have significantly higher AUC scores than the non-neural baseline", we would like to see if less computationally expensive models can perform reasonably well (Paul and Bhaskaran 2018).

This paper aims to evaluate and compare the performance of two models: Naive Bayes Classifier and Logistic Regression, in classifying tweets as either neutral or containing hate speech, as well as the usage of two vectorizers in feature extraction: bag-of-words and term frequency-inverse-document-frequency. Our paper shows that both models were able to effectively identify the presence of hate speech in tweets regardless of the chosen vectorizer, though the vectorizers impact the performance of each model differently. Both models resulted in accuracy scores of 95% and higher, with the Naive Bayes model slightly outperforming Logistic Regression.

Problem Statement

The rise of social media today has led to increasingly rampant hate speech online, as growing connectedness has facilitated the spreading of racist and sexist messages. This has harmful consequences beyond the virtual world - for example, hate speech online has led to hate crimes such as the terror attacks in New Zealand and the London attacks in the UK. As such, it is imperative for social media sites to be able to quickly identify and remove hate speech, and automating this process using machine learning methods removes the time-consuming and labour-intensive element of manual identification.

In this paper, we evaluate the performance of Naive Bayes Classifier and Logistic Regression in classifying a tweet as either neutral or containing hate speech. For the sake of simplicity, the definition of "hate speech" has been narrowed down to including racist or sexist sentiments. In performing feature extraction, we also evaluate the performance of two vectorizers commonly used in document classification: bag-of-words and term frequency-inverse-document-frequency.

Methods

We first split the data into X and y variables: the predictor variable X is the tweet, and the target variable $y \in \{0,1\}$ is the label representing whether the tweet is racist/sexist. $y = 0$ represents a neutral tweet, while $y = 1$ represents a racist/sexist tweet.

For X, a term-document matrix representing the frequency of different words in a tweet is constructed, using a standard vocabulary of 45534 words provided by SciKitLearn. We used two common ways to construct this matrix: using *bag-of-words (BOW)* which simply encodes the number of times a word appears in a tweet, and *term frequency-inverse-document-frequency (tf-idf)*, which divides the number of times a word appears in a specific tweet by the log of the proportion of tweets the word appears in. The vectorizer is set such that only words with a minimum of 2 appearances are included. Additionally, a standard list of stopwords from SKLearn's "english" list are removed, including words such as "and", "the". Upon checking the words with highest frequency of appearing in tweets, this list of stopwords is extended to include "user", "amp" (&), "get", "go" and "one", as these words appear frequently and we assume they are unlikely to influence the sentiment of a tweet, and removing them ensures they are not used as features in prediction.

	word	frequency
13848	user	14083
7820	love	2238
3233	day	1876
425	amp	1432
5844	happy	1367
13247	time	920
7581	life	895
13293	today	857
7613	like	843
8921	new	780
13060	thankful	752
10087	positive	745
5318	get	735
5476	good	691
9729	people	683
1291	bihday	668
9312	one	632
11510	see	602
12012	smile	560
5434	go	524

Figure 1: words sorted by frequency, from which we determined additional stopwords

This term-document matrix is used as our feature vector.

	fashionâ	fat	father	fatherly	fathers	fathersday
0	0.0	0.0	0.284179	0.0	0.0	0.0
1	0.0	0.0	0.000000	0.0	0.0	0.0
2	0.0	0.0	0.000000	0.0	0.0	0.0
3	0.0	0.0	0.000000	0.0	0.0	0.0
4	0.0	0.0	0.000000	0.0	0.0	0.0

Figure 2: part of the term-document matrix constructed by tf-idf vectorizer

The data is then split into train and test sets, with 80% as train set and the remaining 20% as test set. We then made use of two models to classify the tweets: Naive Bayes Classifier and Logistic Regression.

Naive Bayes Classifier

Firstly, we calculated the maximum likelihood prior distribution on class labels, which is the empirical probability mass function: number of class i tweets / total tweets:

prior: [0.93030991 0.06969009]

Notably, the majority of tweets were neutral (0.93), compared to racist/sexist tweets (0.07). Next, class-conditional probabilities for each class is computed as: number of word j in class i tweet / total number of words in class i tweets:

	bag-of-words	tf-idf
y = 0 (neutral)	[1.96858982e-05 1.36057814e-05 1.36057814e-05 ... 1.44554777e-06 1.41203265e-03 7.52566460e-06]	[2.35332325e-05 2.60698465e-05 1.86154175e-05 ... 3.93002460e-06 8.62532019e-04 1.14299811e-05]
y = 1 (racist/sexist)	[1.36649989e-06 1.36649989e-06 1.36649989e-06 ... 1.36649989e-06 1.36649989e-06 7.80930520e-05]	[3.72831320e-06 3.72831320e-06 3.72831320e-06 ... 3.72831320e-06 3.72831320e-06 1.05802587e-04]

We added a pseudocount of 10^{-5} to prevent zero probabilities from occurring.

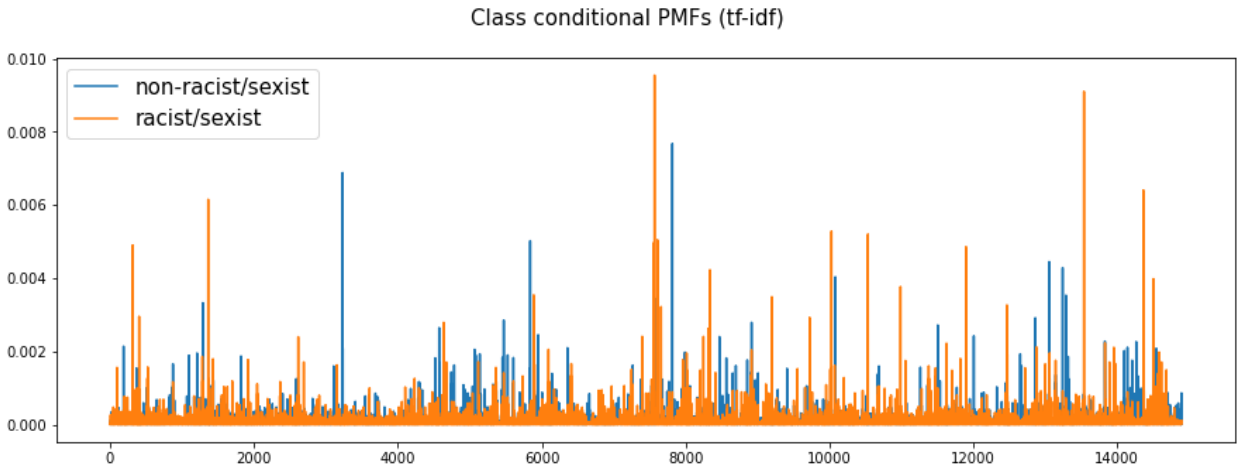

Figure 3: Class-conditional PMF of each class of tweets using tf-idf vectorizer

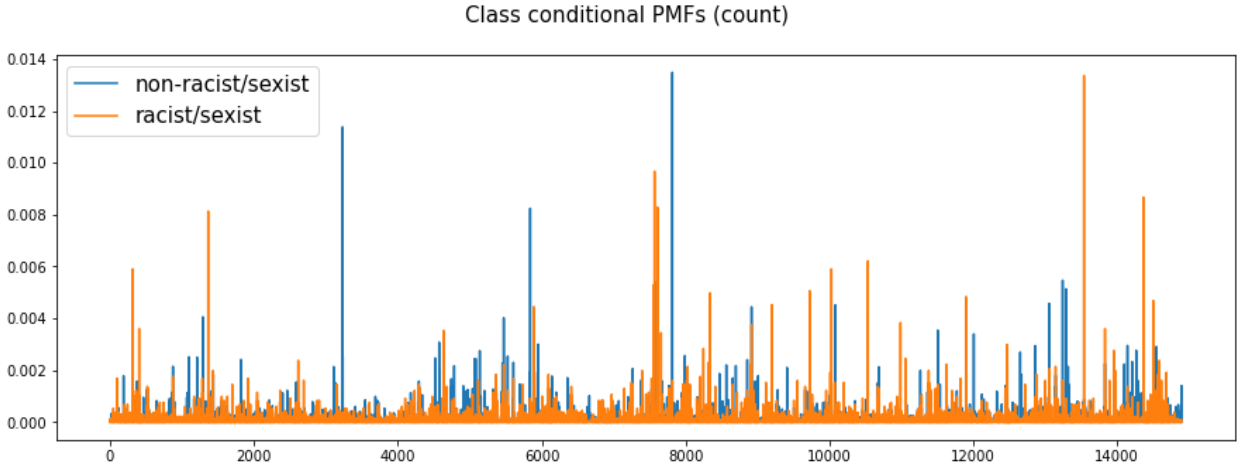

Figure 4: Class-conditional PMF of each class of tweets using BOW vectorizer

We then multiply the class-conditional probabilities by the prior distributions and divide them by the marginal probabilities to compute the predictive probability mass function:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Here we show the predictive PMFs of the first 6 tweets using each vectorizer:

BOW	tf-idf
[9.99983808e-01 1.61921649e-05]	[9.98135667e-01 1.86433278e-03]
[7.26445092e-01 2.73554908e-01]	[8.94887316e-01 1.05112684e-01]
[9.99998840e-01 1.15959126e-06]	[9.99766699e-01 2.33301127e-04]
[9.97455763e-01 2.54423721e-03]	[9.90830327e-01 9.16967288e-03]
[1.00000000e+00 6.75391493e-11]	[9.99233952e-01 7.66047963e-04]
[1.98133500e-05 9.99980187e-01]	[2.65897774e-01 7.34102226e-01]

The model then classifies the tweet by selecting the label with the higher predictive probability. For example, The fifth tweet has higher predictive probability for y = 0 than y = 1, and hence it is classified as a neutral tweet.

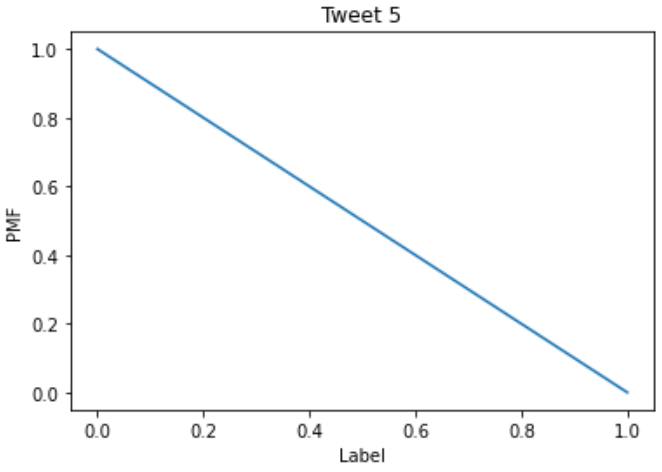

Figure 5: Predictive PMF of Tweet 5

Logistic Regression

For logistic regression, we used *sklearn.linear_model.LogisticRegression.* The term-document matrix is first multiplied with a coefficient vector W, and this matrix is mapped into the Bernoulli success probability p using the sigmoid activation function. W is then optimised using the default Limited-memory BFGS algorithm, and L2-regularization is automatically applied as part of the function.

Here we show the 10 words with highest coefficients. The appearance of these words contribute greatly to a tweet being labelled 1.

	words	coefs
0	allahsoil	4.485203
1	racism	3.130266
2	bigot	2.908237
3	white	2.641452
4	racist	2.620348
5	blacklivesmatter	2.483757
6	maga	2.468139
7	2017	2.331907
8	latest	2.322576
9	equality	2.276078

Figure 6: Top 10 highest coefficient words with their corresponding coefficients

On the other hand, the 10 words with lowest coefficients contribute greatly to a tweet being labelled 0.

	words	coefs
0	bihday	-2.213675
1	orlando	-1.813595
2	day	-1.806850
3	smile	-1.701322
4	hardcore	-1.643043
5	healthy	-1.599646
6	tomorrow	-1.598249
7	friday	-1.553784
8	life	-1.500524
9	weekend	-1.433967

Figure 7: Top 10 lowest coefficient words with their corresponding coefficients

We can see that the coefficients given by the model make sense, as the words with highest coefficients seem to reflect more sensitive topics, whereas the words with lowest coefficients are more neutral and contain more positive words such as “smile” and “healthy”. Lastly, the model makes a prediction based on whether the success probability p exceeds 0.5.

Theory

Naive Bayes Classifier

Using the Naive Bayes model, we classify a tweet’s sentiment by looking at the probability of it being racist/sexist based on the words it contains and assign the label with the higher probability computed. However, instead of modelling the predictive probability directly, we model the class-conditional probability - the probability of a tweet having certain words based on its label as either racist/sexist or neutral.

The output $y \in \{0,1\}$ is the label representing whether a tweet is racist/sexist. It is modelled as a discrete random variable Y with probability mass function (PMF) $[p_1,...,p_k]$ that depends on the observed feature $\phi(x)$ of each tweet, a term-document matrix representing the frequency of different words in a tweet.

$$\mathbf{P_i(Y = i \mid \phi(X)) = p_i \text{ for } i = 1,...,\kappa.}$$

The objective is to model the class-conditional probability - the probability that the observed feature vector $\phi(X) = [\phi_1(X),...,\phi_p(X)]$ equals $[\phi_1,...,\phi_p]$ given that the class label Y is i. In order to do so, we rewrite the above with Bayes' Theorem:

$$\mathbf{P_i(Y = i \mid \phi(X) = [\phi_1,...,\phi_m]) \propto P¡(\phi(X) = [\phi_1,...,\phi_p] \mid Y = i) P(Y = i)}$$

Predictive probability $\propto$ class-conditional probability * prior distribution

Where we can now determine the relative predictive probabilities by multiplying the class-conditional probabilities and prior distributions.

Since the term-document matrix is based on a standard vocabulary of 45534 words provided by SKLearn, the feature vector is of an extremely large dimension p, and the resulting joint distribution has $(p-1)^2$ degrees of freedom to be specified. To reduce the model's complexity, we can make the Naive Bayes assumption, where given the class label Y = i, the features $\phi(X) = [\phi_1(X),...,\phi_p(X)]$ are independent. As such, we only need to model the marginal class-conditional probabilities individually, then multiply all to get the joint class-conditional probability.

Such an assumption is valid in this case, since every word in a tweet can be considered to be independent of other words. The model will not be valid in cases where the words in a given document are not independent of each other.

Modelling the marginal class-conditional probability of the j-th feature given i-th class using the parameter w_{ij}, the joint likelihood of observing the class labels is as follows:

$$L(y_1, \ldots, y_N; \mathbf{W}) := \prod_{s=1}^{N} \prod_{i=1}^{\kappa} \left(\prod_{j=1}^{p} \mathbb{P}\left(\phi_j(\mathbf{X}) = \phi_j(\mathbf{x}_s) \,\middle|\, Y_s = i\right) \mathbb{P}(Y_s = i) \right)^{\mathbf{1}(y_s = i)}.$$

The optimal parameter W can then be found by the maximum likelihood method:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}=(q_i, q_{ij})}{\arg\max} \left[\sum_{s=1}^{N} \sum_{i=1}^{\kappa} M\mathbf{1}(y_s = i) \log q_i + \sum_{s=1}^{N} \sum_{j=1}^{M} \sum_{i=1}^{\kappa} \mathbf{1}(y_s = i) \phi_j(\mathbf{x}_s) \log(q_{i,j}) \right]$$

In which the solution W = (q_i ,$q_{i,j}$) for tweet classification is given by:

$$\hat{q}_i = \frac{1}{N} \sum_{s=1}^{N} \mathbf{1}(y_s = i), \qquad \hat{q}_{i,j} \propto \sum_{s=1}^{N} \mathbf{1}(y_s = i) \phi_j(\mathbf{x}_s) \quad \text{for } j \in \{1, \ldots, M\}.$$

Lastly, multiplying the class-conditional probabilities and prior distributions gives the relative predictive probabilities of a tweet being classified into either class, and a prediction is made based on the maximum a posteriori (MAP) decision rule, which selects the label with the highest predictive probability.

The Naive Bayes classifier is advantageous in that it avoids problems associated with the curse of dimensionality, since the features are assumed to be independent and each distribution can be computed in one-dimension. As such, it can be used on very large datasets with low computational cost. Additionally, even though the predictive probabilities calculated by the model may be inaccurate, by using the MAP decision rule, the classifier makes the correct choice of label as long as the correct label has higher probability than other labels, even if this probability is only slightly higher. However, as mentioned above, this classifier relies on the assumption of independent features, and will not work on data whose features are correlated. Furthermore, Naive Bayes makes use of prior distributions in its computations, which implies that the dataset evaluated should more or less have a ratio of labels representative of reality. As such, our dataset may face the disadvantage that it most likely oversamples racist/sexist tweets.

Logistic Regression

Before discussing the specifics of logistic regression, it is important to understand the general properties of linear classification models. These models are based on decision algorithms which take input vectors x and assign them to one of several classes. Assuming we are working with the tweet dataset, our target variables take a value in {0,1}, representing the extreme value probability that the tweet is racist/sexist.

There are several different approaches which can be taken to construct this classification model. While there exists a more straightforward approach given by the use of a discriminant function to immediately assign labels, logistic regression works best with a separable inference-decision technique. In this strategy, the inference stage creates a model of conditional probabilities, $p(\mathcal{C}_k|\mathbf{x})$ which are optimized using training data, and the decision stage applies this model towards classification of test data. In essence, we classify tweets based on which class label has a higher probability. This technique relies on direct modeling rather than extensive calculations using Bayes theorem to solve for posterior class probabilities.

To compute the logistic regression we utilize input data, features and binary class labels. We consider the output to be a Bernoulli random variable,

$$\Pr(X = 1) = p = 1 - \Pr(X = 0) = 1 - q,$$

with probability p based on the features of our input vector. The conditional probability function previously mentioned is written as a logistic sigmoid function where the input is a function of the feature vector.

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right)$$

This is where classification differs from regression, since we want a nonlinear function to fit to the discrete labels. The logistic sigmoid is a good activation function for this

task as it maps to values in the range [0,1]. This allows us to represent results as probability values and it well fits the output data of values {0,1}.

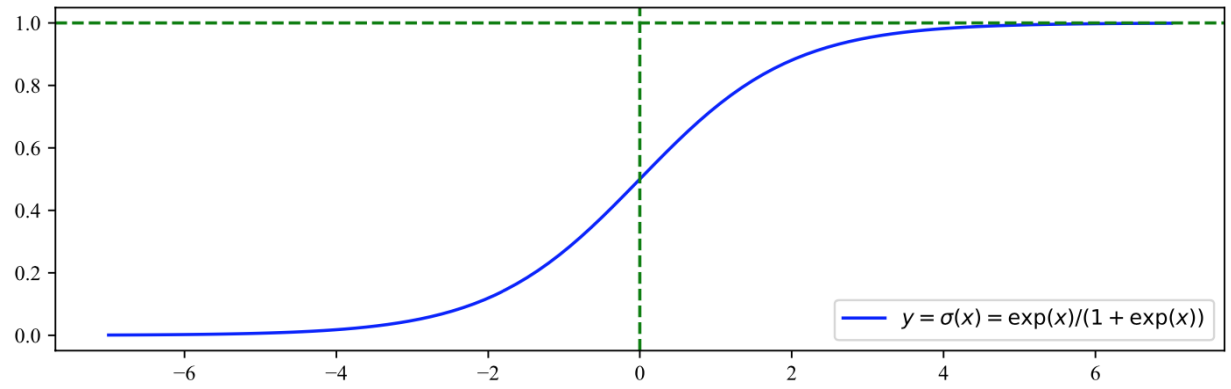

Figure 8: Plot of sigmoid function

To fit the probabilistic model parameters to the training data, we estimate the maximum likelihood that our predicted labels match the true labels. This is expressed through the joint likelihood function,

$$L(y_1,\ldots,y_N;\mathbf{w}) = \mathbb{P}(Y_1 = y_1,\ldots,Y_N = y_N;\mathbf{w}) = \prod_{i=1}^{N} p_i^{y_i}(1-p_i)^{1-y_i}.$$

which can be used to solve for the parameter w by maximizing the log likelihood. This is the same as minimizing the logistic regression loss function,

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min}\left[\ell_{LR}(\mathbf{w}) := \left(\sum_{i=1}^{N}\log\left(1+\exp(\boldsymbol{\phi}(\mathbf{x}_i)^T\mathbf{w})\right)\right) - \mathbf{Y}^T\boldsymbol{\Phi}^T\mathbf{w}\right],$$

This is a convex optimization problem which can be solved with an iterative algorithm. Our model made use of the Limited-memory BFGS algorithm, as this lowered the computational cost than using gradient descent or Newton-Ralphson. This is because L-BFGS requires linear memory and is hence suitable for our problem, which evaluates a large dataset with many variables.

Logistic regression works well for disjoint classes, such as classification problems as it classifies outputs as belonging to only one of the possible classes. Thus it fails in situations with non-discrete or scaled outputs. It also works well for a large number of features, and conversely suffers in datasets that are too small, since overfitting may occur. Other advantages of this method relate to the use of discriminative methods, as accuracy of prediction is more likely than a generative approach which suffers from assumptions about class-conditional density. It can also convey the importance of specific features based on the chosen coefficients and does not depend on a certain distribution of target variables. Disadvantages lie in the assumption that features are highly uncorrelated.

Dataset Description

The dataset is obtained from Kaggle and contains 29530 unique tweets, each labelled with 1 or 0, where 1 represents a tweet with racist or sexist sentiments, and 0 represents a neutral tweet. 93% of the tweets are labelled 0.

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð□□...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...

Figure 9: First 10 tweets in the dataset

```
11966    flaxseed gel/shampoo results!  #peace #love #o...
24368     @user great end to a great day! happy weekend...
31695    @user given what we now know, how the church s...
26812             happy day!  #altwaystoheal  #healthy
23501    18 km in the rain along the thames. perfect st...
24759    #zaynmalik   bull up: you will dominate your b...
22545                  @user true dat ... thanks !   @user
17153    #fruit #tea #peach #drink #yummy   #dayoff #li...
29593     @user @user well here i am...just waiting...a...
20324    okay i think i've got my @user and @user chann...
```

Figure 10: Sample tweets labelled 0 (neutral)

```
31364    america isn't racist. their are a plethora of ...
11515    the latest the indigenous and remote daily!  t...
5128     i think his the worst in everything,  his hatr...
19975    #thanks to all those  #dead #white #dudes from...
7085     happy new year white people elected a  preside...
14082    are you #black &amp;amp; feel like the  are â...
24026    @user after 8 yrs of endless lies, contempt fr...
22285    thought factory: left-right polarisation! #tru...
10852    @user #allahsoil not all muslims hate america....
14530    trump is a cuck  @user #nazi #swastika #ebay! ...
```

Figure 11: Sample tweets labelled 0 (neutral)

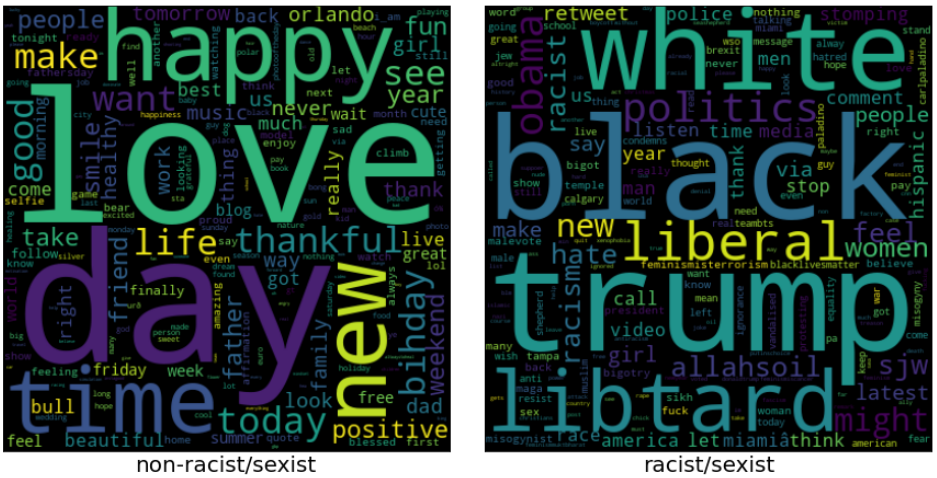

Figure 12: Word Cloud of top words found in each class of tweets in the train set using BOW vectorizer

Results

In our dataset, the Naive Bayes Classification model gave a highly accurate classification of tweets. Both vectorizers produced extremely similar accuracy rates, with 0.962 using the tf-idf vectorizer and 0.961 using the bag-of-words vectorizer.

However, since the priority is on correctly classifying racist/sexist tweets (at the risk of Type II error), the precision and fall out rate metrics are of greater importance to us. While the two accuracies were similar, the tf-idf vectorizer had a much higher precision (0.913) and lower fall out rate (0.004) than bag-of-words, (precision = 0.754, fall out rate = 0.017), indicating it correctly classified more racist/sexist tweets. On the other hand, bag-of-words performed slightly better than tf-idf on specificity, suggesting it had a higher accuracy of classification within the total of those classified as neutral.

Therefore, the implications of our model would strongly vary based on which vectorizer we chose to use. If we used tf-idf, it would suggest that we prioritize a highly accurate subset of racist/sexist tweets at the expense of having mis-classified some as neutral (false-negative). However, if we used bag-of-words, it would suggest that we are more concerned about ensuring accuracy in the true positive and true negative subsets of data, at the expense of letting some mis-classifications into either set. This is evident visually, as the confusion matrix suggests that the bag-of-words vectorizer mis-classified more neutral tweets as racist/sexist than the tf-idf vectorizer.

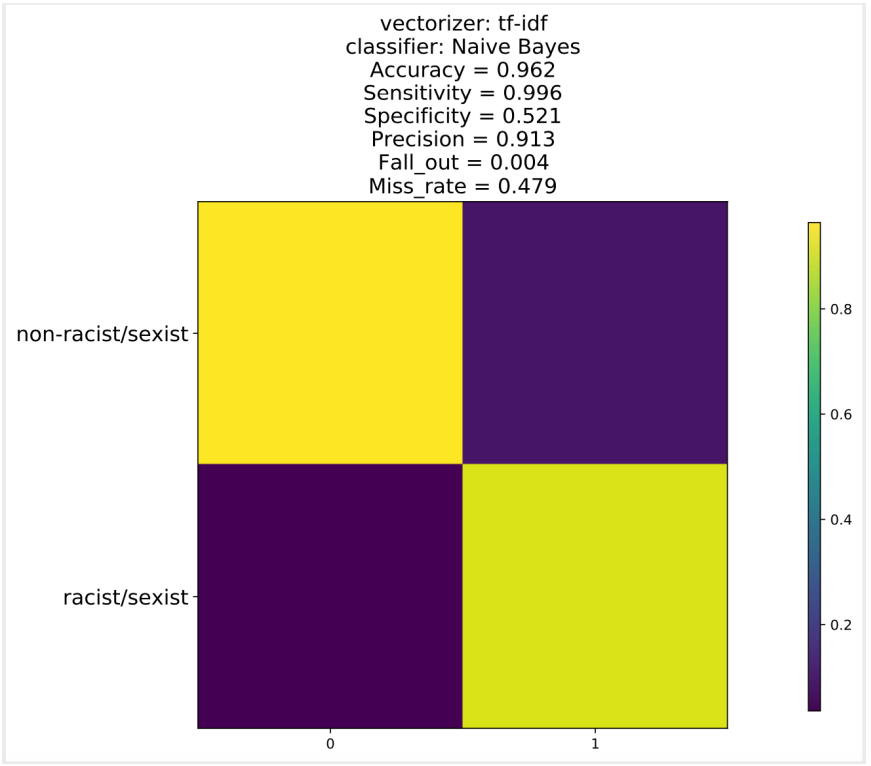

Figure 13: Confusion matrix and accuracy metrics of Naive Bayes model using tf-idf vectorizer

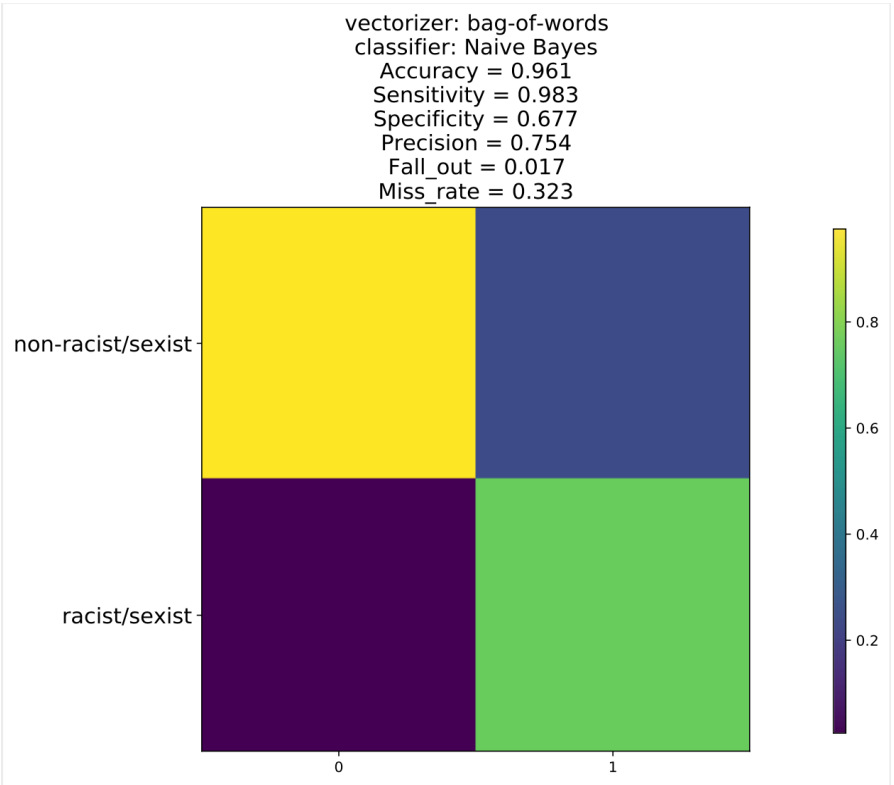

Figure 14: Confusion matrix and accuracy metrics of Naive Bayes model using BOW vectorizer

For our logistic regression model, our results gave similarly high accuracy rates in which the bag-of-words vectorizer performed slightly better with an accuracy score of 0.960 compared to the tf-idf vectorizer's accuracy score of 0.949. This time, the tf-idf vectorizer gave higher rates in terms of sensitivity, precision, and miss_rate whereas the bag-of-words vectorizer produced higher scores on specificity and fall_out statistics. This suggests that the tf-idf vectorizer produced more false negatives and true positives while the bag-of-words vectorizer produced more false positives and true negatives, as evident in the confusion matrix.

Our results indicate that. with logistic regression, if we were to use the bag-of-words method we would have a highly accurate subset of neutral tweets, with many neutral tweets misclassified as racist/sexist, such that we skew towards catching every possible sexist-racist sentiment rather than focusing on a high positive predictive value for racist/sexist tweets as given by tf-idf. This conclusion makes sense given the structure of these two vectorizors. While bag-of-words simply counts the presence of relevant words within each tweet, tf-idf discounts word frequency if it occurs in many tweets. So bag-of-words would be more likely to characterize any tweet with racist phrases as racist/sexist, whereas tf-idf would be weighted towards those with uniquely hateful sentiments.

Tf-idf

```
AUC ===> 0.633
Opt_threshold ===> 1.000
Accuracy ===> 0.949
Sensitivity ===> 0.998
Specificity ===> 0.268
Precision ===> 0.913
Fall_out ===> 0.002
Miss_rate ===> 0.732
Confusion matrix
 ===>
 [[5949   11]
 [ 317  116]]
```

Bag-of-words

```
AUC ===> 0.745
Opt_threshold ===> 1.000
Accuracy ===> 0.960
Sensitivity ===> 0.993
Specificity ===> 0.497
Precision ===> 0.846
Fall_out ===> 0.007
Miss_rate ===> 0.503
Confusion matrix
 ===>
 [[5921   39]
 [ 218  215]]
```

Overall, we found that the naive bayes model performs better than the logistic regression but only by a minute amount. In accordance with the literature, both models performed very well regardless of the chosen vectorizer. However, our results contradict observations that Naive Bayes performs less accurately than logistic regression with larger datasets. This could be explained by our models only accounting for single words, and not n-grams (i.e. sequences of words).

Conclusion

Through our tests, we concluded that both Naive Bayes Classification and Logistic Regression are able to reasonably evaluate the presence of hate speech in tweets. This is evident through our results which exhibited accuracy scores of 95% and greater when trained on 80% of our dataset. That being said, there are some problems that should be acknowledged before fully accepting our models as effective. Most significantly our data is likely imbalanced. Through our dataset, we made the assumption that 7% of our tweets were racist/sexist, but this is most likely not an accurate representation of the true hate-speech percentage on Twitter. This can have critical implications on our Naive Bayes Classification model, because an inaccurate representation of the hate-speech ratio would affect our prior

probabilities and therefore the optimization of our model. Overall, our paper suggests that it is possible to automate a hate-recognition algorithm to efficiently mark and potentially remove hate-speech from online. This ability, through the utilization of a classification model, will be extremely beneficial in our increasingly digital society.

References

Paul, S. (2018). ERASeD : *Exposing Racism And Sexism using Deep Learning*.

Abro, S., Shaikh, S., Ali, Z., Khan, S., Mujtaba, G., & Khand, Z.H. (2020). *Automatic Hate Speech Detection using Machine Learning: A Comparative Study.* International Journal of Advanced Computer Science and Applications, 11.

Greevy, E., & Smeaton, A. (2004). *Classifying racist texts using a support vector machine.* SIGIR '04.

Mondal, M., Silva, L.A., & Benevenuto, F. (2017). *A Measurement Study of Hate Speech in Social Media*. Proceedings of the 28th ACM Conference on Hypertext and Social Media.

MacAvaney, S., Yao, H., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). *Hate speech detection: Challenges and solutions.* PLoS ONE, 14.

Data source:

https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech?select=train.csv