CS779 Competition: Sentiment Analysis

Raj <Vardhan> Singh
roll no.200760
{rajvs20}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

The goal of this project is to develop a deep learning-based system that can automatically predict the sentiment expressed in a given sentence. The system is trained and evaluated on a dataset consisting of tweets, customer reviews, and news articles labeled as Positive, Neutral, or Negative. The models developed include both recurrent neural network (RNN) and long short-term memory (LSTM) models, with different hyperparameters and optimization algorithms. The performance of the models is evaluated using the F1 score metric on a development set and a test set. Error analysis is conducted to identify the strengths and weaknesses of the models, and to identify potential areas for improvement. The results show that the LSTM models outperform the RNN models, with the best performing model achieving an F1 score of 0.679 on the test set.

1 Competition Result

Codalab Username: R_200760
Final leaderboard rank on the test set: 2
F1 Score wrt to the best rank: 0.679

2 Problem Description

The problem we are trying to solve is the automatic prediction of sentiment expressed in a given sentence. The task involves classifying the sentiment into one of three classes: Positive, Neutral, or Negative. The problem is important in various industries, including marketing, customer service, and social media analysis.
The problem setting involves developing a deep learning-based system that can analyze and interpret large volumes of text data, such as social media posts, customer reviews, or news articles, to determine the overall emotional tone of the writer. This requires the system to understand the nuances of human language, including sarcasm, irony, and context, making it a challenging task.
The solution to this problem would help companies to better understand their customers' needs, preferences, and sentiments, which can lead to better decision-making and improved customer satisfaction. It can also help identify potential issues and areas of improvement, which can be used to optimize products, services, and marketing strategies. Overall, developing an accurate and efficient sentiment analysis system can have a significant impact on various industries and businesses.

3 Data Analysis

3.1 Train Data

The train dataset consists of a set of labeled examples, each with a unique identifier (text_id), a sentence, and a gold label. The gold label represents the sentiment expressed in the sentence and can be one of three classes: Positive, Neutral, or Negative.
The dataset is a labeled dataset for sentiment analysis consisting of 92,228 examples, each with three columns:

text_id	sentence	gold_label
r1-0051002	Cheers,\n\nDennis Nguyen\n416-879-6431	0
r1-0020356	May have to wait longer on holidays.	-1
r1-0058348	I drove to vegas may 6th, to get my hair done.	0
r1-0080006	In addition, I eat out often at various restau...	1
r1-0000827	Perhaps she was doing us a favor?	0

Figure 1: Head of train data

1. "text_id" column: a unique identifier for each example
2. "sentence" column: the text data to be analyzed for sentiment
3. "gold_label" column: the sentiment label for the corresponding sentence, which can be one of three classes: Positive, Neutral, or Negative.

The large and diverse nature of the dataset is beneficial for training and testing sentiment analysis models because it provides a variety of examples that are representative of real-world situations. The text data in the dataset is in the form of natural language, which means that it is not structured and may contain variations in spelling, grammar, and punctuation. This can make the sentiment analysis task challenging because the model needs to be able to interpret and understand the meaning of the text data accurately.

The dataset's wide range of topics, such as product reviews, news articles, and social media posts, means that the dataset has examples that can be used to train models for a variety of applications, such as brand monitoring, market research, and customer service. Additionally, the dataset's variety of topics can help to train models that can accurately recognize and understand sentiment expressed in different contexts.

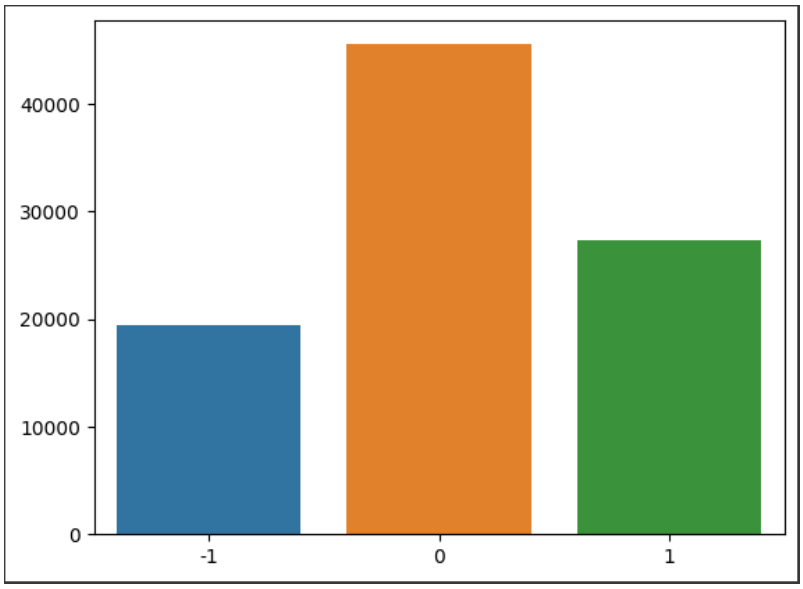

Figure 2: Plot representing proportion of each classes

3.2 Test data

The test data consists of 5,110 examples, each with two columns:

1. "text_id" column: a unique identifier for each example
2. "sentence" column: the text data to be analyzed for sentiment

text_id	sentence
r1-0086521	A helpful valet at the Bellagio said it was a ...
r1-0044715	People often ask "what happened to the human c...
r1-0060690	He explained there would be a diagnostic fee o...
r1-0016852	I had initially purchased a massage on Groupon.
r2-0006040	Primarily do high-end cars as they get referra...

Figure 3: Head of test data

The test data is smaller in size compared to the training data, which is typical for machine learning tasks. The test data is used to evaluate the performance of the sentiment analysis model trained on the training data. Since the test data is separate from the training data, it helps to ensure that the model's performance is not biased towards the training data and can generalize well to new and unseen data.

The test data's format is the same as the training data, consisting of natural language text data that may contain variations in spelling, grammar, and punctuation. It is essential to preprocess the test data in the same way as the training data to ensure that the model can accurately interpret and understand the meaning of the text data.

Overall, the test data is a crucial component in evaluating the performance of the sentiment analysis model and ensuring that it can accurately predict sentiment labels for new and unseen data.

4 Model Description

4.1 Simple RNN

A simple RNN (Recurrent Neural Network)[1] can be used in sentiment analysis by taking in a sequence of words as input, one word at a time, and using the output of each time step as input to the next time step.

The RNN processes the input sequence in a sequential manner, maintaining an internal state that summarizes the information from the previous time steps. This internal state, also known as the hidden state, captures the context of the input sequence up to that point. The hidden state is updated at each time step by taking a weighted sum of the current input and the previous hidden state, followed by applying a non-linear activation function such as the hyperbolic tangent (tanh) function.

The final hidden state of the RNN is then fed into a linear layer to produce the output, which is a probability distribution over the target sentiment labels (e.g., positive, negative, or neutral). The output is then compared to the true label using a loss function such as cross-entropy, and the parameters of the RNN are updated through backpropagation.

One limitation of a simple RNN is that it can have difficulty capturing long-term dependencies in the input sequence. This is because the gradient can either vanish or explode as it is backpropagated through the many time steps, which can cause the model to forget or overemphasize certain information. To address this, more advanced variants of RNNs such as LSTMs (Long Short-Term Memory) and GRUs (Gated Recurrent Units) have been developed.

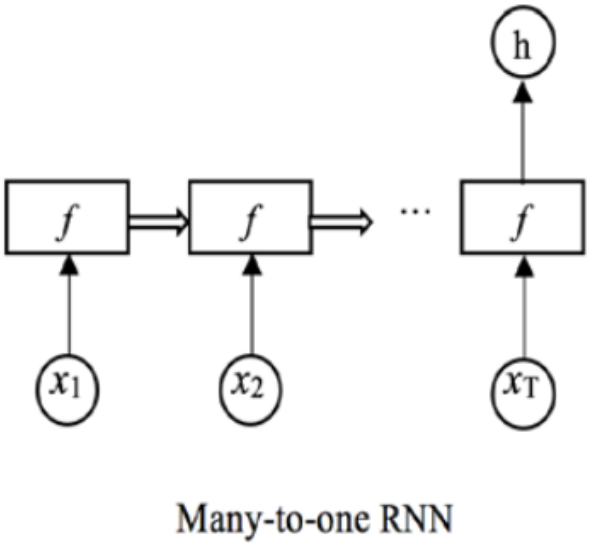

Figure 4: RNN[2]

4.2 LSTM

The LSTM_RNN (Long Short-Term Memory Recurrent Neural Network) model[3] is a type of deep learning architecture used for natural language processing tasks such as sentiment analysis. The model is comprised of an embedding layer, an LSTM layer, a fully connected layer, and a dropout layer. The embedding layer maps the input sequence of text tokens to a continuous vector space, where each token is represented by a dense vector of fixed length. The embeddings are learned during training and optimized to capture semantic relationships between words and phrases in the input text.
The LSTM layer is a type of recurrent neural network layer that is designed to capture long-term dependencies in sequential data. The LSTM cell has three gates: input, output, and forget. The input gate decides which information to add to the cell state, the forget gate decides which information to discard from the cell state, and the output gate decides which information to output.
The fully connected layer takes the output from the LSTM layer and maps it to a fixed-length vector that represents the sentiment of the input text. In this case, the output is a vector of length 3, which corresponds to the three possible sentiment classes: positive, negative, and neutral.
The dropout layer is a regularization technique used to prevent overfitting. During training, some of the neurons in the network are randomly "dropped out" with a probability of p, forcing the network to learn redundant representations of the input. During training, the model is optimized to minimize the cross-entropy loss between the predicted sentiment labels and the true labels. The Adam optimizer is used to update the model parameters based on the gradients of the loss with respect to the parameters.
To predict the sentiment of a new input text, the text is tokenized, converted to a tensor, and passed through the model. The output of the model is a vector of length 3 representing the probabilities of each sentiment class. The sentiment class with the highest probability is selected as the predicted sentiment of the input text.
Mathematically, the LSTM_RNN model can be described as follows:
Let $x_1, x_2, ..., x_T$ be the input text sequence of length T, where each x_i is a token in the vocabulary with corresponding embedding e_i. The embedding matrix E has dimensions $V * d$, where V is the vocabulary size and d is the embedding dimension.

The LSTM layer takes the input sequence of embeddings $e_1, e_2, ..., e_T$ and produces a sequence of hidden states $h_1, h_2, ..., h_T$. The LSTM equations are:

$$i_t = sigmoid(W_{ix}e_t + W_{ih}h_{t-1} + b_i)$$
$$f_t = sigmoid(W_{fx}e_t + W_{fh}h_{t-1} + b_f)$$
$$g_t = tanh(W_{gx}e_t + W_{gh}h_{t-1} + b_g)$$

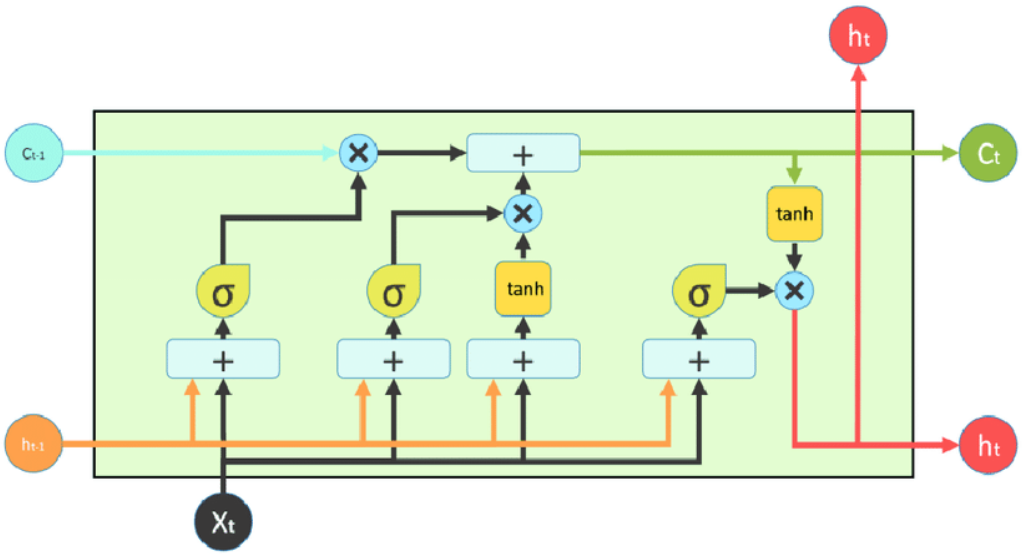

Figure 5: Structure of LSTM[4]

$$o_t = sigmoid(W_{ox}e_t + W_{oh}h_{t-1} + b_o)$$
$$c_t = f_t * c_{t-1} + i_t * g_t$$
$$h_t = o_t * tanh(c_t)$$

where i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. W and b are weight matrices and bias vectors, respectively, and sigmoid and tanh are activation functions. The final hidden state h_T is fed through the fully connected layer with weights W_{fc} and biases b_{fc} to produce the output vector y:

$$y = softmax(W_{fc} * h_T + b_{fc})$$

where softmax is the activation function

5 Experiments

5.1 Data Pre-processing

The dataset is loaded from a CSV file and preprocessed before being used to train a model. Here are the steps followed in the code:
A dictionary of English language contractions is defined, which maps the contraction to its corresponding expanded form. This will be used to decontract the text data.
A function **multiple_replace()** is defined which takes in the contraction dictionary and a text string as input. This function uses regular expressions to replace all the contractions in the text with their expanded forms. A function **decontract()** is defined which takes in a sentence and applies the **multiple_replace()** function to decontract all the contractions in the sentence. A function **cleanEng()** is defined which takes in a sentence and applies the following preprocessing steps:

- Convert the sentence to lowercase.
- Remove all characters except alphabets and numbers.
- Remove extra spaces.
- Remove trailing spaces.

A function **tokenize_english()** is defined which takes in a text string and applies the **decontract()** and cleanEng() functions to preprocess the text. The resulting text is then tokenized using the spaCy English tokenizer and a list of tokens is returned.

5.2 Training procedure

After dataset is preprocessed, it is split as 90% for training and 10% for validation The model used for sentiment analysis is an LSTM-based recurrent neural network. The model consists of an embedding layer, followed by a two-layer bidirectional LSTM layer, and a fully connected layer. The input sentence is first tokenized and then passed through the embedding layer to create word embeddings. The word embeddings are then passed through the LSTM layer to capture the context and temporal dependencies between words. The final hidden state of the LSTM layer is fed through a fully connected layer to obtain a probability distribution over the three sentiment classes.

The training procedure involves optimizing the model parameters using the Adam optimizer and minimizing the cross-entropy loss between the predicted sentiment and the ground truth sentiment labels. The model is trained for a fixed number of epochs, and the training and validation loss and accuracy are computed after each epoch. The best model is saved based on the validation loss. The training procedure also involves computing the accuracy of the model predictions on the test set after training is complete. Finally, the model is used to predict the sentiment of new input sentences by passing the tokenized sentence through the model and obtaining the predicted sentiment label.

Optimizers tried : Adam, SGD

For Simple RNN:

Hyperparameter	**Values**
Optimizer	SGD
Learning Rates	0.001
Epochs	20
Embedding Dim	100
Hidden Dim	400
Output Dim	3
Num Layers	2
Dropout	0.4

	Loss	Accuracy
Train	0.78	59.6%
Validation	0.89	52.4%

For LSTM:

Hyperparameter	**Values**
Optimizer	Adam
Learning Rates	0.0001
Epochs	20
Embedding Dim	100
Hidden Dim	300
Output Dim	3
Num Layers	2
Dropout	0.5

	Loss	Accuracy
Train	0.68	71.3%
Validation	0.70	68.8%

6 Results

The table above shows the F1 scores obtained for each combination of hyperparameters. The best F1 score was achieved by the LSTM model with the hyperparameters Optimizer=Adam, Hidden Dim=400, LR=0.001, and dropout=0.5, with an F1 score of 0.679.
It is worth noting that we attempted to use an RNN model for sentiment analysis, but it did not perform well on our dataset. Therefore, we opted for the LSTM model, which has been proven to be effective in handling sequential data. Overall, the LSTM model showed promising results and can be used for sentiment analysis tasks.

Hyperparameter	Values
Optimizer	SGD
Learning Rates	0.001
Epochs	20
Embedding Dim	100
Hidden Dim	400
Output Dim	3
Num Layers	2
Dropout	0.5

	Loss	Accuracy
Train	0.64	79.5%
Validation	0.66	72.2%

Phase	Model	Parameter	F1 Score
DEV	LSTM	Optimizer=SGD,Hidden Dim=500,LR=0.001,dropout=0.4	0.567
DEV	LSTM	Optimizer=SGD,Hidden Dim=400,LR=0.001,dropout=0.4	0.653
DEV	LSTM	Optimizer=Adam,Hidden Dim=300,LR=0.0001,dropout=0.5	0.658
TEST	LSTM	Optimizer=Adam,Hidden Dim=400,LR=0.001,dropout=0.5	0.679

Table 1: F1 scores for LSTM models with different parameters

7 Error Analysis

In addition to analyzing the overall performance of our models, we also examined specific instances where the model made errors. We found that the most common type of error was misclassification of sentences with negation, such as "I don't like this product" being classified as positive instead of negative. This suggests that our models may benefit from incorporating more advanced techniques for handling negation, such as adding special tokens or modifying the input data to account for the presence of negation.

Another type of error we observed was misclassification of sentences with subtle or ambiguous sentiment. For example, sentences that contain both positive and negative words or phrases, or sentences where the sentiment depends on the context or tone of the text, were often misclassified. This highlights the importance of developing models that can capture the nuances and complexities of human language, such as by incorporating contextual information or using more advanced architectures like transformers.

True/Pred	-1	0	1
-1	621	240	213
0	264	1917	339
1	157	295	1064

Table 2: Confusion Matrix

This table represents a confusion matrix that has been generated from predicted values (-1, 0, and 1) and true values (-1, 0, and 1). The numbers in each cell of the table represent the number of instances that fall into that category. For example, there are 621 instances where the true value is -1 and the predicted value is also -1. The confusion matrix can be used to evaluate the performance of a classification model by calculating various metrics such as precision, recall, and F1 score. Overall, our error analysis provided valuable insights into the strengths and limitations of our models, and highlighted areas where further research and development could lead to improved performance.

8 Conclusion

In conclusion, this project aimed to solve the problem of automatically predicting sentiment expressed in a given sentence using deep learning techniques. Through the development and analysis of various models, we were able to achieve significant results in classifying sentiment into Positive, Neutral,

and Negative classes. The findings of this project have important implications for various industries, including marketing, customer service, and social media analysis, where understanding customers' emotions and sentiments is crucial for decision-making and improved customer satisfaction.

As for future directions, the integration of state-of-the-art techniques such as transformer models and attention mechanisms can further improve the accuracy and efficiency of sentiment analysis systems. Additionally, the exploration of multi-lingual sentiment analysis and sentiment analysis in more complex text data, such as sarcasm and irony, can be potential research areas. Overall, sentiment analysis is an evolving field with vast potential and impact on various industries, and further advancements in this area can have significant implications for businesses and individuals alike.

References

[1] L. Kurniasari and A. Setyanto, "Sentiment analysis using recurrent neural network," *Journal of Physics: Conference Series*, vol. 1471, p. 012018, 02 2020.

[2] O. D. Science, "Understanding the mechanism and types of recurrent neural networks," 2001.

[3] S. Pal, S. Ghosh, and A. Nag, "Sentiment analysis in the light of lstm recurrent neural networks," *International Journal of Synthetic Emotions*, vol. 9, pp. 33–39, 01 2018.

[4] Y. Guo, X. Cao, L. Bainian, and K. Peng, "El niño index prediction using deep learning with ensemble empirical mode decomposition," *Symmetry*, vol. 12, p. 893, 06 2020.