# Depression Detection via Emotion Timeline

Netanel Tamir (ID. 204900062), Shaltiel Shmidman (ID. 206464182)

Submitted as final project report for Practical Topics in Machine Learning, BIU, 2021

## 1 Introduction

Depression is a common mental and behavioural disorder which is estimated to affect over 300 million people worldwide (meaning it affects 1 out of every 22 people, 4.4% of the global population). It serves as a leading cause of disability worldwide and increases the risk of contacting other illnesses such as dementia.

The social stigma around mental illnesses is that it differs from other illnesses and one can overcome them by simply deciding to. This is a major contributing factor to the lack of treatment for those inflicted by such illnesses. In fact, due to this single factor, more than half of the people afflicted by such illnesses do not receive help with their disorders.[1]

The ability to diagnose a person with depression in an external way (e.g. online posts written by the individual) can be utilized in diagnosing this illness in the general population, and by doing so we can make sure more people get the help they need.

The problem of assessing a mental illness from a series of posts of a person in and of itself is quite interesting as there are numerous ways to tackle such a problem. We will show our method in the creation of our data set (based off of SMHD[1]) and in the different models which we will use to solve the problem.

### 1.1 Related Works

There have been multiple attempts at tackling the problem of assessing a certain mental illness throughout the years via text, and a myriad of different techniques have been utilized in these attempts. For example, in this Stanford paper, Detecting Depression Through Tweets[3], the authors created their own data set and compared the accuracy of different models, with a word-based GRU and a word-based CNN being the most accurate. Another Stanford paper Identifying Depression on Social Media[2], utilized many different models such as SVM,

[1]https://en.wikipedia.org/wiki/Depression_(mood)

Logistic Regression, and Neural Networks to identify depression in an individual based on their Reddit comments. The author found that a CNN achieved the best accuracy of all the models tested.

## 2 Solution

### 2.1 General approach

Our approach was to try and flag people that might suffer from depression, by looking solely at how the person feels over time. By doing so, we can provide a practical tool for making diagnosing depression more accessible. Without any tool to help with this task, the system would be overrun trying to identify people with depression, when only 1 out of 22 people are positive for depression. We hope that our tool can narrow down the list of people to a more manageable number whilst maintaining a high recall as to not miss too many cases. In addition, it is quite simple to have a given person log every day how they feel, making this process very practical and scalable.

### 2.2 Data

We began by creating our data set from the SMHD[1] data set we were given. This data-set is a collection of Reddit users, where for each user we have a list of their textual posts on Reddit as well as a label marking their professionally diagnosed mental illness (or if they are a control user). We created a Python script to parse said data, and then convert the textual posts into emotions. Our script uses a model from hugging face, an open-source provider of NLP technologies, to recognize emotion from text[2]. Using this model, each post was converted into one of 6 distinct emotions: Surprise, Sadness, Joy, Love, Anger, Fear. To narrow down the data-set for our task, we removed all examples except those labeled "control" or "depression" (we only looked at single class diagnoses), because in this project we are focused solely on the classification of depression. In addition, to reduce noise in our data, we removed posts that had fewer than 5 characters, and cut off each post at 200 characters. Moreover, because there are many more instances that are labeled "control" than "depression", we uniformly selected only 5% of the control instances. Lastly, due to limited computational power, we only took a sub sample of the posts of each user, when enough posts existed, because of the large amount of posts per user.

After all the work on the original data set, we were left with the following number of instances per label in the train, dev, and test files:

Train: 1,316 depression, 4,427 control.

Dev: 1,308 depression, 4,426 control.

Test: 1,316 depression, 4,431 control.

[2]https://huggingface.co/mrm8488/t5-base-finetuned-emotion

As seen above, each of the files has approximately the same amount of instances, like in SMHD[1]. Also, the ratio between the number of instances per label is kept between the different files.

### 2.3 Implementation

In our implementation, we created a lookup table of size 6 for each emotion, where each row is an embedding of size 20. We then tried 3 different methods for encoding the emotion timeline, followed by a Multi-Layer Perception (MLP) with an output dimension of size 2 - predicting whether the user might have depression or not. We tried 3 different methods for encoding the timeline: Concatenation, CNN, and LSTM.

Concatenation - The most basic approach is to simply concatenate the given timeline into a single vector and then run it through the classifier. Since each user has a different number of time points in their timeline, we had to apply a method of converting each timeline into a constant size vector. Therefore, we split the data points evenly into 10 groups (in chronological order, so that the first group has the first X/10 points, etc.) and then did a weighted average of the different emotions in each group. We then concatenated the 10 groups together and ran that through the MLP.

CNN - The second approach is a more advanced method for dealing with sequences of varying lengths - Convolutional Neural Networks. Using a given window-size, the CNN travels across the timeline producing another vector summarizing the sequence, and then we max-pool the result vector to get the most meaningful value from each dimension. We run multiple CNN's of varying lengths (also known as multiple filters with different kernel-sizes within a CNN), and concatenated the results together, resulting in a constant size encoding. We then feed this final vector into the MLP.

LSTM - Long Short-Term Memory[4] - A type of recurrent neural network capable of learning order dependence in sequence prediction problems. This type of network works well with a varying length sequences, and is capable of modeling long-term dependencies. A LSTM seems to us like the appropriate solution to our problem, and it will be interesting to see how the other models perform relative to it.

### 2.4 Design

Our code was written in a Google Colab notebook, using pytorch as the main ML library. We trained the models until they stopped improving, with a maximum of 40 epochs. Like we previously mentioned, due to the lack of computational power at our disposal, the data set wasn't large enough to create models that were as accurate as we wanted.

All models used the ADAM optimization function and a Cross-Entropy loss function. Additionally, all models are built of the same building blocks: Sequence of emotion embeddings, an encoder, and a neural network consisting of 2 hidden layers, an output layer of size 2, and a tanh activation function between the layers.

Training time was approximately 10 min using the Concatenation encoder, approximately 25 min with the CNN, and significantly longer with the LSTM.

We encountered many difficulties during the process, many of which were mentioned throughout the paper including how we fixed it. The main issue was the limited computational power and long run-time, which we solved by sampling the data and defining our task strictly to have less noise in the data-set. Another implementation issue we encountered was with looking up the embeddings for the emotions in the concatenation encoder. In pytorch (and in Linear Algebra) tensors are required to have constant size in each dimension. Meaning, we can't have a matrix where the first row is of size 5, and the second row is of size 10. In the concatenation encoder we group the emotions into 10 groups, and in any case where the number of emotions isn't divisible by 10, we ended up with one group that has a different number of elements than the rest, resulting in an illegal tensor. We solved this problem with adding filler values to the final group for it to be equal to the rest. The filler values got their own embedding in the lookup table and were fine-tuned accordingly.

## 3 Experimental results

Our data set was evenly split between train, dev, and test sets like in the original data set. In terms of metrics, we focused mainly on recall and precision as opposed to accuracy. As described in our general approach, our goal was to maximize recall while still having a decent precision. This will allow to catch most of those with depression, and send them to professional analysis.

We trained our network using each of the three encoders, and watched the loss progress over the epoch. The results are presented in Figures 1 and 2. As seen here, it seems like the first method of concatenation overfit to the data very fast. The loss on the training data decreases throughout the entire training process, whereas the loss on the dev data starts off decreasing and then increases and mostly stays there. Meaning, the network was learning the training data very well, but wasn't generalizing well to the dev data. On the other hand, the CNN and LSTM didn't seem to overfit, and their loss graphs look similar in training and in dev. They both seemed to perform similar, with the LSTM slightly in the lead.

We now move on to the more interesting metric, which is the precision and recall. Our network produces a vector of size 2, where it predicts the probability of the given timeline indicating depression or not. In order to reach our goal of maximum recall, we didn't just take the higher probability, but rather tested multiple thresholds, determining that our prediction is depression if the depression probability is above the given threshold. We tested thresholds across

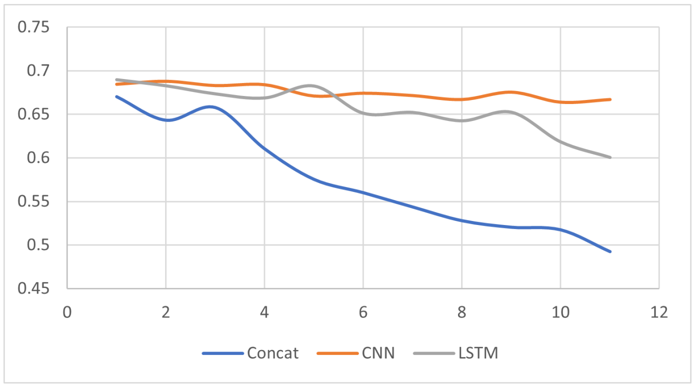

Figure 1: The average loss value on the training set, as a function of the number of epochs trained.

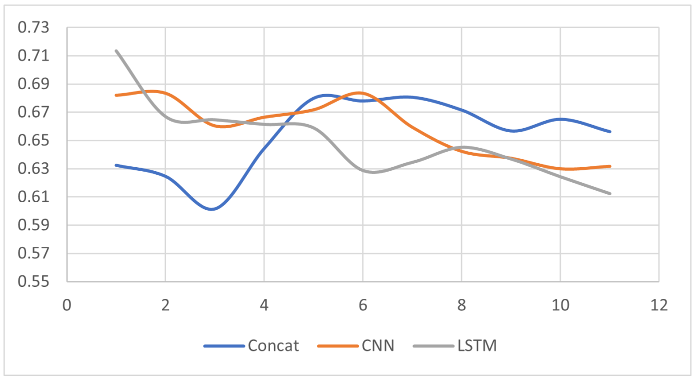

Figure 2: The average loss value on the dev set, as a function of the number of epochs trained.

the distribution, from 10% to 90%. As our threshold got higher, we saw the precision improve and the recall decline, and can be seen in Figure 3. This was as expected, the model is more sure of it's choice so results in fewer mistakes, but it misses any case where it isn't very confident.

In order to choose the ideal threshold, we manually looked at the precision-recall graph of our model on the dev set, and decided based on that the right threshold. We chose a different threshold for each of the three encoders, as they learn and fine-tune differently. After choosing a threshold, we then report the results of the model on the test-set with that threshold. All three models were trained for 10 epochs on the same data. The chosen thresholds and results on the test-set can be seen in Table 1.

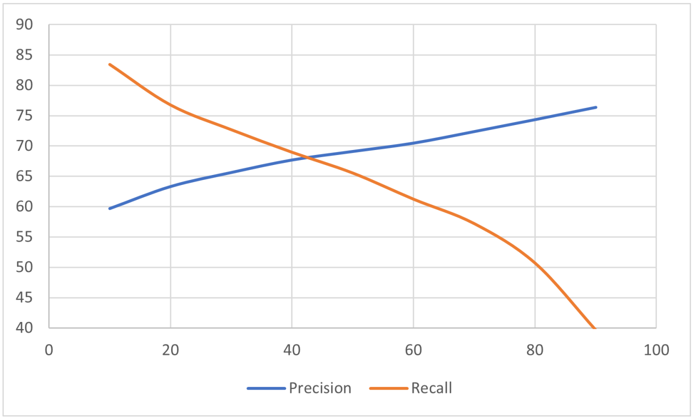

Figure 3: Plotting the precision and recall of our model as a function of the threshold used for determining our classification. The results are on the training set, using the LSTM encoder, after 11 epochs.

| | **Threshold** | **Precision** | **Recall** |
|---|---|---|---|
| **Concatenation** | 0.1 (10%) | 27.41% | 47.14% |
| **CNN** | 0.4 (40%) | 28.44% | 55.72% |
| **LSTM** | 0.1 (10%) | 26.6% | 69.44% |

Table 1: For each of the three encoding methods we experimented with, we list the threshold we chose after 10 epochs, and the results on the test set with that threshold.

## 4 Discussion

The results from our experiments showed that the LSTM worked the best out of the 3 models we tested. That model succeeded the best at maximizing the recall while also keeping a decent precision percentage.

We encountered several difficulties while working on this project, many of which were mentioned previously in the paper. For example, we had to deal with the limited computational power at out disposal when creating our data set from the SMHD[1] data set we were given. That resulted in results that were less than ideal. However our methods seem to work and our models did learn. We believe that more data would improve our results.

In order to help us figure out why one method succeeded better than the other, we tried to visualize our training data. We split our emotion categories into positive (surprise, joy, love) and negative (sadness, anger, fear) emotions. We then sampled two depression and two control examples, and graphed the moving average of their polarized emotion, where a positive emotion is 2 and a negative emotion is 1. The graph can be seen in Figure 4. When viewing these examples, we were able to figure out ourselves without looking at the labels which were the depression cases and which were the control cases. If we tried to express what indicated to us the classification of the case, we realized that the depression cases seem to lean towards the negative emotion, and contain streaks of negativity in a row. These of course are just general indication, and are quite hard to write down as rules. Therefore, our hope was that our models would be able to learn how to classify them correctly.

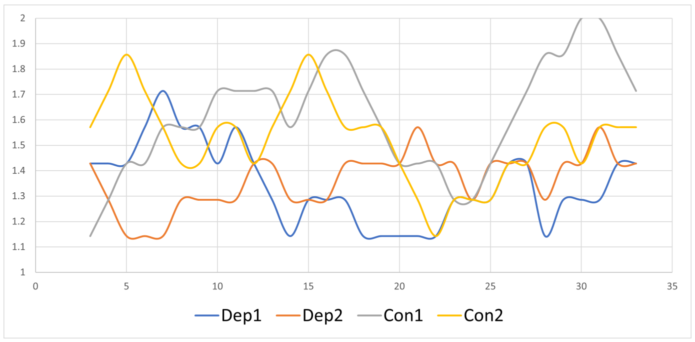

Figure 4: Four sampled examples from our data-set, polarized and placed onto a timeline for visual aid

When viewing our results, we saw that the concatenation method didn't do so well. We theorize that this might be because it was wasn't able to visualize the timeline and see the streaks of negative emotions, but it didn't fail completely possibly because it was able to tell that the person was overall unhappy over time, which is another tell that we as people noticed.

The second method we tried, the CNN, provided much better results. The CNN seems to have more power over the sequences (as expected) and was therefore able to detect any long streaks of sadness, but it was limited to a window each time, and therefore had a hard time finding repeating patterns.

Finally, the LSTM had the best results as we predicted. The LSTM was able to capture the full sequence whilst maintaining knowledge of the general feel of the person over time. These two factors most likely contributed to the LSTM producing the best performance.

To summarize, our results haven't completely solved the problem, but we think that it definitely provided an opening for further research and for practical application. With larger resources we could multiply the size of the data set significantly, and likely achieve more promising results. In addition, even with our results, this research has potential for real life practical use. If people would just log how they are feeling once a day, we would be able to run that through our network, and alert the users that are potentially depressed to go get checked out clinically. This would have far reaching implication if distributed properly - we would be able to identify most depression cases early on, with relatively few false positives on the way.

## 5 Future Work

Of course, our work is far from over. There are many avenues for us to explore in expanding our research. These avenues are split into two categories. The first category is how to improve our data, the main three extra elements we would like to encode in our data in the future are:

[1] Using alternate methods for classifying the emotion of a given text.

[2] Using the actual date+time of the posts in the sequence to provide more context for the network (i.e., to allow the network to know when each post was posted to learn important things such as how often a day the person posted, when were there long gaps, etc.)

[3] Treat different textual elements differently (posts, comments, etc.).

The second category is ways to improve on our experiments and evaluations. We would like to be able to detect a sub sequence of emotions that represent depression in a person across a timeline. In addition, we would like to juxtapose the results of our models with input data of a person's emotions diagnosed by a trained professional as opposed to a self diagnosis.

## 6 Code

The following link directs to a Google Drive folder with the train, dev, and test files, along with the python script used for creating them from the SMHD data set, and the colab file with the entire code for this project:

https://drive.google.com/drive/folders/1DgFelY4TFmnS9NkMZ3OUQXvLCP7DTyZx

## References

[1] Arman Cohan et al. *SMHD: A Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions*. 2018. arXiv: 1806.05258 [cs.CL].

[2] Kali Cornn. *Identifying Depression on Social Media*. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15712307.pdf.

[3] Aileen Wang Diveesh Shingh. *Detecting Depression Through Tweets*. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6879557.pdf.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.