ANALYSIS AND THE APPLICATION OF CLASSIFICATION MODELS IN DETECTING ONLINE HATE SPEECH

MEMBERS

Ami, Julius Lorenz

Capulong, Mark

Cueva, Larry Miguel

Osorio, Bryan

Ore, Gabriel

Panlilio, Andrei Shane

Zabala, Leonard Howell

INTRODUCTION

Freedom of speech or expression is very important for a country to give everyone the right to express their thoughts and opinions about a certain idea or topic. But not every thought or opinion they express is a sensible one. This is where hate speech comes along, it is hate speech when it is directly attacking a person or group of people in an offensive manner.

LITERATURE REVIEW

Title	Description
How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets? Fortuna, P., Soler-Company, J., & Wanner, L. (2021)	The paper discussed the cross-dataset generalization and discussed the possible impact of dataset characteristics and model in generalization.
A Survey on Hate Speech Detection using Natural Language Processing. Schmidt, A. and Wiegand, M. (2017)	Discussed the detection of hate speech in the growing body of social media content in the entire world, to determine the rise of hate speech in the platform.
***Towards generalisable hate speech detection: A review on obstacles and solutions*.** Zubiaga, A., et. al (2021)	Hate speech detection can be challenging due to the limitations of existing NLP techniques, dataset construction, and the characteristics of online hate speech, which are frequently mixed up.
Text Classification Using Support Vector Machine with Mixture of Kernel. Wei, L, Wei, B. and Wang, B. (2012)	Most of the classifiers are based on the methods from information retrieval and the machine learning algorithms that are introduced for text classification purposes.

LITERATURE REVIEW

Title	Description
Analytics of machine learning-based algorithms for text classification. Hassan, S.U., Ahamed, J., Ahmad, K. (2022)	Logistic Regression is quick to train data, works well for categorical data, for Simple parameter Estimation and better for linear data. It is not better for non-linear data and requires a large sample size.
A Survey on Text Classification: From Traditional to Deep Learning. Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S. and He, L.. (2021)	The multilayer perceptron from the study of Alsmadi, et al. (2009) and the recursive neural network from Pouyanfar, et al. (2018) are the first two deep learning approaches used for the text classification task, which improves performance compared with traditional models. CNNs, Recurrent Neural Networks (RNNs), and attention mechanisms are used for text classification. (Qin, L., et al. 2020, Deng, Z., et al. 2021 and Zhao, et al. 2018)

PROBLEM

- Opinions
- Internet and Social Media limits
- Abuse of freedom of Expression
- Combat Hate Speech

DATASET DESCRIPTION

Combined, handpicked, and curated these datasets to form a much larger corpus of words and sentences from Mollas, I., Chrysopoulou, Z., Karlos, S. et al. (2022), Kurrek, J., Saleem, H. M.; Ruths, D. (2019), and T-Davidson. (2020).

- ETHOS Hate Speech Dataset (2022)
- Automated Hate Speech Detection and the Problem of Offensive Language (2019)
- Towards a Comprehensive Taxonomy and Large-Scale Annotated Corpus for Online Slur Usage (2020)

DATASET DESCRIPTION

The combined corpus contained a total of 65780 records of comments drawn from the three sources together with its labels
3 - Homonym which had 1998 records
2 - derogatory which had 22395 records
1 - non-derogatory which had 21644 records
0 - offensive which had 19743 records

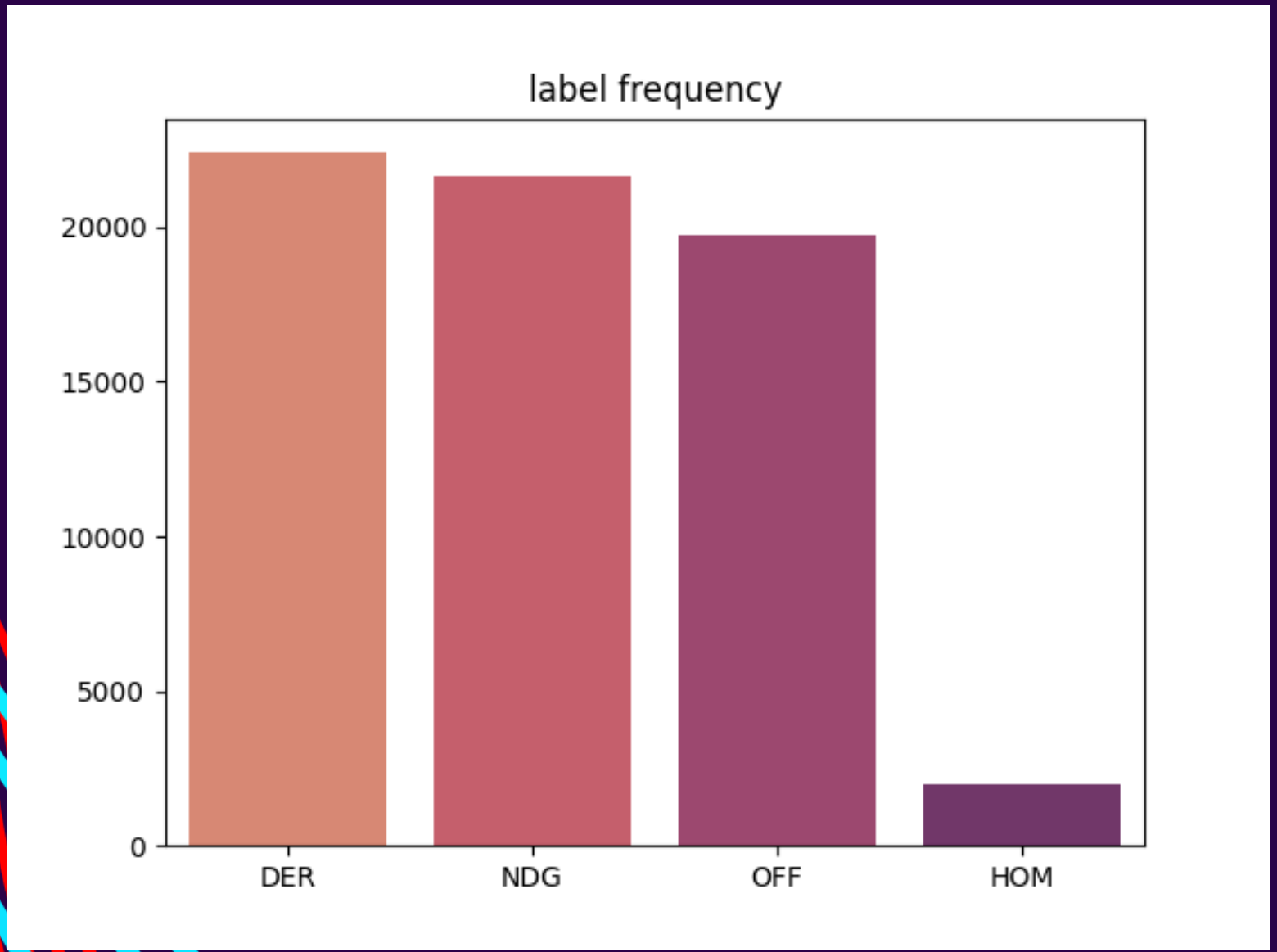
Derogatory by definition is showing a critical and disrespectful attitude to someone. **Non-derogatory** by definition is the opposite of derogatory. **Offensive** by definition are the comments that can cause someone to feel upset and angry, mainly because they are rude. **Homonymous,** on the other hand is defined as a word spelled or pronounced the same but have different meaning.

DATASET DESCRIPTION

	comment	label
0	michelle fuck obama tranny dick bitch	2
1	bitch screaming fuck nigga get mad another bit...	0
2	movie catch young older got gi jane wonder man...	2
3	like bitch u aint first place	0
4	got first time eating dick made someone faggot	2
...	...	...
65775	gtgirls guy tranniesis anyone wont bed hey com...	1
65776	bro rippin apart bird foot u redtail hawk	1
65777	gorillaz like two hit wonder edit three hit wo...	2
65778	goat nitty bitch rubbin beard stomach time jol...	0
65779	faggot point kid dumb high teenager going stup...	2

65780 rows × 2 columns

AFTER PRE-PROCESSING


label frequency
20000
15000
10000
5000
0
DER
NDG
OFF
HOM

PURPOSE

- The main purpose of this experiment is to classify hate speeches using Long Short-Term Memory (LSTM) and Softmax Regression. The experiment is inspired by the study of Muslim, A., et al. (2020), where they used it for the Twitter dataset.
- According to the research of Almeida F. (2019), word embeddings are highly useful in NLP tasks such as parsing and sentiment analysis, chunking, and question and answer. These advancements are now included in several toolkits, such as Word2Vec, Gensim, and GloVe, allowing for more accurate and less time-consuming usage for word embeddings. With those findings, this experiment will also utilize word embeddings, and the results from LSTM and Softmax Regression will be compared to see which methods will provide better results and be able to classify hate speeches correctly.

PURPOSE

Once engineering the dataset was finished, we wanted to answer the following questions

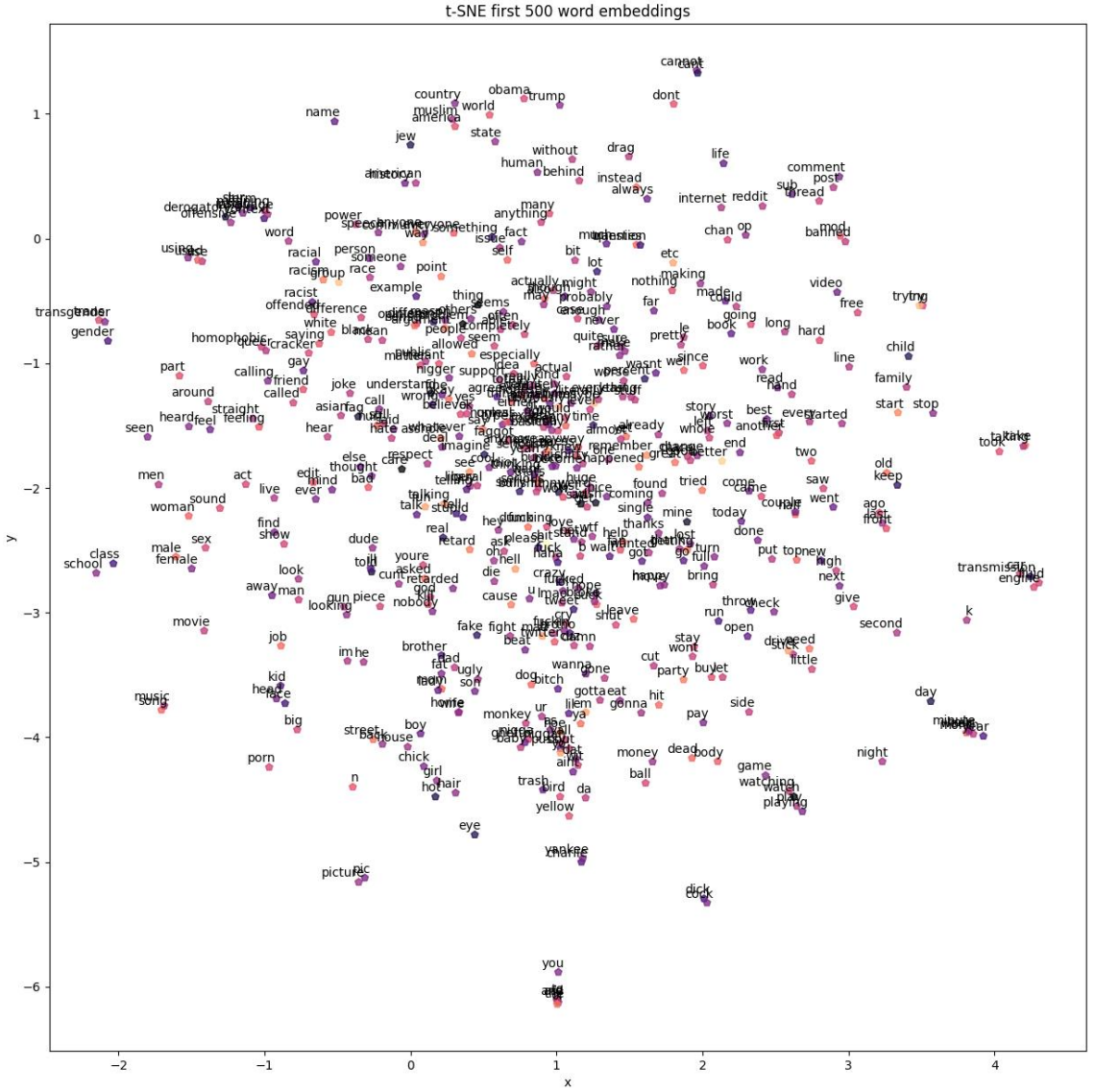
1. What words were most frequently attributed in derogatory comments?
2. What were the percentages of the frequent derogatory comments?
3. What words were most frequently attributed in offensive comments?
4. What were the percentages of the frequent offensive comments?
5. What words were most frequently attributed in non-derogatory comments?
6. What were the percentages of the frequent non-derogatory comments?
7. How best would we be able to automate the detection or even regulate these inflammatory comments?

PROPOSED SOLUTION

- we propose a two classifiers of these comments. The other using a deep learning approach particularly an LSTM Neural Network model and the other a simple generalized version of logistic regression called the Softmax Regression model suited best for multi class classification
- Machine learning models work on numbers the next problem was how would we feed words to a model such that it was able to detect sentences or phrases that were known to either be derogatory, non-derogatory, offensive, or homonymous

PROPOSED SOLUTION

- According to Ameida, F. & Xexeo, G. (2019) words have been found to be useful by representing them as vectors which lend themselves well to be used in many Machine Learning (ML) algorithms and strategies.
- Also stated that words with similar contexts (other words) have the same meaning.


t-SNE first 500 word embeddings
x
y

PROPOSED SOLUTION

- We can represent these embeddings in a lower dimensional such that words can be visualized in a 2D plane or 3D plane. T-distributed Stochastic Neighbor Embedding (T-SNE) is a machine learning algorithm for data visualization, which is based on a nonlinear dimensionality reduction technique.
- More recently ways of creating embeddings have surfaced, Among the most influential models is that of the GloVe model by Pennington, J., et al. (2014).
- Another model proposed was named ELMo (Embedding from language models) looks at the entire sentence as it assigns each word an embedding. Since it uses a bidirectional architecture, the embedding is based on both the next and previous words in the sentence. M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proc. of NAACL, 2018
- Many of the suggested advances seen in the literature have been incorporated in widely used toolkits, such as Word2Vec, gensim, FastText, and GloVe, resulting in ever more accurate and faster word embeddings to be used in NLP, as stated by Ameida, F. & Xexeo, G. (2019)

PROPOSED SOLUTION

- While various models have been proposed to train embeddings of numerous words such as Word2Vec, GloVe, ELMo etc., sometimes sparsity in data can occur.
- Open source and publicly pre-trained embeddings can be of great use
- we propose to use in this problem we can represent the words in our corpus in a vector space such that it is useful later, in the two models we have chosen to classify and detect certain phrases with derogatory, non-derogatory, offensive, or homonymous words.
- Common crawl pre-trained word vectors also by Pennington, J., et al. (2014)

LSTM ARCHITECTURE AND HYPER PARAMETERS

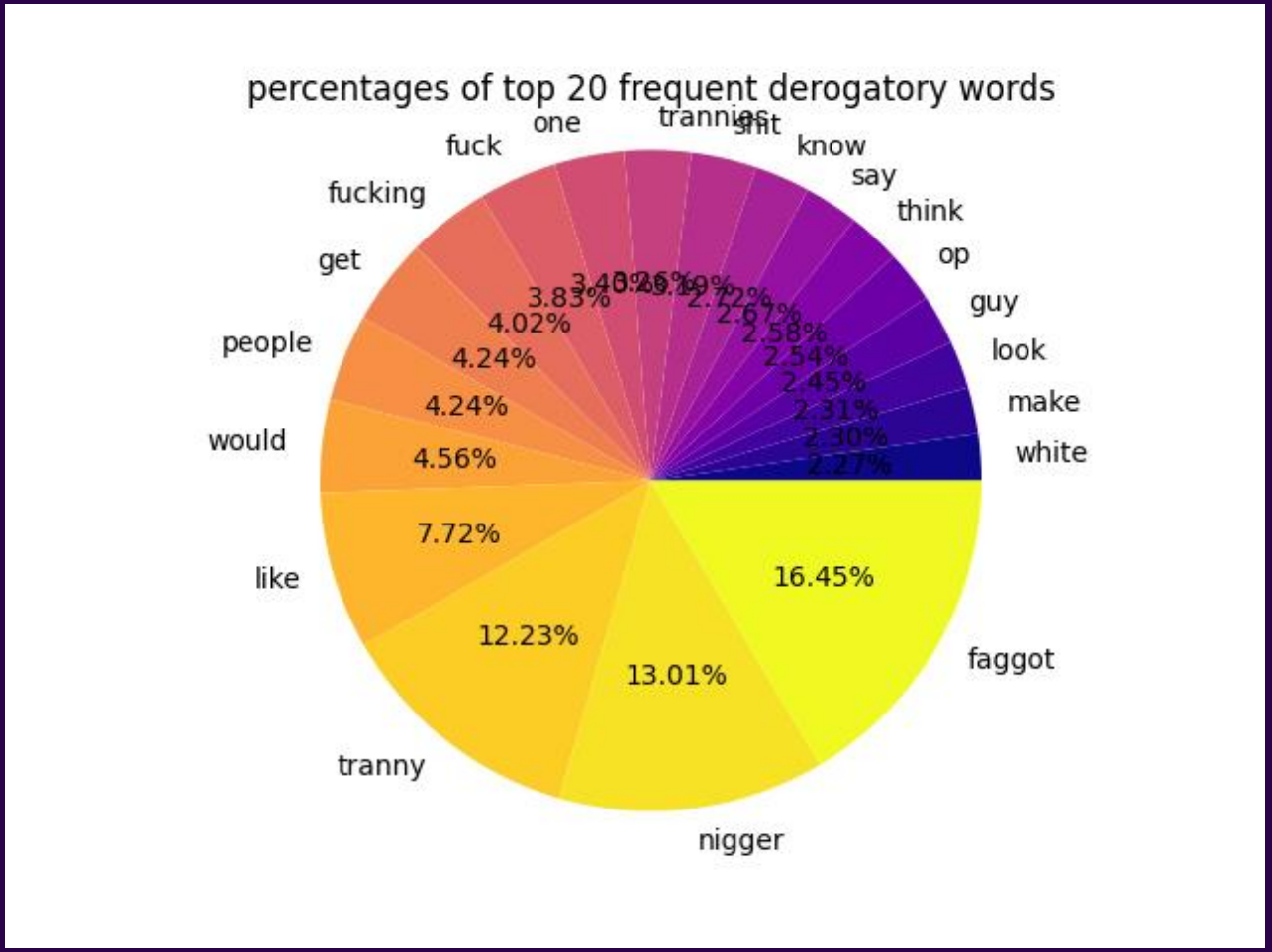
```python
def init_embedding_layer(vocab_len, emb_matrix):
    emb_dim = emb_matrix.shape[1]
    embedding_layer = Embedding(vocab_len, emb_dim, trainable=False)
    embedding_layer.build((None,))
    embedding_layer.set_weights([emb_matrix])

    return embedding_layer


def load_lstm_model(input_shape, vocab_len, emb_matrix):
    """
    Define architecture of LSTM model then return for later training

    Takes in also the embedding layer with weights/coefficients set
    to the pre-trained GloVe embeddings
    """
    print(f'input shape: {input_shape}')

    seqs_padded = Input(shape=(input_shape, ), dtype='int64')
    embeddings = init_embedding_layer(vocab_len, emb_matrix)(seqs_padded)
    A1 = Bidirectional(LSTM(units=16, return_sequences=True))(embeddings)
    D1 = Dropout(0.5)(A1)
    A2 = Bidirectional(LSTM(units=16, return_sequences=False))(D1)
    D2 = Dropout(0.5)(A2)
    Z3 = Dense(units=4)(D2)
    A3 = Activation('softmax')(Z3)

    model = Model(inputs=seqs_padded, outputs=A3, name='hate-speech-lstm')

    model.compile(
        loss=cce_loss(),
        optimizer=Adam(learning_rate=0.001),
        metrics=[cce_metric(), CategoricalAccuracy()]
    )

    return model
```

SOFTMAX ARCHITECTURE AND HYPER PARAMETERS

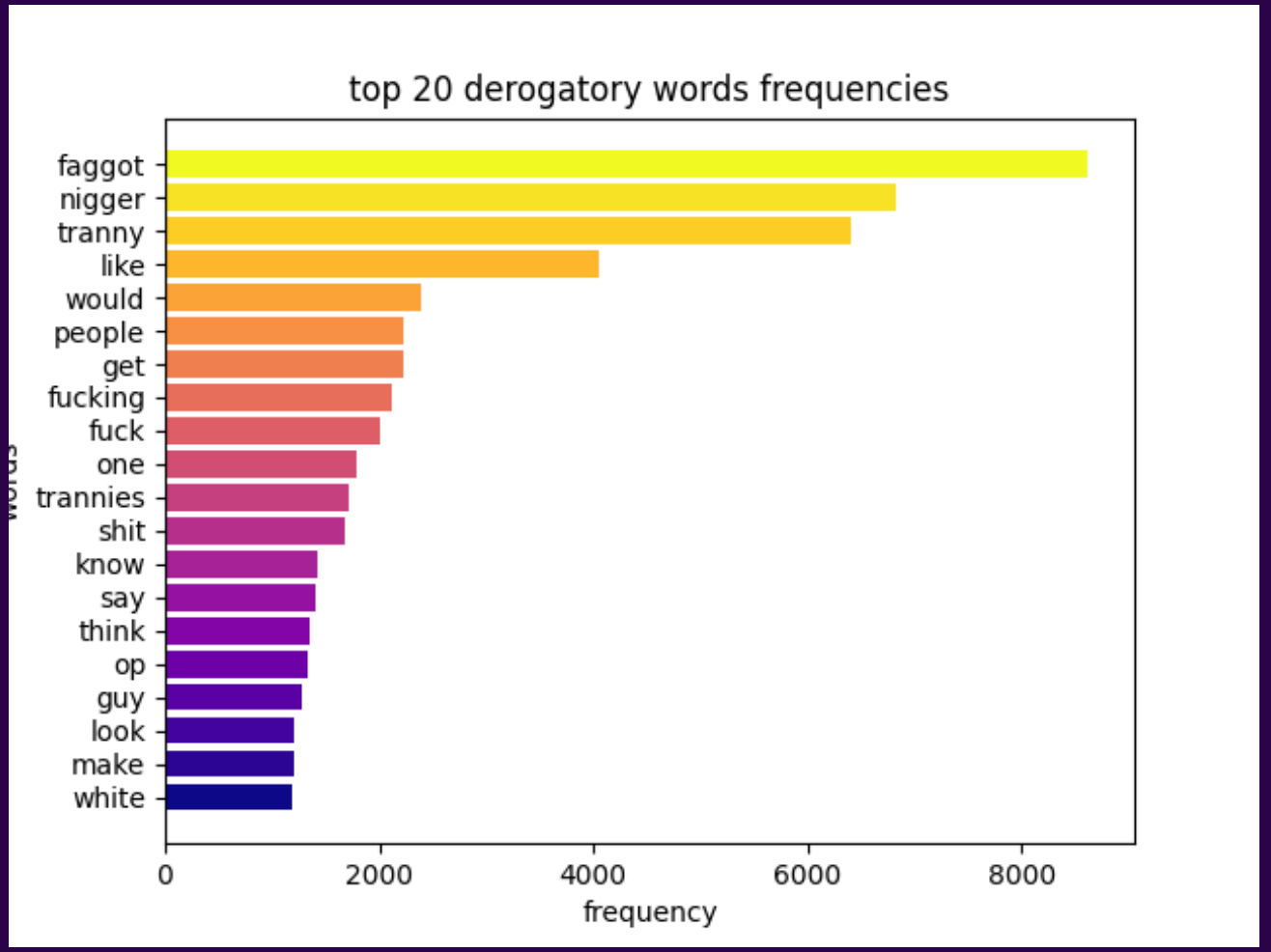

```python
class SoftmaxRegression:
    def __init__(self,
                 train_X,
                 train_Y,
                 val_X,
                 val_Y,
                 epochs=5000,
                 rec_ep_at=500,
                 learning_rate=0.03,
                 regularization='L2',
                 lambda_=0.1,
                 ) -> None:
        self.train_X = train_X
        self.train_Y = train_Y
        self.val_X = val_X
        self.val_Y = val_Y

        self.epochs = epochs
        self.rec_ep_at = rec_ep_at

        # X will be (num_instances x num_features)
        self.num_instances = train_X.shape[0]
        self.num_features = train_X.shape[1]

        # Y will be (num_classes x num_instances)
        self.num_classes = train_Y.shape[1]

        self.learning_rate = learning_rate
        self.lambda_ = lambda_
        self.regularization = regularization

        self.history = {
            'history': {
                'train_loss': [],
                'train_categorical_accuracy': [],
                'val_loss': [],
                'val_categorical_accuracy': []
```

```python
def train(self, show_vars=True):
    # maybe hte bug occurs because Y and X are transposed versions of each other
    train_X = tf.constant(self.train_X)
    train_Y = tf.constant(self.train_Y)
    val_X = tf.constant(self.val_X)
    val_Y = tf.constant(self.val_Y)

    # initializer parameters e.g. because X is (m x nf) and Y is
    # (m x nc) theta is (nf x nc) and beta is (nc x 1)
    THETA = tf.Variable(tf.random.normal(shape=(self.num_features, self.num_classes), dtype=tf.float64))
    BETA = tf.Variable(tf.zeros(shape=(1, self.num_classes), dtype=tf.float64))

    # initialize optimizer
    optimizer = Adam(learning_rate=self.learning_rate)

    # run training
    for epoch in range(self.epochs):
        with GradientTape() as tape:
            # calculate logits or the linear activation of the weights and input
            train_Z = self.linear(train_X, THETA, BETA)
            train_A = tf.nn.softmax(train_Z)

            val_Z = self.linear(val_X, THETA, BETA)
            val_A = tf.nn.softmax(val_Z)

            # calculate the softmax as well as the cross entropy for each logit
            # by passing the logits Z and the real Y output labels
            # pass A to cross entropy function
            train_cost = self.J_cross_entropy(train_A, train_Y) + self.regularizer(THETA)
            val_cost = self.J_cross_entropy(val_A, val_Y)

            train_acc, _ = self.accuracy(train_A, train_Y)
            val_acc, _ = self.accuracy(val_A, val_Y)
```

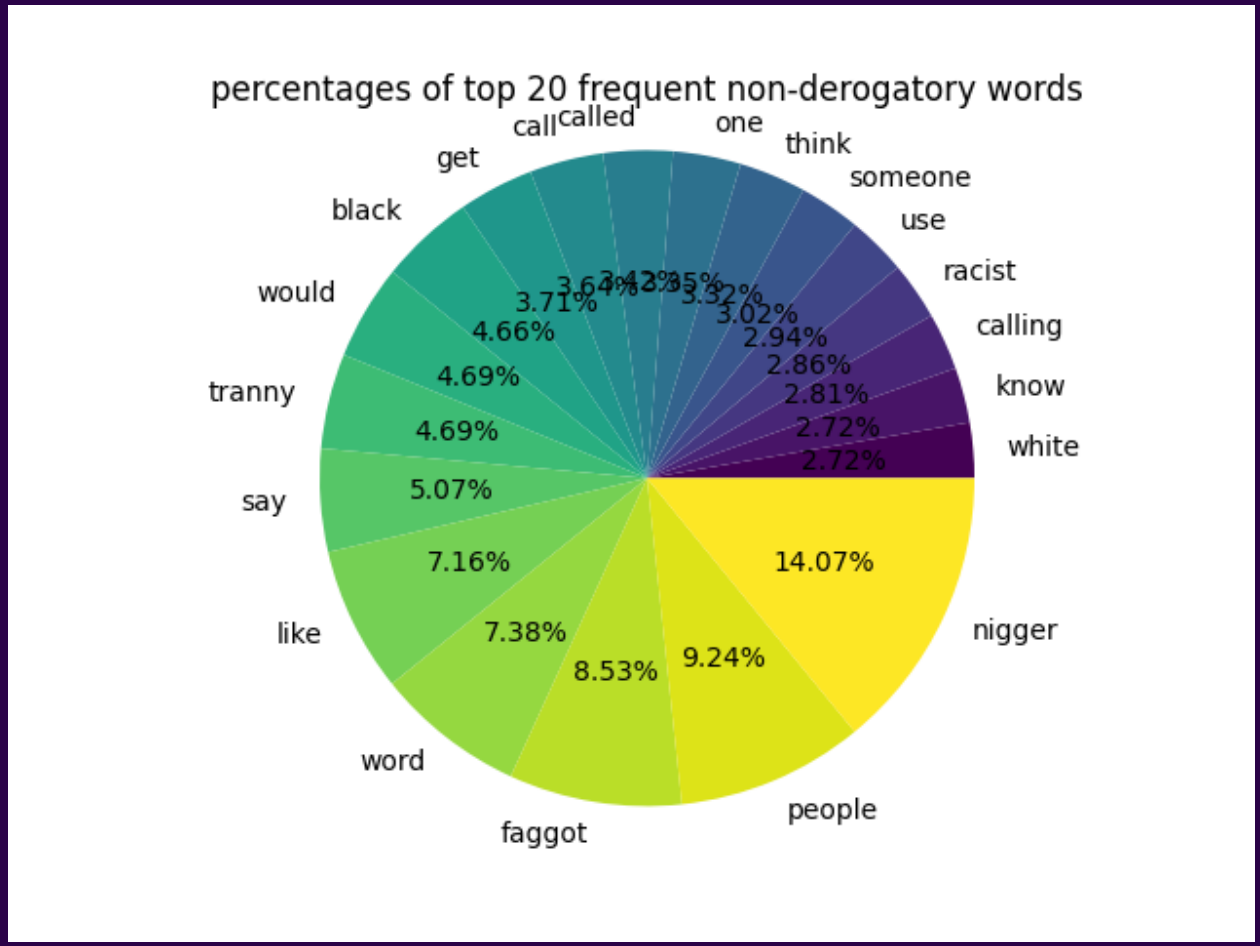
PERCENTAGE OF TOP 20 DEROGATORY WORD FREQUENCIES



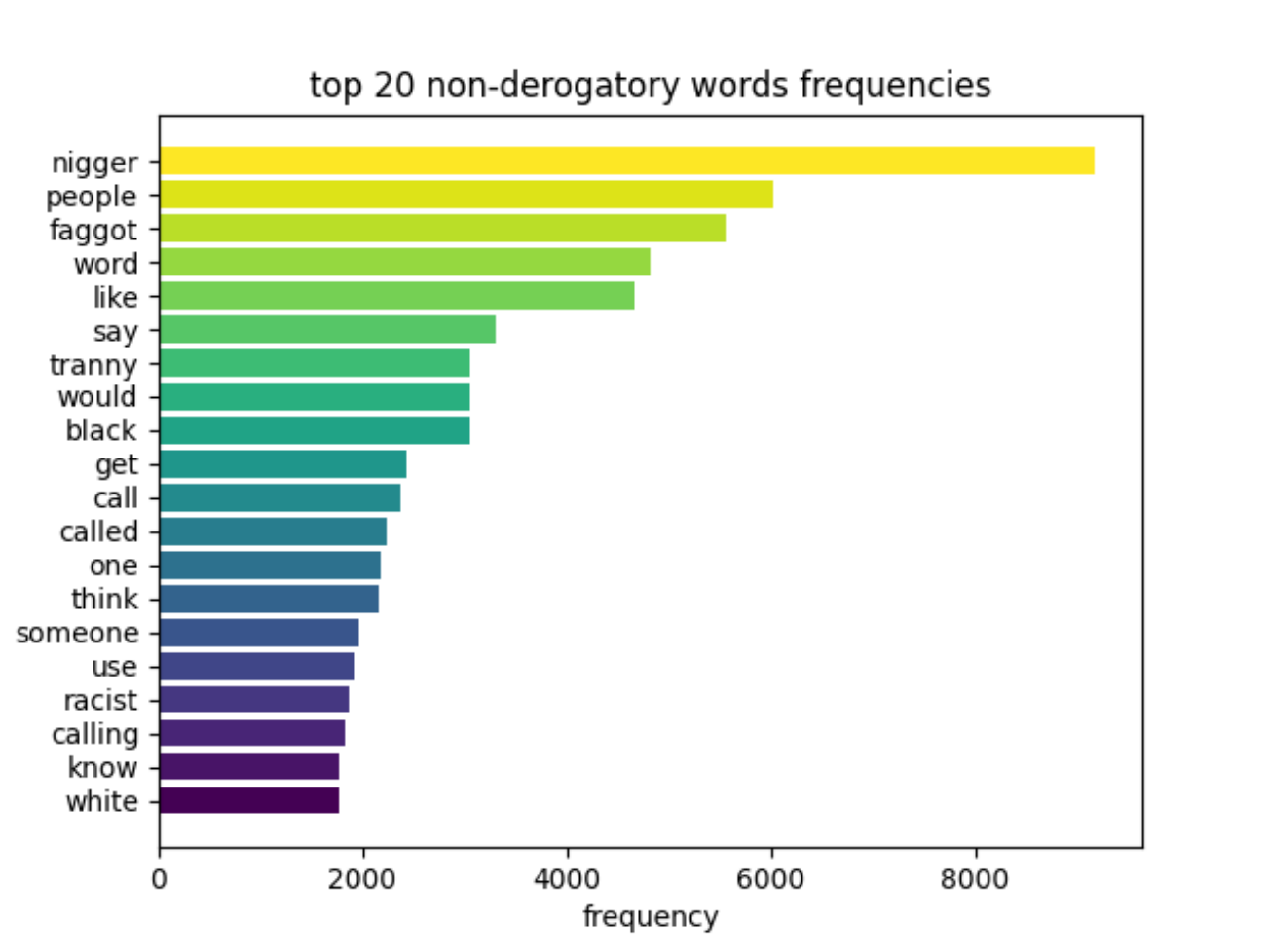
TOP 20 DEROGATORY WORD FREQUENCIES



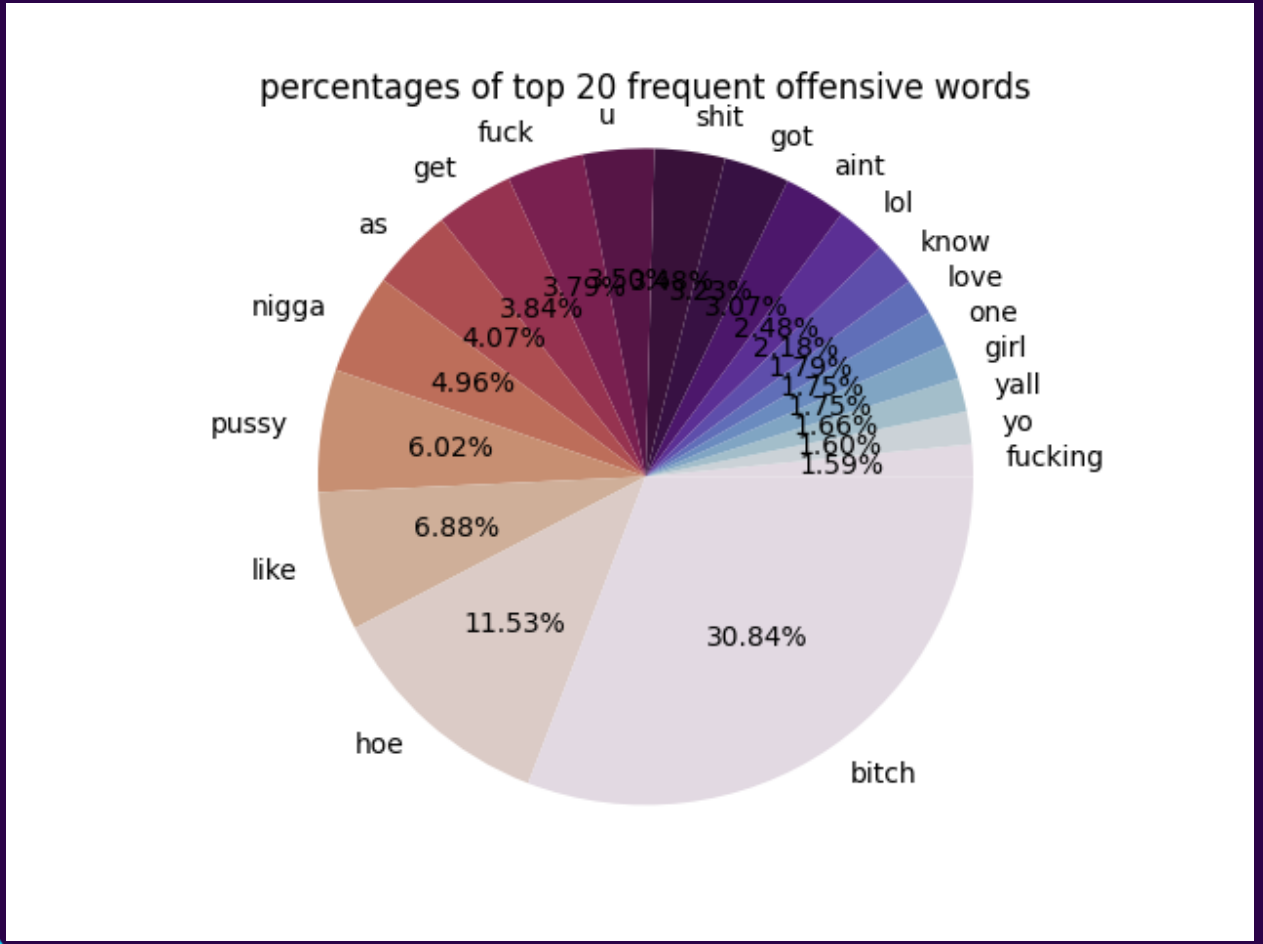
PERCENTAGE OF TOP 20 NON - DEROGATORY WORD FREQUENCIES



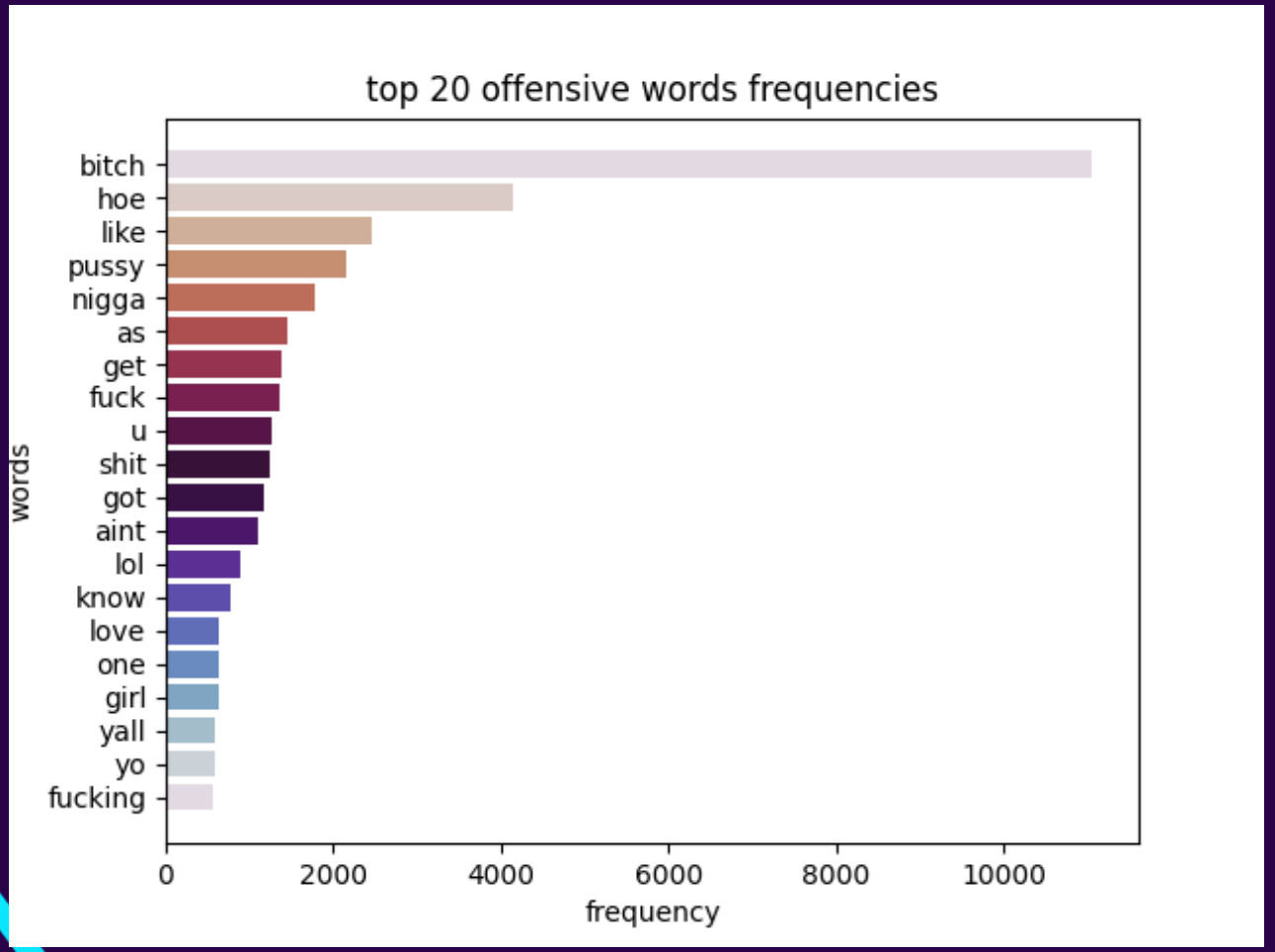
TOP 20 NON-DEROGATORY WORD FREQUENCIES



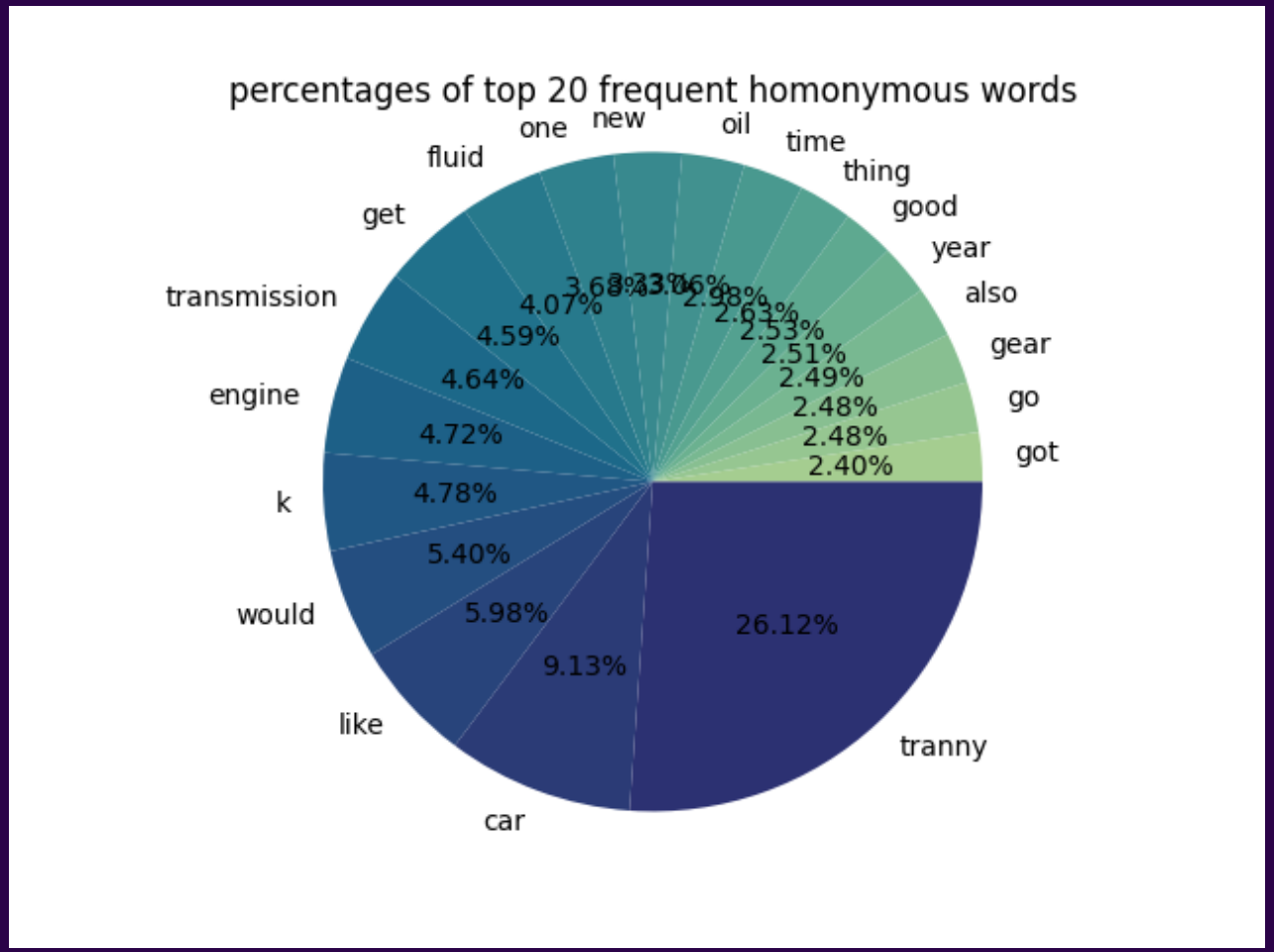
PERCENTAGE OF TOP 20 OFFENSIVE WORD FREQUENCIES



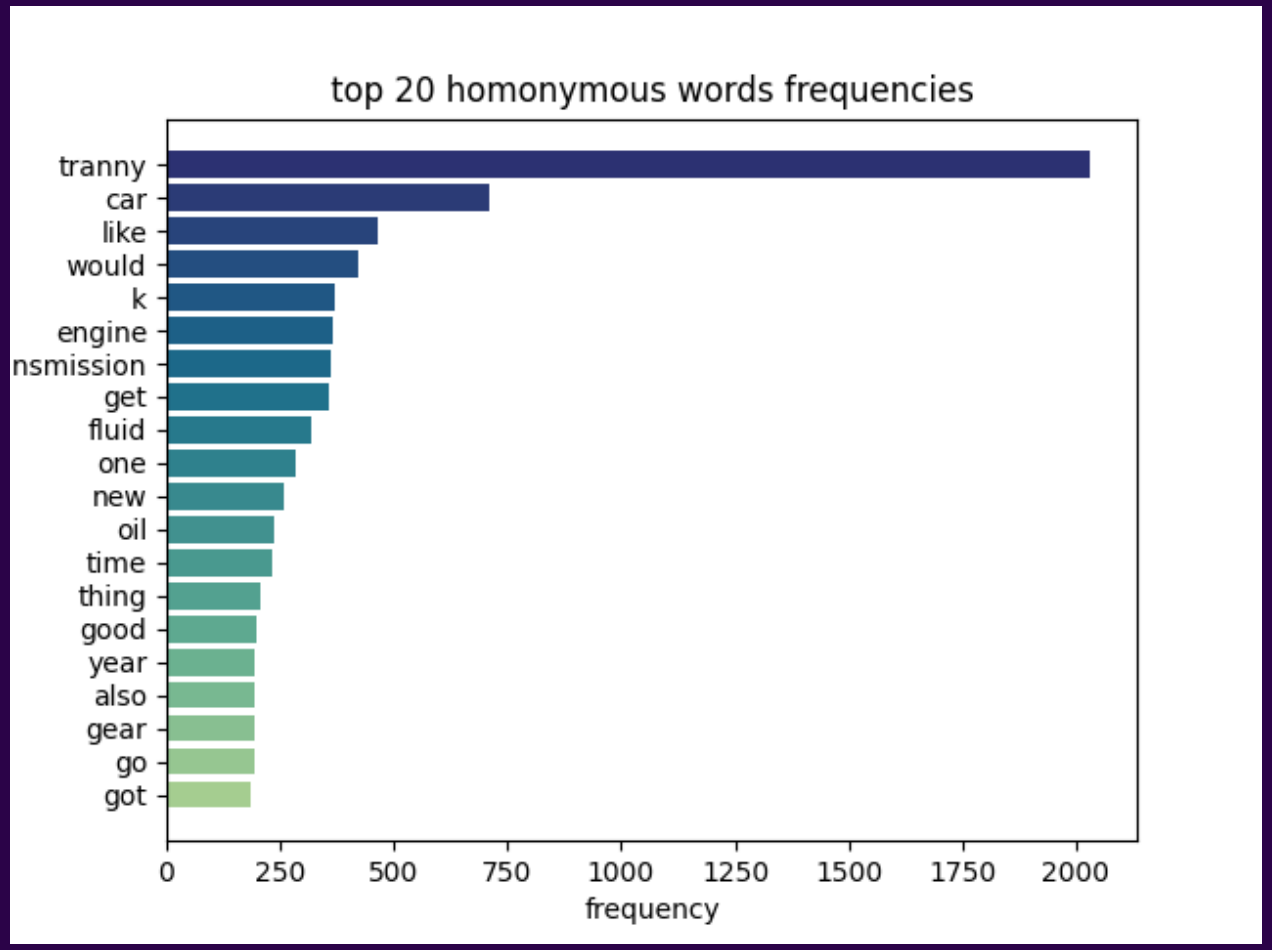
TOP 20 OFFENSIVE WORD FREQUENCIES



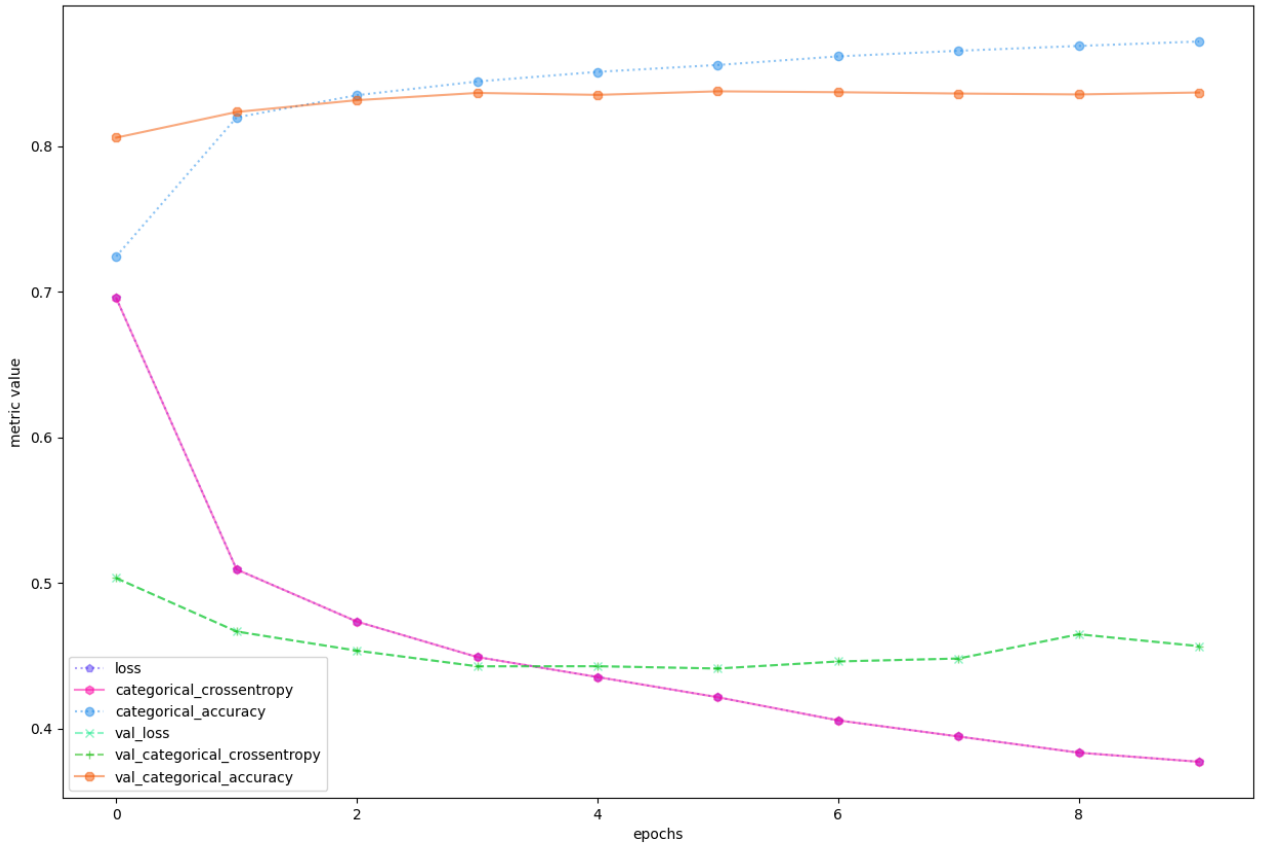
PERCENTAGE OF TOP 20 HOMONYMOUS WORD FREQUENCIES



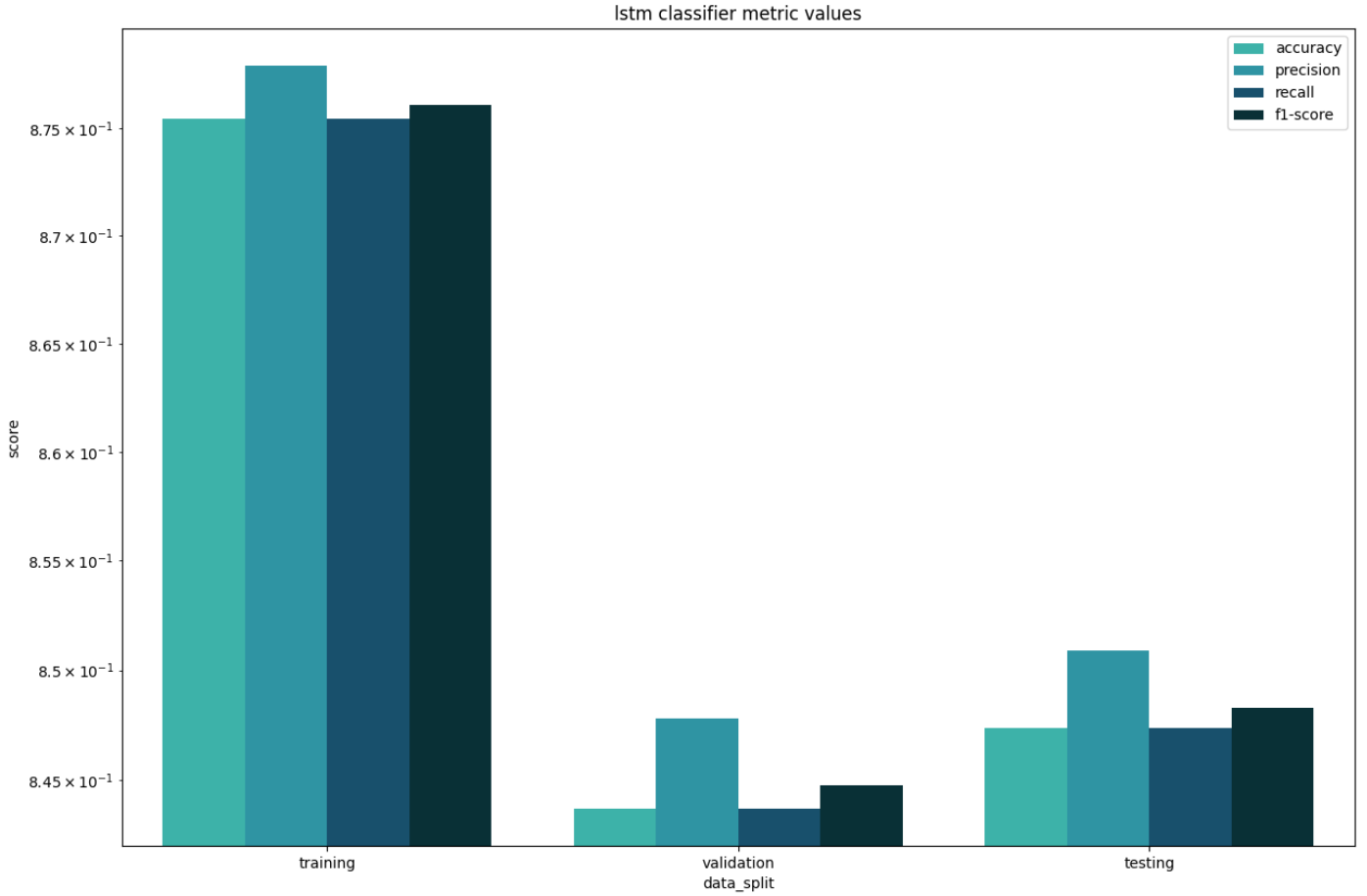
TOP 20 HOMONYMOUS WORD FREQUENCIES



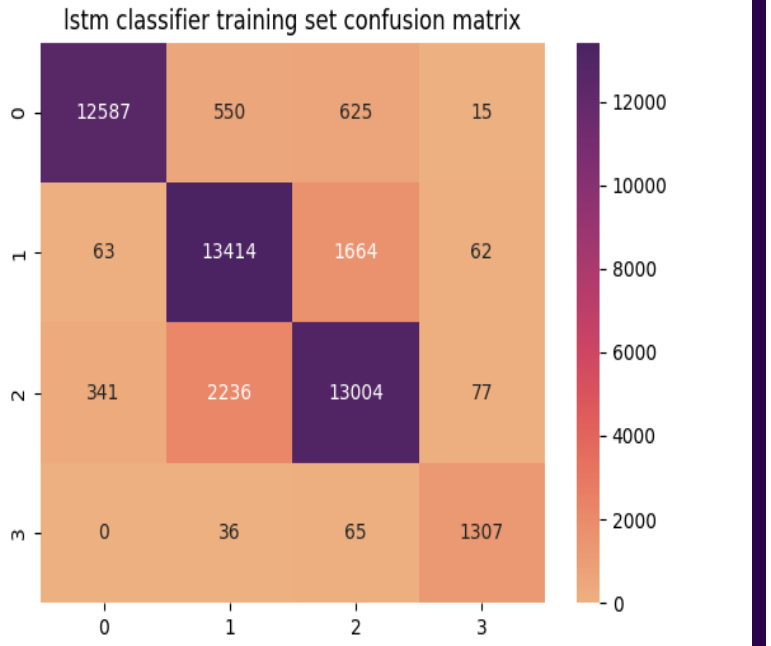
LSTM NEURAL NETWORK RESULTS

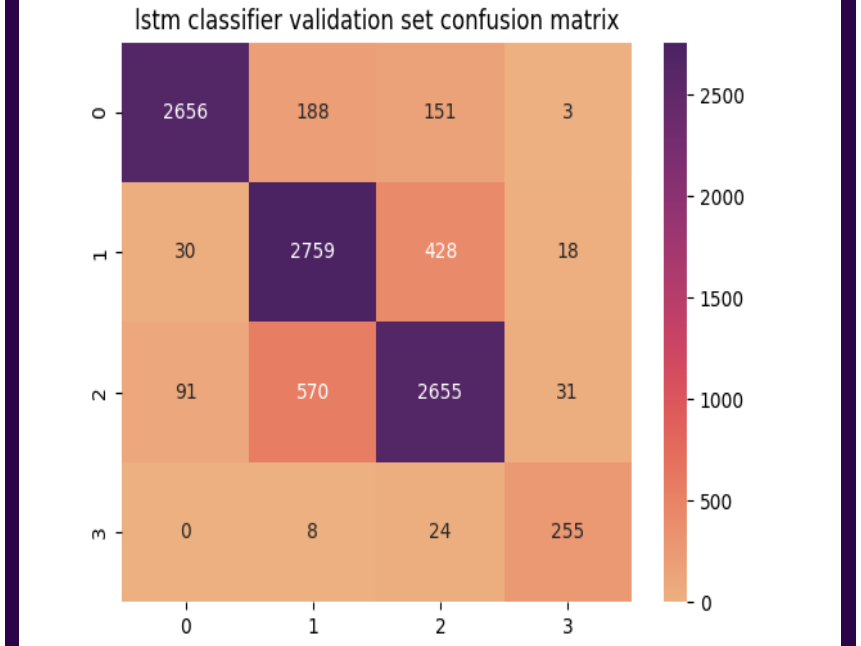


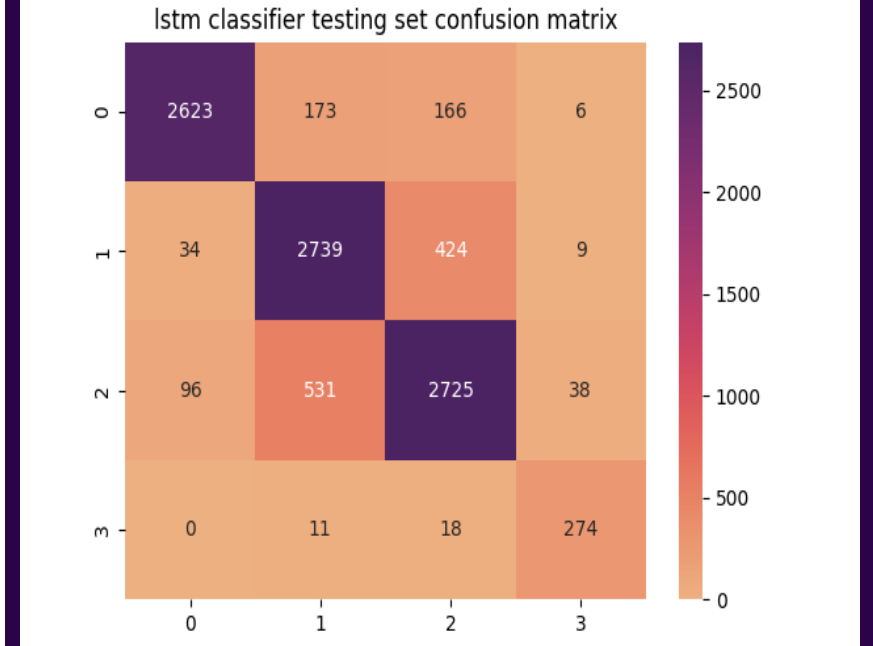
LSTM CLASSIFIER METRIC VALUES



LTSM Classifier Confusion Matrices





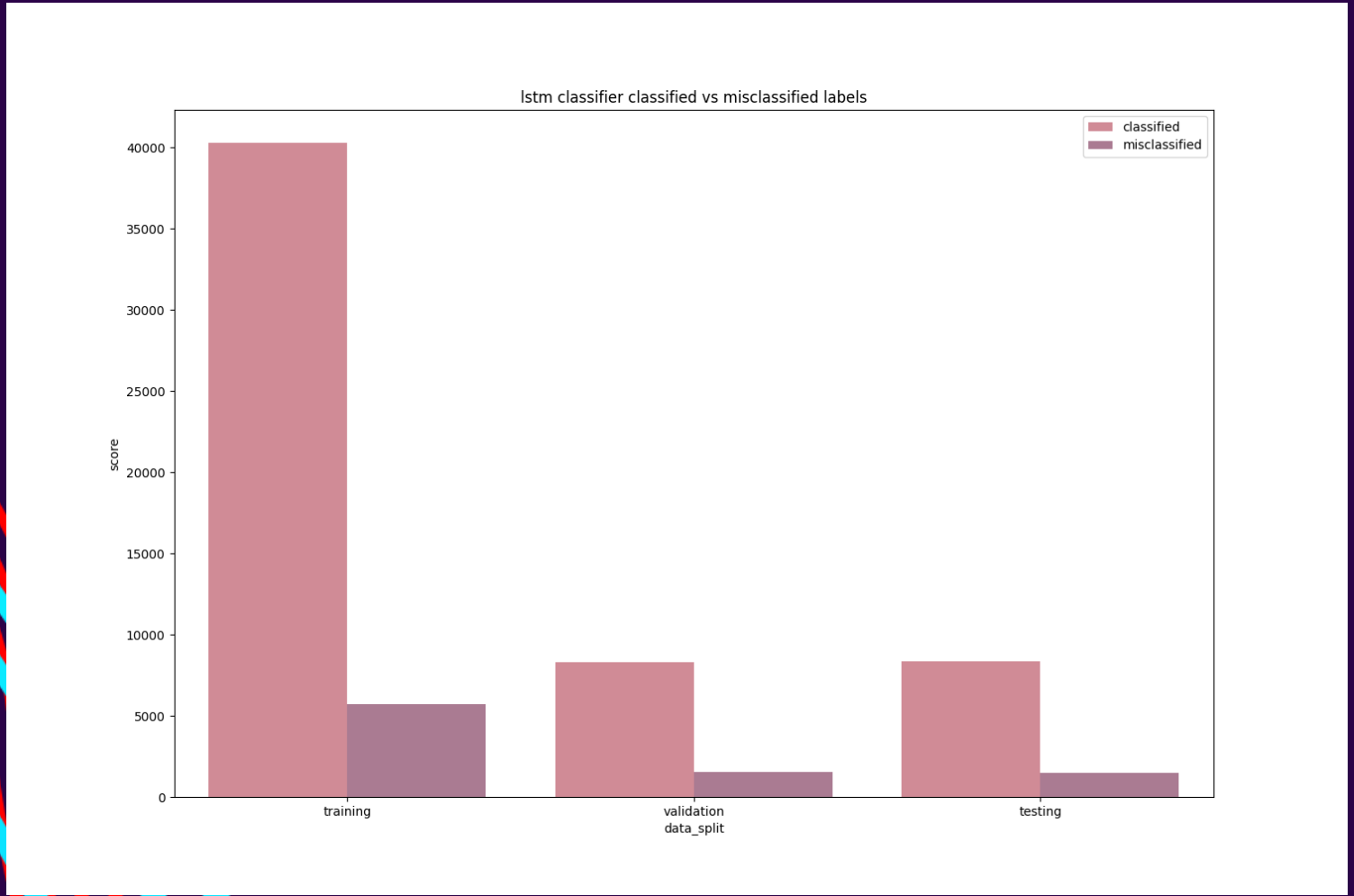


Training
40,312
5,734

Validation
8,325
1,542

Testing
8,361
1,506

LSTM CLASSIFIER CLASSIFIED VS MISCLASSIFIED LABELS



Training

40,312	
5,734	

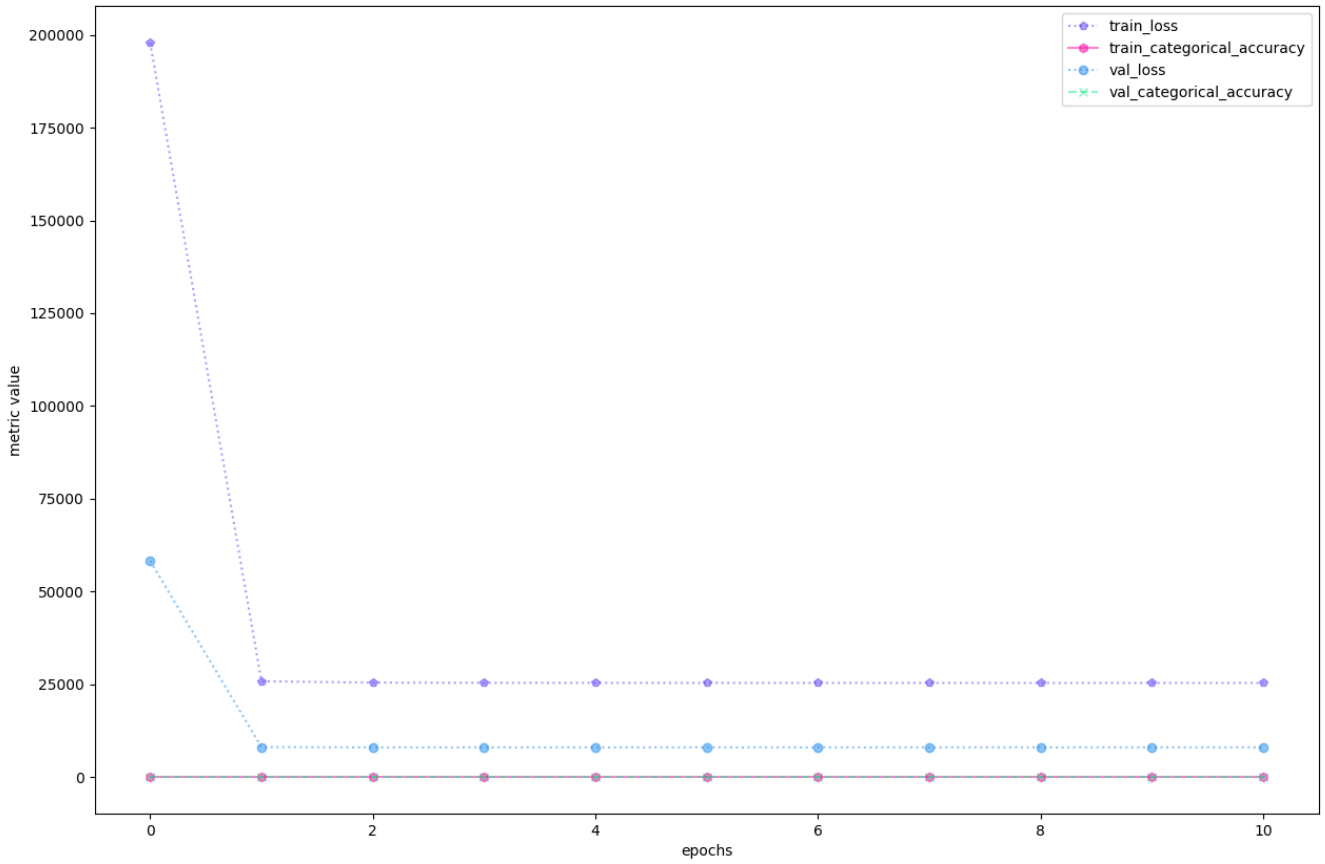
Validation

8,325	
1,542	

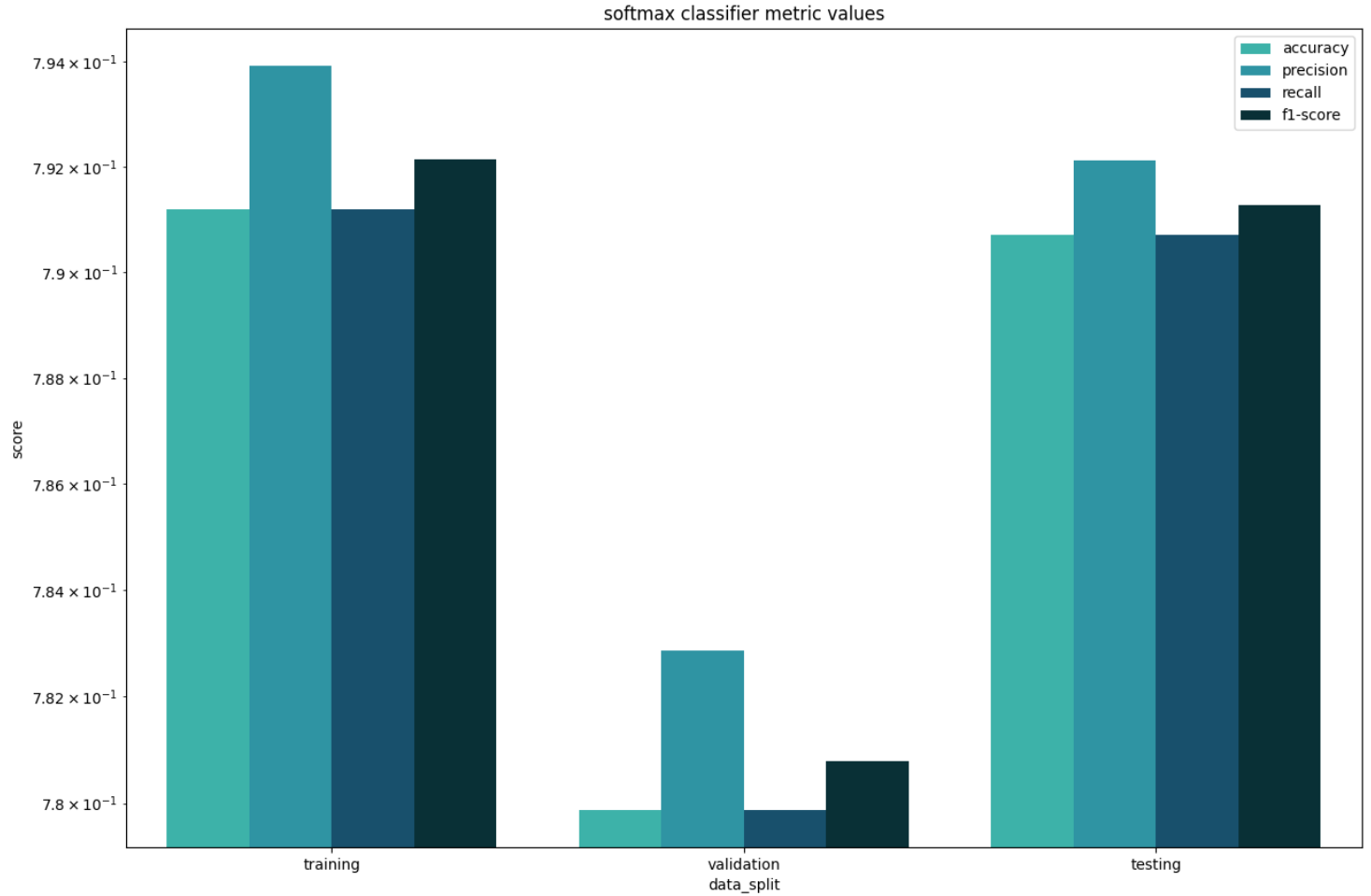
Testing

8,361	
1,506	

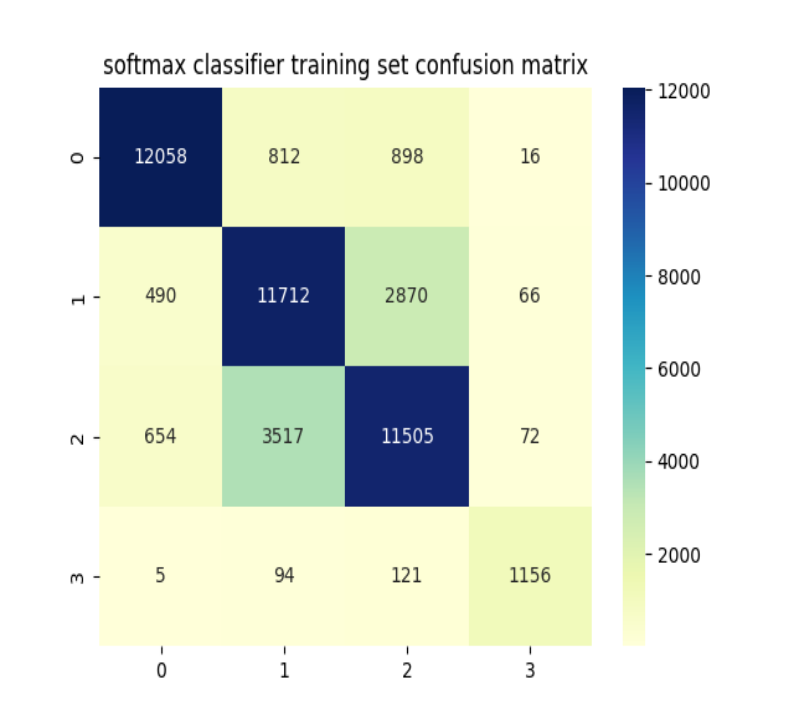
SOFTMAX REGRESSION RESULTS

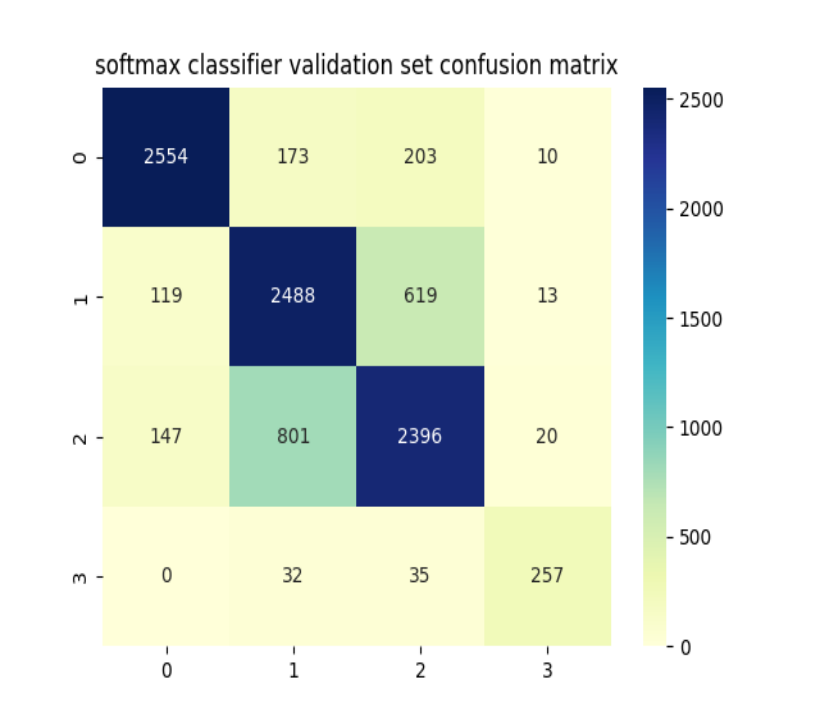


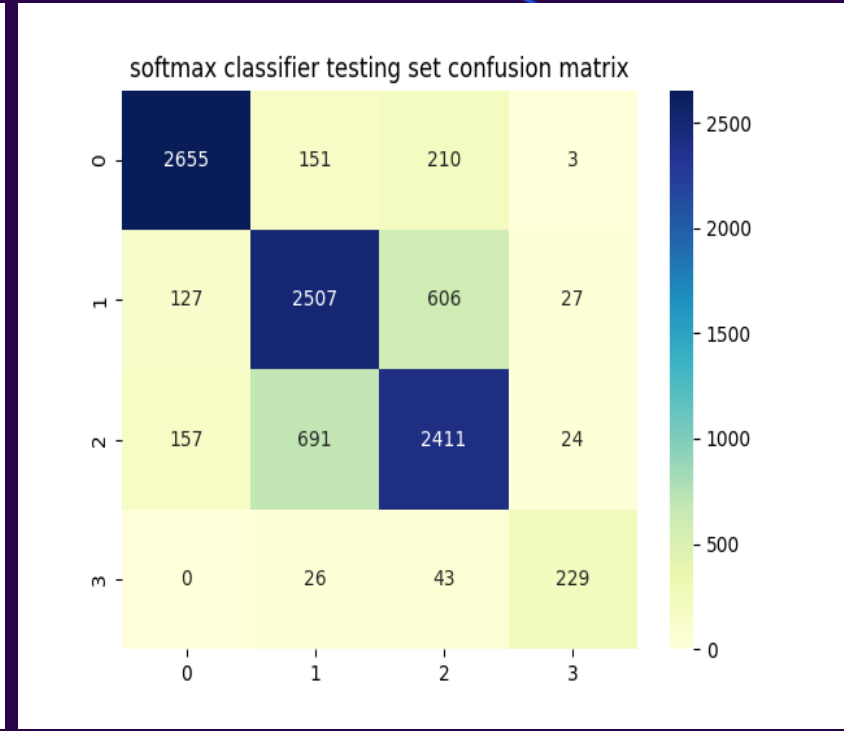
SOFTMAX REGRESSION RESULTS



Softmax Classifier Confusion Matrices

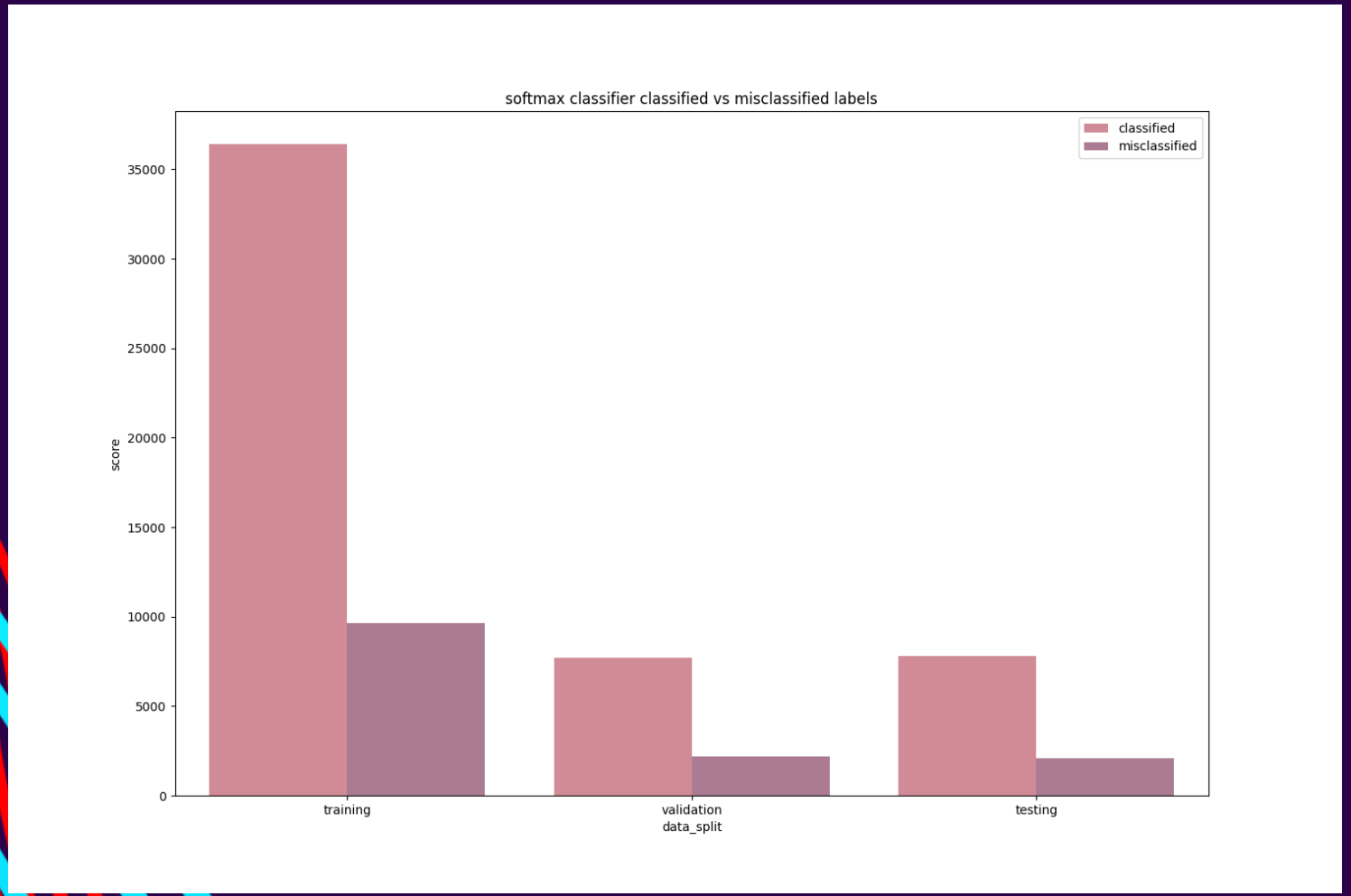






Training	Validation	Testing
36,431	7,695	7,802
9,615	2,172	2,065

SOFTMAX CLASSIFIER CLASSIFIED VS MISCLASSIFIED LABELS



Training

36,431
9,615

Validation

7,695
2,172

Testing

7,802
2,065

SUMMARY OF FINDINGS

- The created program has been able to identify into four categories: **derogatory**, **non-derogatory**, **offensive**, and **homonymous** words. The program was also able to determine the number of words and the percentage of each word.

Derogatory	
faggot	16.45%
nigger	13.01%
tranny	12.23%
fucking	4.02%
fuck	3.83%

Non-Derogatory	
nigger	14.07%
people	9.24%
faggot	8.53%
word	7.38%
like	7.16%

Offensive	
bitch	30.84%
hoe	11.53%
pussy	6.02%
nigga	4.96%
fuck	3.79%

Homonymous	
tranny	26.12%
car	9.13%
like	5.98%
would	5.40%
k	4.78%

SUMMARY OF FINDINGS

LSTM Classifier Performance metrics

	Training	Validation	Testing
Accuracy	87.55%	84.37%	84.74%
Precision	87.80%	84.78%	85.09%
Recall	87.55%	84.37%	84.74%
F1 Score	87.61%	84.48%	84.83%

Softmax Classifier Performance metrics

	Training	Validation	Testing
Accuracy	79.12%	77.99%	79.07%
Precision	79.39%	78.29%	79.21%
Recall	79.12%	77.99%	79.07%
F1 Score	79.21%	78.08%	79.13%

SUMMARY OF FINDINGS

- The LSTM Classifier performed better in classifying hate speeches with an f-score of 87.61% for the training set, 84.48% for the validation set and 84.83% for the testing set. It accurately classified labels 40,312 times during training, 8325 times during validation, and 8361 times during testing. It also misclassified labels 5734 times during training, 1542 times during validation, and 1506 times during testing.

- The Softmax Classifier performed worse with an f-score of 79.21% for the training set, 78.08% for the validation set, and 79.13% for the testing set. It accurately classified labels 36431 times during training, 7695 times during validation, and 7802 times during testing. It also misclassified labels 9615 times during training, 2172 times during validation, and 2065 times during testing.

CONCLUSION

- Based on the gathered data from the experiment, it shows that LSTM has performed better in classifying hate speeches compared to the Softmax Regression approach. This shows that there is a noticeable difference between the results of the two classification models.
- Full code is publicly available on repository: https://github.com/08Aristodemus24/hate-speech-classifier.git

THANK YOU