Task 27: haspeede3

Bruno Francesco Nocera
1863075
nocera.1863075@studenti.uniroma1.it

1 Introduction

The task is binary classification of some tweets, in order to verify if these ones are hate tweet or neutral one. To resolve this task I have tried many approach, but the one that works better is a BiLSTM combined to a word2vec model.

2 Dataset

The data is taken from the haspeede3 dataset. It's split into training and test. I've also splitted the train in train part e validation part. Sample of the input format processed for the task is showed in **table 1**.

3 Preprocessing

I've removed symbol like @ and # because when we tokenize the text, sometimes these symbol are usually spitted alone as token and doesn't make much sense. In some tweets there are url indicated as [URL], I've removed them as well. With these removement I didn't notice a big difference in terms of performance of model, but when remove also emoji and special character the accuracy of the model decrease as we can see in Figure 3 and Figure 4. Thus I guess that the use of special character and in particular of emoji help the model in learn some important pattern. In second step I've choose the tokenizer for the text, the choice was between spacy, nltk and stanza. I've tried all but the difference is relative small as we can see at **table 2** in the case of italian text, thus I've decide to use nlkt because is very fast w.r.t the others. About stopwords, remove them increase the accuracy of the model maybe because lead the model to focus on important part.

4 Architecture and design choices

We can see the architecture at Figure 5. As we can see we start from cleaning text as described before, then through nltk tokenize, we obtain tokens in which the stopwords are removed. Now before pass to the BiLSTM layers, the tokens are embedded via word2vec model. I've choose to use a word2vec model trained on the train set because it allows the embedding layer specifically adapt on the vocabulary present in the training data. In fact this leads a better performance of the model w.r.t the original idea in which we are using a vocabulary from torch text. Thus the result of this layer is passed to an BiLSTM. I've tried both single LSTM and BiLSTM, and I've notice better result on the second one (as we can see in the figure 1 and 2). I guess because BiLSTM is designed to capture both forward and backward dependencies within the text. Note that I've implemented a mechanism of early stopping, in fact when train the network I save the model if the accuracy on validation increase, in fact in the last cells I load the best model in the train w.r.t the valid accumulation before test the model on test set.

5 Baseline

The baseline that I have choose is stratified baseline. I've iterated on the train json file and with a dictionary I've tracked the number of label 0 (neutral) and 1(hate speech), then I've used this distribution to predict on the test data, and I've reached 0.55 of accuracy.

6 Advanced Baseline

The idea for this baseline is the following: tokenize the text, and divide the tokens in 2 sets, good set and bad one. The good set contains token contained in sample with label neutral, on the other hand the token that are contained in the sample with label hate speech are in bad one. If an element are contained in both is removed. Now to evaluate on test, I used a word2vec model trained on the json train file, the idea is to tokenize the test element and for

each token obtained check the most similiar token and count if amoung them there are more in bad or in good set. If there are more in a good set I choose from [neutral,hate] with probabilities [0.8,0.2], because "it's more like that is neutral", while if there are more in bad set viceversa. If there is a pair the probabilities is [0.5,0.5] as expected. With this approach I've obtained about 0.64 of accuracy. Note that the word2vec used for this baseline is the same used to embedding in the principal model, I've tried to use a lighter model with less epochs, and smaller windows to increase the speed of the execution, but there is a decrease of accuracy of 3/4 percentage.

7 Results

From the results **at table 4** we can see that the model reach an accuracy and precision better than the baseline, and even of the advanced baseline, the latter share also word2vec model used to embedding. Despite the accuracy is quite good, I think that the precision is low, and this issue I guess is due to an imbalance of the classes. In fact I have 1715 of hate sample, and 2765 of neutral sample, so the model is better to recognize neutral sample rather than hate sample. Another issue is that the low number of data train w.r.t to test data, this is responsible for a low reliability of the validation, and low representative of the test during the training.

8 How to run the code

The code works in colab, it is enough to press run all button to work as requested. I remain also some of the old code that I've experimented.

LSTM VS BILSTM training

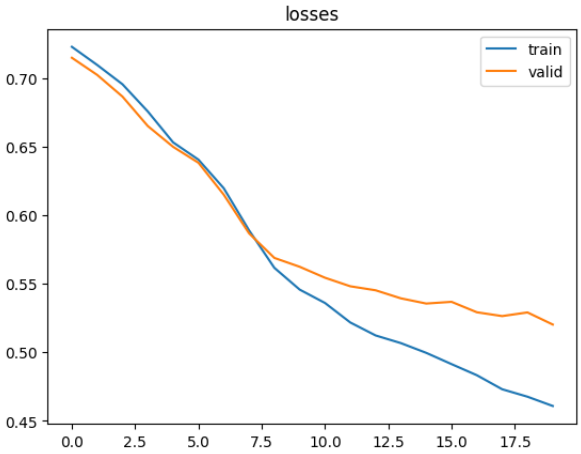

Figure 1: This is an example of training of the model with LSTM. In this case the accuracy reach 70 %

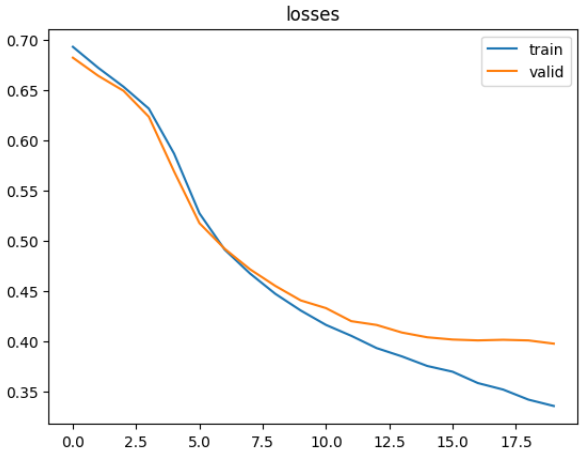

Figure 2: This is an example of training of the model with BiLSTM. In this case the accuracy reach 75 %

Emoji vs non Emoji

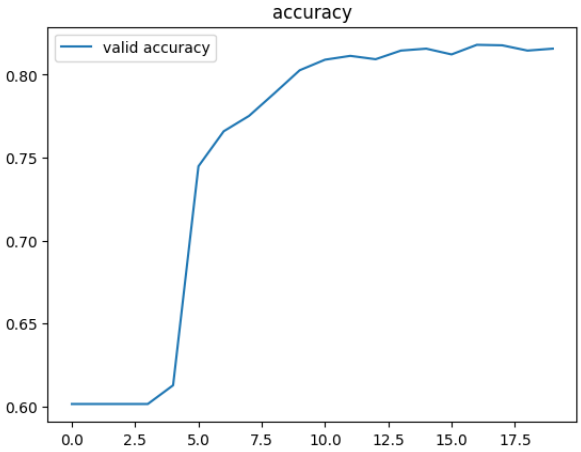

Figure 3: Accuracy on validation test WITH emoji even surpass 80% on validation, while on test reach 75%

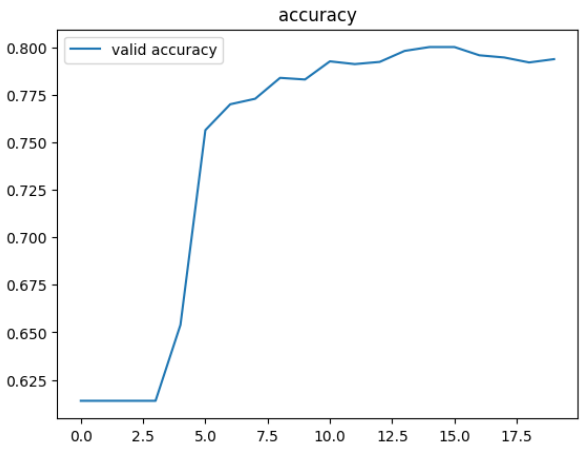

Figure 4: Accuracy on validation test WITHOUT emoji not surpass 80% on validation, while on test reach 70%

Architecture

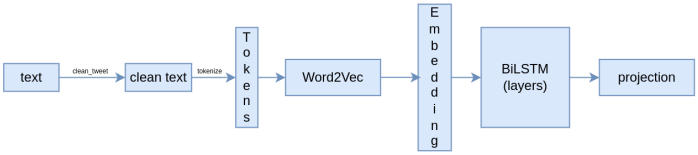

Figure 5: BiLSTM model architecture.

idx	text	tokens	label
"0"	"Con tutte le denunce che si sta beccando, Salvini ..."	['tutte', 'denunce', 'beccando', ',', 'Salvini', ...]	0

Table 1: Examples taken from the training

field	value
original text	Con tutte le denunce che si sta beccando, Salvini rischia
spacy	['denunce', 'beccando', ',', 'Salvini', 'rischia']
nltk	['tutte', 'denunce', 'beccando', ',', 'Salvini', 'rischia']
stanza	['denunce', 'beccando', ',', 'Salvini', 'rischia']

Table 2: Example of tokenizers

Name parameter	best value
vector size	100
hidden dim	32
dropout	0.4
bilstm layers	4
bidirectional	True
learning rate	1e-4
Optimizer	AdamW
learning rate	1e-4

Table 3: Best parameters for the model

model	accuracy	precision
BiLSTM+W2V	0.76	0.43
Advancend baseline	0.66	0.36
Stratified baseline	0.55	0.27

Table 4: Best results for the model and baselines