CS598 DL4HC Reproducibility Project Proposal

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

Presentation link: TODO
Code repository: https://github.com/mich1eal/cs598_dl4hc (Tuohy and Miller, 2022)

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 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

Preliminary results include only the authors' Multi-label LSTM (see Multi-label LSTM for model definition) was implemented. The authors' three remaining models will be implemented in conjunctions with the final submission of this paper, with additional work beyond the original paper's scope presented in Further studies. Once all models have been implemented, the claims involving multiple models (See Addressed claims from the original paper) will be verified. Currently, only nominal values are reported, but implementation of bootstrapped confidence bounds is also planned.

4.1 Hypothesis 1: automatic schema prediction

Preliminary 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 1 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 1: Spearman correlation [95% confidence bounds] for multi-label LSTM model

Preliminary 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.

Overall, the similarity between the published and preliminary reproduced results gives confidence that the general model architecture and training scheme is correct. With additional fine-tuning possible rectification of the differences listed above, a closer match is expected.

4.2 Further studies

The following additional research is planned, to be further defined and implemented with the final submission of this paper:

1. The authors' Multi-label RNN, but without the removal of common words in preprocessing, or with multiple thought records chained together. In the original experiment, most participants gave a sequence of responses to each scenario, and we may find some advantage by chaining each participant's responses together.
2. The above experiment, but using a modern transformer based model such as BERT.
3. Our draft model is highly sensitive to small changes in the learning rate. We plan to try dynamically adjusting the learning rate and to see if it gives better results. We will also vary the number of training epochs.
4. The authors' Multi-label RNN uses a loss function that expects one-hot encoded class labels. This is very different from the actual schema labels, where thought records can match multiple schemas on a 0-4 scale. We may try a loss function that's better suited to modeling these labels. If we treat both labels and predictions as probability distributions, we may be able to use KL divergence.

5 Discussion

TODO

5.1 What was easy

TODO

5.2 What was difficult

TODO

5.3 Recommendations for reproducibility

TODO

6 Communication with original authors

TODO

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`.