Project 2 - Emotion

Henrik Daniel Christensen[hench13@student.sdu.dk]
Frode Engtoft Johansen[fjoha21@student.sdu.dk]

DM873: Deep Learning
University of Southern Denmark, SDU
Department of Mathematics and Computer Science

1 Introduction

The objective of this project is to develop a deep learning model capable of classifying the sentiment of a given text. The model will be trained on The *dair-ai/emotion* dataset from Hugging Face, a dataset of 16k tweets labeled with one of 6 emotions: sadness (0), joy (1), love (2), anger (3), fear (4) and surprise (5).

2 Label Distribution

We started by analyzing the distribution of the labels in the dataset, see Figure 1.

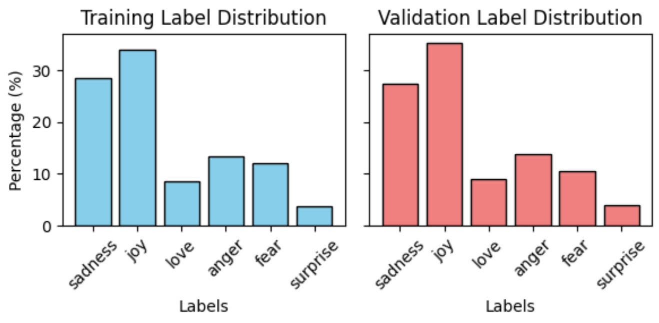

Fig. 1. Distribution of the labels in the dataset.

From the figure, it is clear that the dataset is very unbalanced in the training set, with the majority of the tweets being labeled as joy or sadness. However, the validation set have more and less the same distribution of labels. Therefore, we expect the test set to have a similar distribution.

Of course, this unbalance in the training set will have an impact on the performance of the model, as sentences labeled as joy will be more less likely to be classified correctly. This is something we will have to take into account when evaluating the performance of the model.

3 Tokenizing and Vocabulary

Next, we tokenize the text data for that we created a custom RegexpTokenizer which tokenizes the text by only allowing words, numbers and some special characters. Even though the dataset is rather cleaned. We choose to keep exclamation marks and question marks as they potentially could add value for sentiment analysis.

Next, we check the top 10 most common words in the dataset. First of all, many stopwords are present in the dataset, these could add some value for sentiment analysis, but we expect them to be more noise than signal. Therefore, we remove the stopwords from the dataset. In addition, many words having similar meanings, e.g. 'feel' and 'feeling' are present, therefore these are also removed by stemming the words.

After text cleaning, we could visualize the common words in the sentences by using a WordCloud, see Figure 2.

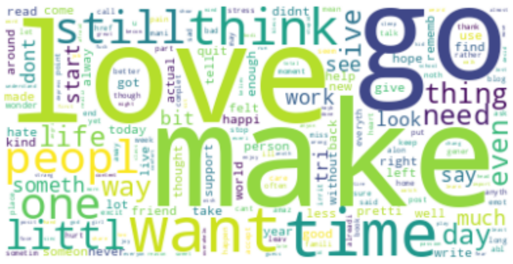

Fig. 2. WordCloud of the vocabulary.

As expected many emotional words are present in the vocabulary, e.g. 'love', 'friend', 'hate', 'never', 'go', etc.

To be able to process the text data, we need to choose a maximum sentence length. A boxplot of the distribution of the lengths of the sequences can be seen in Figure 3.

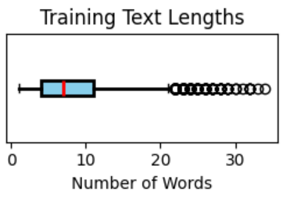

Fig. 3. Distribution of the lengths of the sequences.

From the boxplot, we see that the majority of the sequences have a length of around 8 words. We choose to set the maximum sequence length to 10 words. This means that all sequences longer than 10 words are truncated, and all sequences shorter than 23 words are padded with a special token, '<PAD>', to make them all the same length. In addition, we discovered that some sentences are very short, therefore we choose to remove all sentences with a length less than 3 words.

Having our cleaned and tokenized text data, we build a vocabulary from the training dataset. The vocabulary is built by mapping each unique word to an integer. The vocabulary for the training dataset ended up being 10,336 words.

Then, we convert the text data to integers by mapping each word in the text data to the corresponding integer in the vocabulary and pad the sequences to the maximum sequence length. This is done for both the training, validation and test datasets. The text data is now ready to be used for a model.

4 Models

Two different models were developed for the sentiment analysis task. The first model is a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells, and the second model is a Transformer model.

4.1 Model 1: RNN-LSTM

We choose to start with a 'simple' RNN model with LSTM cells. By using a grid search approach over the hyper-parameters like the number of hidden units, the number of layers, the dropout rate, etc., we found the following model shown in Table 2, having in total 1,117,734 number of parameters.

Embedding w/ 10,336 #embeddings (vocab size) and 75 #embedding dims.
LSTM w/ 10,336 #features and 256 #hidden units in 1 layer
Dropout w/ 50% dropout rate
Linear w/ 256 #features and 6 #output classes

Table 1. RNN-LSTM model.

The model is trained using the AdamW optimizer with a learning rate of 0.001, a batch size of 16. We use the CrossEntropyLoss as the loss function and train the model for 4 epochs. To prevent overfitting, we add a 0.0001 L2-norm penalty to the weights.

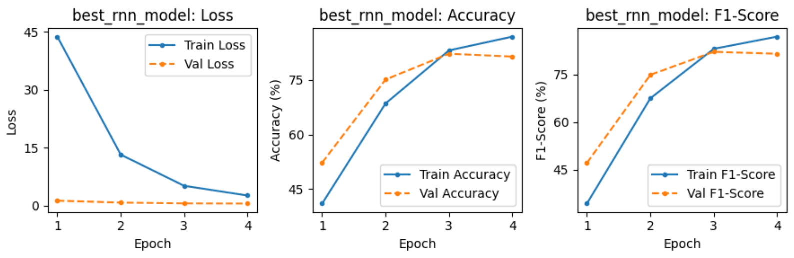

Fig. 4. Training and validation loss, accuracy and F1-score for the RNN-LSTM model.

From Figure 4, we see we hit the inflection point around 3 epochs, which is why we choose stop the training after 4 epochs.

Validation Accuracy	Validation F1-Score
81.4%	81.4%

Table 2. RNN-LSTM model performance on validation set.

4.2 Model 2: Transformer

5 Analysis and Final Prediction

6 Pretrained Model (DistilBERT)

To see how good our model performs compared to a well-known model, we used the pretrained DistilBERT base model (uncased) from Hugging Face. The model is trained on a dataset containing 11,038 books and English Wikipedia having a vocabulary size of 30,000 words.

To fit the model to our task, we trained the model for 5 epochs using a learning rate of $5e-6$. The performance of the model can be seen in Table 3.

Test Accuracy	Test F1-Score
88.9%	88.7%

Table 3. DistilBERT model performance on test set.

7 Conclusion

7.1 Individual Contributions

	Henrik Daniel Christensen	**Frode Engtoft Johansen**
Code	Task 1, 2, 5	Task 3, 4
Report	Section 1, 2, 3, 4, 7	Section 5, 6, 8

Table 4. Individual contributions.