Emotion Detection

Bouruc Petru-Liviu, Dawod Paul-Nasser and Florin Brad*
University of Bucharest, Romania
*Supervisor,Bitdefender, Romania

petru.bouruc@s.unibuc.ro, paul.dawod@s.unibuc.ro, fbrad@bitdefender.com

Facultatea de Matematică şi Informatică
Universitatea din Bucureşti

1. Introduction

1. The main goal of our project was to predict the emotion conveyed, given 3 utterances U1, U2, U3, in those messages: angry, happy, sad, others.
2. Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language.

2. Dataset and Preprocessing

Dataset Description:
For the stats we counted all the text available, that means train.txt, test.txt and dev.txt which can be found in [1]:

Dataset stats	
AVERAGE WORDS PER EXAMPLE	13.39
LOWEST NUMBER OF CHARS IN DATASET	6
HIGHEST NUMBER OF CHARS IN DATASET	834
AVERAGE EMOJIS PER EXAMPLE	0.376

Data preprocessing:
1. Replacing Emoticons and Emojis into normal language- this step was done 2 times because of errors when the found Emoticons where between 2 other Emoticons
2. Removing Ilegal characters that were not recognized by unicode
3. Using TweetTokenizer to split the concatanated Utterances
4. Shortening words that have the same character more than 2 times one after another to 2 characters
5.Using spell.correction on each word

3. Models

Baseline: We trained a feedforward neural network with one hidden layer and a ReLU activation function. The model gets as input the mean value of words features vectors for a dialogue. After some testing, we added a dropout layer to reduce the overfitting of the model. As for optimizer and loss function, we used Adam with weight decay, respectively, CrossEntroyLoss.
RNN Approaches: We have trained an LSTM model. For this, we define a vocabulary, consisting of translating words into numbers. Then to the model we give that matrix of numbers, with padding eventually for some dialogues, and in the first layer, if wanted, it converts numbers to their glove representation. We used almost the same hyperparameters, loss function and optimizer as before, but lowered a little bit the learning rate. Here we also added a dropout for the overfitting.
Glove embeddings vs embeddings from scratch The difference between these two approaches is the processing of embedding. In the first approach we use the glove representation for each word in the dialogue, while for the second, we just use the vocab index.

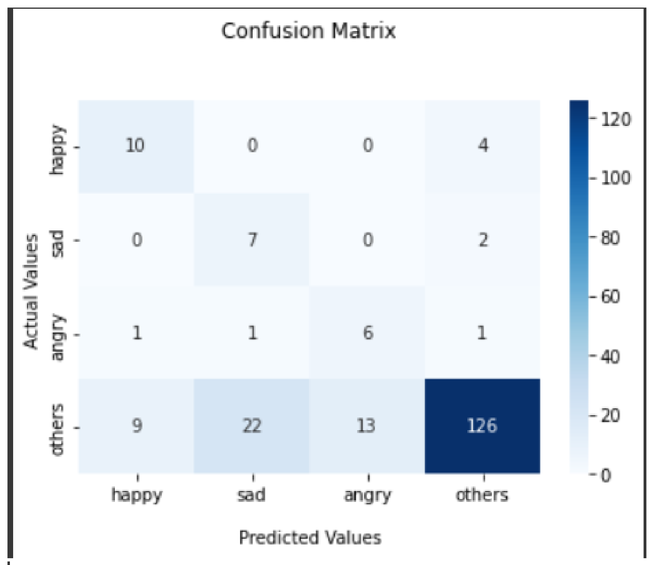
4. Results

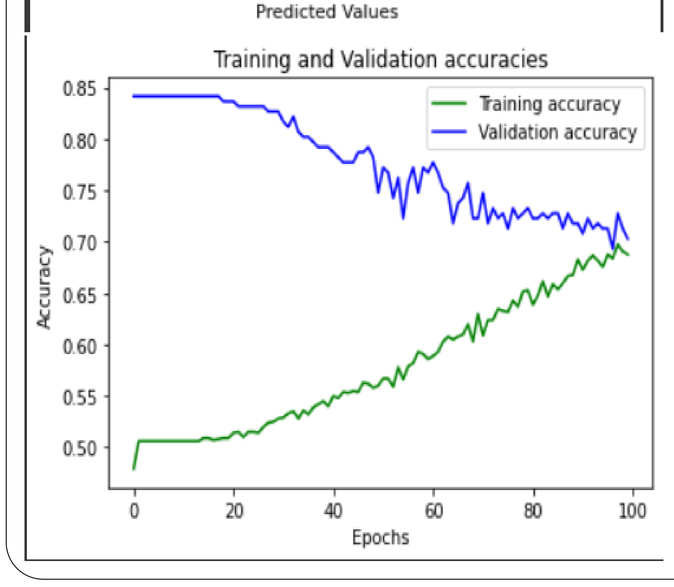
In this section we include things such as:

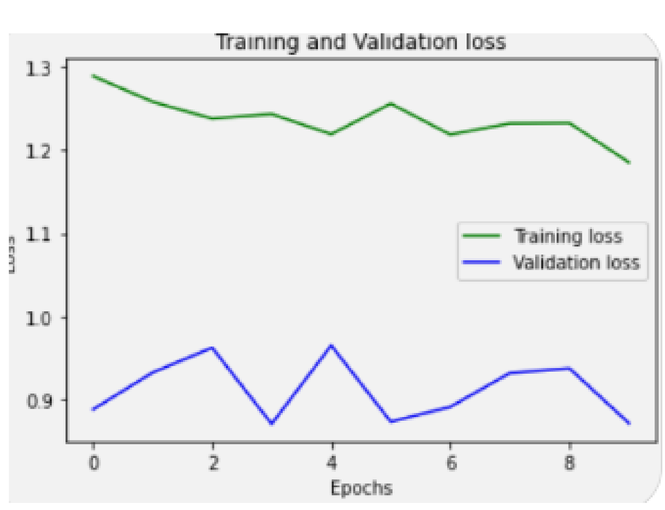
- tables with models and test metrics
- learning curves (train vs validation)
- performance curves for different models
- confusion matrix

-

Architecture	Accuracy	Precision	F1	Recall
FEED-FORWARD	0.79	0.84	0.88	0.93
LSTM WITH GLOVE	0.82	0.83	0.90	0.97
LSTM WITHOUT GLOVE	0.83	0.84	0.90	0.98







5. Conclusion

Using LSTM, we got better results than the simple feedforward, having an accuracy of procent bigger than the feed-forward, surpring was that the model without glove did better than the model with glove The uneven dataset was a problem that can be resolved in the future using distributed weights. Using GloVe that has preprocessed Tweets was helpful given the short lenght of the text, but the context is not the same between individual tweets and direct dialogue between two people.

6. References

1. https://colab.research.google.com/drive/1brb-IwHLgGYWQ86DuNsbz9Ti8AjUcfk?*usp* = *sharing*
2. https://www.aclweb.org/anthology/S19-2005.pdf
3. https://nlp.stanford.edu/projects/glove/
4. https://www.nltk.org/api/nltk.tokenize.html
5. https://radimrehurek.com/gensim/models/word2vec.html