# Identifying emotions in the Hebrew language

Naor Sasi 302727052

2023 May 28

## 1 Introduction

Natural language processing (NLP) is widely used in many fields such as sentiment analysis, translation, etc.
Emotion classification is one of the most challenging NLP tasks due to the subjective nature of emotions and the difficulty of identifying emotions from textual data. Identifying these emotions can show expected behaviors, mental states, etc. in which people are. In this work I will examine the emotions (expectation, happiness, trust, fear, surprise, sadness, disgust, anger) in the Hebrew language using AlephBERT.
For the purpose of the analysis I used 3 datasets. One of a data set for training (comments published in ynet news articles, Israel Hayom, and in rooms related to Corona). The second is a test containing a random set of notes taken from the dataset (notes related to covid). The third examines a random group of comments published in response to articles not related to Corona but from the same news sites.

## 2 Challenge

At first I analyzed the array and realized that an unbalanced array has multiple emotions for each example. There are emotions with less than 200 examples (which is very low for the classification tasks of the machine) and there are some emotions around 400 examples and there is an emotion with over 1000 examples. However I continue to see the accuracy can bring from this data set. In addition I will check on a data set where every example has a single emotion.

## 3 Access

I started with the text processing that will be customized: downloading duplicate data and columns that I don't use (by the DATA-NLP notebook).
I divide the training into 2. One is learning about the data after the text has been personalized and using the AlephBERT tokenizer. The second is a study

of data after personalization and that we have downloaded links, website addresses, special characters and stop-words.
For the classification I use:
Logistic Regression - simple basic learning along with finding the best hyperparameters by GridSearch.
Random Forest, Linear Support Vector classification (Linear SVC) and stacked them together using StackingClassifier.
Deep learning - learning using the AlephBERT model without direction and with subtle direction.

## 4 Results

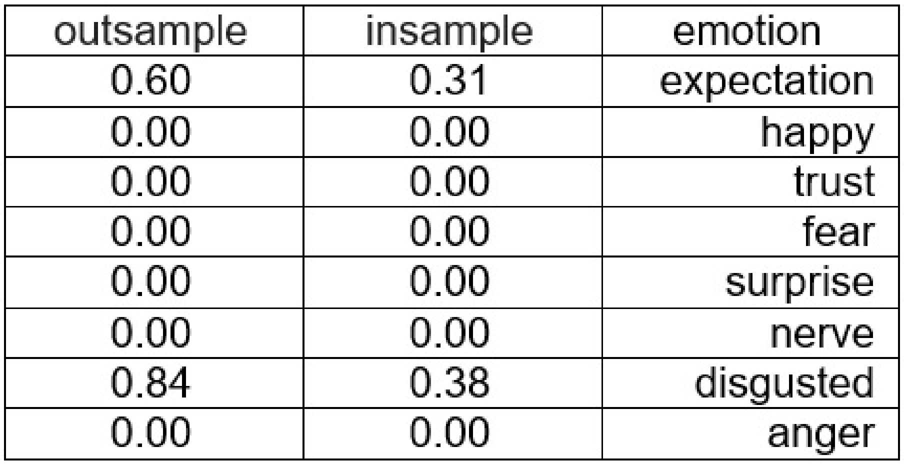
The results of the sentence-based tasks of sentiment analysis:
Model quality table according to the F1-score index on the data after the text has been customized and AlephBERT's tokenizer has been used.

| | Logistic Regression | +Random Forest Linear SVC | AlephBERT Without additional learning layers | AlephBERT With the addition of learning layers |
|---|---|---|---|---|
| Train | 0.6 | 0.53 | 0.99 | 0.94 |
| Val | 0.45 | 0.44 | 0.42 | 0.42 |
| Test insample | 0.27 | 0.27 | 0.29 | 0.33 |
| Test outsample | 0.62 | 0.62 | 0.71 | 0.66 |

Model quality table according to the F1-score index on the data after the text has been personalized and we have downloaded links, website addresses, special characters and stop-words.

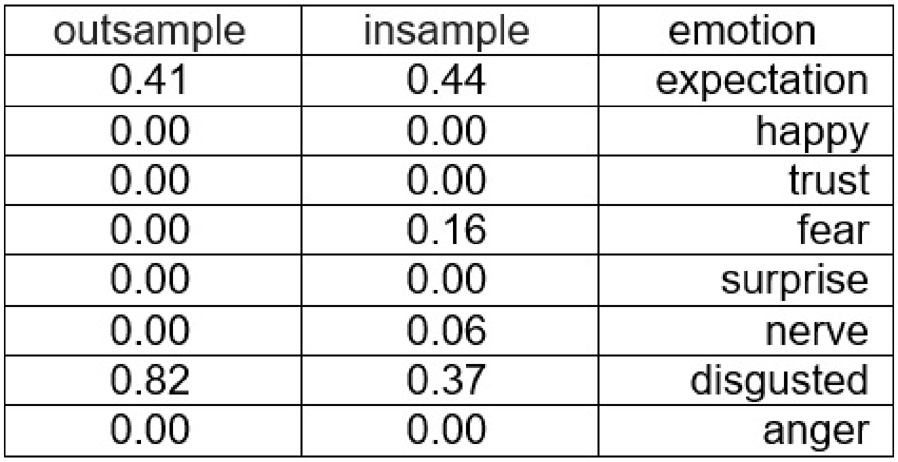
| | Logistic Regression | +Random Forest Linear SVC | AlephBERT Without additional learning layers | AlephBERT With the addition of learning layers |
|---|---|---|---|---|
| Train | 0.59 | 0.52 | 0.4 | 0.4 |
| Val | 0.44 | 0.41 | 0.4 | 0.4 |
| Test insample | 0.23 | 0.05 | 0.22 | 0.22 |
| Test outsample | 0.61 | 0.01 | 0.67 | 0.67 |

The deep-learned AlephBERT-based classifier produced better accuracy than logistic regression and random forest with a linear support vector classifier.
AlephBERT without additional learning layers according to the F1-score index.



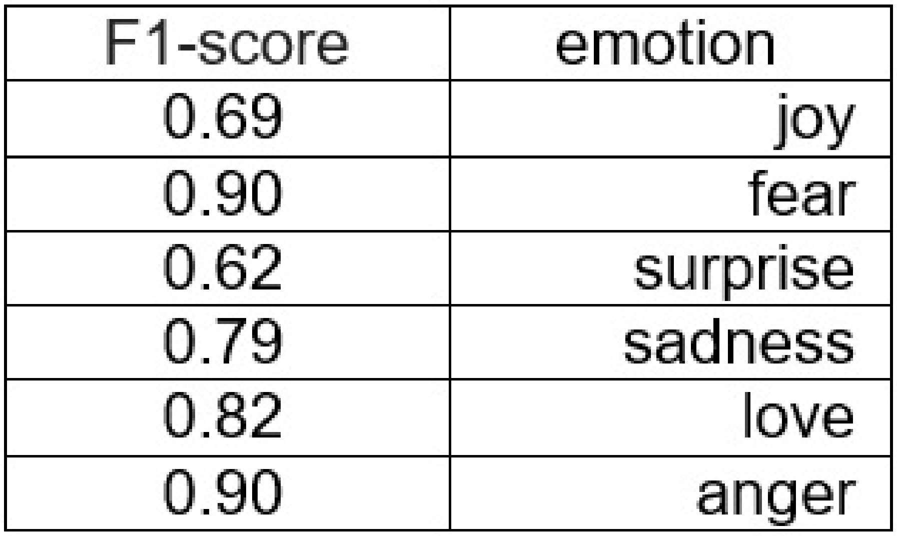
| outsample | insample | emotion |
|---|---|---|
| 0.60 | 0.31 | expectation |
| 0.00 | 0.00 | happy |
| 0.00 | 0.00 | trust |
| 0.00 | 0.00 | fear |
| 0.00 | 0.00 | surprise |
| 0.00 | 0.00 | nerve |
| 0.84 | 0.38 | disgusted |
| 0.00 | 0.00 | anger |

AlephBERT with added learning layers according to the F1-score index.



| outsample | insample | emotion |
|---|---|---|
| 0.41 | 0.44 | expectation |
| 0.00 | 0.00 | happy |
| 0.00 | 0.00 | trust |
| 0.00 | 0.16 | fear |
| 0.00 | 0.00 | surprise |
| 0.00 | 0.06 | nerve |
| 0.82 | 0.37 | disgusted |
| 0.00 | 0.00 | anger |

AlephBERT that each example has a single emotion (NLP$_{N}EWfile$).



| F1-score | emotion |
|---|---|
| 0.69 | joy |
| 0.90 | fear |
| 0.62 | surprise |
| 0.79 | sadness |
| 0.82 | love |
| 0.90 | anger |

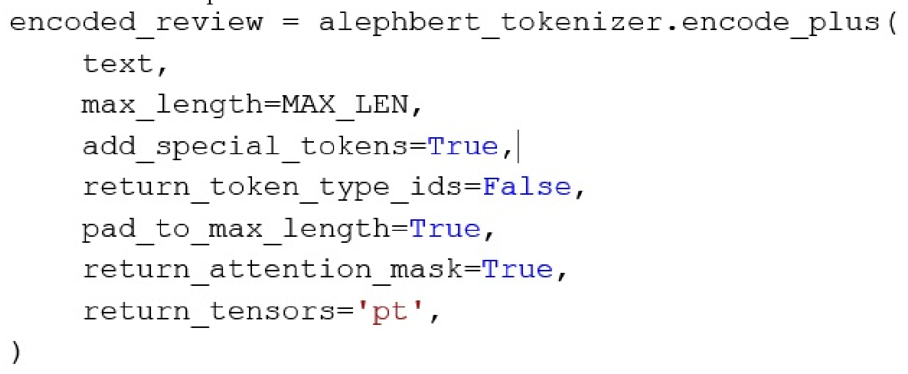
## 5 Conclusion

Due to the cleanliness of the information and that each sentence can have several emotions, the information is unbalanced.
The learning was performed on one emotion per sentence which was tested without good success.
I was not able to identify the emotions with the reduced number of examples in learning without additional layers.
On the other hand, with the addition of layers, we were able to identify 2 more emotions, but in total, the identification was not successful.
A test on information with explicit sentences for a single emotion yielded good results.
It is necessary to analyze the sentences with several labels.

## 6 Details about the work

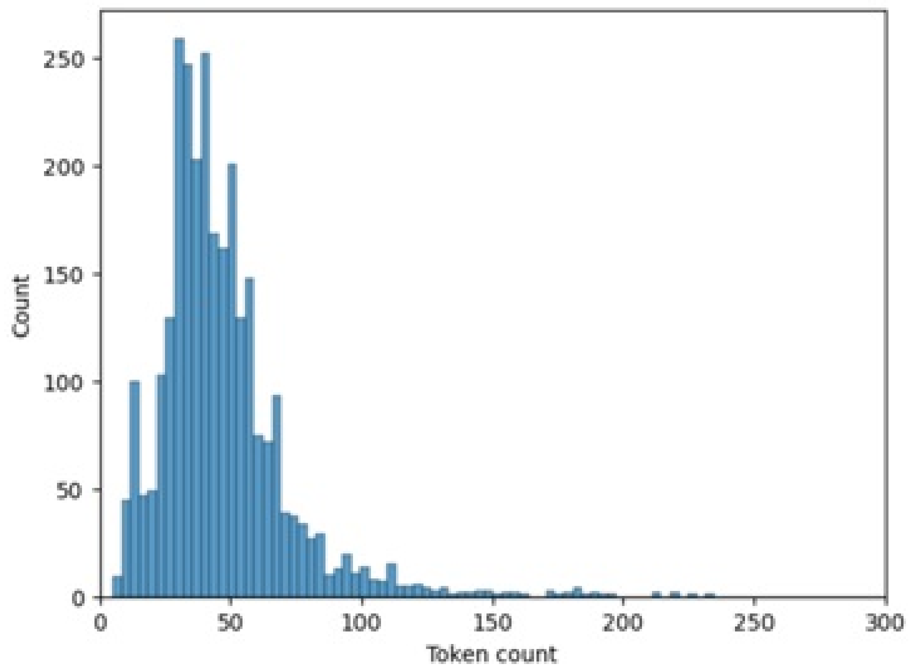
The Transformers library provides a wide variety of Transformer models including AlephBERT.
It works with TensorFlow and PyTorch It also includes pre-built tokenizers that do the tasks for me.
I used the basic AlephBERT and built on top of sentiment classification.
Loading the model: alephbert = BertModel.from_pretrained('onlplab/alephbert-base')
AlephBERT requirements:

Add special tokens to separate sentences, pass sequences of fixed length (insert padding), an array of 0 (surface token) and 1 (real token) called attention mask.
Loading a pre-trained BertTokenizer.
Use this to convert text to tokens and then from tokens to unique integers (IDS).
alephbert_tokenizer = BertTokenizerFast.from_pretrained('onlplab/alephbert-base')
There are special tokens:
[SEP] – marking for the end of a sentence.
[CLS] - Added this token to the beginning of each sentence, so AlephBERT knows I'm doing a classification.
[PAD] – the marking is special for padding.
[UNK] - AlephBERT understands Tokens that were in the training set. Everything else I coded unknown.
I used the encode_plus method which does all this work.



```
encoded_review = alephbert_tokenizer.encode_plus(
      text,
      max_length=MAX_LEN,
      add_special_tokens=True,
      return_token_type_ids=False,
      pad_to_max_length=True,
      return_attention_mask=True,
      return_tensors='pt',
)
```

You can see that most of the reviews contain less than 80 Tokens but I chose 200 to be the safest.



GPReviewDataset – Creating a PyTorch dataset.
SentimentClassifer - I used it in a sentiment analysis model and transferred it to sentiment analysis with small changes. This is the creation of a basic model used in AlephBERT.
EmotionClassifier - a model for recognizing emotions with added layers.
.to(device) transfer to gpu.
device = torch.device(”cuda:0” if torch.cuda.is$_{a}vailable()else”cpu”$)
$AdamW–Optimizerwithconstantweightlossthatcanbeusedtofine–tunemodels.$
$Train_{e}poch–trainingourmodelforoneepoch.$
$Evel_{m}odel–modelevaluation.$

## 7 Sources

The information I used for training and checking the quality of the model:
https://github.com/avichaychriqui/HeBERT/blob/main/data.zip
https://huggingface.co/onlplab/alephbert-base
https://arxiv.org/abs/2104.04052