Classifying Emotions in Real-Time

Stanford CS224N {Custom} Project

Arnav Joshi
Department of Computer Science
Stanford University
ajoshi21@stanford.edu

Ore Popoola
Department of Computer Science
Stanford University
ore@stanford.edu

Abstract

Currently, deep learning systems have difficulty understanding human emotion in real-time. This difficulty has negative implications in a variety of real-word situations such as chatbots and virtual assistants [1]. The goal of this project is to resolve this situation by building a system that can understand human emotion in real world dialogues. To tackle this problem, we take advantage of the EmotionLines corpus which consists of dialogues labeled by utterance. We define our task to be real-time utterance-level emotion recognition (ULER), with real-time meaning that our system only can see previous utterances within a dialogue. Ultimately, we were able to both build a series of multi-level models and fine-tune BERT on a few different tasks to achieve improvement on the CNN baseline from the EmotionLines paper. Finally, in anticipation of future work, we collected a dataset of brief dialogues between users and virtual assistants labeled by errors. Our hope is that by using real-time ULER, future systems can learn to associate user emotions, such as surprise or anger, with virtual assistant errors.

1 Key Information to include

- External mentor (if you have any):Parastoo Abahi and Jackie Yang

2 Introduction

Emotion recognition is a difficult problem primarily due to the contextual nature of emotion. Specifically, common utterances like "what?" can have totally different meanings in various situations. For example, in many cases, "what?" would indicate surprise but it could also be neutral or even representative of anger, joy, or fear. As such, incorporating context is essential when attempting to accurately classify emotion, which is why we chose to work with the EmotionLines corpus. EmotionLines opts to label utterances within the context of the surrounding dialogue and recommends two baselines: one that makes decisions on the utterance alone (without context) and a bidirectional one that uses both prior and future context within the dialogue. While we also want to incorporate context, we decided to address the problem of real-time emotion classification, and, as such, only incorporate prior context. We made this decision because, in many real world situations, it is impossible to incorporate future statements when making emotion classification decisions. Systems like chatbots and virtual assistants that are tasked with communicating effectively with users must be in tune with the users emotional state and have no ability to see the future.

We hypothesize that using more recent developments in NLP such as BERT [2] and GPT[3], we can build systems that demonstrate improvement on current utterance-level emotion recognition (ULER) systems and effectively incorporate prior dialogue context.

Fundamentally, building such systems is critical because NLP systems that interact with users perform better when they have empathy. Specifically, being able to pick up on emotional cues can allow systems to better respond to users and better resemble human interaction. One specific example of

this paradigm is in relation to virtual assistants. In this area, a large problem is the inability to detect user frustration and identify the mistakes that caused this frustration[1]. Having effective real-world emotion detection systems presents the opportunity to detect when a user responds with anger or confusion, and, hence, correct the mistake before the conversation is derailed entirely. In fact, we look into this problem specifically and collect a dataset of virtual assistant errors within the context of user-assistant dialogues. Although we could not collect enough data to fine-tune and evaluate our models on this dataset specifically, we provide the created dataset to aid future work.

3 Related Work

In Ekman et al. [4], Ekman develops a series of studies for the identification of 6 basic emotions of anger, disgust,fear, happiness, sadness, and surprise. This model of universal human emotion is used in a variety of works. Gordeev et al.[5], for example, constructs a CNN with sliding window and max-pooling that is used classify statements as having aggression. This model, however, is limited by the fact it predicts a single binary output instead of classifying a whole range of emotions. Furthermore, this approach can only handle texts of a predetermined size in contrast to solutions that, for example, involve RNNs. Later, Batbaatar et al. 2019 [6] expands upon the idea of using CNN's for emotion classification by proposing SENN (Semantic-Emotion Neural Network). This overarching model includes two sub-networks, a BiLSTM-RNN and a CNN. The networks work side-by-side: the BiLSTM produces "semantic" word embeddings from the sentence, while the CNN produces "emotional" word embeddings of the sentence. These resulting outputs are concatenated before being put through a feed forward layer and softmax. The CNN incorporates a variable window size, and experiments with different filter sizes, with the best performing models having a filter size of 100.

Ultimately, the approach that we take is inspired by the paper for the EmotionLines Dataset[7]. Published in 2018, Emotionlines is a 29,000 dialogue-based set of text utterances labeled on the context of the dialogue. The paper presents strong baselines and the use of the CNN-BiLSTM structure[7]. In comparison, to the Batbaatar et al., the CNN and BiLSTM in the EmotionLines paper are organized into a single multi-level model to allow for contextual understanding. Specifically, the CNN is used to generate general utterance embeddings and the BiLSTM incorporates context from the surrounding dialoge.

In this paper, we take advantage of the advent of transformer-based models[8]. Specifically, we make use of GPT-2 [9], which, since being initially published by OpenAI in 2018, has performed well on a wide variety of language modeling tasks. Much like GPT, BERT[2], makes use of a stack of transformer encoders to build powerful contextual representations of language through bidirectional attention. Both these architectures are pertinent to our work in different ways.

4 Approach

4.1 Task

As discussed, our task involves real-time ULER. Specifically, given a set of dialogues D, each with an ordered set of utterances U_i, our goal is to correctly label each utterance u_j^i given all prior utterances within U_i as context.

4.2 Baseline

The original EmotionLines paper offered two baselines: a CNN and a CNN-BiLSTM[7]. We decided to use the CNN as the baseline because our proposed task, predicting emotions from utterances in real-time, does not allow for looking at future information in the dialogue. Thus, the CNN-BiLSTM model has access to more data at each step than we do. The CNN model from the EmotionLines paper has a weighted accuracy of 0.592; our final models should demonstrate significant gains on this number.

4.3 BERT

4.3.1 Utterance-Level BERT

To make a simple improvement on the baseline, we fine-tuned a pretrained BERT architecture to make utterance-level emotion predictions without any surrounding dialogue context. Specifically, this model defined the training set as datapoints (u_i, e_i) where u_i constitutes an utterance and e_i is the corresponding emotion. The notion of dialogue was discarded entirely. Each utterance was tokenized into tokens w_j^i and put through a BERT-Base-Uncased layer from TensorFlow Hub[10]. Then, the final encoder output of the [CLS] token was taken as a summary and put through two dense layers, the first with 0.5 dropout and the second with 0.4 dropout and softmax activation. The resulting logits were used to predict e_i.

We employed BERT because we felt that the BERT architecture provided an effective method for generating utterance representations.

4.3.2 Contextual BERT

To improve our system, we then aimed to effectively incorporate prior dialogue context to make better predictions. To this end, we decided to take advantage of the short length of the dialogues. All dialogues are under 512 words, which is the max input sequence length of the BERT-Base model[2]. Specifically, we chose to put entire dialogues through a BERT architecture, allowing the model to fine-tune on understanding and using context within a dialogue. However, in line with our original task, we only want to incorporate context from prior utterances.

As such, we built a unique method for defining training examples from the original dataset. Specifically, consider n training dialogues, each with d_i utterances for $1 \leq i \leq n$. We defined d_i examples for each dialogue with example k consisting of the first k utterances for all $1 \leq k \leq d_i$. These k utterances were tokenized and concatenated with [SEP] tokens between utterances. The intended output, then, was the emotion label for the final utterance within the input example. To urge the model to understand this paradigm, the tokens corresponding to the last utterance in the example are encoded with a token type ID of 1 instead of 0. As expected, there are $d_i * n$ total training examples.

Through this method, BERT could bidirectionally attend to all previous utterances for context as well as the utterance it is attempting to classify.

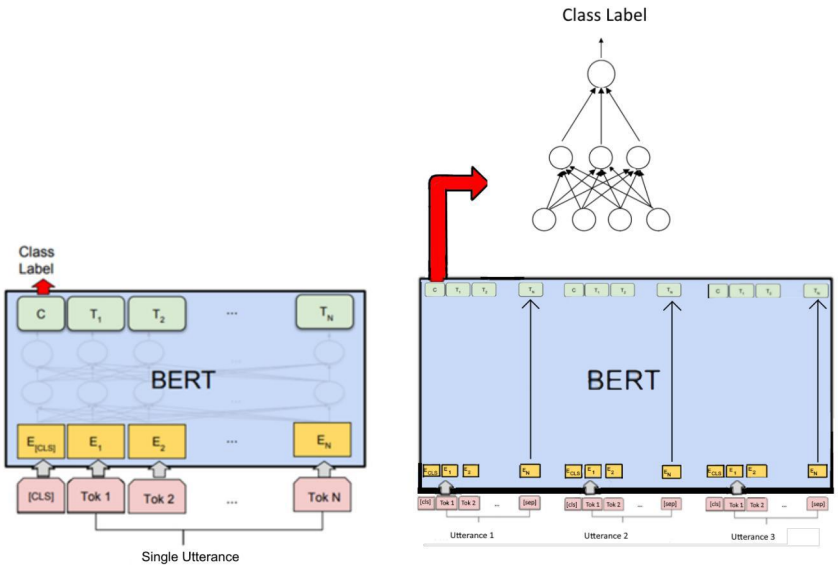

Figure 1: BERT (left) and Contextual-BERT (right) models (Source: Modified from [2])

4.4 Multi-Layer Models

Additionally, inspired by the initial paper[2], we decided to build a series of multi-level models. The general intuition behind these models is to build a lower-level architecture that can produce utterance embeddings and an upper-level architecture that can take these embeddings and use prior utterances to make contextual ULER predictions.

In all cases, the lower level of our model takes a dialogue d_i and parses it into utterances u_j^i, each with tokens w_k^{ij}. For each utterance, the corresponding tokens are converted into word embeddings $e_{w_k^{ij}}$ and put through a model to generate utterance summaries $s_{u_j^i}$. Now, the upper level of our model can take these summaries and build contextual utterance embeddings $e_{u_j^i}$, using prior dialogue context, to make final emotion classification predictions by utterance.

4.4.1 BERT-LSTM

In regards to the model architecture, we used the Tensorflow Hub tokenizer[10] and BERT-Base-Uncased English model to generate utterance summaries from individual utterances. These summaries were put through an additional non-linearity in the form of a linear layer, which, combined with the BERT architecture, constitutes the lower component of the model[2].

The upper component of the model took these summaries and put them through a LSTM[11] to generate final outputs over each utterance. These time-step outputs were put through a final linear layer with softmax to generate classification logits.

To generate summaries, we attempted to use both the standard [CLS] token and max pooling over the final encoder outputs of each word in the given utterance. Max pooling has been shown to be effective at extracting features of the utterance[12]. This feature extraction is useful because emotion is often conveyed by a few distinct words within an utterance.

Furthermore, we ran models that relied on dropout as a form of regularization and ones that relied on layer normalization instead. The models optionally had regularization after the linear layer of the lower model and after the LSTM layer in the upper model.

To train this architecture, we experimented with three different training strategies. First, we attempted the default strategy of training the entire BERT-LSTM model in one session.

Next, we tried to pretrain the lower BERT model and then finetune the BERT-LSTM as one contiguous structure. The pretraining objective was simply to train the model to predict emotion from individual utterances. The structure of this pretraining objective was very similar to our first BERT model. Then, the weights from the BERT layer within this pre-training architecture were frozen and put into the BERT-LSTM prior to final fine-tuning. The intuition behind this approach was to ensure that the lower model could effectively generate utterance summaries prior to training the contextual component.

Finally, we tried fine-tuning the BERT-LSTM architecture for some number of epochs and then freezing the lower BERT model and fine-tuning the LSTM by itself. Intuitively, we felt that since the BERT model was starting with pre-trained weights, it would be able to converge to an appropriate solution after fewer epochs with a lower learning rate. Thus, by training the entire BERT-LSTM model for a few epochs and then freezing BERT and continuing to train with a higher learning rate, we aimed to improve convergence and simultaneously improve the efficiency of training.

4.4.2 BERT-GPT

We then attempted to improve on the BERT-LSTM architecture by replacing the LSTM with a transformer decoder stack. Specifically, we opted to use the GPT-2 architecture provided by the HuggingFace library[13]. GPT-2's attention mask ensures that, once again, the upper level model can only attend to prior utterances. Thus, we can conform to the initial task specification.

In regards to training, we used the first two methods for training the BERT-GPT model. Specifically, we attempted training the entire BERT-GPT at once, and we tried pre-training the lower BERT component and then finetuning the entire BERT-GPT architecture together.

In this case, we decided to train the GPT-2 architecture from scratch because we felt that the input utterance embeddings would be too different from the token embeddings that GPT-2 was pretrained on.

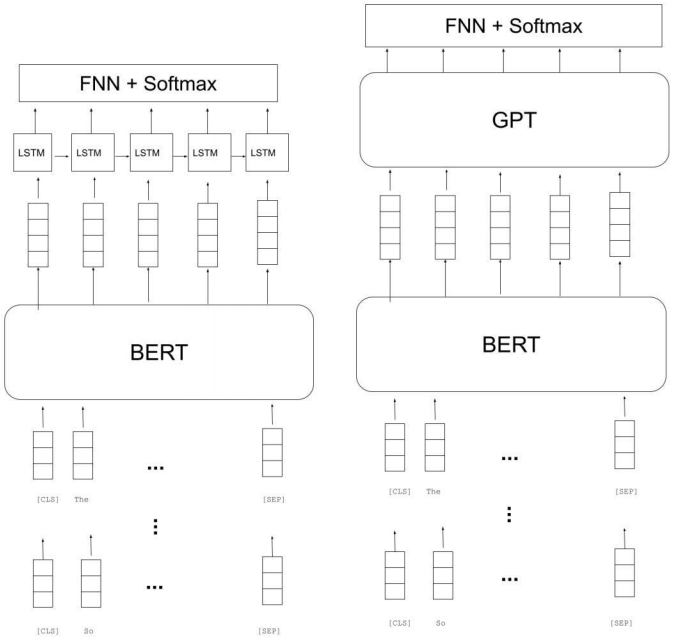

Figure 2: Illustration of Multi-Level Model Architectures

5 Experiments

5.1 Data

5.1.1 EmotionLines Dataset

We chose to work with the EmotionLines dataset [7], which consists of dialogues from the Friends TV Show labeled by utterance.

	# of Utterances	Utterance Length	Emotion Label Distribution (%)							
			Neu	Joy	Sad	Fea	Ang	Sur	Dis	Non
Friends	14,503	10.67	45.03	11.79	3.43	1.70	5.23	11.43	2.28	19.11

Table 1: Friends portion of EmotionLines dataset distribution [7]

5.1.2 Virtual Assistant Dataset

Our goal, initially, was to use our real-time emotion detection system to understand when a virtual assistant has made an error based on the emotional quality of a user response (anger, confusion, etc.). As such, we collected a dataset by transcribing user interactions with a virtual assistant. Participants were given a visual prompt and told to ask a question to a virtual assistant (for example, a timer and number 5 with a command to set a timer for 5 minutes). The system responded correctly at times and incorrectly at other times. For example, the system may respond "Okay, setting an alarm for 5 AM, in response to a prompt for a 5 minute timer. The participant was then told to respond naturally to the error. Ultimately the sentiment of the responses ranged from anger (No, that is not what I said), to

humor (Haha, No I meant Justin Trudeau, not the Intruder). When an error occured and there was no discernible emotion, this usually indicated the user responded telling the assistant to repeat itself.

While we were able to collect a meaningful amount of data, circumstances outside of our control meant that we were not able to collect enough data to both fine-tune and ultimately evaluate our model. As such, we decided to test our models on the test dataset provided by EmotionLines. We hope that future work can expand on the data that we collected and use it in more meaningful ways. Our collection efforts, the resulting data, and a few specific examples can be found in the Appendix section.

5.2 Evaluation method

In our paper, we opt to use the weighted accuracy metric suggested in the original EmotionLines paper.

$$\text{WA} = \sum_{l \in C} s_l a_l \tag{1}$$

where a_l denotes the accuracy of emotion class l and s_l denotes the percentage of utterances in emotion class l. [7] For loss we are using categorical cross-entropy.

$$loss = \frac{1}{U_i} \sum_{j=1}^{U_i} y_j \cdot log(\hat{y_j}) \tag{2}$$

where U_i is the number of utterances in the given training batch which is always a single dialogue.[7]

5.3 Experimental details

For our multi-level models, we ran 26 total experiments while tweaking aspects of the model architecture along with a few hyperparameters. Specifically, we tested different ways of producing utterance summaries, different methods of regularization, and different amounts of training. In regards to utterance summaries, we tested both using the [CLS] token output from the final encoder layer of BERT and max-pooling over the encoder outputs of all tokens in the final layer of BERT. Furthermore, we chose to add regularization after the linear layer from the lower model and after context head in the upper model. We tested both using layer normalization and various amounts of dropout. Finally, we adjusted the training epochs as necessary, based on the loss, although these numbers were much more standardized.

Additionally, we ran models with both types of context heads, LSTM and GPT, and with the different training methodologies outlined in the approach section. All of these models were run with a training batch defined to be a single dialogue, which consisted of approximately 15 utterances on average.

6 Results

The better performing multi-level models are featured below. Training type 1 corresponds to pre-training the lower-level BERT model and then fine-tuning the entire model architecture. Type 2 corresponds to training the entire model as one with no pre-training. Finally, Type 3 corresponds to training the model for some epochs, freezing the lower-level BERT and continuing to finetune the upper-level context head.

Each of the LSTM models has 128 hidden units per layer with a single layer, except the third model in table 2 which has two layers. Furthermore, each was trained with an Adam [14] optimizer with a 10^{-5} learning rate. The one exception was when the model was trained as one contiguous block at first and then the lower-level model was frozen to allow for further fine-tuning of the context head. In this case, the second part of the training session had a $3 \cdot 10^{-5}$ learning rate. Additionally, all models have a linear layer after the BERT summary except for the fourth model. Finally, the GPT architecture is set up with 4 layers and 4 attention heads.

As is obvious from the table 2, our multi-level models that performed well are disproportionately those with LSTM context-heads instead of GPT context-heads. This is both because the GPT models generally performed worse, and, by virtue of this result, we ran more LSTM models in total. This is

Context Head	Training Type	Regularization	Epochs	Summarization	WA
LSTM	1	Layer Norm	(2, 3)	[CLS]	**0.588**
LSTM	1	Layer Norm	(2, 3)	Max Pool	0.570
LSTM	1	Layer Norm	(2, 2)	[CLS]	0.568
LSTM	2	N/A	5	Max Pool	**0.589**
LSTM	2	Dropout (0,0.5)	5	Max Pool	0.565
LSTM	3	Dropout (0.5,0.2)	(3,7)	[CLS]	**0.577**
GPT	1	Dropout (0.5, 0.4)	3	Max Pool	0.548

Table 2: Multi-level models with strong performance

discussed further in the results section.

We also trained a BERT and Contextual-BERT model. Our aim was to train more Contextual-BERT models, but, due to the inefficiency in the definition of training examples, there was an extensive amount of time required to do even a single session. Specifically, since each training example consists of the first k utterances within a training set, utterances are effectively repeated during training. Thus, training took a approximately 12 hours on the GPU.

Model Type	Epochs	Learning Rate	WA
BERT	2	10^{-5}	0.575
Contextual-BERT	2	10^{-5}	0.576

Table 3: BERT Models

Finally, we compare our best contextual model (the BERT-LSTM) against the CNN baseline presented in the paper on the held-out test set. As expected, the BERT-LSTM shows significant improvement.

Model	WA
CNN (Chen et.al)	0.592
BERT-LSTM (ours)	0.623

Table 4: Test Set Comparisons

One interesting note is that the jump in performance between our BERT and multi-level models is not as large as the performance boost between the CNN and CNN-BiLSTM architectures in the original EmotionLines paper [7]. One might reasonably expect a similar increase in performance because, like our models, the original CNN makes predictions purely based on utterance and the CNN-BiLSTM incorporates surrounding context. However, there are two key differences between our models and the EmotionLines models, causing this discrepancy. Specifically, our lower-level BERT model is significantly more powerful than the CNN proposed by EmotionLines, so it might be able to learn some of the things that the BiLSTM learns in the CNN-BiLSTM model. Furthermore, since our task mandates only using prior utterances, we cannot take advantage of bidirectional utterance context gained by the BiLSTM. Taken together, these differences explain the smaller performance boost.

6.1 LSTM vs GPT

In regards to the two types of context heads – LSTM and GPT – we noticed that the LSTM generally has equal or better performance. This is likely due to the size of our dataset; with only roughly

14,000 utterances, training a GPT based stack proved to be difficult. Indeed, research suggests that for smaller datasets training transfomers from scratch is not nearly as effective[15].

We considered using a pretrained GPT model and substituting our own embeddings generated by the lower-level BERT model, but this approach was abandoned quickly. Perhaps future work could dedicate more time to this method.

6.2 BERT vs Contextual-BERT

The Contextual-BERT model only demonstrated a very marginal improvement on the basic BERT architecture. One possible reason for this could be related to difficulties resulting from the unconventional structure of the data. Specifically, during BERT pre-training, the token type IDs were used to identify two separate sentences to determine if the sentences are sequential [2]. In our model, however, the token type IDs were used to segregate the prior dialogue context from the actual utterance that the model should classify. These use cases may be too different for the gradients to lead to an effective solution.

7 Analysis

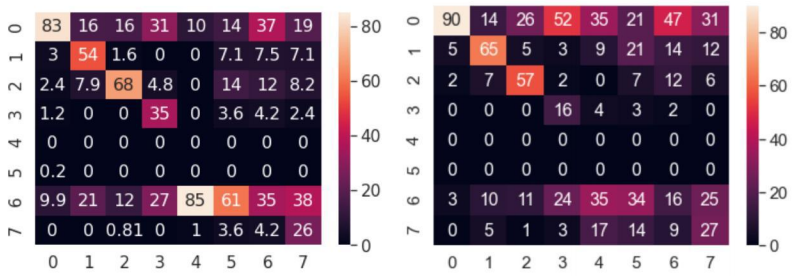

Figure 3: BERT-LSTM (left) and non-contextual BERT (right) heatmaps

We analyzed the results from our best contextual model, the BERT-LSTM, and our non-contextual BERT model on the dev set. The presented heat maps show the accuracy of this BERT-LSTM model compared to the accuracy of the BERT model. The columns represent the correct labels and the rows represent the actual labels and the values are normalized into percentages by column. The numeric labels correspond to emotions as follows: zero is neutral, one is surprise, two is joy, three is sadness, four is disgust, five is fear, six is non-neutral, seven is anger.

When we analyzed our results, we discovered a few patterns that were consistent across the variety of models that we built. One of these key similarities had to do with the non-neutral label in our dataset. The dataset used this eight extra label for situations in which the MTurkers that defined ground labels were able to identify emotion but could not classify it into a specific type of emotion. Both our models that incorporated dialogue context and the basic BERT model seem to have relied on this category in a similar way. In our BERT-LSTM model that had the best performance on the dev set, for example, the emotion 'surprise' within the dev set was accurately classified 54% of the time but was given the label 'non-neutral' and additional 21% of the time. This pattern hold across other emotions and reflects real-world explainability: the model recognized emotion but cannot quite categorize it completely. Simultaneously, while this same model only classifies 35% of non-neutral labels correctly, it predicts neutral 37% of the time. This seems to suggest that many of the utterances labeled non-neutral in the original data may have been those that had only a slight emotional tinge and are similar to those labeled neutral.

Additionally, we can see the contextual models do indeed make use of prior context. A perfect example of this is the utterance "what?". This individual utterance occurs a total of 17 times in the dev set, 10 of which have the label "surprise". The BERT model that does not incorporate context simply predicts "surprise" every time and, as a result, gets some wrong. The best BERT-LSTM, on the other hand, occasionally picks "neutral" or "non-neutral" and gets some correct that the BERT model did not.

Furthermore, as seen with the heat maps, the model best BERT-LSTM models performs very poorly on fear and disgust, which are both underrepresented in the dataset. This is because our loss function and accuracy metrics

8 Conclusion

In this project, we have evaluated several approaches for classifying emotion by utterance using prior context. Training on the EmotionLines-Friends dataset, we have built several strong models for emotion recognition applicable to error detection that account for context inside a dialogue. We compare the various architectures we developed towards this task, having a BERT-LSTM configuration with no pretraining as the highest performing classifier with a Weighted Average of 0.589. We build a small evaluative dataset of context-based user interaction with a virtual assistant, which was size-limited by collaborators failing to deliver.

Some future work includes expanding the evaluative dataset for interactions between virtual assistants and humans, to get a more accurate understanding of the effectiveness. With enough samples, it could be expanded to the point it could be used for training an improved model. Other areas to look into are evaluating the performance of a pretrained GPT model with Bert embeddings.

References

[1] Waiber Suhm. Multimodal error correction for speech user interfaces. *ACM Transactions on Computer-Human Interaction*, 8:1.6:1.1–1.6:1.64, February 2001.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. cite arxiv:1810.04805Comment: 13 pages.

[3] A. Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[4] P. Ekman and W.V. Friesen. Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17(2):124–129.

[5] Rodmonga Potapova and Denis Gordeev. Detecting state of aggression in sentences using cnn, 2016.

[6] E. Batbaatar, M. Li, and K. H. Ryu. Semantic-emotion neural network for emotion recognition from text. *IEEE Access*, 7:111866–111878, 2019.

[7] Chao-Chun Hsu, Sheng-Yen Chen, Chuan-Chun Kuo, Ting-Hao (Kenneth) Huang, and Lun-Wei Ku. Emotionlines: An emotion corpus of multi-party conversations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, 2018.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[12] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, 2016.

[13] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[15] Jordan J. Bird, Diego R. Faria, Anikó Ekárt, Cristiano Premebida, and Pedro P. S. Ayrosa. Lstm and gpt-2 synthetic speech transfer learning for speaker recognition to overcome data scarcity, 2020.

A Appendix

Below is an example from the extra dataset that we built. A conversation begins with a user is prompted through images to give a specific command. After giving the command, the a virtual assistant is made to occasionally intentionally fail, and the user response is given. As the machine response, as and the user query cannot "fail" or "succeed", the label is placed on the User Response for the dialogue.

User Query:	Can you show me a picture of Joshua Tree	LABEL N/A
Assistant Response:	Here is a picture of Justin Pierre James Trudeau, a Canadian politician	LABEL: N/A
User Response:	Not Justin Trudeau, Joshua Tree please	Label: FAILURE