Automatic Psychological Text Analysis using Recurrent Neural Networks

Suo Xian Zhang[1] , **Merijn Bruijnes**[1] , **Willem-Paul Brinkman**[1]
[1]TU Delft

Abstract

Schema therapy is a type of psychological treatment for people suffering from personality disorders. A schema is a core psychological state of mind that influences external behaviour through the development of coping styles. Current schema therapy is human-processed and therefore time inefficient. Enabling automatic schema classification helps the overall goal of creating a chatbot that can classify schema modes from conversations. The goal of this research was to optimize a Recurrent Neural Network (RNN) model to classify patient's schema modes from a dataset containing a recent emotional story from participants and are labeled with SMI questionnaire answers. Three RNN models were created: a binary classification Multilabel RNN, a binary classification Per-Schema RNN and a ordinal classification Per-Schema RNN. The results have shown that the binary classification Multilabel model scores an average F1-score of 0.48. The binary classification Per-Schema model scores an average F1-score of 0.49. While ordinal classification Per-Schema model performs with an average Spearman Coefficient of 0.15.

1 Introduction

A schema is a core psychological state of mind that influence external behaviour through the development of coping styles, which are strategical responses to certain schemas [1]. Coping styles are behaviour based and part of response, whereas schemas contain memories, emotions, bodily sensations, and cognition. Schema modes represent the momentary emotional and cognitive states and coping responses that are active at a given point in time. Emotional events can trigger these schema modes. Additionally, an individual may shift from one mode into another.

Schema therapy, introduced by Young in 2003 [1], is a type of psychological treatment for people suffering from personality disorders. The goal of this treatment is to help patients deal with the unhealthy memories that create these schemas and replace them with healthier patterns of thought and behaviour [2]. However, there are some downsides to the current schema therapy treatment.

First, the current schema therapy is time expensive. Schema therapy requires the therapist to assess the patient's schema modes. To assess the patient's schema modes Short Schema Mode Index (SMI) is used. SMI is a 118 item questionnaire, which is scored using a 6-point scale ranging from 'never or hardly ever' to 'always'. The 118 items on the questionnaire relate to 1 of the 14 schema modes [2]. The SMI questionnaire takes 40 minutes to fill. In addition to the questionnaire, 3-6 sessions between the patient and the therapist are necessary to establish the complete schema model. In other words, schema therapy is time expensive.

Second, as the SMI questionnaire takes long, it is generally taken once. The SMI questionnaire tries to capture the schema modes for a period of time. Therefore, the results are a single static measurement of what is actually a dynamic system of schema modes [3].

Third, it has been found that the technological state of e-mental health systems is limited. Technology is often used as a platform to deliver information to the patient or to the therapist. When a patient is asked to provide textual information to the system, this information is usually processed by a human, by a guided system, or not at all. Human processing is costly and offers no advantage over traditional paper-based workbooks [4].

Automatically analyzing textual information and classifying schema modes can help therapists by efficiently finding the right treatment for patients. This is a step towards more affordable and efficient mental healthcare.

This research should support the overarching goal [1] to create a chatbot that holds conversations with patients. The chatbot should be able to detect the schemas for a patient from the conversation and suggest appropriate treatment options when an unhealthy schema mode is detected.

1.1 Related work

Natural language processing for cognitive therapy: extracting schemas from thought records [4] shows that preprocessed Natural Language Processing (NLP) can extract schema modes from thought records. The thought record utterances were manually labelled with schemas from the schema rubric developed by Millings and Carnelley [5] with some modifications. Burger used the Recurrent Neural Net-

[1]https://projectforum.tudelft.nl/course_editions/39/projects/973

works (RNN), Support Vector Machine (SVM) and k-Nearest Neighbors (KNN) classifiers for the schema classification. However, Burger did not aim for optimal classification as the purpose of her research was to explore the possibilities for automatically classifying mental health data.

Schema mode assessment through a conversational agent [3] used the questions of the Short Schema Mode Index (SMI) questionnaire for a conversational agent that interviews persons. The conversational agent asked for a recent emotional event, then followed with questions from the SMI questionnaire corresponding to the schema modes that the agent thinks are relevant. One of Allaart's conclusions stated that "a schema mode that was confirmed by the conversational agent was 5.20 times more likely to be confirmed by the SMI questionnaire too." However, Allaart mentions that the text analysis algorithm of his research is currently lacking.

Combating Depression in Students using an Intelligent ChatBot: A Cognitive Behavioral Therapy [6] used a chat-bot that asks persons a view questions and identifies the emotions of the persons to calculate the percentage of negativity in the chat. Finally, the mental status of the person is classified to normal, stressed or depressed. Patel used Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Hierarchical Attention Network (HAN). However, Patel also mentioned that the text classification methods could use accuracy.

Understanding the relationship between patient language and outcomes in internet-enabled cognitive behavioural therapy: A deep learning approach to automatic coding of session transcripts [7] The aim of this research was to determine the association between utterances and clinical outcomes by developing a deep learning model to automatically categorize patient utterances during text-based internet-enabled Cognitive Behavioural Therapy into one or more of five categories. The research concluded that the deep learning model provided an effective means of automatically classifying patient text utterances.

1.2 Research Question

The main question this research shall answer is:

How well can a schema be automatically classified from a text using RNN?

The following sub-questions should help tackle the research in a step-by-step manner.

1. What pre-processing steps on the data are necessary for an optimal classification by RNN?
2. What are the optimal hyper-parameters in the RNN model?
3. What are the differences between the results of three methods, RNN, KNN and SVM?

The research in the Related Work section support the possibility to create a chat-bot that can autonomously detect schema modes from users. However, a better classifier is needed [3] to be able to classify schema modes from conversations with a chatbot. In this research, a RNN classifier is used to classify unstructured emotional stories of users to detect their schemas. The classifier shall be trained, evaluated and optimized on the data. The RNN classifier shall classify patient schema modes from stories to 7 schemas from the SMI questionnaire [3].

Section 2 presents the Methodology, with background information on RNNs, word embeddings and the methods used. Section 3 presents the Experimental Setup and Results. Section 4 will discuss the results and compare them to research from the Related Work section. Section 5 offers Concluding Remarks with Limitations as well as Future Work. Lastly, in Section 6, I will reflect on the ethical aspects of my research and discuss the reproducibility of my methods.

2 Methodology

This section introduces background information on RNNs and the methods used to answer the research question. First, the background section is presented with a description of RNNs and Long-Short Term Memory (LSTM). Followed by an explanation of word embeddings. Second, the methods are introduced with an introduction to the dataset, pre-processing methods, RNN classifier and evaluation metrics.

2.1 Background

Recurrent Neural Networks

The task of classifying text utterances is also known as a Natural Language Processing (NLP) problem. Recurrent Neural Networks, or RNNs, were designed to work with NLP and sequence prediction problems [8]. RNNs in general have also received the most success in this area [9] [10].

A RNN is a classification model that uses a network containing recurrent connections that allow the network's hidden units to see its own previous output, and the resulting behavior can be influenced by previous responses. These recurrent connections is what gives the network memory [11]. RNNs must learn which past inputs have to be stored to produce the desired output. The current error signal has to "flow back in time" over feedback connections to past inputs for building up an adequate input storage[12].

However, RNNs have a downside. In the long-term the weights changes can become very small, so small that the weight does not change anymore. This phenomenon is therefore also known as the vanishing gradient . Long Short-Term Memory (LSTM) [13] can solve this problem.

Long Short-Term Memory

To avoid the vanishing gradient phenomenon, LSTMs are used for this research. They are particularly well-suited to natural language tasks because they can encode context and word order when they are important for the task [7] and LSTMs do not suffer from the vanishing gradient phenomenon.

The LSTM model is a type of RNN network with memory cells as filters to capture the relevant previous state information. While a normal RNN model only uses the input and hidden state, the LSTM model has a forget gate, input gate, output gate and cell state.

Word embedding

For the RNN to be able to process the data, the textual data needs to be transformed into something that the model

can understand. This way the deep learning model can learn hidden patterns in the textual data [14]. This can be done by transforming each word to a word embedding, that captures the semantics of the word in a general sense [7]. Word embeddings give us a way to use an efficient, dense representation in which similar words have a similar encoding. Word embeddings are created by first creating a vocabulary of the dataset. This is done by first mapping each word to their unique integer index. Lastly, each word index in the vocabulary is mapped to their word vector, resulting in a two-dimensional embedding matrix. Word vectors can be manually created and trained or pre-trained word vectors can be used. Afterwards, each word is mapped to their word vector.

2.2 Methods

Dataset

The dataset contains a recent emotional story from the participants and are labelled with the answers to related questions from the SMI questionnaire that the participants filled in afterwards. The participants answered the questions with a score on a 1-6 scale. The questions for the 7 schema modes: Angry, Detached, Happy, Healthy, Impulsive, Punishing and Vulnerable, were used. Each schema has 10 questions related to it. Besides Impulsive, which has 8 questions and Detached which as 9 questions. In total there were 67 questions. Therefore, each text utterance contains a total of 67 answers relating to the questions from the SMI questionnaire, as well as the binary label confirming the schema.

For this research the data is classified to the 7 schema modes. First, binary classification (does or does not reflect schema) will be performed. The reason for binary classification is motivated by the dataset, which contains binary labels for each schema. Afterwards, ordinal classification (0 - 3) will be performed. Where 0 stands for nothing to do with the schema and 3 stands for perfect fitting with the schema. For each text utterance, the questions for each schema are averaged and rounded up. This results in stories with 7 labels relating to the 7 schema modes.

The dataset is split as follows: 72% train, 8% validation and 20% test.

Pre-processing

Before the model classifies data, it is common to first preprocess the data. Pre-processing is one of the most important steps in any machine learning or deep learning task [14]. One of the first steps is to remove sentences in the dataset that add no value to the prediction of schema modes (e.g.: Good bye, stupid bot) and response words (e.g.: OK, yes, no, quit). Furthermore, the following pre-processing steps are applied as well: [15] [4]:

- Lower-casing
- Splitting of contractions, using the contractions package.
- Removal of stopwords, using the stopwords list from NLTK(Natural Language Toolkit).
- Removal of unnecessary white space.
- Lemmatization, using the WordNetLemmatizer from NLTK.

Word embedding

For this research project the word vectors that are used were pre-trained by Mikolov on 100 billion words of Google News. The word vectors were trained with the Continuous Bag-Of-Words (CBOW) model. Each vector has a dimensionality of 300. Due to the training of high dimensionality word vectors on a large amount of data, the resulting vectors can be used to notice subtle relationships between words [16]. In other words, the pre-trained word vectors are of high quality and will ensure maximum accuracy.

RNN classifier

Inspiration for the LSTM model has been taken from Burger's research [4]. The LSTM model for this research will have one input embedding layer, one bidirectional hidden layer, a dropout layer and one output layer. First, all models will be run with identical parameters, then Talos hyperparameter optimization will be run on the models to find the best parameters for the best results.

1) Embedding Layer The Embedding Layer can learn word embeddings for the input. However, it is quite inefficient. Therefore, it will be seeded with the embedding matrix and the word embeddings will not be updated. The layer will take as arguments: the input dimension, output dimension and the weights, input length of the input vectors and a trainable parameter.

2) Bidirectional LSTM Hidden Layer The LSTM Layer takes as an argument the number of nodes the layer should have. This layer transforms the inputs that are entered for the final classification with regard to the error rate [13].

3) Dropout Layer The Dropout Layer takes an argument between 0-1 and randomly drops the input units to 0 with the given frequency. This helps prevent overfitting.

Evaluation metrics

Binary For evaluation on the binary labels, the accuracy_score[2] method from scikit-learn is used. This function computes the overall effectiveness of a classifier: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true. Furthermore, the F1-score is used, which describes system performance using a scale from zero to one. F-score itself is derived from precision and recall. Precision describes the proportion of entities which a system returns that are correct. Recall describes the proportion of all entities that potentially should be found, that a given system actually returns [17]. For the F1-score calculation, the classificiation_report[3] method from scikit-learn is used. This report also returns: precision, recall, micro average, and macro average. Micro average will aggregate the contributions of all classes to compute the average metric, this means that the micro average takes a bias to more populated classes. Macro average will compute the metric independently for each class and then take the average, treating each class equally, meaning it biases the least populated classes.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[3]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

In addition, confusion matrices are used to visualise the false/true positives and false/true negatives.

Lastly, ROC-curve is used as well. ROC graphs are two-dimensional graphs in which true positive rate is plotted on the Y axis and false positive rate is plotted on the X axis. An ROC graph depicts relative tradeoffs between benefits (true positives) and costs (false positives) [18].

Both the ROC-curve and the F1-score are considered a standard nowadays [19].

Ordinal For evaluation on the ordinal labels the Spearman rank-order correlation coefficient is used. Spearman [20] rank correlation is a non-parametric statistic that allows an investigator to describe the strength of an association between two variables X and Y. The Spearman coefficient was picked as it was also used in Burger's research[4].

3 Experimental Setup and Results

This section describes the experimental setup, followed by the discussion of results. Experiments are performed to evaluate the RNN model for multi-label prediction of the schema classification problem on text utterances.

3.1 Dataset

The pre-processing step of removing sentences that added no value to the classification, resulted in a reduction of rows in the dataset. The dataset was reduced from 1793 rows to 1375.

3.2 Word Embedding

First, the vocabulary was created. The data was tokenized and padded. Resulting in a max vector of size 500. Therefore, all vectors were padded to a length of 500. In total there are 4553 unique words in the vocabulary.

The embedding matrix was created by mapping each word in the vocabulary to its word embedding. This resulted in an embedding matrix of shape: (4554, 300). Words that were not in the pre-trained word2vec vectors were mapped to a null embedding. In total there were 184 null embeddings.

3.3 LSTM Setup

For the RNN modeling the Tensorflow implementation of Keras is used. provides an implementation of Keras[4], which is one of the most used deep learning frameworks. Specifically Sequential from Keras is used, which can stack layers to create a RNN model. First, the binary Multilabel model, binary Per-Schema model and ordinal Per-Schema model will be created as follows:

1) Embedding Layer The Embedding Layer was given the following parameters: as input dimension the vocabulary size, as output dimension the size of the word2vec vectors, as weights the embedding matrix, as input length the max length of the input vectors and trainable was be set to false, so the word embeddings were not be updated. The parameters for the Embedding Layer were identical for the Binary Multilabel Model, Binary Per-Schema Model and the Ordinal Per-Schema Model.

[4]https://keras.io/

2) Bidirectional LSTM Layer For the Bidirectional LSTM Layer 300 units were selected.

3) Dropout Layer For the Dropout Layer a dropout value of 0.5 was selected.

4) Compile The compile function from Sequential configures the model for training. For the compile function from Sequential, the arguments: optimizer, loss and metrics were used. For the optimization argument: 'Adam' was selected.

5) Fit For fitting the model, the fit function for Sequential was used. For the batch_size argument: 32 was used.

3.4 Talos Hyperparameter optimization

Talos hyper-parameter optimization was run on the Binary Multilabel Model, Binary Per-Schema Model and the Ordinal Per-Schema Model. The following parameters were optimized: for the hidden layer units 100, 200 and 300 units were considered; for the dropout layer, the dropout values 0.1, 0.2, 0.5 were considered; for compiling, the optimization argument: 'rmsprop' and 'Adam' were considered; for fitting, batch size 32 and 64 were considered.

3.5 Binary Multilabel Model

Talos hyperparameter optimization was run on the model. The following parameters were the most optimal with a mean absolute error (MAE) of: 0.004137 for 300 hidden layer units, a dropout value of 0.1, optimization function rmsprop and a batch size of 32. Furthermore, the output layer has 7 output nodes. For the compiling, the loss function binary crossentropy was used and for metrics the mean absolute error.

However, Talos showed a MAE of 0.004420 for a dropout value of 0.1, 200 hidden layer units and activation function rmsprop, a MAE of 0.004481 for a dropout value of 0.1, 200 hidden layer units and activation function Adam, and a MAE of 0.004564 for a dropout value of 0.1, 300 hidden layer units and activation function Adam. The difference in MAE is very small. The complete results are available on GitHub.

3.6 Binary Per-Schema Model

For every schema, Talos hyperparameter optimization was run on the related model. For each parameter, the value that was presented the most in the results was selected for the final model. The complete best parameter results for each schema will be included in Appendix A. The final optimized model used 200 hidden layer units, a dropout value of 0.1, activation function rmsprop and a batch size of 32 was selected.

3.7 Ordinal Per-Schema Model

For hyper-parameter optimization, the results from Burger's paper have been used. This means 300 units for the hidden layer, a dropout value of 0.1, optimization function Adam and a batch size of 32 was picked.

3.8 Results

This section presents the results for the Binary Multilabel Model, the Binary Per-Schema Model and the Ordinal Per-Schema model. For the Binary models, accuracy, F1-score, confusion matrices and ROC curves are presented. See the Appendix B and C for the confusion matrices and the full

classification report. For the Ordinal model, the Spearman Correlation is calculated and the output table will be in the Appendix D.

Binary Multilabel Model

Schema	Before	After
Vulnerable	0.64	0.59
Angry	0.59	0.59
Impulsive	0.75	0.76
Happy	0.63	0.62
Detached	0.55	0.58
Punishing	0.74	0.74
Healthy	0.92	0.91

Table 1: Accuracy Multilabel Model Optimization

Table 1 shows the accuracy calculated with the accuracy method from the scikit-learn package before and after hyper-parameter optimization. The accuracy increased slightly for the impulsive schema and increased for the detached schema. On the other hand, the accuracy decreased significantly for the vulnerable schema and the accuracy decreased slightly for 2 schemas: happy and healthy. The accuracy stayed identical for 2 schemas: angry and punishing. The results show an accuracy of 59% for vulnerable, 59% for angry, 76% for impulsive, 62% for happy, 58% for detached, 74% for punishing and 91% for healthy.

Schema	Before	After
Vulnerable	0.39	0.33
Angry	0.45	0.43
Impulsive	0.23	0.15
Happy	0.75	0.74
Detached	0.30	0.37
Punishing	0.32	0.27
Healthy	0.96	0.95
Micro avg	0.64	0.63
Macro avg	0.49	0.46
Weighted avg	0.62	0.61
Samples avg	0.66	0.65

Table 2: F1-Score Multilabel Model Optimization

Table 2 shows the F1-score before and after hyper-parameter optimization calculated with classification_report from scikit-learn. The F1-score increased only for the detached schema. While, the F1-scores decreased significantly for 3 schemas: vulnerable, impulsive and punishing. The F1-scores decreased slightly for 3 schemas: angry, happy, healthy. The results show an F1-score of 0.39 for vulnerable, 0.45 for angry, 0.23 for impulsive, 0.75 for happy, 0.30 for detached, 0.32 for punishing and 0.96 for healthy. From the confusion matrices it appears that the TN and FN rate increased for most schemas. While, the TP and FP rate decreased for most schemas. Therefore, it can be assumed that the model after hyper-parameter optimization increased the false classification, while the true classifications decreased.

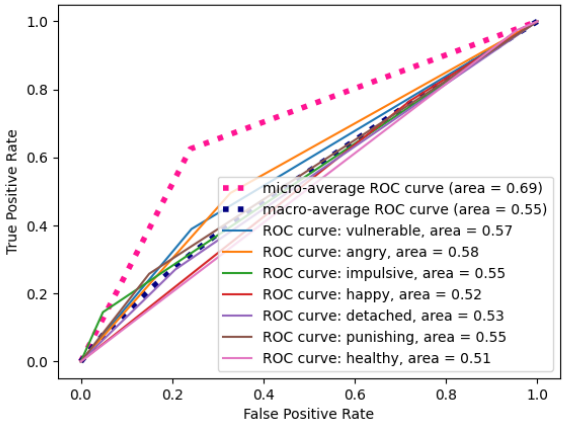

Figure 1: ROC curves, after hyper-parameter optimization

Lastly, Figure 1 shows the ROC curves as well as micro and macro average of all schemas after hyper-parameter optimization. After hyper-parameter optimization the AUC value increased for 4 schemas: vulnerable, impulsive, happy and detached. While it decreased for 1 schema: punishing schema and stayed identical for 2 schemas: angry and healthy. Both micro and macro average increased after hyper-parameter optimization. The results show that the AUC values are 0.69 for the micro-average, 0.55 for the macro-average, 0.57 for vulnerable, 0.58 for angry, 0.55 for impulsive, 0.52 for happy, 0.53 for detached, 0.55 for punishing and 0.51 for healthy.

Based on the accuracy and F1-score, the model before hyper-parameter optimization is the most optimal. Although the AUC value increases for the macro and micro ROC curve, it increases with only 0.02 for both macro and micro.

Binary Per-Schema Model

Schema	Before	After
Vulnerable	0.64	0.63
Angry	0.64	0.62
Impulsive	0.79	0.73
Happy	0.73	0.65
Detached	0.66	0.61
Punishing	0.79	0.77
Healthy	0.92	0.91

Table 3: Accuracy Per-Schema Model Optimization

Table 3 shows the accuracy calculated with the accuracy method from the scikit-learn package before and after hyper-parameter optimization. The accuracy decreased for all 7 schemas. The final results conclude that the accuracy for classifying schemas from the text utterances is 63% for vulnerable, 62% for angry, 73% for impulsive, 65% for happy, 61% for detached, 77% for punishing and 91% for healthy.

Table 4 shows the F1-score calculated with classification_report from scikit-learn before and after hyper-parameter optimization. After hyper-parameter optimization the F1-

Schema	Before	After
Vulnerable	0.36	0.38
Angry	0.15	0.48
Impulsive	0.00	0.17
Happy	0.84	0.77
Detached	0.00	0.36
Punishing	0.03	0.34
Healthy	0.96	0.95
Micro avg	0.66	0.66
Macro avg	0.33	0.49
Weighted avg	0.54	0.63
Samples avg	0.70	0.66

Table 4: F1-Score Per-Schema Model Optimization

score increased for 5 schemas: vulnerable, angry, impulsive, detached and punishing. While, the F1-score decreased slightly for the healthy schema and significantly the happy schema. The final results conclude that the F1 score for schema classification is 0.63 for vulnerable, 0.62 for angry, 0.73 for impulsive, 0.65 for happy, 0.61 for detached, 0.77 for punishing and 0.91 for healthy. Although the accuracy decreased for all schemas, the F1-scores did increase.

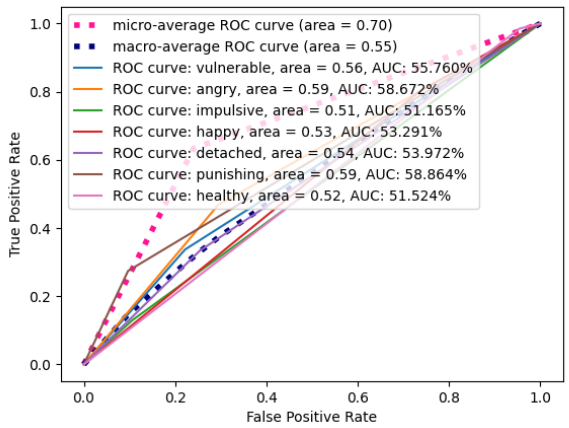

Figure 2: ROC curves for the Per-Schema Binary model, after hyper-parameter optimization

Lastly, Figure 2 shows the ROC curves after hyper-parameter optimization of all schemas, as well as micro and macro average. After hyper-parameter optimization the AUC value increased for 4 schemas: vulnerable, impulsive, happy and detached. While it decreased for the punishing schema and stayed identical for 2 schemas: angry and healthy. Both micro and macro average increased after hyper-parameter optimization. The final results show that the AUC values are 0.70 for the micro-average, 0.55 for the macro-average, 0.56 for vulnerable, 0.59 for angry, 0.51 for impulsive, 0.53 for happy, 0.54 for detached, 0.59 for punishing and 0.52 for healthy.

Based on the results of the confusion matrices, the TP and FP rate increases while the TN and FN rate decreases. In addition the AUC value increases for the schemas that also increase on TP. Therefore, it is assumed that the positive classification increases and the the negative classifications decreases.

Ordinal Per-Schema Model

Schema	Before	After
Vulnerable	0.164	0.278
Angry	NaN	0.176
Impulsive	NaN	0.0418
Happy	0.075	-0.057
Detached	0.209	0.245
Punishing	NaN	0.266
Healthy	0.112	0.092

Table 5: Spearman Rank-Order Coefficient Per-Schema Model

Table 5 presents the Spearman rank-order coefficient before and after hyper-parameter optimization.

Before optimization there were 3 schemas: angry, impulsive and punishing with NaN as value. A reason for this could be that the standard deviation is 0, due to no variation in the either the predicted labels or actual labels.

It is not possible to compare the performance for the classes: angry, impulsive and punishing as these had NaN as value before optimization. However, the performance for the 2 schemas: happy and healthy decreased. While the performance for the 2 schemas: vulnerable and detached increased. The final Spearman Coefficients are 0.278 for vulnerable, 0.176 for angry, 0.0418 for impulsive, -0.057 for happy, 0.245 for detached, 0.266 for punishing and 0.092 for healthy.

3.9 Comparison RNN, KNN and SVM

Schema	SVM	kNN	RNN
Vulnerable	0.27	0.34	0.38
Angry	0.38	0.40	0.48
Impulsive	0.07	0.13	0.17
Happy	0.75	0.80	0.77
Detached	0.18	0.35	0.36
Punishing	0.2	0.22	0.34
Healthy	0.93	0.96	0.95
micro avg	0.63	0.66	0.66
macro avg	0.40	0.46	0.49
weighted avg	0.57	0.62	0.63
samples avg	0.66	0.68	0.66

Table 6: Comparison between SVM, kNN and RNN using F1 Score

The optimized Binary Per-Schema Model has been compared to the Binary kNN and Binary SVM model. Table 6 compares the F1-scores. RNN outperforms the other models for 5 schemas: vulnerable, angry, impulsive, detached and punishing. Although, RNN outperforms kNN for the schema detached, it is only with a difference of 0.1. kNN outperforms the other models for 2 schemas: happy with a F1-score of 0.8 and healthy with a F1-score of 0.96 . However, the kNN

outperforms RNN with a difference of 0.1 for the schema healthy. The micro avg for kNN and RNN are both 0.66, while the micro avg for SVM is 0.63. The macro avg as well as the weighted average for RNN is ranked the highest between the three models with a value of 0.49 compared to 0.40 for SVM and 0.46 for kNN.

Appendix D presents the micro and macro averages of RNN, kNN and SVM compared to eachother. The micro-average of SVM is significantly better than RNN and kNN. The macro-average of RNN is slightly better than the other models. This means that SVM is better at classifying the most populated classes, while it is not very good at classifying the least populated classes.

Schema	SVM	kNN	RNN
Vulnerable	0.078	0.13	0.28
Angry	0.023	0.08	0.18
Impulsive	0.0033	0.12	0.042
Happy	0.12	0.06	-0.057
Detached	-0.090	0.08	0.24
Punishing	0.074	0.09	0.27
Healthy	0.020	0.06	0.09

Table 7: Comparison between SVM, kNN and RNN using Spearman Correlation

The Ordinal Per-Schema Model has been compared to the Ordinal kNN and Ordinal SVM model. Table 7 contains a comparison of the Spearman rank-order correlation between RNN, kNN and SVM. RNN outperforms the other models for 5 schemas: vulnerable, angry, detached, punishing and healthy. For the schema happy, SVM outperforms the other models with a Spearman Correlation of 0.12, where RNN has a value of -0.057 and kNN has a value of 0.06.

4 Discussion

This section will discuss the results presented in the previous sections and compare the results to the findings of some papers from the related work section.

After hyper-parameter optimization for the binary classification Multilabel model, the performance accuracy, F1-score and confusion matrices and has shown to be worse than the model before hyper-parameter optimization. Other parameter values from Talos scored a MAE within a range of 0.0004 compared to the MAE of the model that was used. Different parameters may have been more efficient, because the MAE did not differ strongly.

After optimizing the hyper-parameters for the binary classification Per-Schema model, the accuracies for all 7 schemas decreased. A reason for the lower accuracy could be because the optimal hyper-parameters for each schema were logged and the majority value had been picked for each hyper-parameter. However, these hyper-parameters may not have worked together as optimal. Contrary to the accuracy, the F1-score increased significantly for 5 schemas and only decreased for 2 schemas. The 2 schemas that were decreased were healthy and happy. Based on the confusion matrices, the conclusion can be drawn that the classification for the 2 schemas (happy and healthy) with the most data has decreased, but the classification for the schemas with less data instances has increased.

In Allaart's research [3] the accuracy is 42% for vulnerable, 65% for angry, 34% for impulsive, 39% for happy, 66% for detached, 47% for punishing and 80% for healthy. Averaged it is 53% for all 7 schemas. Whereas for this research the average accuracy for all 7 schemas is 72%. Furthermore, all schemas have an accuracy higher than 60%. Therefore, this research has managed to improve the classification Allaart's study.

The Spearman rank order correlation coefficient for the Ordinal Per-Schema Model is a very low as none of the schemas have a Spearman Correlation higher than 0.3. However, an ordinal model is more realistic as a patient can be scored from 0-6 for a schema. It is not possible to compare to Burger's [4] as the schema modes are not identical. However, Burger managed to achieve an F1-score of more than 0.5 for 6 classes. Whereas in this research the F1-score did not achieve more than 0.3 for any schemas.

5 Conclusion and Future Work

1) What pre-processing steps on the data are necessary for an optimal classification by RNN? Removing data instances that do not contain information for classification, lower-casing, splitting of contractions, removal of stopwords, removal of unnecessary white space and lemmatization have been selected for this research and have proven to increase accuracy compared to Allaart's study [3].

2) What are the optimal hyper-parameters in the RNN model? The Binary Multilabel model before the hyper-parameter optimization, with a hidden layer with 300 units, a dropout value of 0.5, optimization function 'Adam' and a batch size of 32 is the most optimal.

For the Binary Per-Schema model the accuracy for all schemas decreased after the Talos hyper-parameter optimization. However, the F1-score increased for most of the schemas, except happy and healthy. Therefore, a hidden layer with 200 units, a dropout value of 0.1, optimization function 'rmsprop' and a batch size of 32 is most optimal.

Lastly, for the Ordinal Per-Schema Model the Talos optimization output from Burger was used. A dropout rate of 0.1, optimization function 'Adam' and a batch size of 32. Although the Spearman Coefficient decreased for healthy and happy, it increased for the other schemas. Therefore, the optimized model with a hidden layer with 300 units, a dropout value of 0.1, optimization function 'Adam' and a batch size of 32 is most optimal.

3) What are the differences between the results of three methods, RNN, KNN and SVM? Based on the results it can be concluded that RNN classifies the best for schema with less data instances. SVM classifies the best for healthy and happy, based on the micro average. And kNN and RNN perform similar based on the macro and micro average.

For the ordinal classification kNN performs the best for the impulsive schema and SVM performs the best for the happy schema. RNN performs the best for the other 5 schema. It should also be noted that before the optimiza-

tion of the RNN, the happy schema outperformed the other models.

How well can a schema be automatically classified from a text using RNN?

The Binary Multilabel Model has an overall accuracy of 70%. The model performs with an accuracy of 91% for the healthy schema. Based on the F1-score, I can conclude that the Binary Multilabel Model classifies the happy and healthy schema well, with a F1-score of 0.74 for the happy schema and a F1-score of 0.95 for the healthy schema. However, the other schemas: vulnerable, angry, impulsive, detached and punishing are classifying poorly with a F1-score of less than 0.5.

The Binary Per-Schema Model has an overall accuracy of 68%. The model performs with an accuracy of 91% for the healthy schema. Whereas other schemas have an accuracy below 80%. Based on the F1-score, I can conclude that the Binary Per-Schema Model classifies the happy and healthy schema well, with a F1-score of 0.77 for the happy schema and a F1-score of 0.95 for the healthy schema. However, the other schemas: vulnerable, angry, impulsive, detached and punishing are classifying poorly with a F1-score of less than 0.5.

The Spearman rank order correlation coefficient performs poorly with an average Spearman Correlation of 0.15. However, an ordinal model is more realistic as a patient can be scored from 0-6 for a schema.

To summarize, this research has used a RNN classifier to detect schema modes from stories. Results have shown that the dataset is biased. This has limited the performance of this research. The binary classification has proven to be quite accurate regarding the accuracy for the Per-Schema model with an accuracy of 72%. The F1-score is less, as the average is 0.49. However, the ordinal classification performs poor as none of the schemas achieved a Spearman Correlation of higher than 0.3. This research has improved the classification algorithm of Allaart's study [3]. However, it did not manage to outperform the classification of Burger's study [4].

5.1 Limitations

This research has several limitations regarding the dataset and the purpose of the SMI. This resulted in the classification algorithm not performing as well as expected, regarding the F1-scores, ROC curves and Spearman Rank-Order Coefficient.

1) Biased dataset For the vulnerable and detached schema, 33% is labelled as 1 and 67% is labelled as 0. For the angry schema 38% is labelled as 1 and 62% is labelled as 0. For the impulsive and punishing schema 21% is labelled as 1 and 79% is labelled as 0. For the schema happy 73% of the dataset is labelled as 1, the remaining is labelled as 0. For the schema healthy 92% of the dataset is labelled as 1, the remaining is labelled as 0. Thus, therefore a classifier that always predicts 1 can achieve 92% accuracy for the schema healthy and a classifier that always predicts 0 can achieve 60% accuracy for the other schemas.

2) Low Correlation The SMI captures a general schema of the patient over a period of time. The data contains a story about a momentarily event and labels reflecting the schema of the patient based on the SMI questionnaire that was filled in after the story. This can result in a low correlation between the story and schema (see point 2), resulting in bad classification. For some data instances the text utterance and label does not reflect each other. See Appendix E for two examples.

3) Small dataset After cleaning the dataset there were 1375 instances left. However, this is not enough to create a good classifier [21].

5.2 Future Work

This research can be improved in two areas. First of all the dataset can be improved. First, the results can be more reliable by ensuring a balanced dataset. This means that there are equal negative and positive data instances for each schema. Second, the data instances should be increased so the classifier has more data to train upon. Third, the data can be manually labelled. This way the story and schemas will represent each other and errors such as in Appendix E will not occur. Lastly, as mentioned in the limitations, the SMI questionnaire represents schema modes for a period of time. Instead of using one story to predict the schema for a patient. It is possible to use several stories from several conversations over time for a patient. This way the schema modes actually represent a period of time. The second area that could be improved upon is the classifier. First, the models used in this research can use different parameters from the Talos results in Appendix A. Second, additional pre-processing can be explored.

6 Responsible Research

For this research project I indirectly worked together with Budi Han, Jeongwoo Park, Jahson Binda and Jimmy Lam. Our research is supporting an "overarching goal to create a chatbot that can autonomously detect schemas from users through 'casual conversation or storytelling' and suggest appropriate treatment options when an unhealthy schema mode is detected." [5] I have collaborated more tightly with the former two. The reason for this is because one of our subquestion requires a comparison of our classification models. Between the three of us, the performance results between the models were shared. Part of the code for pre-processing is also shared, as all three of us needed the same pre-processing to ensure the performance comparison was made with equal circumstances.

Furthermore, we all got provided the same data set by the supervisors. The data was obtained from Allaart's study [3]. The dataset is not shared outside of the research of me and my peers. The data was shared on Microsoft Teams files in the designated team for this project. The provided raw data was transformed as mentioned in the methodology. For preprocessing, it was necessary to take out data instances that provided no use to the classification of schemas. No further modifications have been done on the dataset. The final dataset used for this research is available on GitHub [6]. One important issue was the fact that the dataset was biased to be la-

[5] https://projectforum.tudelft.nl/course_editions/39/projects/973

[6] https://github.com/Mirijam1/Automatic_Psychological_Text_Analysis_RNN

belled as 1 for the healthy and happy schemas and labelled as 0 for the vulnerable, angry, impulsive, detached and punishing schemas. This is also explained in the limitations section. Therefore, the results may be biased towards either the 0 or 1 label depending on the schema.

Additionally, the work from Burger was provided by the supervisors. A Jupyter notebook containing her code and results for KNN, SVM and RNN was shared. I have used the RNN model from her Jupyter notebook as inspiration and improved from there. None of Burger's code has been directly copied. The final RNN classification models are also shared on GitHub.

Lastly, the Talos hyper-parameter optimization outputs and all performance results for the Binary Multilabel Model, Binary Per-Schema Model and the Ordinal Per-Schema Model are included in the Appendix

6.1 Reproducibility

The modified dataset is available on GitHub. All RNN models will be are available as a Jupyter Notebook and as a Python file. To ensure reproducibility, it is recommended to follow the experimental setup or to use the Jupyter Notebook on GitHub. To make sure the correct dependencies are installed, a requirement.txt file is included. This will ensure the right versions are installed and environment is created.

It is generally recommended to run the model many times (30+) to get stable and reproducible results[7]. It is also recommended to compare the results of the models to be able to draw conclusions on the variability between the results. However, this research has three models: Binary Multilabel RNN, Binary Per-Schema RNN, Ordinal Per-Schema RNN. To run each model for 30 times takes a very long time and it was not possible to do this in time. As it was not possible to run the model 30 times, all models have been run with epochs of 100 to make sure the models and results are stable.

References

[1] J. E. Young, J. S. Klosko, and M. E. Weishaar, *Schema Therapy: A Practitioner's Guide*. Guilford Press.

[2] J. Lobbestael, M. Vreeswijk, P. Spinhoven, E. Schouten, and A. Arntz, "Reliability and validity of the short schema mode inventory (SMI)," vol. 38, pp. 437–58.

[3] D. Allaart, "Schema mode assessment through a conversational agent."

[4] F. Burger, M. A. Neerincx, and W.-P. Brinkman, "Natural language processing for cognitive therapy: extracting schemas from thought records."

[5] A. Millings and K. B. Carnelley, "Core belief content examined in a large sample of patients using online cognitive behaviour therapy," vol. 186, pp. 275–283.

[6] F. Patel, R. Thakore, I. Nandwani, and S. K. Bharti, "Combating depression in students using an intelligent ChatBot: A cognitive behavioral therapy," in *2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1–4. ISSN: 2325-9418.

[7] M. P. Ewbank, R. Cummins, V. Tablan, A. Catarino, S. Buchholz, and A. D. Blackwell, "Understanding the relationship between patient language and outcomes in internet-enabled cognitive behavioural therapy: A deep learning approach to automatic coding of session transcripts," vol. 31, no. 3, pp. 300–312.

[8] J. Brownlee, "When to use MLP, CNN, and RNN neural networks."

[9] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding,"

[10] J. Brownlee, "Crash course in recurrent neural networks for deep learning."

[11] J. L. Elman, "Finding structure in time," vol. 14, no. 2, pp. 179–211.

[12] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," vol. 06, no. 2, pp. 107–116. Publisher: World Scientific Publishing Co.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, pp. 1735–80.

[14] J. Hassan and U. Shoaib, "Multi-class review rating classification using deep recurrent neural network," vol. 51.

[15] D. Can, D. C. Atkins, and S. S. Narayanan, "A dialog act tagging approach to behavioral coding: A case study of addiction counseling conversations," p. 5.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," p. 9.

[17] L. Derczynski, "Complementarity, f-score, and NLP evaluation," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 261–266, European Language Resources Association (ELRA).

[18] T. Fawcett, "An introduction to ROC analysis," vol. 27, no. 8, pp. 861–874.

[19] A. Maratea, A. Petrosino, and M. Manzo, "Adjusted f-measure and kernel scaling for imbalanced data learning," vol. 257, pp. 331–341.

[20] C. Spearman, "The proof and measurement of association between two things. by c. spearman, 1904," vol. 100, no. 3, pp. 441–471.

[21] J. O'Dwyer Wha Binda, "Active learning in reducing human labeling for automatic psychological text classification."

[7] urlhttps://machinelearningmastery.com/reproducible-results-neural-networks-keras/

Appendices

A Talos results

A.1 Binary Per-Schema

The schema Vulnerable had a MAE of 0.001820 with a dropout value of 0.1, 200 hidden layer units, activation function rmsprop, and batch size of 32.
The schema Angry had a MAE of of 0.001760 with a dropout value of 0.1, 200 hidden layer units, activation function rmsprop, and batch size of 32.
The schema Impulsive had a MAE of of 0.002297 with a dropout value of 0.1, 300 hidden layer units, activation function rmsprop, and batch size of 32.
The schema Happy had a MAE of of 0.001769 with a dropout value of 0.2, 100 hidden layer units, activation function Adam, and batch size 32.
The schema Detached had a MAE of of 0.001708 with a dropout value of 0.1, 200 hidden layer units, activation function rmsprop, and batch size of 32.
The schema Punishing had a MAE of of 0.001661 with a dropout value of 0.5, 200 hidden layer units, activation function rmsprop, and batch size of 32.
The schema Healthy had a MAE of of 8.785876e-09 with a dropout value of 0.1, 300 hidden layer units, activation function rmsprop, and a batch size of 32.

B Binary Multilabel Model results

B.1 Confusion Matrices

Schema	TP	FP	FN	TN
Vulnerable	33	43	62	151
Angry	50	61	59	119
Impulsive	11	22	51	205
Happy	160	58	50	21
Detached	27	60	69	133
Punishing	18	32	44	195
Healthy	266	21	1	1

Table 8: Confusion Matrices Multilabel Model Before Optimization

Schema	TP	FP	FN	TN
Vulnerable	29	52	66	142
Angry	46	57	63	123
Impulsive	6	13	56	214
Happy	155	55	55	24
Detached	36	62	60	131
Punishing	14	26	48	201
Healthy	263	21	4	1

Table 9: Confusion Matrices Multilabel Model After Optimization

B.2 Classification report

	precision	recall	f1-score	support
vulnerable	0.43	0.35	0.39	95
angry	0.45	0.46	0.45	109
impulsive	0.33	0.18	0.23	62
happy	0.73	0.76	0.75	210
detached	0.31	0.28	0.30	96
punishing	0.36	0.29	0.32	62
healthy	0.93	1.00	0.96	267
micro avg	0.66	0.63	0.64	901
macro avg	0.51	0.47	0.49	901
weighted avg	0.63	0.63	0.62	901
samples avg	0.71	0.69	0.66	901

Figure 3: Classification report before

	precision	recall	f1-score	support
vulnerable	0.36	0.31	0.33	95
angry	0.45	0.42	0.43	109
impulsive	0.32	0.10	0.15	62
happy	0.74	0.74	0.74	210
detached	0.37	0.38	0.37	96
punishing	0.35	0.23	0.27	62
healthy	0.93	0.99	0.95	267
micro avg	0.66	0.61	0.63	901
macro avg	0.50	0.45	0.46	901
weighted avg	0.62	0.61	0.61	901
samples avg	0.71	0.68	0.65	901

Figure 4: Classification report after

B.3 ROC curve

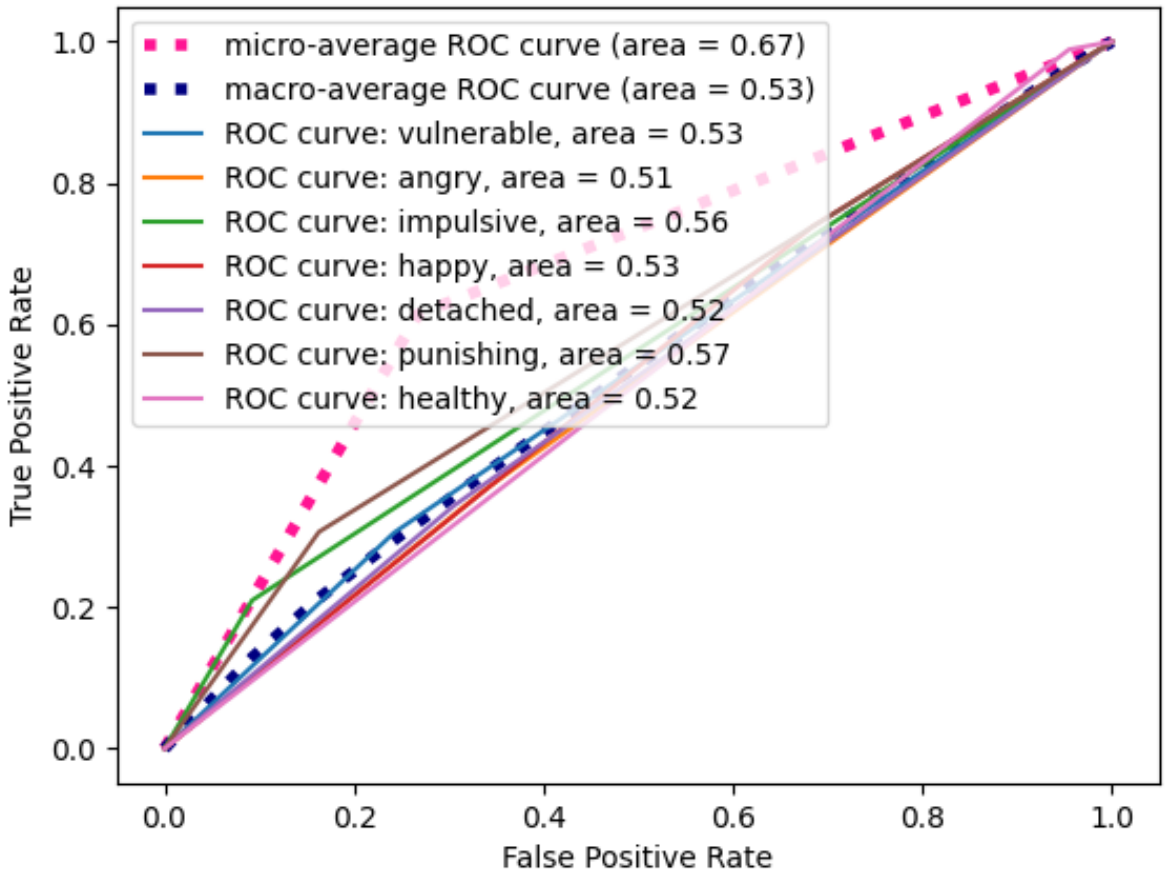

Figure 5: ROC curves, before hyper-parameter optimization

C Binary Per-Schema Model results

C.1 Confusion Matrices

Schema	TP	FP	FN	TN
Vulnerable	30	40	65	154
Angry	9	4	100	176
Impulsive	0	0	62	227
Happy	199	66	11	13
Detached	0	1	96	192
Punishing	1	1	61	226
Healthy	267	22	0	0

Table 10: Confusion Matrices Per-Schema Model Before Optimization

Schema	TP	FP	FN	TN
Vulnerable	32	43	63	151
Angry	51	53	58	127
Impulsive	8	24	54	203
Happy	168	58	42	21
Detached	32	49	64	144
Punishing	17	22	45	205
Healthy	263	21	4	1

Table 11: Confusion Matrices Per-Schema Model After Optimization

C.2 Classification report

	precision	recall	f1-score	support
vulnerable	0.43	0.32	0.36	95
angry	0.69	0.08	0.15	109
impulsive	0.00	0.00	0.00	62
happy	0.75	0.95	0.84	210
detached	0.00	0.00	0.00	96
punishing	0.50	0.02	0.03	62
healthy	0.92	1.00	0.96	267
micro avg	0.79	0.56	0.66	901
macro avg	0.47	0.34	0.33	901
weighted avg	0.61	0.56	0.54	901
samples avg	0.81	0.66	0.70	901

Figure 6: Classification report before

	precision	recall	f1-score	support
vulnerable	0.43	0.34	0.38	95
angry	0.49	0.47	0.48	109
impulsive	0.25	0.13	0.17	62
happy	0.74	0.80	0.77	210
detached	0.40	0.33	0.36	96
punishing	0.44	0.27	0.34	62
healthy	0.93	0.99	0.95	267
micro avg	0.68	0.63	0.66	901
macro avg	0.52	0.48	0.49	901
weighted avg	0.64	0.63	0.63	901
samples avg	0.71	0.69	0.66	901

Figure 7: Classification report after

C.3 ROC curve

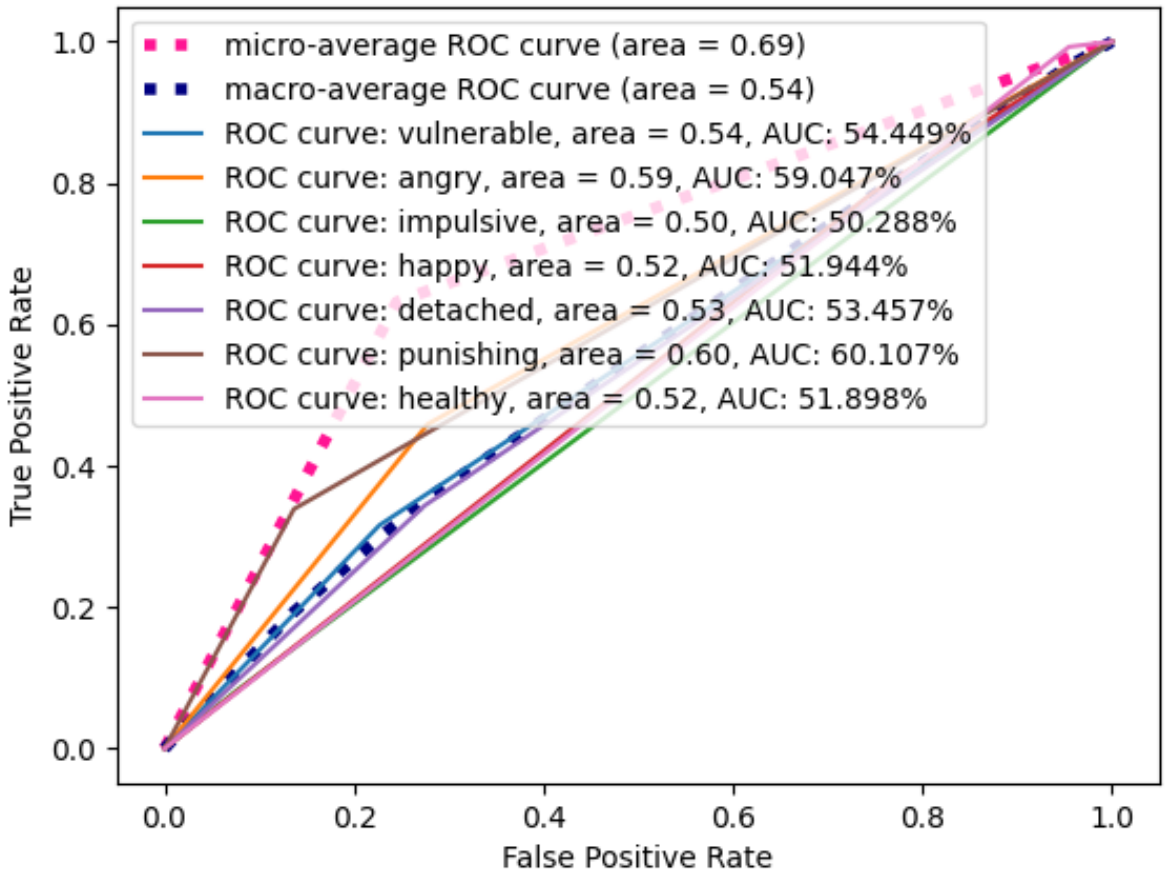

Figure 8: ROC curves for the Per-Schema Binary model, before hyper-parameter optimization

D Comparison RNN, KNN and SVM

D.1 ROC curves

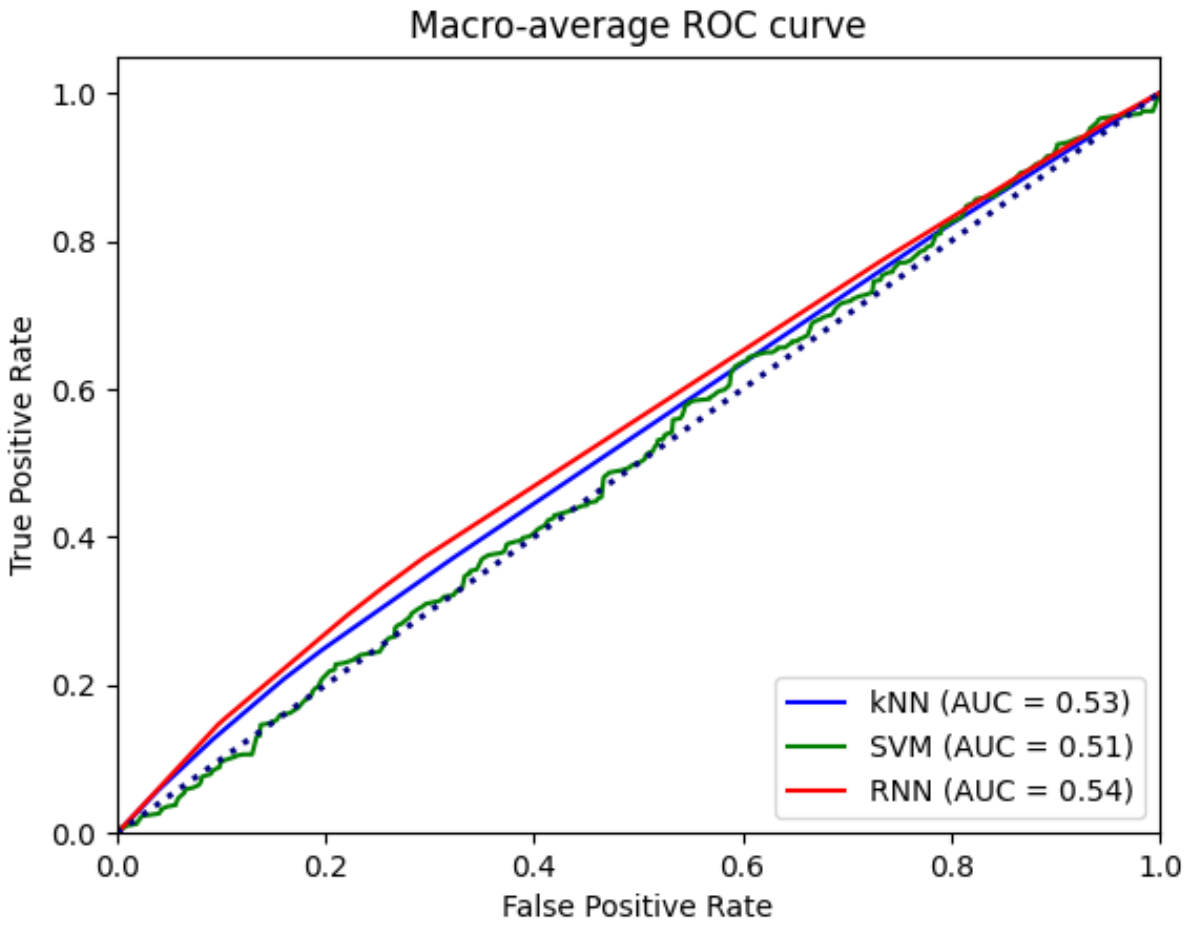

Figure 9: Macro avg curves for RNN, KNN and SVM

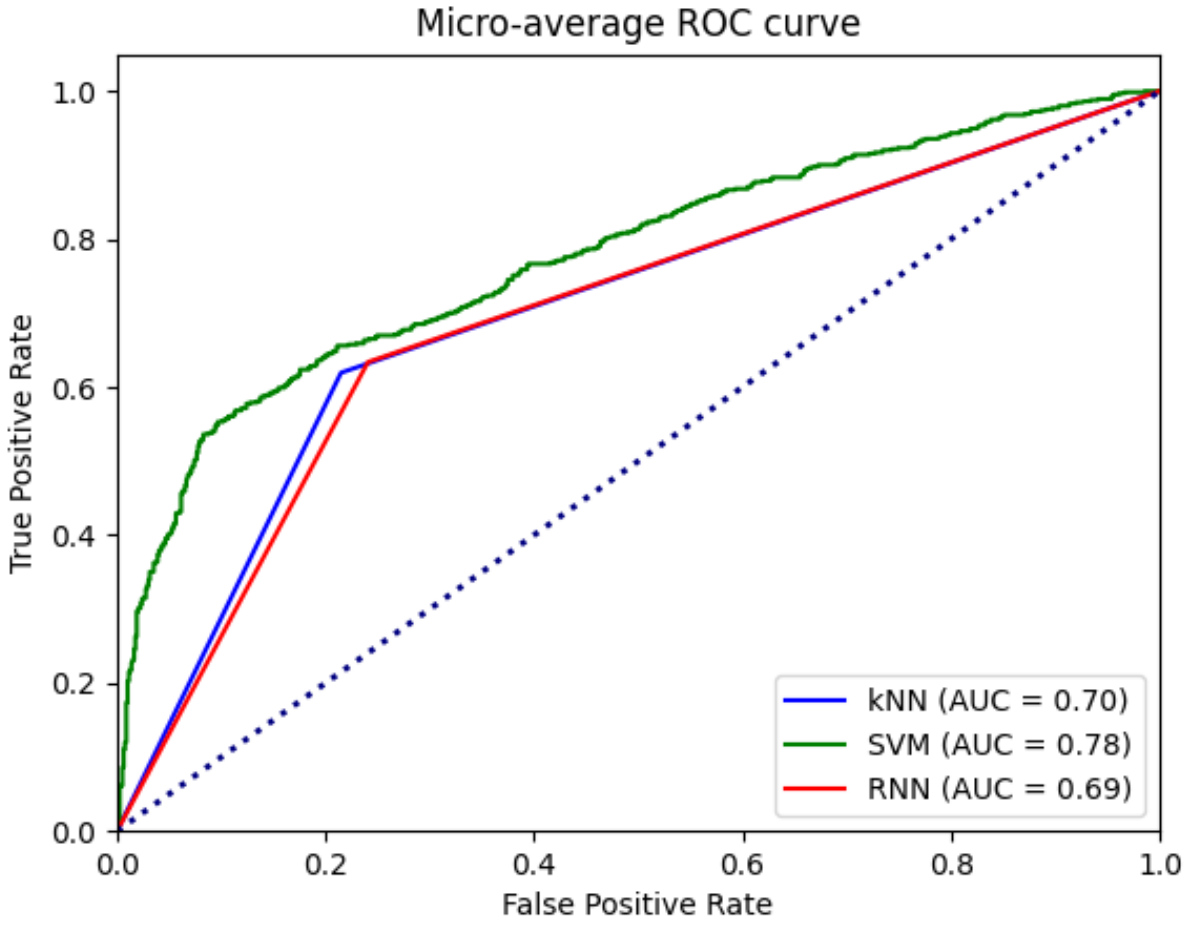

Figure 10: Micro avg curves for RNN, KNN and SVM

E Wrongly labeled text utterances

Text	is_vulnerable	is_angry	is_impulsive	is_happy	is_detached	is_punishing	is_healthy
hello, billy. i recently had to shut my childcare business for the week. i was upset and angry as it meant i had to let down 6 families and refund their money. me and my boss lost money and it was a sad time for both of us. it was a stressful time and we had to contact a lot of people in order to find out if we could re open.	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE

Figure 11: Wrongly labelled text utterance 1

Text	is_vulnerable	is_angry	is_impulsive	is_happy	is_detached	is_punishing	is_healthy
i have not been feeling myself over the past few weeks and i have been finding myself not being able to complete tasks, i have been putting off a job that i said i would do for a friend that would help their business. i can't stop thinking about the fact that i haven't done it and it makes me feel really anxious and really bad but i can't bring myself to do it. i keep receiving emails from the friend but i have just felt so anxious i haven't been able to open them. i finally opened them and did the work and sent it back and i felt so relived but annoyed at myself for not doing it sooner and leaving it so long.	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

Figure 12: Wrongly labelled text utterance 2