CS598 DL4HC Reproducibility Project Proposal

Michael Miller and Kurt Tuohy
{msmille3, ktuohy}@illinois.edu

Group ID: 127
Paper ID: 117, Difficulty: easy
Presentation link: https://youtu.be/Q-s3llfATW8
Code repository: https://github.com/mich1eal/cs598_dl4hc

1 Introduction

The majority of text written by patients in psychological care is only interpreted by human therapists (Burger et al., 2021). In their paper, Burger et al. begin to explore the potential of NLP as an aid to cognitive therapy. They seek to determine whether NLP techniques can reliably classify patient writing.

A useful cognitive therapy exercise involves having patients identify the thoughts that underlie their reactions to potentially distressing situations. The authors attempt to classify a set of these written thoughts (or "utterances") into schemas.

"Schemas" refer to core beliefs that function as the infrastructure to one's views of the world. A standard set of schemas defined by Millings and Carnelley was used.

Burger et al. establish a baseline for applying NLP to cognitive therapy. They intend for future researchers to leverage their paper and data to continue to explore applications of NLP in this field. In this paper, we verify the results of Burger et al., and implement several design modifications as further studies.

2 Scope of reproducibility

In this paper, we test the central hypothesis of Burger et al. We verify that schemas can be automatically predicted from patient thought records, and that this can be done with accuracy greater than random chance.

We re-implement four models used in the paper:

1. k-nearest neighbors (classifier and regression algorithms)
2. Support vector machines (classifier and regression algorithms)
3. Multi-label LSTM, i.e. a single model to output predictions for all schemas (regression algorithm)
4. Per-schema LSTMs, i.e. one LSTM for each schema (classification algorithm)

In order to generate comparable results, we adopt a number of the design choices of Burger et al.. We use pre-trained GLoVe embeddings (Pennington et al., 2014) in all of the above models.

We also measure model performance using Spearman's rank-order correlation between predictions and ground truth.

2.1 Addressed claims from the original paper

Because of our similar design, we expect to obtain similar results to Burger et al.. In particular, we will verify the following three claims:

1. Per-schema LSTMs outperform other tested models on all schemas except "Power and Control" and "Meta-Cognition".
2. Each regression model outperforms its corresponding classifier on the "Power and Control" schema.
3. Each classifier outperforms the corresponding regression model on the "Health" schema.

3 Methodology

3.1 Model descriptions

3.1.1 K-nearest Neighbors

Burger et al. generated kNN models for selected values of k from 2 to 100. These models used the cosine distance between thought-record embeddings as the distance metric.

To generate predictions, the classification models computed the statistical mode of the k nearest neighbors' labels. The regression models used the average of the nearest neighbors' labels.

k=4 gave the best results for classification models, and k=5 was best for regression models.

3.1.2 Support Vector Machine

Burger et al. implemented both Support Vector Machine Classifier (SVM-C) and Regression (SVM-R) models. For both implementations, a separate classifier was created for each schema. Each SVM outputs a value for its schema, and all nine were evaluated together using Spearman's correlation coefficient. Burger et al. selected Radial Basis Function (RBF) as the most effective kernel. RBF has one trainable parameter per input dimension, plus the SVM's bias term. Utterance-level GLoVE embeddings with 100 dimensions were used as inputs. As a result, nine SVM models have a total of 909 trainable parameters.

3.1.3 Multi-label LSTM

A single model was trained to output predictions for all nine schemas.

For this model, the researchers obtained the best performance from a bidirectional Long Short Term Memory (LSTM) Recurrent Neural Network (RNN) with 100 parameters in its hidden state.

The model also included an embedding layer, which used GLoVE embeddings on a per-word basis. Following the LSTM, a 10% dropout layer fed a single fully connected layer, which reduced the output to the 9 classes. Finally, a sigmoid activation function output per-class probabilities.

The model was trained using cross entropy loss with the Adam optimizer and a default learning rate of 0.001.

The number of trainable parameters was 162,609. The embedding involved 262,400 non-trainable parameters, for a total of 425,009.

3.1.4 Per-schema LSTMs

A total of nine models were trained, one for each schema. Model architecture is similar to the LSTM defined above, with the following differences:

Instead of 9 output classes, each model used 4. This matched the researcher's 0-3 scale for rating thought records against schemas, where 0 meant ”no correspondence with schema” and 3 meant ”complete correspondence with schema”.

Instead of a sigmoid activation, these models used softmax, since points on the rating scale are mutually exclusive.

Each per-schema model had 161,604 trainable parameters, with the same number of non-trainable parameters as the multi-label model. The total was 424,004.

3.1.5 Note on embeddings

The researchers created custom per-thought-record embeddings for the kNN and SVM models. In contrast, the RNN models used basic per-word GLoVe embeddings as weights.

For the custom embeddings, each word's GLoVe embedding was multiplied by the word's TF-IDF weight. The resulting vector was divided by its 2-norm. Then, for each thought record, the applicable word vectors were summed and averaged. This gave one embedding per thought record.

3.2 Data descriptions

The data used in the original paper is available online (Burger, 2021). The data is referenced at multiple phases:

1. Fully raw - data directly pulled from the authors' survey. This data contains sensitive information and is not published.

2. Anonymized raw - data that has been anonymized. This data is otherwise in its original form. It includes participant IDs, the situational scenarios that participants responded to, the thought records themselves, other grouping information, and the ground-truth correspondence scores for each schema on a 0-3 scale.

3. Preprocessed - thought records have been normalized, spell-checked, stripped of common words, and split into test/validation/train sets. Thought records have also been decoupled from contextual information like participant IDs and situational scenarios.

Dataset 3 was used to recreate authors' original results. By taking advantage of the authors' preprocessing, fewer confounding variables will exist in the output.

Dataset 2 will be used for additional ablations. It is hypothesized that keeping common words in the data may improve performance, especially when using larger, more expressive models.

3.3 Hyperparameters

For their RNN models, Burger et al. conducted a large parameter sweep to optimize hyperparameters. It is not our intent to duplicate this effort. We will instead focus on verifying the authors' result using their optimal hyperparameters.

For our ablations, we intend to perform a hyperparameter sweep.

3.4 Implementation

The authors' code is available in its entirety online (Burger, 2021). While the authors' code was referenced, the code for this paper is original.

The codebase of this paper is available online (Tuohy and Miller, 2022). All scripting code is written in Python. A number of python modules were used, in particular, we used pyTorch as our primary deep learning module. A full list of Python module dependencies in available in the repository.

Code was executed on machines running Windows 10 using Python 3.10. GPUs were deemed unnecessary for training (see Computational requirements).

During implementation, a number of documentation sources were reviewed. These include documentation for individual Python modules, online tutorials and forums, and course material from this and other classes.

3.5 Computational requirements

In our proposal, we stated that a relatively small dataset (5747 thought records) and relatively simple models were used. The largest time requirements of Burger et al. (2021) exist because they trained hundreds of RNN models to decrease the role of chance in the final results. We planned to reduce the number of models trained while still confirming key results, if computation time became prohibitive. As such, we planned execution using standard computers without access to GPUs, TPUs, or computational clusters.

In general, this approach appears to be succeeding. Training an individual RNN takes 7-10 minutes on a conventional CPU. This performance is satisfactory for development and validation. Final model evaluation may be run overnight but is not expected to become an inhibiting factor.

The paper's original Jupyter notebook tracked clock time but not CPU usage. Each of their RNNs required about 2.6 minutes to train, which is on the same order of magnitude as our training times.

The script to run our single draft model consumes about 570 MB of RAM, which includes GLoVe embeddings. In comparison, the original notebook consumes about 834 MB before training any models.

Based on partial reruns of the paper's original Jupyter notebook, each additional RNN should require about 26 MB. The kNN and SVM models should be smaller yet.

As models are finalized, code will be added to capture more accurate metrics about runtime to be documented in the final report.

4 Results

The paper's original model types were all reimplemented in pyTorch. Following are the models' point estimates of Spearman correlation coefficients for predicted schemas vs. ground-truth values. We compare them to the original paper's 95% confidence intervals.

Additional work beyond the original paper's scope is presented in Further studies.

4.1 Model Results Comparison

4.1.1 k-Nearest Neighbors

Our kNN models are well-behaved, with almost all r values falling within paper's the 95% confidence bounds. Our classifier originally returned much lower values until we discovered that two utterances with only unknown tokens were being classified as the nearest neighbors of all other utterances, due to the model's cosine distance measure. Results improved dramatically after we removed those two utterances.

	kNN Classification	
Schema	Published	Reproduced
Attachment	0.55 [0.51, 0.60]	0.59
Competence	0.69 [0.64, 0.73]	0.64
Global self-eval	0.40 [0.33, 0.46]	0.54
Health	0.74 [0.65, 0.81]	0.59
Power & ctrl	0.11 [0.02, 0.18]	0.18
Meta-cognition	nan [0.00, 1.00]	nan
Other people	0.28 [0.00, 1.00]	-0.01
Hopelessness	0.48 [0.44, 0.55]	0.59
Other's views	0.45 [0.41, 0.51]	0.42

Table 1: Spearman correlation [95% confidence bounds] for k-nearest neighbors classifier

kNN Regression		
Schema	Published	Reproduced
Attachment	0.63 [0.59, 0.65]	0.62
Competence	0.66 [0.63, 0.69]	0.68
Global self-eval	0.41 [0.36, 0.46]	0.47
Health	0.53 [0.44, 0.60]	0.53
Power & ctrl	0.23 [0.17, 0.27]	0.23
Meta-cognition	0.10 [0.01, 0.20]	0.02
Other people	0.24 [0.17, 0.31]	0.14
Hopelessness	0.51 [0.47, 0.56]	0.50
Other's views	0.46 [0.42, 0.50]	0.45

Table 2: Spearman correlation [95% confidence bounds] for k-nearest neighbors regressor

4.1.2 Support Vector Machine

SVMs are also well-behaved, with most r values either within or very close to the original 95% confidence bounds.

SVC (Classification)		
Schema	Published	Reproduced
Attachment	0.65 [0.61, 0.68]	0.66
Competence	0.68 [0.65, 0.72]	0.68
Global self-eval	0.36 [0.31, 0.40]	0.54
Health	0.73 [0.65, 0.81]	0.63
Power & ctrl	nan [0.00, 1.00]	0.10
Meta-cognition	nan [0.00, 1.00]	nan
Other people	nan [0.00, 1.00]	-0.01
Hopelessness	0.49 [0.43, 0.53]	0.58
Other's views	0.48 [0.43, 0.53]	0.43

Table 3: Spearman correlation [95% confidence bounds] for support vector machine classifier

SVR (Regression)		
Schema	Published	Reproduced
Attachment	0.68 [0.65, 0.70]	0.68
Competence	0.64 [0.61, 0.67]	0.67
Global self-eval	0.49 [0.45, 0.52]	0.48
Health	0.35 [0.31, 0.40]	0.38
Power & ctrl	0.31 [0.26, 0.35]	0.22
Meta-cognition	0.11 [0.06, 0.16]	0.07
Other people	0.19 [0.14, 0.24]	0.13
Hopelessness	0.54 [0.51, 0.57]	0.48
Other's views	0.52 [0.48, 0.55]	0.49

Table 4: Spearman correlation [95% confidence bounds] for support vector machine regressor

4.1.3 Multi-label LSTM

Results for the Multi-label LSTM generally approximate the published values, with four out of nine schemas (Global self-evaluation, Health, Other people, and Other's views on self) falling within the authors' 95% confidence bounds. The full list of results is presented in Table 5 below.

Schema	Published	Reproduced
Attachment	0.67 [0.66, 0.72]	0.63
Competence	0.66 [0.64, 0.69]	0.63
Global self-eval	0.49 [0.45, 0.53]	0.49
Health	0.35 [0.31, 0.39]	0.35
Power & ctrl	0.31 [0.27, 0.34]	0.06
Meta-cognition	0.11 [0.06, 0.14]	0.03
Other people	0.16 [0.10, 0.20]	0.11
Hopelessness	0.53 [0.50, 0.56]	0.44
Other's views	0.50 [0.47, 0.54]	0.49

Table 5: Spearman correlation [95% confidence bounds] for multi-label LSTM model

Results show some variation between published data and reproduced data. These variations are attributed to a number of factors:

- A random partition of train/validation/test sets was used. Burger et al. created sets of samples with equal distributions of each schema. This may explain the much lower scores we obtain for schemas that were sparsely represented in the source data, such as "Power and Control" and "Meta-cognition."
- Model architecture was implemented using Pytorch instead of TensorFlow. This includes the LSTM, loss function, optimizer, etc.
- Unique padding, unknown, and End of Sentence tokens were implemented. Burger et al. only used padding tokens, with unknown tokens handled through TensorFlow's default functionality.
- The original Keras models used categorical cross entropy, which has no direct pyTorch equivalent. In its place we used CrossEntropyLoss. We also doubled the researchers' learning rate to 0.002.

4.1.4 Per-schema LSTMs

These models predicted the rating of each utterance against each separate schema, on the 0-3 rating scale. Our models predicted the majority class for all utterances, i.e. that the utterances were not related to any schema. This is why it scores ”best” on rarely-occurring schemas like Power & Control.

Schema	Published	Result
Attachment	0.73 [0.70, 0.76]	0.52
Competence	0.76 [0.72, 0.79]	0.56
Global self-eval	0.58 [0.54, 0.63]	0.54
Health	0.75 [0.65, 0.82]	0.69
Power & ctrl	0.28 [0.20, 0.35]	0.65
Meta-cognition	-0.01 [0.00, -0.01]	0.73
Other people	0.22 [0.07, 0.33]	0.72
Hopelessness	0.63 [0.56, 0.68]	0.65
Other’s views	0.58 [0.52, 0.63]	0.64

Table 6: Spearman correlation [95% confidence bounds] for per-schema LSTM model

We tried several measures to change this behavior, including:

- assigning prior weights to each rating based on their inverse frequencies, to make the models less likely to predict 0
- removing utterances of 1-2 words, on the hypothesis that bidirectional LSTMs would not function well on such short inputs
- avoiding special end-of-line markers to better match the original paper’s code

Unfortunately, none of these approaches improved results.

4.2 Claim Results

Each subsection in Section 4.1 highlights discrepancies discovered between published and reproduced models. Each subsection also provides hypothesis as to why these discrepancies occur. When combined, the discrepancies mean that many of the original paper’s claims could not be verified.

Claim 1 (per-schema LSTMs): Due to issues described in Section 4.1.4, our per-schema LSTM model varied the most from the original authors. In general, our model performed poorly, and did not receive the highest score for any Schema.

Claim 2 (regression models for Power and Control): Our findings confirm this result.

Claim 3 (classifiers for Health): Our findings confirm this result.

4.3 Further studies

The following additional studies were conducted to explore hypotheses outside of those of the original authors.

4.3.1 Relaxed Preprocessing

RNNs combined with GLoVE word embeddings are able to capture a significant amount of information. Burger et al. preprocessed their text inputs to a high degree, removing English stop words, punctuation, and contractions. We hypothesized that leaving these words in place would allow GLoVE and our multilabel RNN model access to more information, possibly making it more expressive. For example, we theorized that “I can’t” might carry some semantic difference to “I cannot”, and “a problem” might have different meaning from “the problem”.

We ran the model described in Section 4.1.3 above, using our own preprocessing. We allowed for a slightly larger vocabulary (2100 words) to account for the inclusion of a larger vocabulary. Our preprocessing did not remove stop words, and it allowed common punctuation and contractions.

The results of this experiment are shown in Table 7 below.

Schema	Baseline	Relaxed
Attachment	0.63	0.60
Competence	0.63	0.61
Global self-eval	0.49	0.43
Health	0.35	0.30
Power & ctrl	0.06	0.22
Meta-cognition	0.03	0.04
Other people	0.11	0.13
Hopelessness	0.44	0.35
Other’s views	0.49	0.51

Table 7: Spearman correlation for baseline and relaxed preprocessing

Overall, relaxing preprocessing constraints made little difference. Four Schemas (Power & Control, Meta-cognition, Other people, and Other’s views) made small improvements, while five worsened.

Stop words typically carry little meaning. We theorize that the benefit of including more information was tempered by the additional noise they introduced.

4.3.2 Scenario-level Preprocessing

We theorized multiple utterances from the same patient replying to the same prompt might contain information that would be useful when analyzed together.

We ran the model described in Section 4.1.3 above. We used the same preprocessing described in Section 4.3.1. This time, we concatenated all utterances from each patient for each scenario. We took the mean of the schema scores for the same groups. To account for the combined utterances, we increased the maximum length our model considered from 25 to 45 tokens.

The results of this experiment are shown in Table 8 below.

Schema	Baseline	Scenario
Attachment	0.63	0.50
Competence	0.63	0.71
Global self-eval	0.49	0.09
Health	0.35	0.12
Power & ctrl	0.06	0.08
Meta-cognition	0.03	-0.13
Other people	0.11	-0.11
Hopelessness	0.44	-0.04
Other's views	0.49	0.38

Table 8: Spearman correlation for baseline and scenario utterances

Using scenario-level utterances resulted in significant changes to most schemas. Only one Schema (Competence) showed better results, with all others worsening. We hypothesize that this is due to the significant reduction in training samples we encountered when we combined utterances. These results are also consistent with the original authors' finding that utterance/schema correlation did not improve when considering later utterances on the same scenarios.

5 Discussion

5.1 What was easy

Thanks to Burger et al., the paper's data and codebase were well-documented and easily accessible. Acquiring the material was straightforward.

5.2 What was difficult

We struggled to precisely replicate the authors results. We intentionally switched python packages, but we received more variance than expected. We also noted large changes in results based on hyperparameter changes, such as learning rate, sample, etc.

Despite the paper's transparency with code and data, it proved difficult to rerun the original Jupyter notebook and replicate the paper's results. On a Windows machine, the notebook's Python code had multiple indentation errors. Attempts to fix the errors allowed us to train the multi-label RNNs, but the models made predictions that had nearly zero correlation to ground truth.

The authors' saved-to-disk models worked as expected, however.

Finally, our per-schema RNNs only behaved as majority-class classifiers. See 4.1.4 for details.

5.3 Recommendations for reproducibility

Burger et al. sets an excellent example for reproducibility. We recommend other authors follow this example.

6 Communication with original authors

The corresponding author was contacted on 5/1/2021. We offered to share our project results upon completion, and we detailed the issues we encountered rerunning the paper's Jupyter notebook (see 5.2).

The author's response was almost immediate, requesting information about the operating system we used. She noted that one instance of trouble in the authors' original reproduction efforts may have been related to differences in machine environments.

Correspondence will be ongoing.

References

Franziska Burger. 2021. Dataset and analyses for extracting schemas from thought records using natural language processing.

Franziska Burger, Mark A. Neerincx, and Willem-Paul Brinkman. 2021. Natural language processing for cognitive therapy: Extracting schemas from thought records. *PLOS ONE*, 16(10).

Abigail Millings and Katherine B Carnelley. 2015. Core belief content examined in a large sample of patients using online cognitive behaviour therapy. *Journal of Affective Disorders*, 186:275–283.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. pages 1532–1543.

Kurt Tuohy and Michael Miller. 2022. Cs598 deep learning for healthcare final project. https://github.com/mich1eal/cs598_dl4hc.