# Emotion Detection using BiLSTM models

Onescu Iancu-Gabriel, Bertalan Victor, Florin Brad*
University of Bucharest, Romania
*Bitdefender, Romania

onescu.iancu@gmail.com, victorbertalan@gmail.com, fbrad@bitdefender.com

UNIVERSITY OF BUCHAREST
VIRTUTE ET SAPIENTIA

## 1. Introduction

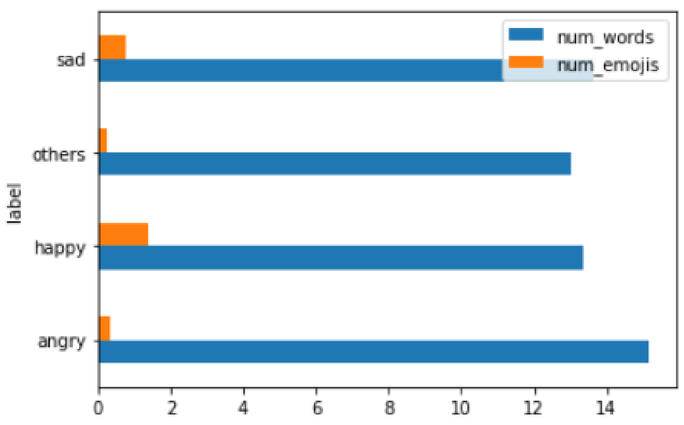
Given a series of short conversation between an end user and an agent we set to train a model that can predict what emotion the user expressed.
The challenges come from the fact that the conversations were recorded in an environment where natural, day-to-day language was used, meaning the preprocessing of the data was a crucial point in the task. Moreover, when texting in a non-formal environment the users are prone to making more spelling errors or typos.

## 2. Dataset and Preprocessing

**Dataset Description:**
The dataset is represented trough a series of short conversations between an user and an agent consisting of three lines in the following order: user, agent, user. Each such conversation is classified in one of the following classes: sad, angry, happy, other. This labeling shows the emotion the user exhibited during the conversation which is what we are trying to predict.

| Dataset stats | |
|---|---|
| AVERAGE NUMBER OF WORDS: | 13.57 |
| NUMBER OF WORDS IN LONGEST SAMPLE: | 164 |
| NUMBER OF WORDS IN SHORTES SAMPLE: | 3 |
| AVERAGE NUMBER OF EMOJIS IN TRAIN SET | 0.5214 |
| AVERAGE NUMBER OF EMOJIS IN VAL SET | 0.3208 |

Dataset stats for particular classes




We can see that happy people tend to use more emojis, while angry people smash their keyboard in an attempt to fix their frustrations
**Data preprocessing:** We will now describe the entire preprocessing pipeline:

- merged each conversation into a single text corpus
- replaced numbers with text
- removed liks
- converted emojis and emoticons to textual form (e.g.: 😂 →smiley face with tears of joy)
- normalized whitespaces
- expanded words (e.g.: We're →We are)
- corrected words (using the symspell module on a pretrained dictionary)
- tokenized input
- removed stopwords

In the end we had 30160 training examples, 2755 validation examples, and the test we set apart contained 5509 examples.

## 3. Models

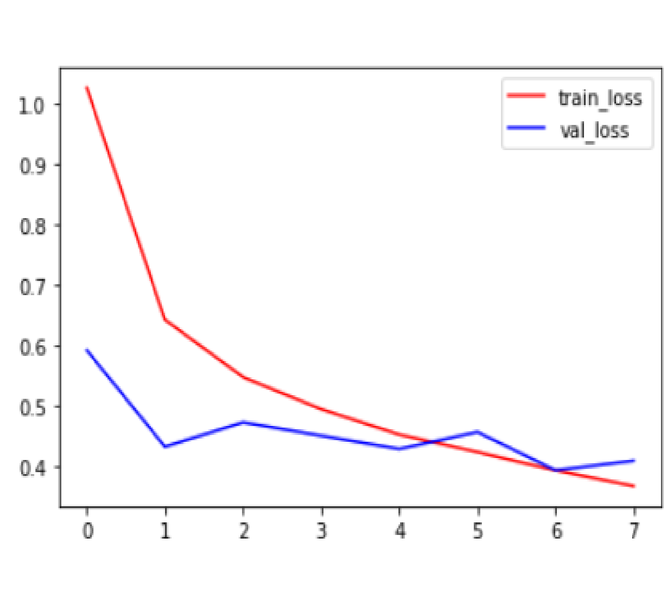
Models trained and hyperparameters chosen. For the word embeddings a pretrained GloVe model was chosen with a vector size of 300 items. We excluded the words that were not found in the dictionary for the baseline model, but trained an embedding layer for the lstm models. In the end we agreed to make a comparisson between the self-learned embeddings and the ones that GloVe provides.
**Baseline**: Multi-layered perceptron with a single hidden layer
**LSTM:** LSTM model. Hyperparams : optimizer: Adam, hidden size: 256, non-linearity: TanH, number of epochs: 10, batch size: 16, timesteps:50, dropout: 0.5
**BiLSTM:** Bidirectional LSTM model with only one set of layers. Hyperparams : optimizer: Adam, hidden size: 128, non-linearity: TanH, number of epochs: 8, batch size: 16, timesteps: 50, dropout=0.5

## 4. Results

The models are trained from scratch, and as previously explained the LSTM with GloVe uses the pretrained vectors for the respective positions in the embedding look-up table, the other one learns all the features by itself.
The following results were compiled using the weighted accuracy, precision, F1 and recall.

| Architecture | Accuracy | Precision | F1 | Recall |
|---|---|---|---|---|
| FEED-FORWARD | 0.66 | 0.85 | 0.71 | 0.66 |
| LSTM WITHOUT GLOVE | 0.82 | 0.89 | 0.84 | 0.82 |
| LSTM WITH GLOVE | 0.84 | 0.88 | 0.85 | 0.84 |
| BILSTM | **0.86** | **0.89** | **0.87** | **0.86** |

From the following confusion matrix we see that a lot of examples from the "other" class were miss classified. We could reduce the number of instances, but the result would hurt real applications because most of the conversation are just "small-talk" with no real emotion involved.


The loss shows that the model is prone to overfitting fast, even though the sizes for the hidden layers we tried were 128 and 256, with the former being the better choice. From this follows the fact that deeper architectures showed no improvement at the expense of training time.

**Some missclassified examples.**

*Where are you now?*
*At home just about to have breakfast...*
*what are you eating?*
*predicted:**sad**, actual:**other***

*Yeah mee too*
*Me too! All my friends are also excited*
*Ohhh so funny*
*predicted:**angry**, actual:**happy***

**Some well classified examples.**

*That was a joke btw...*
*it was*
*Yes smiley-face*
*predicted:**happy**, actual:**happy***

*First you hurt me*
*okay*
*So I talked rude*
*predicted:**angry**, actual:**angry***

Apart from the hyperparameters tuning we tried to study how a loss function with custom weights would perform. The method we chose was to divide the number of occurences for the class that has the most examples with the number of occurences for each class. But the results were a disappointing as the models have shown a decreased accuracy by approximately 2%.

We also tried to eliminate certain steps in the preprocessing pipeline and found out that the accuracy will decrease if we eliminate the emojis and emoticons instead of converting them to their textual form. This shows that users tend to express their feelings easier this way than trough words.

## 5. Conclusion and future improvements

Once again BiLSTM proves itself superior to it's one-directional counterparts. Although the architectures were not very deep they were prone to overfitting very fast, party because after preprocessing most of the examples were left with just a few words. This was seen in the choice of number of timesteps: 50. A larger value would result in an overfit even though the model should learn that the padding instances are not important for classification.