12/10/2022

SUPERVISED LEARNING

Texte classification

ÉCOLE D'INGÉNIEUR·E·S

Creating the future together

Gedeon PASSO-KAFACK

Link to the repository: https://github.com/GoblinPakage/Supervised-Learning

Table of contents

Introduction

Describe the dataset and task

- What is the data about, who collected it and how?

The GoEmotions dataset contains 58k carefully curated Reddit comments labeled for 27 emotion categories or Neutral. The emotion categories are admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realization, relief, remorse, sadness, surprise.

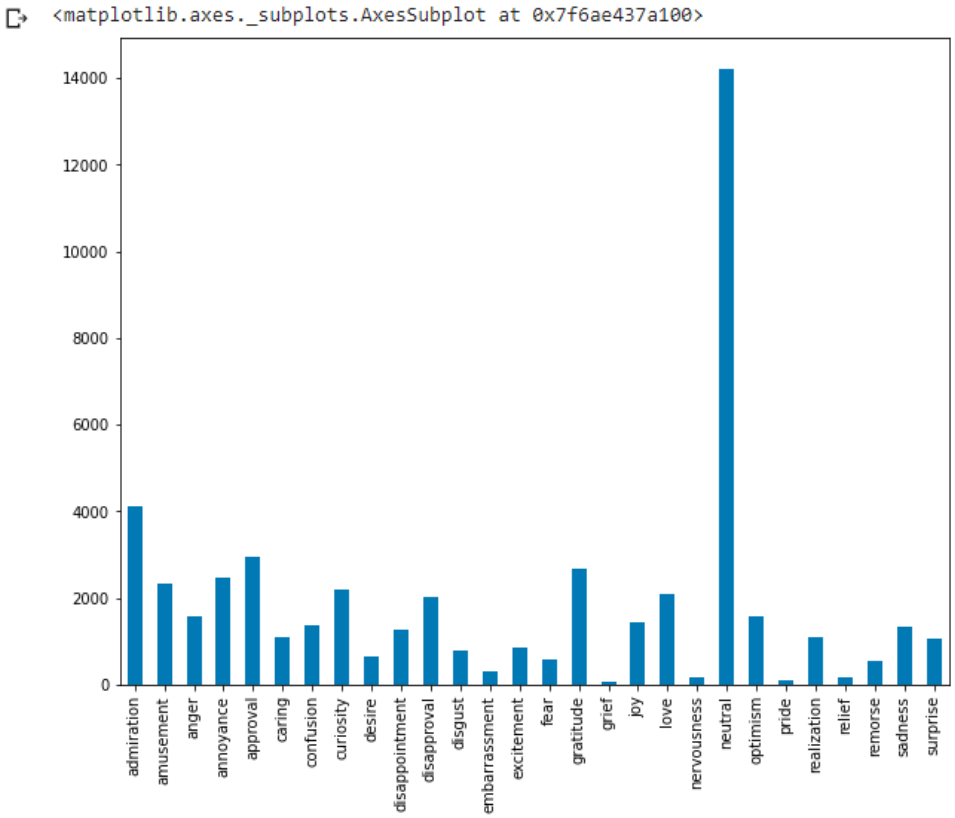


Figure 1: Proportion of presence for each emotion in the dataset

- What is the task targeted by your experiments?

In this dataset we have 27 emotions categories. We tried to make a multi-classification with 6 of this emotion plus neutral: admiration, anger, disappointment, excitement, curiosity, fear, neutral.

We tried in regard of the diagram below to take emotion that regroup this sentiment positive negative ambiguous.

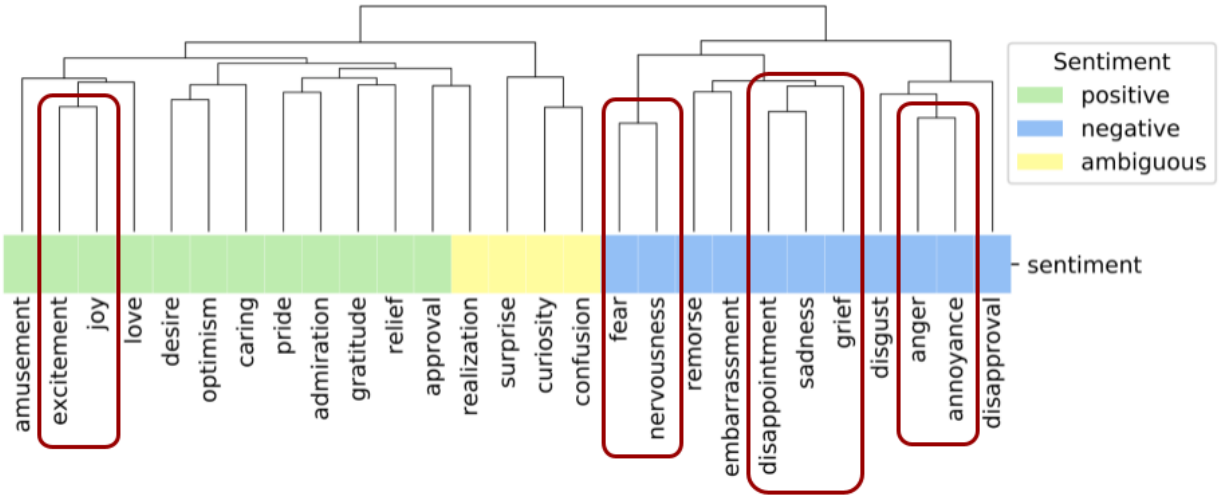


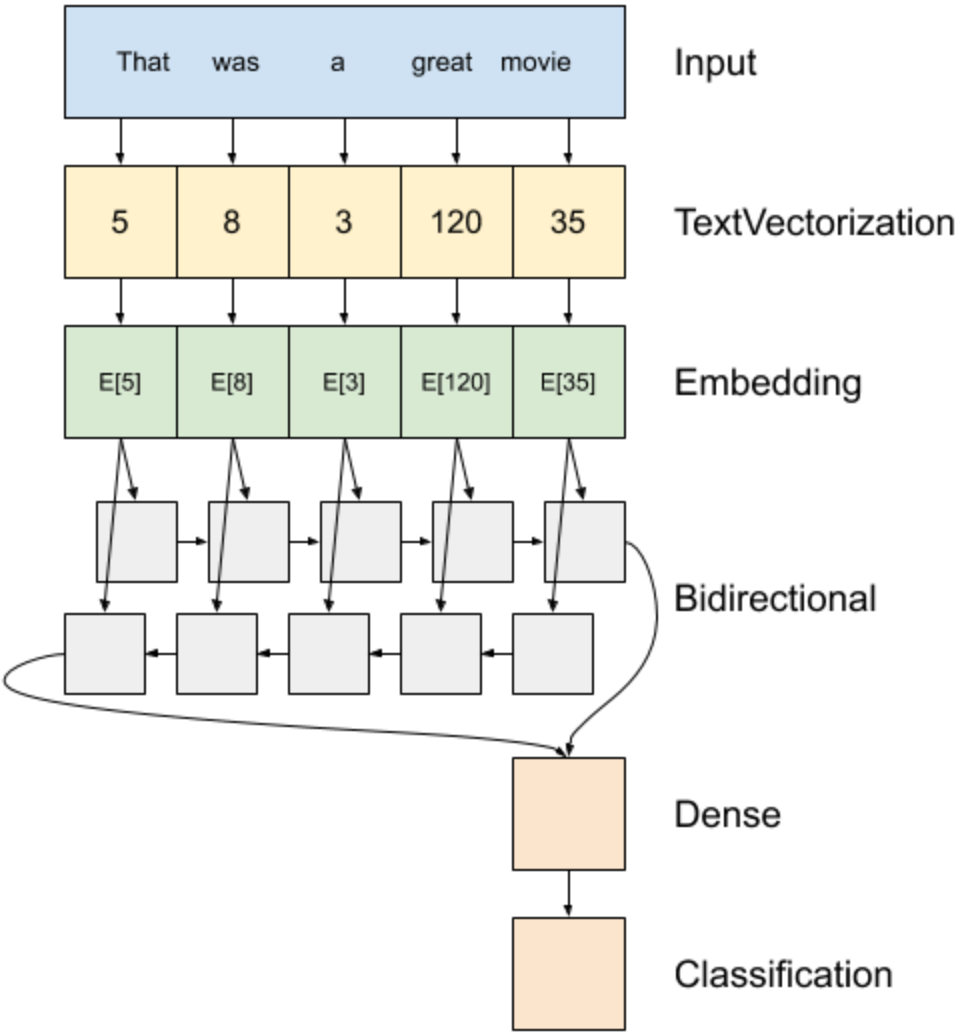
Figure 2: Emotion group by sentiment

- **Summarize your objectives** and what you aim to achieve with your experiments

In conclusion we tried to predict et sentiment in the comment of people using 6 emotions. ‘This type of data can be valuable for building expressive conversational agents, as well as for suggesting contextual emojis, and is a particularly interesting area of future work.’ (GOOGLE, n.d.)

Proposed pipeline

- Briefly describe and **justify** the models you used and how you applied it/them to your task



Above is a diagram of the model. This model can be build as a `tf.keras.Sequential`.

1. The first layer is the `encoder`, which converts the text to a sequence of token indices.

2. After the encoder is an `embedding` layer, that converts the sequences of word indices to sequences of vectors.

3. A recurrent neural network (RNN) processes sequence input by iterating through the elements. RNNs pass the outputs from one timestep to their input on the next timestep.

Here we use a recurrent layer of the LSTM type.

4. Additionally, we use the tf.keras.layers.Bidirectional wrapper. This propagates the input forward and backwards through the RNN layer and then concatenates the final output. It is often used with text sequences (but should not be used with time series since it breaks causality).

- The main advantage of a bidirectional RNN is that the signal from the beginning of the input doesn't need to be processed all the way through every timestep to affect the output.
- The main disadvantage of a bidirectional RNN is that you can't efficiently compute online predictions for a word stream since new words keep getting added at the end of the sequence.

5. After the RNN has converted the sequence to a single vector the two `layers.Dense` do some final processing, and convert from this vector representation to a single logit as the classification output. (GOOGLE, n.d.)

- Describe what your baseline model is and the more complex models you intend to try

So, my baseline model is one that the tutorial of tensorflow present and that we seen in class. I adapt this with some pre-processing tasks to the data and this is what I get for the test:

```
[85] sample_My_example="Whaou! It is beautiful!"
     prediction2=model2.predict(np.array([sample_My_example]))
     prediction2

     1/1 [==============================] - 6s 6s/step
     array([[9.7801363e-01, 6.3568586e-03, 3.1051313e-04, 1.2562207e-03,
             3.2307953e-03, 1.2297310e-03, 9.6022049e-03]], dtype=float32)
```

Figure 3: First prediction

There we see that the array represents the prediction for each emotion that we choose above. At the end, the predict that 'admiration' correspond to the comment: 'Whaou! It is beautiful'. This work with the evaluation of 0.45 accuracy of the test.

```
prediction1=model2.evaluate(test_tfds)

181/181 [==============================] - 3s 7ms/step - loss: 0.1970 - accuracy: 0.4572
```

Figure 4: First Test evaluation

Then my complex model, as in the tutorial we have two models one with bidirectional 'return_sequences = false' and the second we true. The accuracy with 'return_sequences = true' was more than false so we kept it as our complex model and tried to ameliorate it:

- First by adding a 'softmax' activation for output layer which is used to predict a multinomial probability distribution. We have a discret vatiable emotion within this case 7 values.
- Also, with the declaration "some parameters, like the pre-trained word embeddings or the last layer of the network, have a large impact on the performance, while other parameters" (Gurevych, 2017). We already adjust the last layer. For the embeddings as we couldn't transform the text in ASCI. We think that is the first amelioration to do to improve the model, we will find a way to change the comment text to ASCI later.
- Also added regularize and dropout layer

- Describe and **justify** any data pre-processing you had to do

For the data pre-processing part, we must do:

- The emotion feature has the type of Boolean when we load it. So, we must transform it, in int [0,1] which replace [false, true]:
  - We changed the dataset to dataframe which easier for us to make modification to the feature.
  - We changed value of Boolean feature to int value.

	admiration	excitement	anger	disappointment	curiosity	fear	neutral
0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1
4	0	0	0	0	1	0	0

Figure 5: One hot encoding

- As we said above, we had chosen only 6 emotions plus 'neutral' so after transform the emotion label we drop the feature that we don't need.
- After we change the final dataframe to tensor dataset:
  - We sliced the comment feature and the emotion feature so that we could made targets
  - We use "tf.data.Dataset.from_tensor_slices" to achieve the operation.
- We done the same thing to validation and test dataset

Experimental methodology

- Describe and **justify** the methodology used to test your pipeline
  - Metrics you've evaluated

    - Accuracy: Calculates how often predictions equal labels.
    - BinaryCrossentropy: Computes the crossentropy metric between the labels and predictions.
    - Adam optimizer:

  - Hyper-parameter choice criteria (which were set to the framework default, which were tuned using grid/random/other search method)

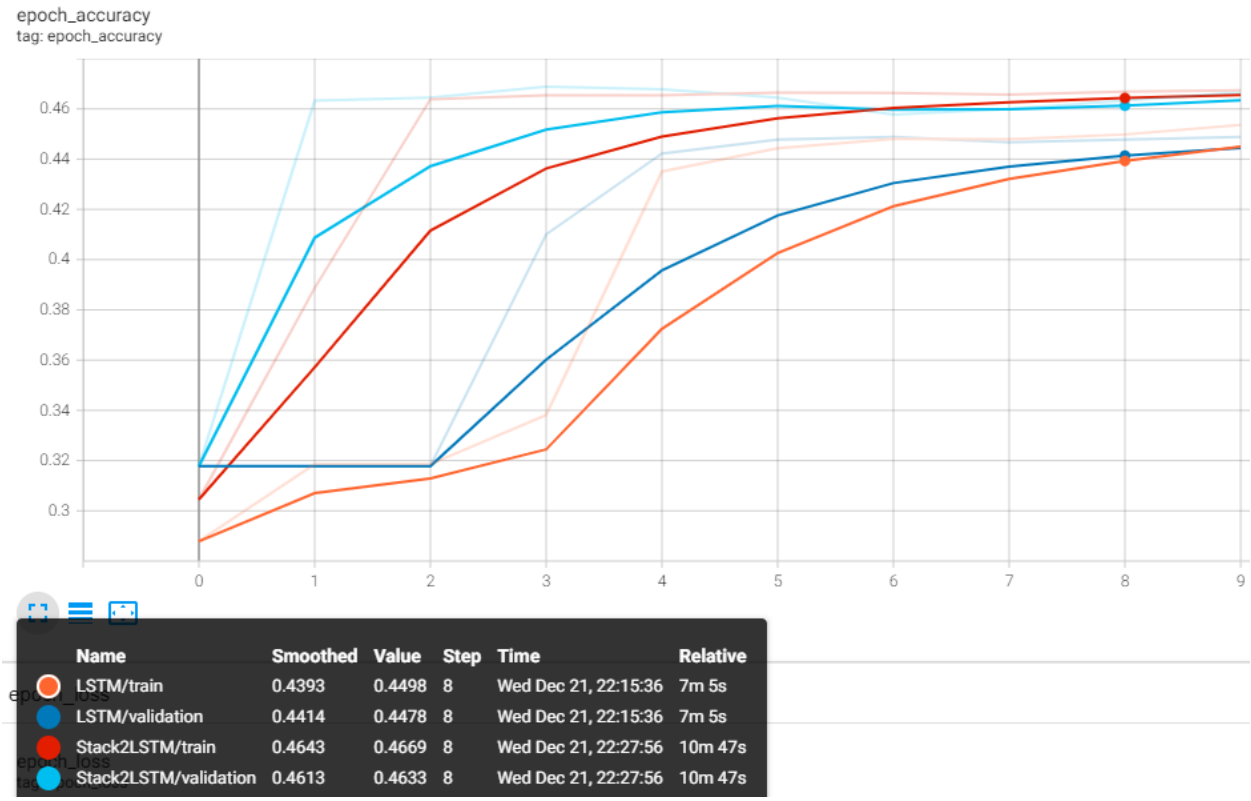
    - Layer: [1, 2] Find by search
    - Neuron: [64, 128]
    - Epoch: [10, 15, 20] random
    - Dropout: [0.01, 0.25, 0.5] random, search

- Cite your GitHub repo for any implementation details

  https://github.com/GoblinPakage/Supervised-Learning

Results and discussion

- Present the **main relevant results** of your experiments
  - Use **tables and/or plots** to summarize the obtained performances on the train/valid/test sets



Name	Smoothed	Value	Step	Time	Relative
LSTM/train	0.4393	0.4498	8	Wed Dec 21, 22:15:36	7m 5s
LSTM/validation	0.4414	0.4478	8	Wed Dec 21, 22:15:36	7m 5s
Stack2LSTM/train	0.4643	0.4669	8	Wed Dec 21, 22:27:56	10m 47s
Stack2LSTM/validation	0.4613	0.4633	8	Wed Dec 21, 22:27:56	10m 47s

Figure 6: First optimisation

- Based on your results, **explain your reasoning** behind changes you made to improve your performance

  First think made us improve our model is the result of accuracy. The second was the course but also the publication and web site such as (Gurevych, 2017)

- **Analyse and compare the results between baseline and proposed models**, discuss which is the best model for the task

  The best model is the baseline present by TensorFlow we just have to accommodate it to our context and make sure to not overfitting or underfitting.

- Comment on the **main difficulties** encountered in running your experiments and/or to improve your models

The main difficulties encountered:

- Handle tensor dataset like we saw I must use 7 features as targets label. The way to do that take me a lot of time. And without that I couldn't used the model.
- Find the good parameter and the good function for my last layer so that I have good output of the 7 features.

Conclusion and next steps

This was a great work for me. It helped me re-understand wat I thought I understand. Also, I work with TensorFlow dataset that not easy at first. Furthermore, with this we expect for a comment come out with a least 3 types of emotion in the comment.

- Comment on eventual extra experiments or procedures you could perform to:
  - refine your decision,
    - About embedding another type of embedding.
    - Add cross-validation
  - to improve your model's performance
    - Use all the features and after selecting which emotion we want to predict

References

GOOGLE. (n.d.). *Google Research*. Retrieved from ai.googleblog.com: https://ai.googleblog.com/2021/10/goemotions-dataset-for-fine-grained.html

GOOGLE. (n.d.). *tensorflow*. Retrieved from tensorflow: https://www.tensorflow.org/text/tutorials/text_classification_rnn

Gurevych, N. R. (2017). *www.semanticscholar.org.* Retrieved from www.semanticscholar.org: https://www.semanticscholar.org/paper/Optimal-Hyperparameters-for-Deep-LSTM-Networks-for-Reimers-Gurevych/1929540803b36222b406cc0aeaa549789e7eba56