ÖZYEĞİN UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

CS 449

2020 Fall

PROJECT REPORT

Hate Speech Detection

By

Cem Denizsel

Mustafa Alper Sayan

Abstract

Hate speech detection is a hot topic in the field of Natural language processing. In this project we aim to solve this problem by implementing different approaches to experience the validity and feasibility of different methods. To achieve this goal we implemented basic machine learning algorithms such as perceptron, Support vector machines, decision trees and random forests. We also used more advanced techniques such as Recurrent neural networks, transformer networks and long short term memory systems to better predict the results. Out of these options, the Bidirectional LSTM model achieved the best results. We used SemEval 2019 Task 5 - Shared Task on Multilingual Detection of Hate dataset which consists of 9000 instances with binary labels. We prepared our own dataset which consists of 1000 turkish tweets in a similar manner. And achieved similar results to the original english dataset.

I. Introduction

Hate speech is a growing problem in social media platforms such as twitter, facebook, instagram. There is an increasing number of people using these platforms and thereby increased activity. This makes moderating these platforms increasingly difficult. Because of these reasons researchers are implementing different approaches to tackle this problem. Numerous competitions have been arranged to find optimal solutions using the field of Natural Language processing. In this project we used the dataset given by SemEval. SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems; it evolved from the Senseval word sense evaluation series. The evaluations are intended to explore the nature of meaning in language. While meaning is intuitive to humans, transferring those intuitions to computational analysis has proved elusive [1]. We prepared our solution according to the task-5 hate speech subtask A. The problem described formally as a binary classification where models have to predict a tweet in English with a given target is hateful or not. The dataset provided consists of 9000 tweets and 3 labels. We also prepared our own dataset in turkish language consisting of 1000 tweets since this type of dataset was not available open source. We are concerned about the label hate speech and prepare our models accordingly. First we tried basic machine learning algorithms to set a baseline for our more complicated models; tried methods were SVM`s, perceptron, decision tree and random forest algorithms. This method on average performed close to random which is at %50 accuracy. Our best performing model is a type of RNN(Recurrent Neural Network) which is known to work best with sequential data and fact is proven by our results as well.

II. Background

We mainly used two libraries to help us in our project. Sklearn and Tensorflow. Scikit-learn is a free software machine learning library for the Python programming language.[2] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. We used sklearns utility methods such as train_test_split, accuracy_score, f1_score, precision, grid_search. We used their model libraries : logistic regression, SVM, decision tree, random forest.

From tensorflow we used Keras API which is the easiest way to prepare neural networks. [3] TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015. We did not use Tensorflow by its own rather we have used API build on top of it Keras which is much more easier. [4] Keras is an open-source software library that provides a Python interface for artificial neural networks.

Keras acts as an interface for the TensorFlow library. From Keras we used numerous layers: Dense, Flatten, LSTM, Dropout, Activation, Embedding, Bidirectional. A dense layer is a fully connected layer from the previous layer. Flatten layer flattens the input. Dropout layer is used to combat overfitting. LSTM is a complicated layer and will be explained in the following sections. Embedding layer is used to convert integer arrays into dense vectors. Bidirectional layer will be explained in the next sections as well. From Keras preprocessing we used tokenizer and pad sequences. Tokenizer vectorizes the text inputs. Padding makes our corpus have the same dimensions as the previously set size so our data is ready for training.

We also used NLTK.[5] NLTK is The Natural Language Toolkit, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language. NLTK is used to preprocess our data. We used it to clear stopwords both in English and in Turkish.

III. Methodology

1. SVM

Support vector machines are supervised models that analyze classification and regression. Given a set of training samples, each marked as belonging to one of two categories. SVM aims to maximize the distance between two classes. If linear classification is not performed, SVM maps the inputs into high dimensional spaces.

For computing SVM classifier we need to minimize the loss function:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(\mathbf{w}^T\mathbf{x}_i-b)\right)\right]+\lambda\|\mathbf{w}\|^2$$

Where n equals to number of samples, y equals to output, w equals weight, b equals intercept, lambda equals alpha value. For each i ϵ {1, 2, 3 , ..., n}, we introduce a new variable c_i=max(0,1-y_i (w^T x_i-b)). Then The equation above can be written as:

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i+\lambda\|\mathbf{w}\|^2$$
$$\text{subject to } y_i(\mathbf{w}^T\mathbf{x}_i-b)\geq 1-\zeta_i \text{ and } \zeta_i\geq 0, \text{ for all } i.$$

Solving equation above gives the figure below:

$$\text{maximize } f(c_1\ldots c_n)=\sum_{i=1}^{n}c_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}y_i c_i(\mathbf{x}_i^T\mathbf{x}_j)y_j c_j,$$
$$\text{subject to } \sum_{i=1}^{n}c_i y_i=0, \text{ and } 0\leq c_i\leq\frac{1}{2n\lambda} \text{ for all } i.$$

Now we need to learn a classification rule for the transformed data points ϕ((x_i)). Furthermore, we should satisfy the following kernel function k(x_i,x_j)= ϕ(x_i)* ϕ(x_j) . W in the transformed space satisfies:

$$\mathbf{w} = \sum_{i=1}^{n} c_i y_i \varphi(\mathbf{x}_i)$$

Then we must solve the following optimization problems:

$$\begin{aligned}\text{maximize } f(c_1 \dots c_n) &= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) y_j c_j \\ &= \sum_{i=1}^{n} c_i \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i k(\mathbf{x}_i, \mathbf{x}_j) y_j c_j \\ \text{subject to } &\sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \le c_i \le \frac{1}{2n\lambda} \text{ for all } i.\end{aligned}$$

$$\begin{aligned}b = \mathbf{w}^T \varphi(\mathbf{x}_i) \quad y_i &= \left[\sum_{j=1}^{n} c_j y_j \varphi(\mathbf{x}_j) \cdot \varphi(\mathbf{x}_i)\right] \quad y_i \\ &= \left[\sum_{j=1}^{n} c_j y_j k(\mathbf{x}_j, \mathbf{x}_i)\right] \quad y_i.\end{aligned}$$

$$\mathbf{z} \mapsto \operatorname{sgn}(\mathbf{w}^T \varphi(\mathbf{z}) - b) = \operatorname{sgn}\left(\left[\sum_{i=1}^{n} c_i y_i k(\mathbf{x}_i, \mathbf{z})\right] - b\right).$$

SVM hyper parameters

- C: Regularization parameter. Regularization is Inversely proportional with c value
- Kernel linear: degree of the kernel function is 1
- Kernel poly: Kernel is a polynomial function
- Kernel sigmoid: Linear classifier
- Kernel rbf: Radial basis function kernel
- Degree: Degree of the polynomial function
- Gamma scale: 1 / (n_features * X.var()) for gamma value
- Gamma auto: 1 / n_features

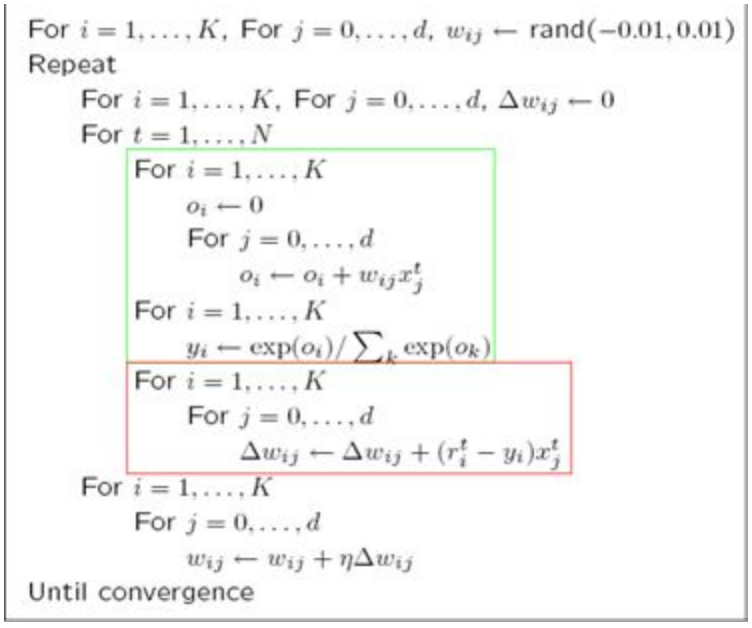
2. Perceptron

The perceptron is a type of supervised learning. It decides whether an input belongs to the class by using weight coefficients. It is a parametric method. We can then define an activation function ϕ(z) that takes a linear combination of certain input values x and a corresponding weight vector w, where z is the so-called net input (z = w1 x +...+ wmx)

Now, if the activation of a particular sample $x^{(i)}$, that is, the output of $\phi(z)$, is greater than a defined threshold θ, we predict class 1 and class -1, otherwise, in the perceptron algorithm, the activation function $\phi(\cdot)$ is a simple *unit step function*, which is sometimes also called the *Heaviside step function*:

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ -1 & \text{otherwise} \end{cases}$$

Perceptron Pseudocode



Where the green part is forward propagation and red part is backpropagation. Pseudocode shows the number of classes > 2. Y in the pseudocode equals sigmoid function. K equals to number of classes, j equals to number of dimensions, N equals to number of sam

Perceptron Hyper Parameters

- Penalty L2: L2 regularization is called ridge regression. Formula is as below:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2$$

  ples, w equals to weight coefficients.

- Penalty L1: L1 regularization is called lasso regularization. Formula is below:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} |w_j|$$

- Where y equals to actual output, M equals to number of samples, P equals to number of dimensions, w equals to weight coefficients, x equals to input, lambda equals to regularization term.
- Penalty elastic net: used for mixing l1 and l2 regularization with different importance
- Penalty None: No regularization should be used
- Alpha: Regularization term. It can be seen in the example formula L1 above.
- L1 ratio: Determines the importance of l1, only used when regularization term is elastic net
- Fit intercept: Used to estimate the intercept
- Max iter: Determines the number of epochs
- Tol: The spotting criterion

3. Decision Tree

Decision tree algorithm is a predictive model used in machine learning. It uses a decision tree to go from observations about a sample to predict a samples output value. There are two main type of decision trees namely decision tree classifier and decision tree regression. In this assignment we used a decision tree classifier. This type of tree is split using the following method: training examples are broken down into smaller subsets and using this method the tree is built. The aim of this method is to partition the data into sparse or dense clusters. Decision trees are built using divide and conquer algorithm to be more specific it is called recursive partitioning.

Steps of decision tree algorithm

- Select a test for the root node. Create branch for each possible outcome of the test
- Split instances into subsets
- Repeat recursively for each branch
- Stop when all the leaf nodes are pure. Or specified depth is reached

Advantages

- Time complexity is O(Nkd) which is better than KNN and svm also random forests where N is number of training samples, k is number of features, d is the depth of the decision tree
- Interpretability
- Excludes unimportant features

Disadvantages

- If max depth is not specified or not set appropriately the tree might over fit training instances.
- They are biased to split on features having large levels
- Small changes in the dataset can result in different trees.
- Large trees are not easy to interpret

4. Random forest

Random forests are an ensemble method for classification, they are constructed using multiple decision trees to combat overfitting. To put more simply they are collections of decision trees with different methods to determine the correct class, one of them being a major vote.

5. Cross Entropy Loss

[6] Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .01 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0. This method penalizes confident predictions much more than unconfident ones. In our case of this project we used binary cross entropy loss since we have binary labels. This result is calculated as follows where y equals label 0 or 1, p equals prediction confidence value:

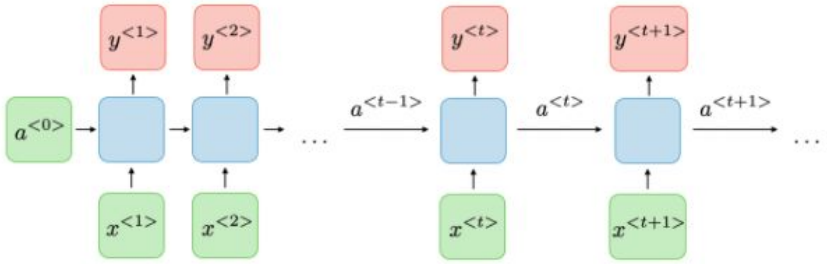
$$-(y\log(p)+(1-y)\log(1-p))$$

6. Adam optimizer

[7] The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Adam optimizer is easy to implement, computationally effective, little memory requirement and good with noisy data. This last part is especially important for our purposes since we do not have large amounts of data to work with and syntactic errors occur frequently in social media platforms.

7. [8] Recurrent Neural Network

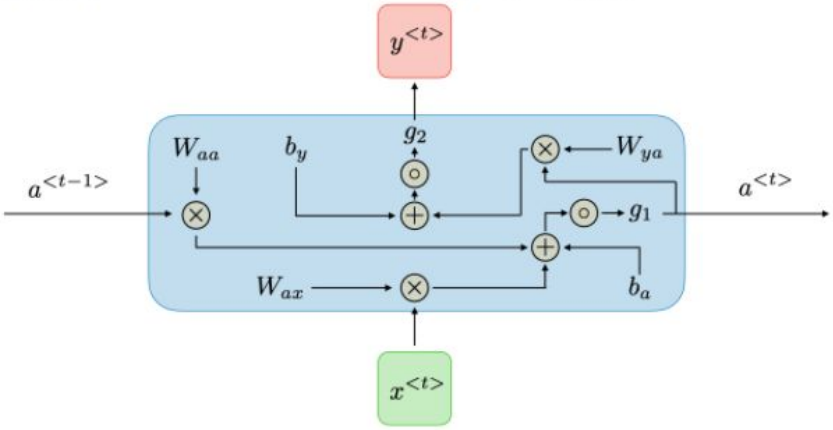
Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states. In our case we are doing a type of sentiment analysis therefore we are using many to one type of RNN. They are typically as follows:



For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

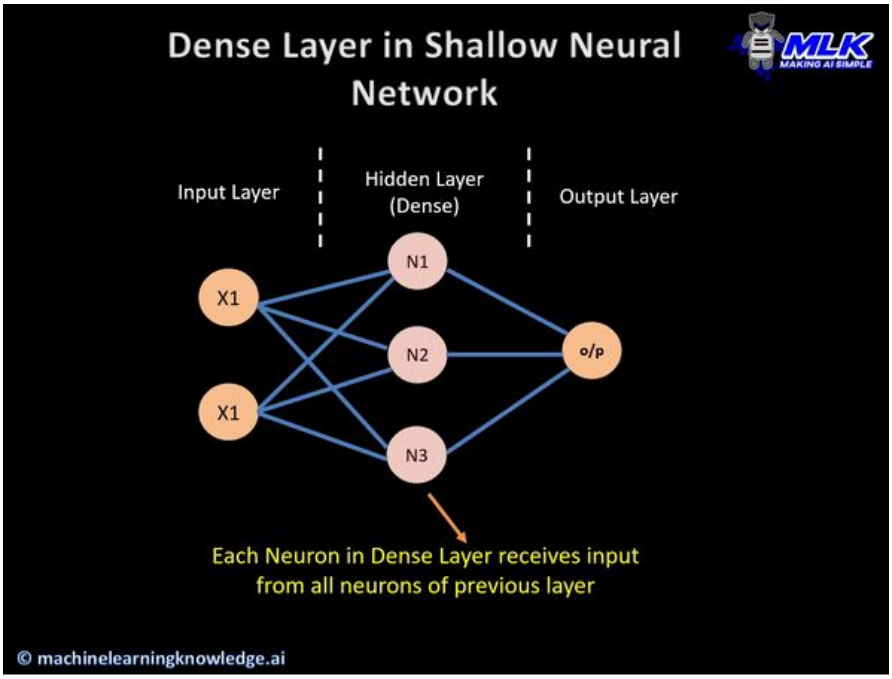
where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.



- Advantages
  - Inputs can be at any size
  - Model size does not increase with the size of inputs
  - Computation takes history into account
  - Weights are shared across time
- Disadvantages
  - Slow computation
  - Has short memory
  - Cannot consider future input when making calculations

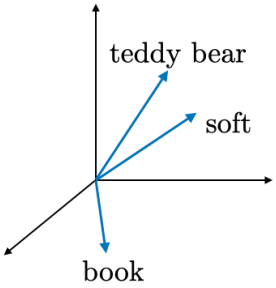
8. Dense Layer

[9] The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models. An illustration is as follows:



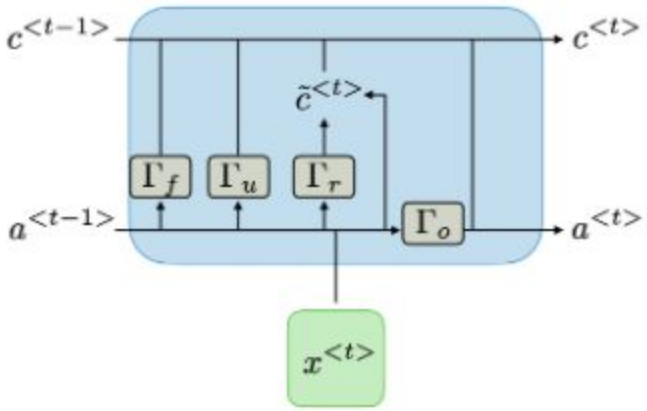
9. [8] Embedding Layer

Word2vec is a framework aimed at learning word embeddings by estimating the likelihood that a given word is surrounded by other words. Popular models include skip-gram, negative sampling and CBOW. An illustration is as follows:

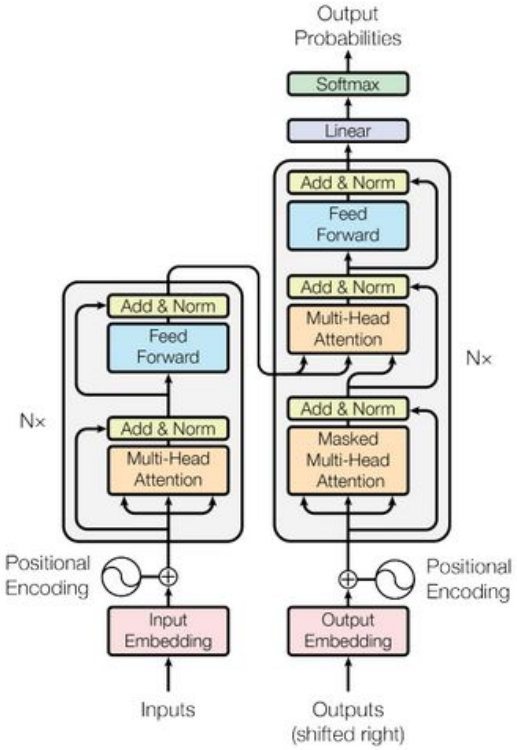


10. LSTM

LSTMs are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. All recurrent neural networks have the form of a chain of repeating modules of neural networks. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. The model structure can be seen below:



11. Transformer



[10]Like recurrent neural networks (RNNs), Transformers are designed to handle sequential data, such as natural language, for tasks such as translation and text summarization. The model structure can be seen on the left hand side.However, unlike RNNs, Transformers do not require that the sequential data be processed in order. For example, if the input data is a natural language sentence, the Transformer does not need to process the beginning of it before the end. Due to this feature, the Transformer allows for much more parallelization than RNNs and therefore reduced training times.Transformers have

rapidly become the model of choice for NLP problems, replacing older recurrent neural network models such as the long short-term memory (LSTM). Since the Transformer model facilitates more parallelization during training, it has enabled training on larger datasets than was possible before it was introduced. Although we do not have much data that needs to be paralized we still wanted to experiment with the performance of the model. Because in both datasets size is relatively small. The model did not perform as expected.

IV. Solution Approach

<u>All the results shown IV. are from English dataset. In section V. we also show the results we have obtained from our turkish dataset.</u>

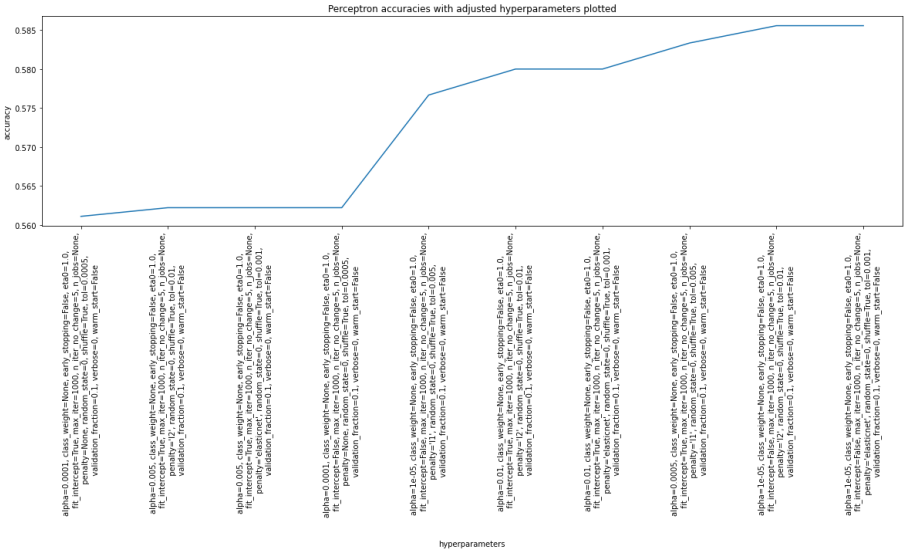
1. Basic Algorithms

a. SVM

We used svm to get an understanding of if there was a need to apply more advanced methods. The algorithm failed to converge on our tests. Therefore we do not know the model's performance.

b. Perceptron

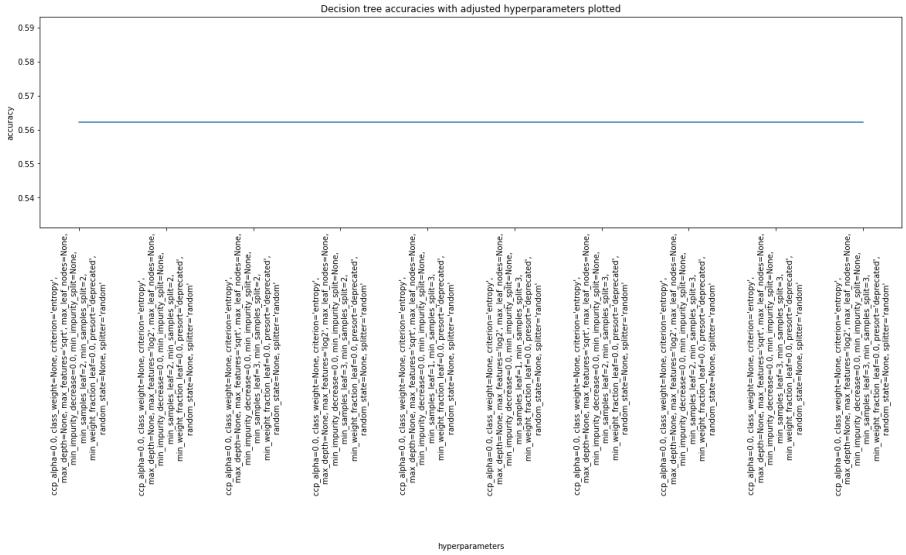
We tried approximately 100 perceptron algorithms to find the optimal algorithm in the plot below the top performing 10 algorithms can be seen:



As can be seen on the above plot we reached 58% accuracy which is slightly above random accuracy. Our algorithm definitely learned something but this algorithm is not suitable for classification hate speech in a feasible way. Therefore should not be used.

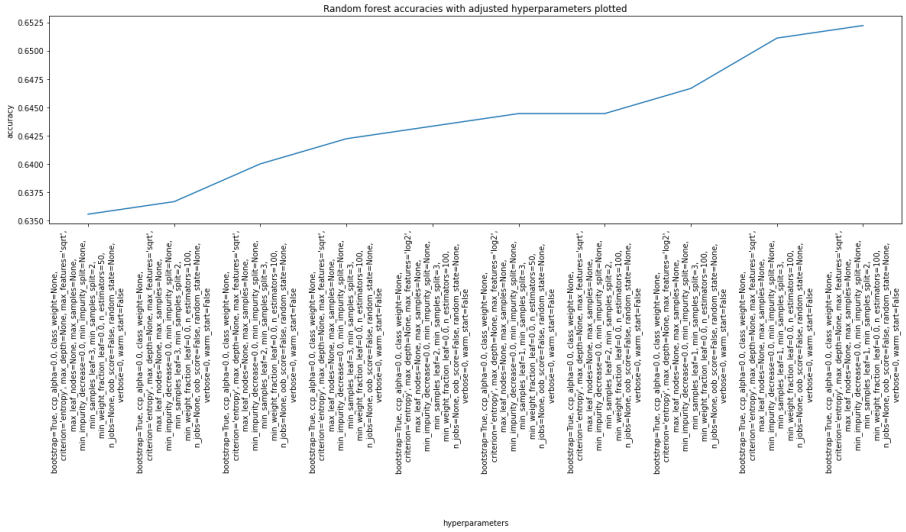
c. Decision Tree

This algorithm slightly outperforms perceptron algorithm but not by much. Since we did not get optimal results. The integrability of the decision tree was not taken into consideration. In the plot below the accuracies can be seen:



d. Random Forest

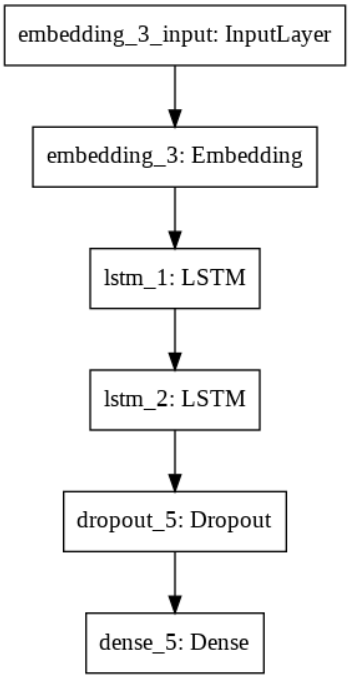
Random forest algorithm has given surprising results given the fact that it outperformed some of the more complex neural network models. In the table below the acquired results can be seen:



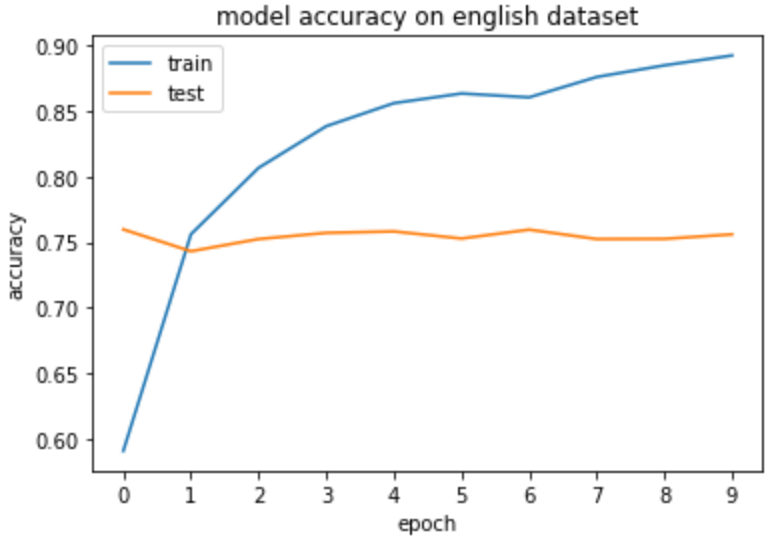
2. Deep neural networks

a. 2-LSTM model

We first started off by training our dataset on this model. The layers of our model can be seen below:

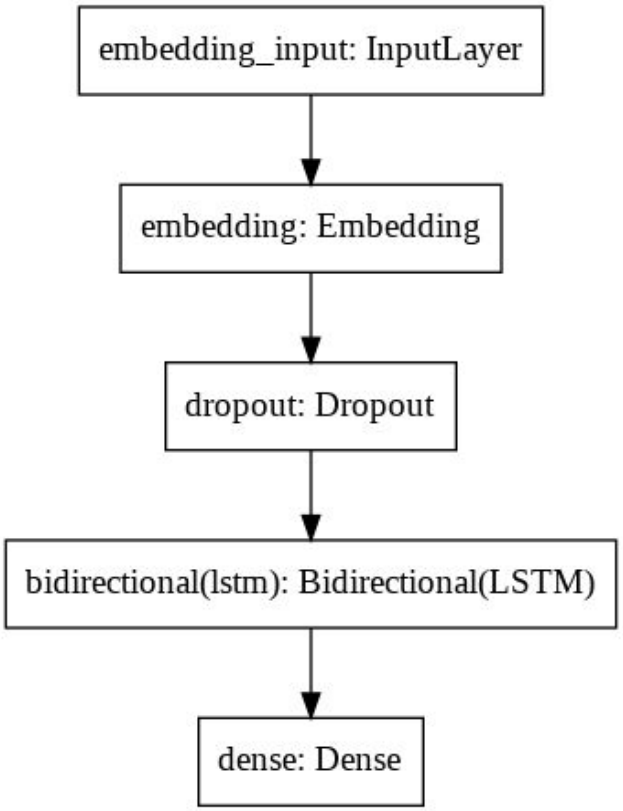


It has an embedding layer to convert arrays into vectors. It has 2 LSTM layers, we have started with one layer but seen that 2 layers give us better results. We use dropout since our dataset is relatively small.This model outperformed the basic machine learning algorithms by approximately %15. This was a significant improvement from the basic algorithms while maintaining a relatively simple deep learning model. The plot below shows accuracy with changing epochs:

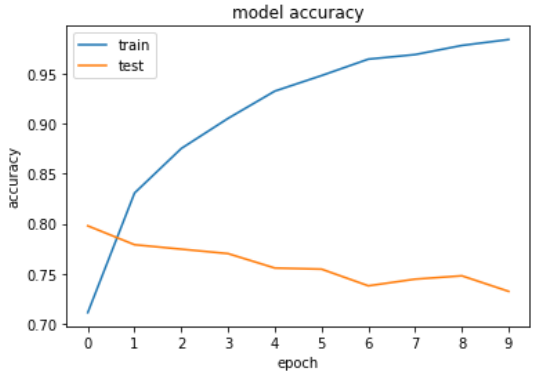


This model is consistently outperformed by other models tested with between 5-7%. This gives us the reasoning that going deeper with LSTM layers will not yield an optimal result.

b. 1-Bidirectional LSTM model

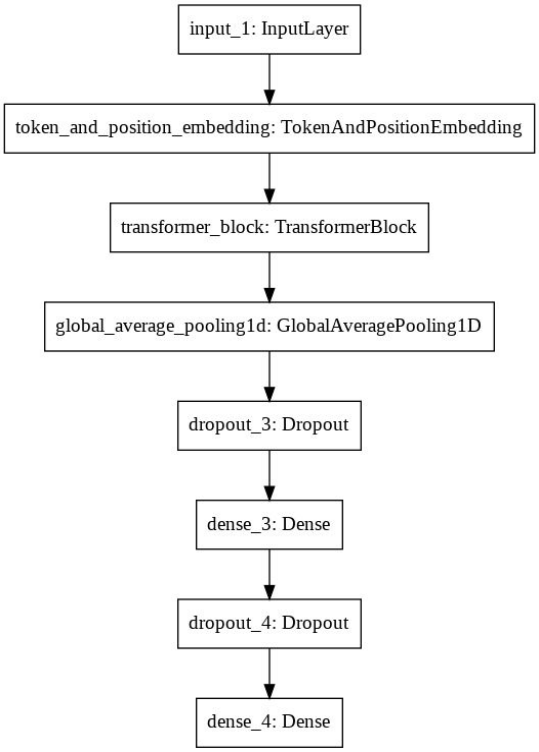


This model is similar to our last model with difference being the bidirectional LSTM layer. This model model performs better approximately %2-%3 than all the other models consistently. With bidirectional LSTM we are able to go back and forward in time and this affects our models weights. This seems to be an important method when dealing with text classification as well. We are using a sigmoid activation function in our model. The results we have obtained can be seen below:

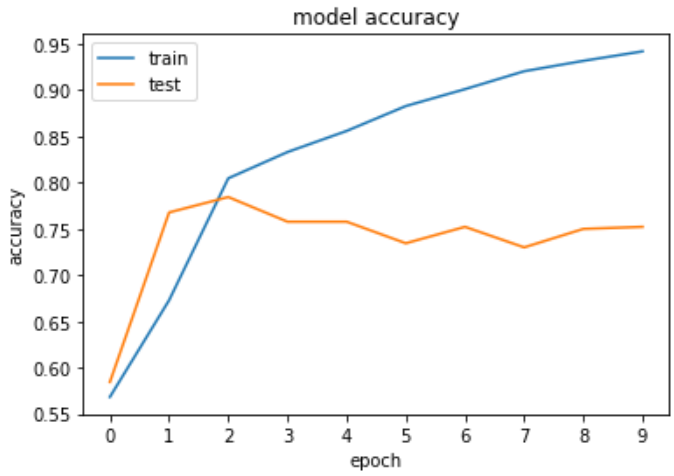


As we can see from the plot above it is very easy to overfit our model. After 2 epochs this can be seen as a result.

c. RNN Transformer model



We tried more complicated models in our solution approach as well. Notice that the plot on the left hand side does not cover all the layers in the model with detail. The transformer model performed similarly to Bi-directional LSTM model. Because of this reason we do not see it as necessary to much deeper in terms of layers. Rather we would need more clever ways combat overfitting since overfitting after 2 or 3 epochs is not ideal. The result we obtained from this model can be seen in the below plot:

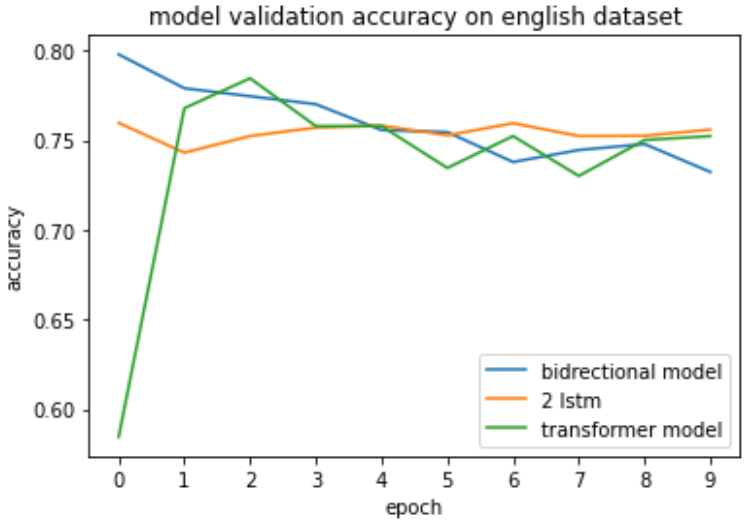


The plot above shows similar results to our two models. It overfits after 3 epochs. It has similar performance to Bidirectional LSTM model therefore we would recommend to use that version since it is a much more simple model with higher interpretability.

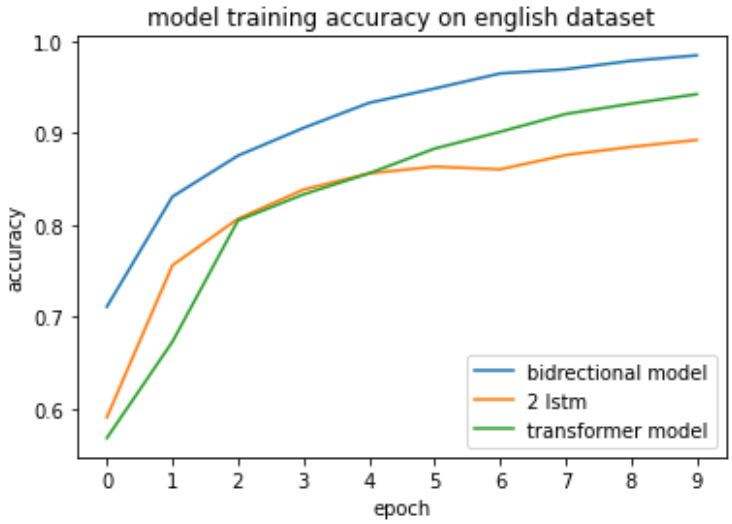
V. Results and Discussion

<u>Since deep learning methods performed better we did not train basic machine learning algorithms on turkish dataset</u>

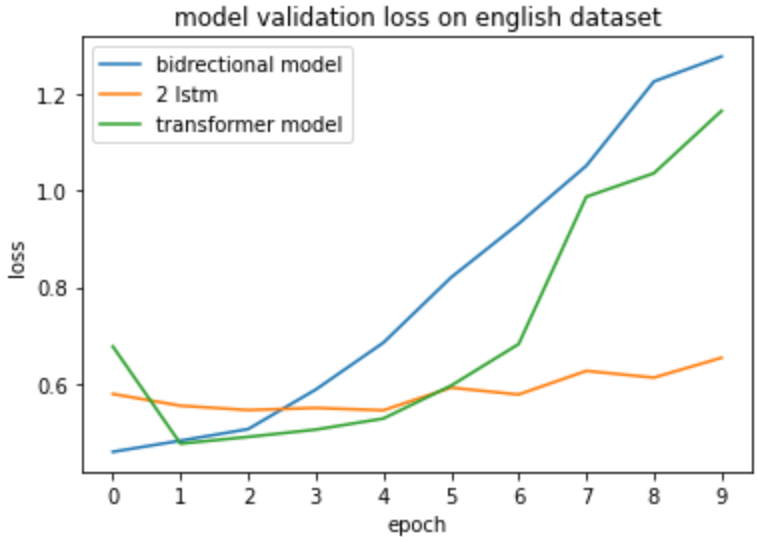
1. On the plot below we can see that our models are relatively similar but bidirectional LSTM model has the best performance



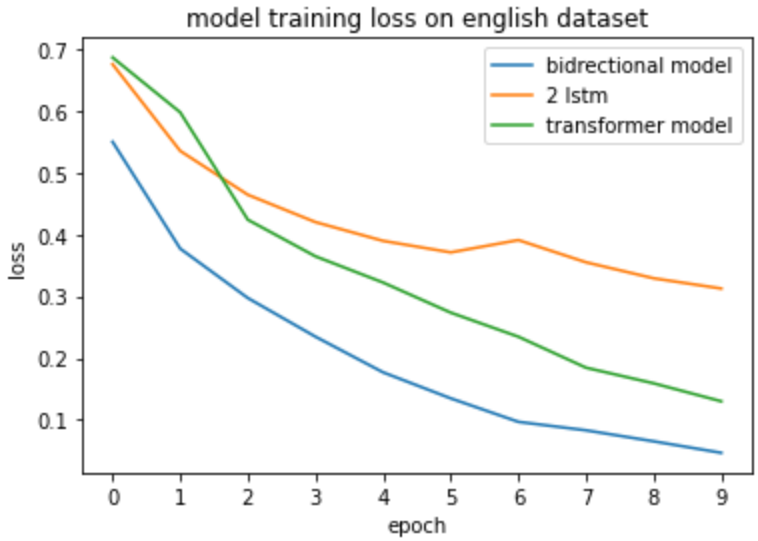
2. As can be seen from the plot below it can be seen that after 3 epochs we are overfitting and we should be careful



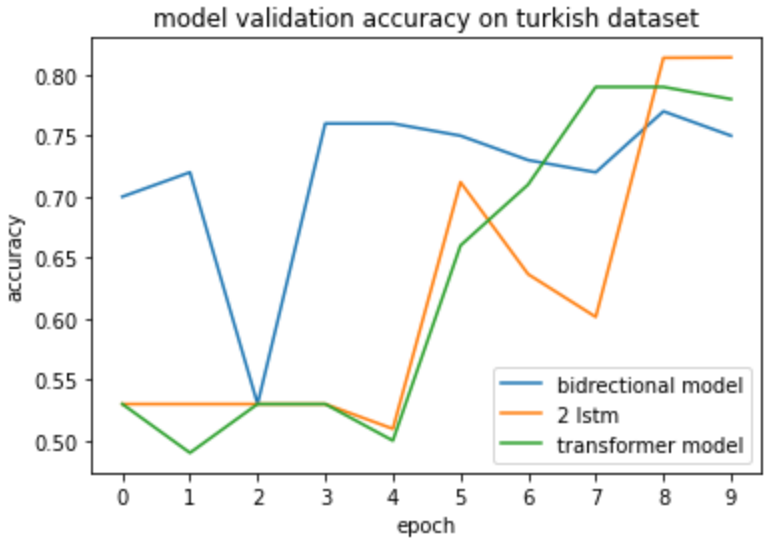
3. Again from the plot below we can see that our top performing models behave similar



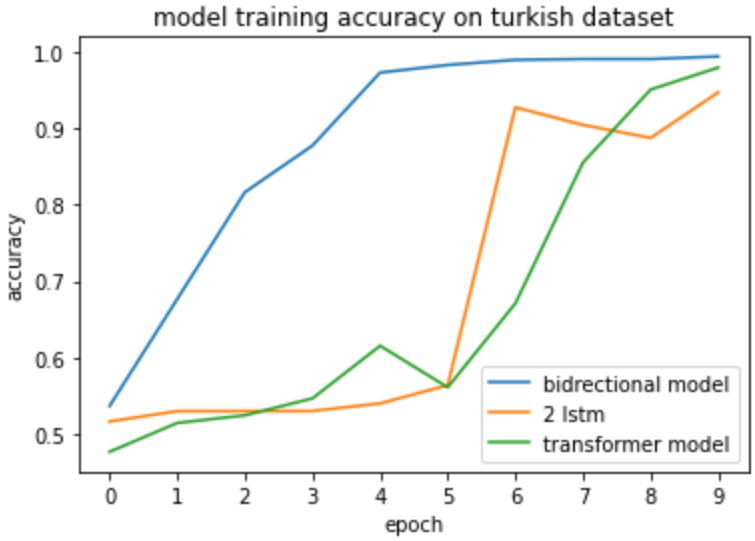
4. Training loss plot can be seen below:



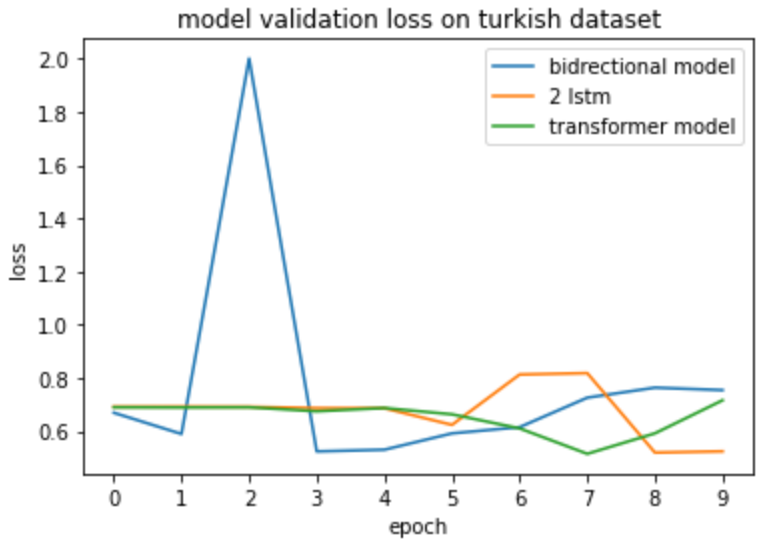
5. In the plot below we can see validation accuracy values. The zigzag nature of the plot should be attributed to the low number of samples 100. In this dataset 2 lstm models seem to perform the best.



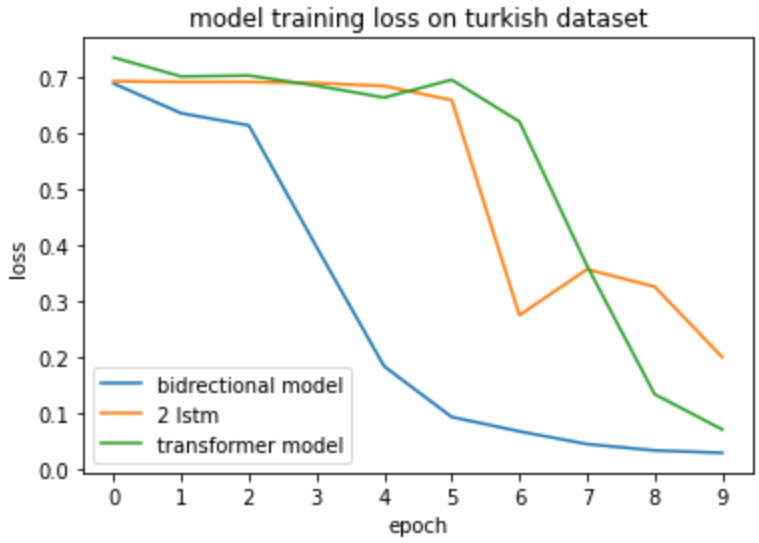
6. Bidirectional model overfits rather quickly more parameter tuning may be done to improve the performance of the model.



7. The spike in bidirectional model should be attributed to low number of test samples



8. Again from this plot we can confirm that the bidirectional model is overfitting we need apply more regularization



VI. Conclusion

In this project we have performed text classification using the Semeval dataset and the dataset we have created. When preparing Turkish dataset we used twitter and I was shocked to see the number of hate speech tweets. This validated the necessity of developing hate speech detection algorithms. For preparing the Turkish dataset we mainly focused on hate speeches in politics, religion, racism. Therefore our dataset is biased towards these categories. We tried numerous algorithms such as perceptrons, decision trees, random forests. We have seen that these algorithms even after parameter tuning do not perform well, they slightly outperform random evaluation. In this category the best performing algorithm was random forest with %63 accuracy, not bad for a simple machine learning algorithm. We tried 3 different deep learning algorithms; these algorithms included LSTM layers, Bidirectional-LSTM layers and transformer models. Out of these algorithms the best performing algorithm was different for english and turkish datasets. In the English dataset bidirectional-LSTM algorithm performed best with approximately %80 accuracy. In the Turkish dataset 2 layered LSTM model performed best with approximately %81 accuracy. This means that we were successful in preparing the Turkish dataset since both accuracy value of the datasets are pretty close. Further parameter tuning could have been done to find the best hyperparameters but we were satisfied with the results we have got.

References

[1] “SemEval,” 11-Dec-2020. [Online]. Available: https://en.wikipedia.org/wiki/SemEval . [Accessed: 17-Jan-2021].

[2] 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 É. Duchesnay, “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, 01-Jan-1970. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html . [Accessed: 17-Jan-2021].

[3] “TensorFlow,” Wikipedia, 12-Jan-2021. [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow . [Accessed: 17-Jan-2021].

[4] “Keras,” *Wikipedia*, 16-Dec-2020. [Online]. Available: https://en.wikipedia.org/wiki/Keras . [Accessed: 17-Jan-2021].

[5] “Natural Language Toolkit,” Wikipedia, 10-Dec-2020. [Online]. Available: https://en.wikipedia.org/wiki/Natural_Language_Toolkit . [Accessed: 17-Jan-2021].

[6] “Loss Functions¶,” Loss Functions - ML Glossary documentation. [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#:~:text=Cross%2Dentropy%20loss%2C%20or%20log,diverges%20from%20the%20actual%20labe l. [Accessed: 17-Jan-2021].

[7] J. Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,” *Machine Learning Mastery*, 12-Jan-2021. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20a%20replacement%20optimization,sparse%20gradients%20on%20noisy%20problems. [Accessed: 17-Jan-2021].

[8] “Recurrent Neural Networks cheatsheet Star,” CS 230 - Recurrent Neural Networks Cheatsheet. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks . [Accessed: 17-Jan-2021].

[9] P. S. -, By, -, P. S. I. am P. Sharma, P. Sharma, and I. am P. Sharma, “Keras Dense Layer Explained for Beginners,” MLK - Machine Learning Knowledge, 20-Oct-2020. [Online]. Available: https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/#:~:text=The%20dense%20layer%20is%20a,performs%20a%20matrix%2Dvector%20multiplication . [Accessed: 17-Jan-2021].

[10] “Transformer (machine learning model),” Wikipedia, 12-Jan-2021. [Online]. Available: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model) . [Accessed: 18-Jan-2021].