Multi-Label Emotion Regression - LSTM vs. BERT

Tomer Lieber (ID. 312485782)
Moti (Mordechay) Goldklang (ID. 037983582)
Submitted as final project report for the NLP course, IDC, 2020

1 Introduction

1.1 General

During the NLP course we learned a variety of approaches and models. We started with probabilistic models like n-gram and HMM which were more common in the past. We continued with classic NLP deep learning models such as RNN and LSTM, and finished with transfer learning and state of the art models like BERT and GPT.
As part of the final project we decided, in collaboration with the lecturer Dr. Kfir Bar, to make a comprehensive comparison between a classic deep learning model and a state of the art model on a specific dataset. The LSTM was chosen as the classic deep learning model, BERT as the state of art model and EmoBank as the dataset. The purpose of the comparison is to experience first hand the performance of each method on a new problem and try to conclude interesting insights.

1.2 EmoBank Dataset

EmoBank is a repository that contains 10k English sentences. Each sentence was manually annotated with emotion according to the psychological VAD scheme. The V, A and D represent Valence (positive vs. negative), Arousal (calm vs. excited), and Dominance (being controlled vs. being in control). Each of those take numeric values from [1, 5].

For example, the most extreme positive and negative sentences are:
“lol Wonderful Simply Superb!” (Valence: 4.6)
“Fuck You!” (Valence: 1.2)

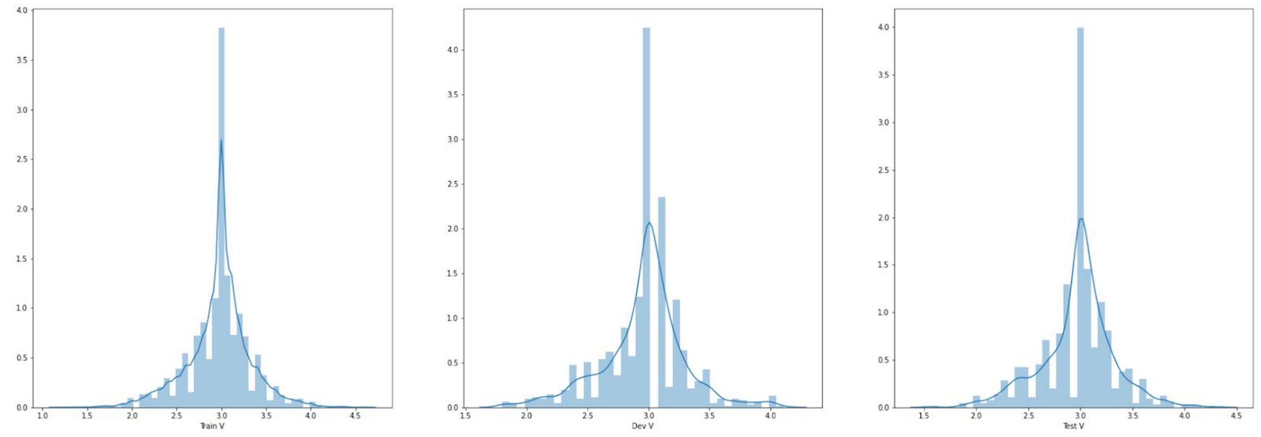
The most extreme arousing and non-arousing sentences are:
"My God, yes, yes, yes!" (Arousal: 4.4)
“I was feeling calm and private that night.” (Arousal: 1.8)

The most extreme dominating and dominated sentences are:
"NO" (Dominance: 4.2)
“Hands closed on my neck and I felt my spine cr...” (Dominance: 2.0)

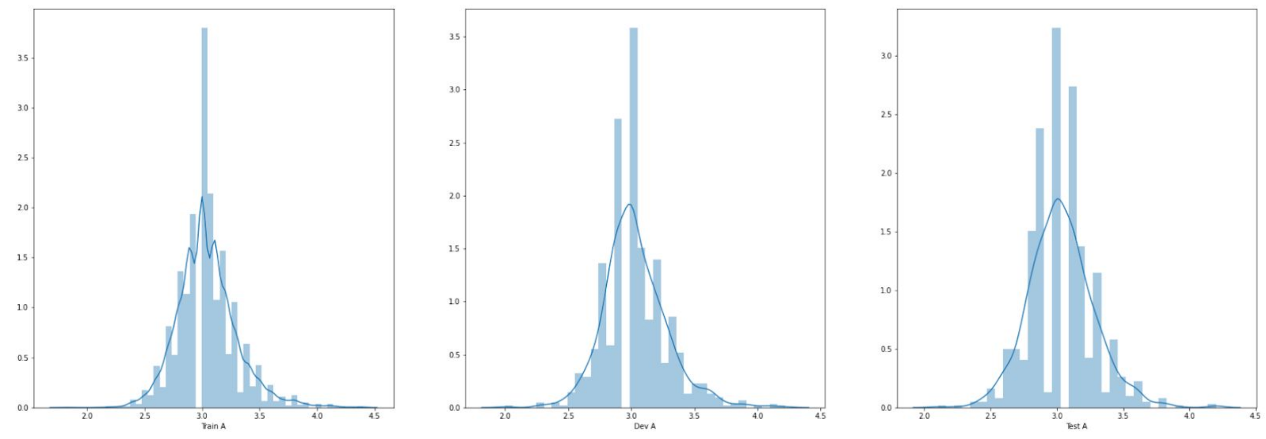
Each sentence in this repository was annotated twice: the emotion which is expressed by the writer, and the emotion which is perceived by the readers. The dataset that we are working on used a weighted average of these values.

The data splits were given and splitted into 3 groups: train (8062 sentences), dev and test (1000 sentences each).
When examining the distributions for our labels, we could see that it was more or less the same normal distributions on each one of our splits:

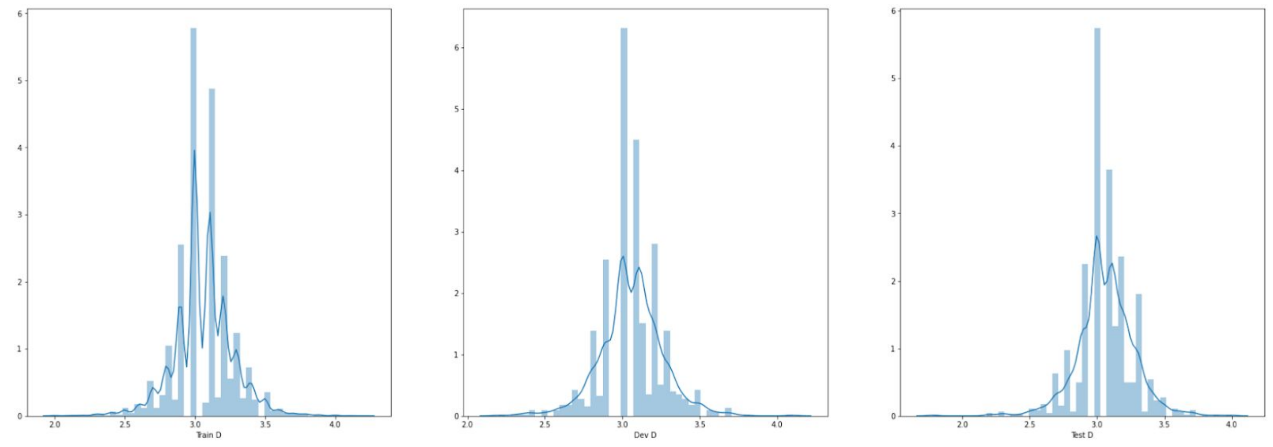
Valence (Train, Dev, Test)



Arousal (Train, Dev, Test)



Dominance (Train, Dev, Test)



	Valence (avg/std)	Arousal (avg/std)	Dominance (avg/std)
Train	2.97 / 0.35	3.04 / 0.26	3.06 / 0.21
Dev	2.96 / 0.33	3.03 / 0.25	3.06 / 0.20
Test	2.97 / 0.34	3.03 / 0.24	3.064 / 0.21

In our project, we examined the performance of LSTM and BERT models to predict the valence, arousal and dominance values. We trained on the train set, fined tune on the dev set and measured the final performance on the test set.

1.3 LSTM

LSTM, which stands for long short term memories, was proposed by Hochreiter and Schmidhuber in 1997. The model is a special type of RNN and designed to model sequences. Unlike traditional RNNs, LSTM is using input gate, output gate and forget gate in order to allow long-distance dependencies to be remembered when necessary and to handle the vanishing/exploding gradient problem often occurring on RNN. it’s very popular nowadays and usually works great.

1.4 BERT

BERT is an NLP deep learning model that was published by Google in late 2018. The model was pre-trained on a huge amount of text from Wikipedia to provide a richer understanding of language. It can be fine tuned to produce state of the art results in many common NLP tasks. The architecture of BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformer’s layers where every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.

1.5 Related work

Toward Dimensional Emotion Detection from Categorical Emotion Annotations
https://arxiv.org/pdf/1911.02499.pdf

2 Solution

2.1 General

In order to evaluate our models, we created two separate Google Collab notebooks. One for the LSTM and another for BERT. We used huggingface’s NLP library called Transformers and PyTorch library. The code was written in Python. The training time we specify assumes that the code runs on GPU Tesla 4, as we got from Google.
EmoBank is a regression task, therefore we used Mean Squared Error loss function to measure the error of our distance from the actual V,A,D values of the sentences.

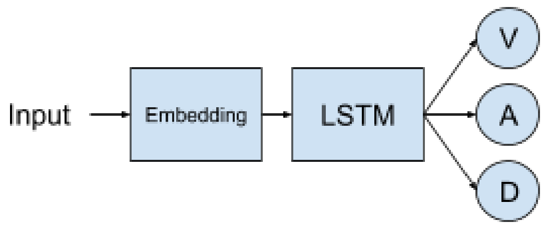
2.2 Baseline

In order to evaluate the contribution of our model's complexity to the results, we have created a simple model that predicts the result (3,3,3) to any sentence (which should be quite accurate when looking at the distributions shown above).
This model's score was 0.075 on the dev set and 0.076 on the test set.

2.3 LSTM

We tokenized the sentences using spacy basic english tokenizer and encoded them using a similar vocabulary we used in class.
Using pytorch, we built an LSTM model with the following architecture:

- Embedding
- LSTM
- Linear to 3 outputs (V,A,D)



In order to best evaluate the LSTM ability to tackle this problem we tuned the following hyper parameters:

- Number of epochs: 1-9
- Learning Rate (Adam): 0.0001, 0.001, 0.01
- LSTM hidden size: 500, 800
- LSTM layers: 1, 2, 3
- LSTM number of directions: 1, 2
- External or internal embeddings

hyper parameters we didn't explore yet, but think that needs further exploration:

- Batch size
- Optimizer
- Other external embeddings

One of the key challenges that we faced was a very long hyper parameter tuning running time (eventually ~14 hours) and reoccurring environment disconnections due to lack of GPU quota from Google.
In order to solve that, we saved each configuration result to a file so on the next run we can skip it and continue to the next configuration. Furthermore, we used multiple google accounts in order to get more GPU quota.

When looking at the results, the best model was trained using the following hyperparameters:

Learning rate (Adam)	0.0001
LSTM hidden size	500
LSTM layers	3
LSTM num of directions:	2
Number of epochs	3
Embeddings	No

This model's score was 0.065 on the dev set and 0.069 on the test set.

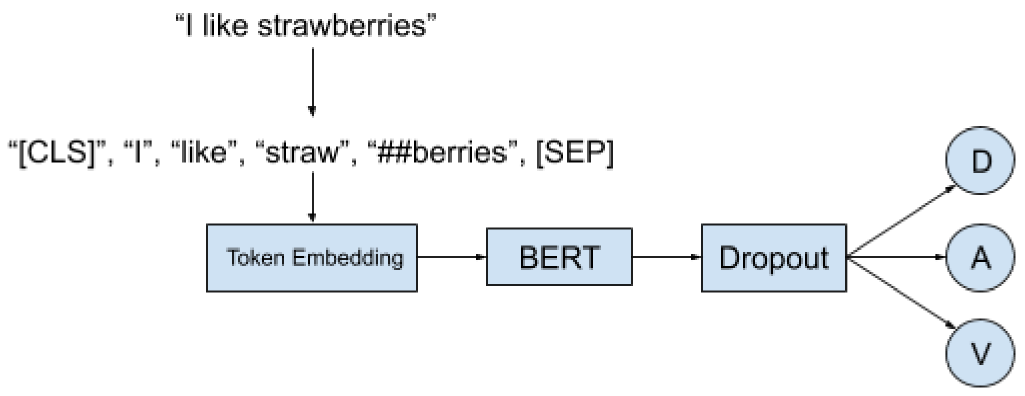
2.4 BERT

First, we transformed our dataset into the format that BERT can be trained on.
We used BERT tokenizer in order to: and then mapped the tokens to their index in the tokenizer vocabulary.

1. Split the sentences into tokens using a method called WordPiece tokenization.
2. Add special [CLS] and [Sep] tokens.
3. Pad all sentences to the same length using [PAD] tokens.
4. Create the attention mask which explicitly differentiate real tokens from [PAD] tokens.
5. Map the tokens to their IDs according to the Token Embeddings layer that include a vocabulary which is 30522 words in size.

The result is a 768-dimensional vector for each token. This is the input representation that is passed to the BERT model.

Second, we created a BERT model with a dropout layer in the middle and a linear layer on top with 3 outputs.
Google released a few variations of BERT models. Our model is based on a variant called "bert-base-uncased". It's the smaller of two available sizes and ignores casing.
Here is a diagram describing the BERT process:



Third, we trained and fine tuned our model on the EmoBank task. The hyper parameters we examined are:

- Batch Size: 16, 32
- Learning rate (Adam): 5e-5, 3e-5, 2e-5
- Number of epochs: 1-9

One of the challenges we faced is that the Huggingface library didn't implement a model designed for a regression task with 3 outputs (Unlike for a classification task). Therefore, we were required to build a new model based on BERT with a linear layer on top with 3 outputs.

Each combination of hyperparameters took about 24 minutes to train on all 9 epochs. So in total it took 2 hours and 40 minutes to execute all the experience.
There is a technique called Smart Batching we didn't explore yet. This technique could dramatically reduce BERT's training time by creating batches of samples with different sequence lengths.

When looked at the result the best model was trained using the following hyperparameters:

Batch Size	32
Learning rate (Adam)	2e-5
Number of epochs	7

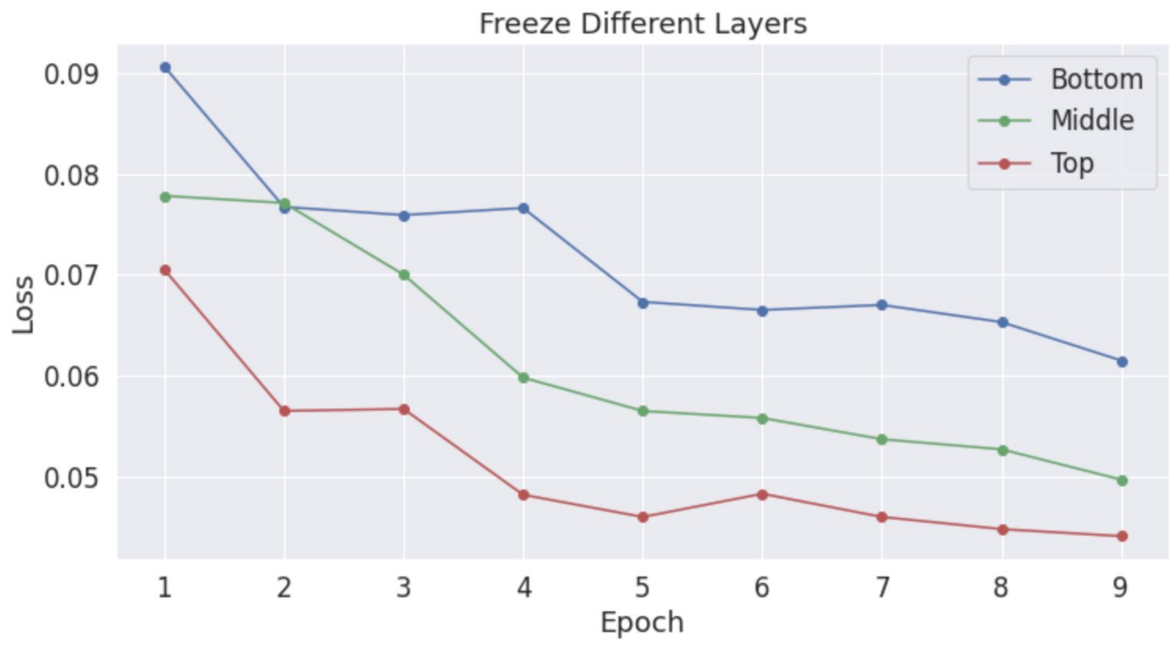
This model's score was 0.042 on the dev set and 0.043 on the test set.

In addition, we checked results of the above hyperparameters on the BERT variant that doesn't ignore casing ("bert-base-cased").
The model's score was 0.052 on dev set and 0.050 on test set.
Put attention that the results of the cased model are worse than the uncased model. In the beginning, we were surprised by the results because we expected the opposite. We expected that, because there are a lot of meanings to capital letters. For example, it indicates names of people, countries and even expressions. Therefore, we thought it might make our model to learn more complex and useful patterns in the data.
After we saw the results and discussed it, we assumed that we got less good results because we trained our model on a small dataset (8062 sentences) and the fact that we allowed capital letters made the variety of different sentences even bigger. So there is not enough support for our model to learn complex patterns and even the simple patterns have less support right now, So we got less good results.

Finally, with the same best hyperparameters we tested the impact on the performance of freezing different layers in the BERT model. BERT is built from one Embedding layer and twelve Transformer layers. We made 3 options:

1. We freezed all the Transformer layers, but the first two.
2. We freezed all the Transformer layers, but two in the middle
3. We freezed all the Transformer layers, but the last two.

We demonstrated the results in the following graph. The options 1, 2 and 3 represented by the blue, green and red line respectively. The x axis is the number of epochs and the y axis is the validation loss.



We can see that the **blue line has the highest validation loss and the red line has the lowest validation loss**. The results reinforced the hypothesis that the upper layers of deep neural networks have more impact on complex features like emotions.

3 Discussion

We made a comprehensive comparison between LSTM (a classic deep learning model) and BERT (a state of the art mode) by training each model on the EmoBank dataset. We fine tuned a variety of hyperparameters to get the best results.
The BERT model's performance was significantly better than the LSTM model. The MSE on the testing set of BERT was 0.043 vs. 0.069 of LSTM. We still believe there is still room for further exploration around hyperparameters we haven't explored, as described above, in order to get even better results. The baseline MSE was 0.076, which was not surprising based on the data distributions we demonstrated.
In addition, when comparing case sensitive and non-case sensitive BERT models, we have found that the non-case sensitive model outperformed the case sensitive one. Our assumption is that this was due to lack of support when treating capitalized and non-capitalized words.
Finally, we tested the impact on the performance of freezing different layers in the BERT model. The result of the third option where all Transformer layers were freezed but the last two was the best. By this we have reinforced the hypothesis that the upper layers of deep neural networks have more impact on complex features like emotions.
We hope that future researchers and companies will find our work beneficial and help them focus more on state of the art solutions instead of traditional methods.

4 Code

The link to the EmoBank dataset:
https://github.com/JULIELab/EmoBank

The link to the LSTM and Baseline Model project:
https://colab.research.google.com/drive/1ieAxM-7oS2Jbh8iBstToNRsKyIUV5Rg_?usp=sharing

The link to the BERT project:
https://colab.research.google.com/drive/1fFTd0Ql_oppAbz9nS9yviIKTFNqo4Gal?usp=sharing

5 References:

- EMOBANK: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis
  https://www.aclweb.org/anthology/E17-2092.pdf

- BERT Word Embedding Tutorial
  https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/

- PyTorch Documentation
  https://pytorch.org/

- Huggingface Transformers
  https://huggingface.co/transformers/