ACARIS: Improving Conversational AI and Human Social Skills using User Embeddings

Simon Slamka
OngakkenAI

June 1, 2023

Abstract

In this paper, we propose ACARIS, the Advanced Communication Augmentor and Relational Insights System, a system utilizing a novel method to analyze emotional state, intent, and interest of text communication parties. ACARIS is being built with the goal of improving social skills of humans, while also improving the performance of human-facing AI systems. We go over our approach, including the initialization of user embeddings from message features, concatenation of user embeddings with word embeddings, modifications of the BERT architecture, and the training and evaluation processes. We also go over the results of our experiments, which demonstrate the effectiveness of our method.

1 Introduction

1.1 Keywords

ACARIS, Conversational AI, Social Skills, User Embeddings, BERT, DistilBERT, Sentiment Analysis, Intent Classification, Emotion Recognition, Interest Recognition

1.2 Definitions

- **ACARIS** - the Advanced Communication Augmentor and Relational Insights System
- **NLP** - Natural Language Processing - a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data
- **Deep Learning** - a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input

- **SVM** - Support Vector Machine - a supervised machine learning model that uses classification algorithms for two-group classification problems
- **DT** - Decision Tree -
- **LogReg** - Logistic Regression - a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist
- **RF** - Random Forest - an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees
- **TF-IDF** - Term Frequency - Inverse Document Frequency - a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus
- **the Transformer** - a deep learning architecture introduced in 2017, used primarily in the field of NLP[1]
- **BERT** - Bidirectional Encoder Representations from Transformers - a transformer-based machine learning model for natural language processing pre-training developed by Google[2]
- **DistilBERT** - DistilBERT is a smaller, faster, cheaper version of BERT developed by HuggingFace[3]
- **Vector** - a quantity that has both magnitude and direction
- **Vector Space** - a collection of vectors, which may be added together and multiplied ("scaled") by numbers, called scalars
- **Embedding Space** - a vector space with a coordinate for each word in the vocabulary, such that words that share common contexts in the corpus are located close to one another in the space
- **Word Embedding** - A vector representation of a word's meaning
- **User Embedding** - A vector representation of a user's personality, emotional state, intent, and interest
- **ReLU** - Rectified Linear Unit - an activation function that returns the input value if it is positive, otherwise it returns zero
- **Loss** - a number indicating how bad the model's prediction was on a single example
- **Cross-Entropy Loss** - a loss function used for classification problems

- **Adam** - an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights
- **Ablation Study** - a study in which specific components of a system are removed to analyze their impact on the overall performance

1.3 Motivation

I've always had an issue with interpersonal relationships. I could never fully understand the way people work on a social level. I understand the technical and biological fundamentals, and superficially, how the mind works, but not how all this becomes so much more complex when actually talking and dealing with people. Sometimes, I feel that another person and I broadcast on different frequencies, because we are unable to either understand one another or maintain a (romantic) relationship for an extended period of time. I have tried for many years to find a technical solution to this problem, but only after coming to Denmark, I acquired the core knowledge to try and accomplish this ambitious goal by really getting into it and self-studying as much as I could.
I firmly believe that all things in nature are governed by rules and these rules can be observed, measured, and then, based on that data, predicted. This includes human behavior and emotions. We're nothing more than complex electrobiochemical machines driven by electrical impulses and hormones. Apart from that, even if we don't understand our own minds yet, we still have many, many years of knowledge about how human personalities work and how we make decisions based on what happens to or around us. On a fundamental level, this doesn't change. Sure, they say that we're all different. However, the core remains. We all share many attributes that make us human. I believe that we, given enough data, can use these attributes to predict human behavior, reactions, emotional state, and intent.

1.4 Hypothesis

Given enough conversational data per person, human behavior (emotional state, intent) in text communication can be predicted with a high level of confidence ($> 80\%$) due to the fact that humans are, on a fundamental level, very similar to one another.

Additionally, we posit that person-specific performance improvements can be achieved by individualizing predictions by using a unique vector representation of a person's personality, emotional state, intent, and interest, which we call a user embedding. The postulate stems in the core concept of neural networks being universal function approximators. Therefore, in theory, a neural network should be able to learn to associate a user embedding with a person's behavior in text communication and use that information to improve its predictions.

1.5 Premise

Interpersonal communication has always been an integral part of human lives and is critical from the moment we're born. With the rise of the Internet and, subsequently, social media and other forms of online text communication, the manner in which we talk has changed dramatically. This has led to a drop in social skills in humans, particularly those of the last generation. ACARIS attempts to adjust for this by providing a way to analyze the emotional state, intent, and interest of text communication parties, providing them with a way to improve their social skills, while also improving the performance of conversational AI systems, which are becoming increasingly prevalent in our society, and their ability to understand human emotions, intent, and interest is becoming more and more important, especially in AI systems that directly interact with humans, such as digital assistants, chatbots, and others.

1.6 Literature Review and Related Work

1.6.1 Studied Literature

- **Machine Learning with PyTorch and Scikit-Learn**[4] by Yuxi Liu, Vahid Mirjalili, Sebastian Raschka
- **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow**[5] by Aurélien Géron
- **Transformers for Natural Language Processing**[6] by Denis Rothman, Antonio Gulli
-

1.6.2 Related Work

- **Human-AI Collaboration Enables More Empathic Conversations in Text-based Peer-to-Peer Mental Health Support**[7]
- **Context Matters: Recovering Human Semantic Structure from Machine Learning Analysis of Large-Scale Text Corpora**[8]

1.7 Initial Attempts

Our initial attempts of implementing ACARIS included the use of SVMs, DTs, LogReg, and RFs, each paired with TF-IDF in the preprocessing stage. None of these methods have proven to be accurate enough (barely reaching 50% accuracy) for large amounts of complex language, especially casual language that often includes slang, sarcasm, emoji, and similar elements. The primary issue was the lack of these methods' ability to capture semantical and contextual aspects of our dataset. SVMs, DTs, LogReg, and RFs weren't built specifically for sequential data. For sequential data, RNNs and CNNs were classically used. However, due to the long-term dependency problem of RNNs, we made the

decision to not use them, as we believed that they would not be able to capture the long-term dependencies in our dataset. Additionally, we weren't able to guarantee that during inference, there would be no long-term dependencies. Another issue was TF-IDF's lack of word order understanding. Knowing that the Transformer[1] architecture is capable of capturing these aspects due to its positional encodinds and self-attention mechanism, we decided to use it as the basis for ACARIS.

1.8 Short Intro on Transformers

The Transformer is a deep learning model architecture proposed in 2017 by a team of Google Brain/Google Research researchers. It is mainly used in NLP, but it has also found its way into numerous vision projects (Latent Diffusion[9]/Stable Diffusion/LLaVA[10], to name a few). The architecture aims to replace RNNs by using attentions mechanisms (self-attention, in particular) and positional encodings to model long-term dependencies in sequential data.

2 Methodology

In this section, we provide a detailed description of our approach, including the computation of user embeddings, concatenation with word embeddings, modifications to the model architecture, and the training and evaluation processes.

2.1 Dataset

To build our dataset, we collected 100 thousand messages from our Discord server, classified them into 3 classes (pos, neg, neu), and augmented them with the GoEmotions[11] dataset, which contains 58 thousand messages classified into 27 classes. We squashed the 27 classes into 3 classes (pos, neg, neu) and concatenated the two datasets. To minimize error caused by input data, we removed all users with less than 25 messages, URLs, markdown code blocks, Discord mentions and custom emoji, and attachments. The resulting dataset was split into 75% training, 10% validation, and 15% test sets.

2.2 User Embeddings Initialization

For each user u, we loop through all messages m sent by u and extract these features:

- **Mean word count (per all m)** - $\overline{w} = \frac{1}{N}\sum_{i=1}^{N} w_i$, where w_i is the word count of i-th message and N is the total number of messages
- **Vocabulary richness (per all m)** - $r = \frac{U}{W}$, where U is the number of unique words and W is the total number of words
- **Mean emoji count (per all m)** - $\overline{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$, where e_i is the emoji count of i-th message and N is the total number of messages

- **Mean emoticon count (per all m)** - $\overline{em} = \frac{1}{N}\sum_{i=1}^{N} em_i$, where e_i is the emoticon count of i-th message and N is the total number of messages
- **Mean punctuation count (per all m)** - $\overline{p} = \frac{1}{N}\sum_{i=1}^{N} p_i$, where p_i is the punctuation count of i-th message and N is the total number of messages
- **Mean sentiment score (per all m)** - $\overline{s} = \frac{1}{N}\sum_{i=1}^{N} s_i$, where s_i is the sentiment score of i-th message and N is the total number of messages
- **Dominant/prevalent topics (per all m)** - the most common topics in all messages
- **Mean response time (per all m)** - $\overline{r} = \frac{1}{N}\sum_{i=1}^{N} r_i$, where r_i is the response time of i-th message and N is the total number of messages
- **Mean message count (per day)** - $\overline{m} = \frac{1}{D}\sum_{i=1}^{D} m_i$, where m_i is the message count of i-th day and D is the total number of days
- **Mean links (per all m)** - $\overline{l} = \frac{1}{N}\sum_{i=1}^{N} l_i$, where l_i is the link count of i-th message and N is the total number of messages
- **Mean markdown code snippet sections (per all m)** - $\overline{c} = \frac{1}{N}\sum_{i=1}^{N} c_i$, where c_i is the markdown code snippet section count of i-th message and N is the total number of messages
- **Mean abbreviations and acronyms (per all m)** - $\overline{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$, where a_i is the abbreviation and acronym count of i-th message and N is the total number of messages
- **Mean hashtag count (per all m)** - $\overline{h} = \frac{1}{N}\sum_{i=1}^{N} h_i$, where h_i is the hashtag count of i-th message and N is the total number of messages

These features were selected based on what we thought would most accurately represent a user's text communication behavior. We projected that no two users would have the same or closely similar values for all of these features, which would allow us to differentiate between them. Then, we concatenate all the features into a single vector e_u (of size d_e):

$$\begin{aligned}&\texttt{from torch import cat}\\&e_u = \texttt{cat}([\overline{w}, r, \overline{e}, \overline{em}, \overline{p}, \overline{s}, \overline{r}, \overline{m}, \overline{l}, \overline{c}, \overline{a}, \overline{h}], \dim = -1)\end{aligned} \tag{1}$$

2.3 Concatenation of User and Word Embeddings

Given a message m with its word embedding e_w (of size d_m) from a BERT-like pre-trained model, which, in our case, was DistilBERT[3], we concatenate the user embedding e_u to the word embedding e_w to create a joint representation e_{joint} (of size $d_e + d_m$):

$$\begin{aligned}&\texttt{from torch import cat}\\&e_{\text{joint}} = \texttt{cat}([e_u, e_w], \dim = -1)\end{aligned} \tag{2}$$

2.4 Model Architecture Modification

To accommodate the concatenated embeddings, we create a custom input layer with the new input size $d_e + d_m$ within a BERT-flavored model. An alternative that we thought of using was to use a separate fully-connected layer (with ReLU activation) to map the concatenated embeddings to a compatible size:

$$\begin{aligned}
&\texttt{from torch import randn} \\
&\texttt{from torch import relu} \\
&\texttt{from torch import matmul} \\
&W1 = \texttt{randn}((d_e + d_m, d_m)) \\
&b1 = \texttt{randn}(d_m) \\
&e_{\text{map}} = \texttt{relu}(\texttt{matmul}(e_{\text{joint}}, W1) + b1)
\end{aligned} \tag{3}$$

Then, pass e_{map} through the existing model architecture.

2.5 Training with User Embeddings

For each training step, we pass the concatenated embeddings e_{joint} through the model and compute the cross-entropy loss L. Then, we backpropagate the gradients from the loss to the model parameters and update the parameters using Adam. The model thus learns to associate every message m with a user embedding e_u and a sentiment label y.

2.6 Updating User Embeddings

The user embeddings e_u are not updated during training. Instead, they are precomputed during data preprocessing and stored in an embedding matrix. We repeat training every week, using new data from the previous week to update the user embeddings e_u. The reason for this decision is that we want the user embeddings e_u in a static state to prevent any potential issues with losses. We considered updating the user embeddings e_u during training, but we decided against it because we believe that it would be too computationally expensive and would not provide any significant benefits. It could actually worsen the model's performance, as the user embeddings e_u would be constantly changing, which could lead to the model being unable to learn to associate messages m with user embeddings e_u.

3 Evaluation

NaN

4 Validation

To validate our approach, we compared the performance of our model to the performance of a baseline model. We used the same model, DistilBERT, but without the user embeddings e_u. We fine-tuned both models on the same dataset and compared their performance on the same test set. The base model performed as follows (mean scores over 5 training runs):

- **Accuracy** ≈ 0.95
- $\textbf{Precision}_{\text{mean of all three scores}} = 0.93$
- $\textbf{Recall}_{\text{mean of all three scores}} = 0.93$
- $\textbf{F1}_{\text{mean of all three scores}} \approx 0.93$

The model with user embeddings e_u performed as follows (mean scores over 5 training runs):

- **Accuracy** ≈ 0.97
- $\textbf{Precision}_{\text{mean of all three scores}} = \text{NaN}$
- $\textbf{Recall}_{\text{mean of all three scores}} = \text{NaN}$
- $\textbf{F1}_{\text{mean of all three scores}} \approx \text{NaN}$

5 Results

NaN

6 Conclusion

NaN

References

[1] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[3] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].

[4] Sebastian Raschka et al. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd, 2022.

[5] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.

[6] D. Rothman and A. Gulli. *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, and GPT-3*. Packt Publishing, 2022. ISBN: 9781803243481. URL: https://books.google.de/books?id=u9FjEAAAQBAJ.

[7] Ashish Sharma et al. *Human-AI Collaboration Enables More Empathic Conversations in Text-based Peer-to-Peer Mental Health Support*. 2022. arXiv: 2203.15144 [cs.CL].

[8] Marius Cătălin Iordan et al. *Context Matters: Recovering Human Semantic Structure from Machine Learning Analysis of Large-Scale Text Corpora*. 2022. DOI: https://doi.org/10.1111/cogs.13085. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cogs.13085. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cogs.13085.

[9] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[10] Haotian Liu et al. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV].

[11] Dorottya Demszky et al. *GoEmotions: A Dataset of Fine-Grained Emotions*. 2020.