Fine-Tuning Conversational Agents: Modeling Personality through Character-Based Dialogue

Matteo Fare' ID: 989345

University of Milan, Department of Computer Science.

Contributing authors: matteo.fare1@studenti.unimi.it;

Abstract

This work explores the integration of personality into conversational agents by fine-tuning `DialoGPT-small` on two contrasting datasets: the Cornell Movie-Dialogs Corpus and character-specific dialogue from the Friends TV show. The goal is to compare generic versus persona-specific behavior in dialogue generation. Using Hugging Face's `Transformers` and `PyTorch`, a full training pipeline was implemented. Evaluation includes Loss, Perplexity, BLEU, ROUGE-L, BERTScore, and emotion alignment via a DistilRoBERTa-based classifier grounded in Ekman's six basic emotions.

Keywords: Chatbot, DialoGPT, Dialogue generation, Personality modeling, Transformers

1 Introduction

Conversational Agents (CAs), commonly known as chatbots, are systems designed to engage users via natural language, simulating human-like interactions. Within the fields of **Artificial Intelligence** (AI) and **Natural Language Processing** (NLP), CAs represent a particularly complex challenge due to the nuanced and context-sensitive nature of human language.

The objective of this work is to explore how **personality** can be embedded into a neural chatbot's behavior through dataset-specific fine-tuning. Personality, while abstract, can be approximated through consistent linguistic patterns, emotional tones, and thematic preferences. This study adopts a data-driven approach using the pre-trained **DialoGPT-small** model, a dialogue-specialized variant of `GPT-2`, trained on

147 million Reddit conversations. This model is known for its conversational fluency and contextual relevance, making it a robust base for persona modeling.

This project builds upon the recent survey by Sutcliffe [1], which classifies techniques for embedding personality, persona, and user profiling within conversational agents.

Two distinct sources are employed for training:

- The **Cornell Movie-Dialogs Corpus**, used to learn a general "cinematic character" style without targeting a specific identity.
- The **Friends TV Show** dataset from Kaggle, from which character-specific subsets for **Joey Tribbiani** and **Phoebe Buffay** were extracted to construct more explicitly defined personas.

The fine-tuning pipeline is implemented using Hugging Face's `Transformers` library and PyTorch, with data preprocessing routines to clean, segment, and contextualize dialogue for training. Models are optimized over five epochs using causal language modeling objectives and evaluated using standard metrics: **Loss**, **Perplexity**, **BLEU**, **ROUGE-L**, and **BERTScore**. In addition, emotional alignment is evaluated using a DistilRoBERTa-based classifier grounded in Ekman's six basic emotions.

Results indicate moderate success in personality alignment, in particular with Joey and Phoebe bots exhibiting character-consistent traits in both lexical choices and emotional tone. Limitations are observed in the form of low metric scores. Finally, an interactive terminal-based interface was developed to allow users to engage with the trained bots.

2 Background and Evolution of Conversational Agents

Conversational agents (CAs), also known as **chatbots**, are software designed to engage in **natural language dialogue** with humans. Their primary function is to interpret user inputs and generate coherent, context-aware responses. This concept is deeply rooted in the history of **Artificial Intelligence (AI)**, most notably illustrated by the **Turing Test** proposed by Alan Turing in 1950 [2]. The test suggests that a machine demonstrates human-level intelligence if it can engage in conversation indistinguishably from a human interlocutor.

Early CA systems were symbolic and **rule-based**. A well-known example is **ELIZA** [3], developed in the 1960s, which simulated a psychotherapist by leveraging pattern-matching rules.

A major turning point came with the rise of **data-driven** approaches and the adoption of **neural networks**. In particular, **Sequence-to-Sequence (seq2seq)** models [4], originally developed for **machine translation**, enabled end-to-end training of dialogue systems. These models map input sequences to output sequences using recurrent architectures, but often suffer from limitations such as poor handling of long-term dependencies and lack of consistency. Notably, seq2seq-based architectures were employed in early works on personality-aware dialogue generation, such as in the project by Nguyen et al. [5], which served as a foundational inspiration for the present study.

The issues of seq2seq, were alleviated by the introduction of the **Transformer** architecture [6], which relies on **self-attention mechanisms** to capture global context and enable parallel sequence processing. This architecture became the backbone of several large-scale **pretrained language models**, including **BERT** [7], **GPT-2** [8], and **GPT-3** [9]. More recently, **GPT-4** [10] has emerged as a state-of-the-art autoregressive language model, demonstrating advanced reasoning, contextual adaptation, and few-shot learning abilities across diverse domains.

A notable adaptation for dialogue is **DialoGPT** [11], which fine-tunes GPT-2 on 147 million Reddit conversations to specialize in open-domain human-like dialogue. In this project, DialoGPT-small—the smallest variant of the model with 117M parameters (compared to the medium and large versions)—serves as the language model that was further fine-tuned on curated datasets to imbue distinct personalities into the conversational agent.

3 Datasets

To instill personalities into a conversational language model, this project utilizes two datasets: the Cornell Movie-Dialogs Corpus and a dataset derived from the Friends TV Series Screenplay, available on Kaggle.

3.1 Cornell Movie-Dialog Corpus

The **Cornell Movie-Dialog Corpus** [12] is a well-established dataset that comprises over 220,000 conversational exchanges extracted from movie scripts, featuring more than 300 characters across 600 films. This corpus captures cinematic language rich in dramatic tone, diverse topics, and informal conversation patterns. The preprocessing pipeline, implemented in the notebook `load_and_process_movie_corpus`, involved the following key stages:

- **Parsing and Structuring Conversations:** Dialogue lines were grouped by `conversation_id` using the function `parse_conversations()`, which ensures chronological consistency by sorting by line ID.
- **Utterance Processing:** Texts were normalized using `clean_dialogue()`, which expanded contractions, removed noise, and standardized punctuation.
- **Context-Response Pair:** The function `create_context_response_dataframe()` generated training pairs using a sliding window (default: 5 turns of context) to simulate multi-turn conversations, aligning with the format required for causal language modeling.
- **Splitting and Exporting:** Using `split_dataset()`, the full corpus was divided into training (80%), validation (10%), and test (10%) sets for evaluation, and then saved in CSV format via `save_splits()` and stored for later.

To have a grasp at the linguistic characteristics of the corpus, are proposed some visual analysis:

- **Word Count Distribution:** The function `plot_text_length_distribution()` plots a histogram about the utterance lengths. As shown in Figure A1, the distribution of utterance lengths exhibits a strong skew toward brevity, with most lines containing fewer than 15 words.
- **Conversation Length Analysis:** `plot_conversation_length_analysis()` produces a scatter plot of conversation lengths. Figure A2 reveals that the majority of conversations are short (under 10 turns), specifically the most of them endures only two turns.
- **WordCloud:** Using `plot_wordcloud()`, a visual summary of lexical frequencies in the corpus was generated, as shown in Figure A3.

3.2 Friends TV Series Dataset

The **Friends TV Series Dataset** comprises the complete screenplay scripts from all ten seasons of the American sitcom *Friends*, sourced from a publicly available Kaggle repository [13]. The dataset features high-quality, well-structured dialogue attributed to distinct characters, making it a valuable resource for training conversational agents with clearly defined personalities. The Preprocessing of the dataset was implemented in the notebook `load_and_process_friends_dataset`, following similar steps to the processing of the previous dataset, as reported below:

- **Loading script lines:** The full script was loaded and iterated over using `load_friends_dataset()`, extracting each line of dialogue.
- **Utterance Processing**: The `process_friends_dataset()` processes and refines unprocessed script data by eliminating stage directions, extraneous whitespace, and parenthetical annotations. It standardizes special characters and excludes improperly formatted lines. Dialogue lines are parsed to distinguish speakers from their utterances, and only legitimate main characters are preserved using `extract_main_characters()`. Lines involving multiple speakers are replicated for each speaker. Further normalization is applied to maintain uniform punctuation and spacing.
- **Context-Response Pair**: In a similar way to the previously discussed dataset, the function `generate_character_context_csv()` reformats the dialogue into a response-context structure tailored for each selected main character.
- **Splitting and Exporting**: Again, each character-specific dataset was split into training (80%), validation (10%), and test (10%) sets for evaluation, and then saved in CSV format via `save_splits()` and stored for later.

Also in this case, to analyze the linguistic structure of the processed data, several visualizations were created:

- **Word Count Distribution:** The function `plot_text_length_distribution_friends()` plots a histogram about the utterance lengths. As shown in Figure A4, most utterances ranged between 5–20 words.
- **Character Appearance Frequency:** `plot_count_character_appearances()` showed the distribution of utterances per character. The result is visible in Figure A5.

- **WordCloud:** For each selected character, `plot_wordcloud()` was used to extract and visualize the dominant lexical features, as shown in Figure A6 and in Figure A7.

4 Brief System Description

This project is built upon the **Transformers** library developed by Hugging Face [14], which provides a unified API[1] for state-of-the-art transformer-based models. The library includes implementations of major architectures such as BERT, GPT, T5, and notably, **DialoGPT**, which serves as the core of this work.

Transformers offers a modular and extensible interface for model configuration, tokenization, training, and inference. It is widely adopted both in academic research and industry applications due to its ease of use, extensibility, and integration with PyTorch and TensorFlow. Overall, the Transformers library was designed to be extensible by researchers and simple for practitioners, characteristics which made it suitable for the project tasks.

5 Training Process: building the chatbot

To develop a personality-aware neural conversational agent, this project fine-tuned a pre-trained generative language model on preprocessed dialogue datasets. The fine-tuning process aimed to adapt the generic conversational ability of the base model to exhibit character-specific traits using the Cornell Movie-Dialogs Corpus and targeted subsets of the *Friends* TV series scripts.

5.1 Model Architecture: DialoGPT-small

The foundational model employed in this project is **DialoGPT-small**[2], a member of the DialoGPT series [11], which stands for Dialogue Generative Pre-trained Transformer, developed by Microsoft Research as a dialogue-optimized extension of OpenAI's **GPT-2** architecture [8].

DialoGPT was fine-tuned on a massive dataset comprising over **147 million multi-turn conversation pairs** extracted from Reddit. This corpus spans a broad range of topics and conversational styles, allowing the model to internalize diverse linguistic patterns, social cues, and context-dependent language use.

The **small** variant contains approximately **117 million parameters**, which includes 12 Transformer decoder layers, 12 self-attention heads per layer, and an embedding size of 768. Despite being the smallest model in the DialoGPT family (which also includes medium and large variants with 345M and 762M parameters, respectively), DialoGPT-small strikes a balance between computational efficiency and dialogue generation quality, making it well-suited for lightweight applications and prototyping. The small version was selected in this project due to time and practical constraints associated with working on **Google Colab**, which imposes memory and computational limits.

---

[1] Transformers GitHub repository: https://github.com/huggingface/transformers

[2] DialoGPT-small on HuggingFace: https://huggingface.co/microsoft/DialoGPT-small

5.2 Data Preparation and Tokenization

Each input sample consists of a context-response pair, where the **context includes up to five preceding utterances**, and the **response is the character's next line**. These were converted into token sequences using the `AutoTokenizer` associated with DialoGPT. Tokenization was handled by the function `tokenize_and_flatten_conversation()`, which encodes each utterance, appends an end-of-sequence token (EOS), and flattens the sequence in reverse order—prioritizing the most recent dialogue turns.

Token sequences were padded on the **left**, consistent with GPT-style models, which predict tokens from left to right. Padding was applied dynamically during batch collation using the PyTorch utility `pad_sequence()`.

5.3 Dataset Wrapping and Batching

Samples were wrapped using the `ConversationDataset` class, which converted context-response rows into padded sequences. These were further processed into uniform batches via the `collate_function()` using `torch.nn.utils.pad_sequence`. The dataset was then served to the training loop via PyTorch `DataLoader`s, with batch sizes of 8 for training and 16 for evaluation.

5.4 Training Configuration and Optimization

The training pipeline was executed using the function `perform_training_process()`, which followed a standard transformer fine-tuning scheme:

- **Loss Function:** The model was trained using the standard causal language modeling objective where inputs and labels are identical.
- **Optimizer:** The `AdamW` optimizer was configured with a learning rate of $2\text{e}{-5}$, $\varepsilon = 1\text{e}{-8}$, and a weight decay of 0.01, excluding bias and normalization layers.
- **Scheduler:** A linear learning rate scheduler with warm-up was configured using `get_linear_schedule_with_warmup()`.
- **Mixed Precision:** To improve performance, `torch.cuda.amp` was used with automatic mixed precision via `autocast()` and `GradScaler()`.
- **Epochs:** Training was run for `5 epochs`, with evaluation occurring at the end of each epoch.

5.5 Training Results

As stated before, the training process was carried out over five epochs for each dataset, with consistent configuration parameters to enable direct comparison. During each epoch, both the training loss and validation loss were recorded, along with perplexity—a commonly used metric to measure the model's confidence in generating the correct token sequence.

Cornell Dataset.

For the **Cornell Movie-Dialogs Corpus**, the model showed a steady improvement in performance over time. The training loss decreased from 2.1186 to 1.9130, while validation loss improved from 1.6262 to 1.5899 across the five epochs, as shown in Figure A8, with the perplexity that reached a final value of 5.1074 as display in Figure A9. The detailed numerical evolution is presented in Table A1.

Joey Dataset.

In the Joey-specific dataset, the model began with a higher training loss of 2.4902. Over the epochs, the training loss decreased to 1.9698, and the validation loss dropped from 1.8555 to 1.7481. Perplexity similarly improved, ending at 5.9122. These results, shown in Figures A10 and A11, are detailed numerically in Table A2.

Phoebe Dataset.

The model trained on the Phoebe dataset exhibited similar dynamics. Training loss dropped from 2.4891 to 1.9539, while validation loss improved from 1.8906 to 1.7867. The final perplexity was 6.1099. As illustrated in Figures A12 and A13, these results indicate that while the model did learn from the data, Phoebe's dialogue may have introduced more stylistic or semantic variation, reflected in slightly higher perplexity. Full epoch-wise results are reported in Table A3.

5.6 Execution Strategy and Saving

The full training process was managed through the function `execute_training_on_all_dataset()`, which looped through the three datasets defined in `DATASETS_ID = {"cornel", "Joey", "Phoebe"}`, applying the same training strategy. Additionally, the training procedure can be conducted on an individual dataset utilizing functions `execute_training_on_single_dataset()` and `prompt_and_execute_training()`, which afford the user the capability to select the specific dataset for training.

Post-training, the model weights, tokenizer configuration, and vocabulary were saved via `save_model_and_tokenizer()`, exploiting the `save_pretrained` method, that permits reproducibility and later use for inference and evaluation.

6 Evaluation and Personality Assessment

To evaluating the performance and personality alignment of the fine-tuned models, a two-fold evaluation process was conducted: (1) classical language modeling metrics on a test set, and (2) affective and semantic analysis of generated responses.

6.1 Test Set Evaluation

After the completion of training, all three models were evaluated on their respective test splits. This process was handled by the function `perform_model_evaluation()`, which computes the average **loss** and **perplexity** over the test data. The results are as follows:

- **Cornell model:** Loss = 1.5592, Perplexity = 4.9392
- **Joey model:** Loss = 1.6936, Perplexity = 5.5974

- **Phoebe model:** Loss = 1.7258, Perplexity = 5.7866

These results confirm that the Cornell model, trained on a broad cinematic dialogue corpus, generalizes slightly better than character-specific models in terms of next-token prediction accuracy, probably simply due to the larger size of the dataset, which allows for better style matching. Nonetheless, the Joey and Phoebe models exhibit reasonable perplexity levels.

6.2 Linguistic Fidelity Metrics

To evaluate the semantic similarity between generated outputs and ground truth responses, predictions were obtained using the function `generate_predictions()`. These were then compared to reference responses using three standard **NLP** metrics:

- **BLEU**: Measures token-level precision of n-gram overlaps.
- **ROUGE-L**: Captures the longest common subsequence (LCS) between predicted and reference texts.
- **BERTScore (F1)**: Uses contextual embeddings to compare similarity at a deeper semantic level.

The following scores were observed:

- **Cornell:** `BLEU` = 0.0012, `ROUGE-L` = 0.0901, `BERTScore (F1)` = 0.1077
- **Joey:** `BLEU` = 0.0034, `ROUGE-L` = 0.0882, `BERTScore (F1)` = 0.1196
- **Phoebe:** `BLEU` = 0.0026, `ROUGE-L` = 0.0836, `BERTScore (F1)` = 0.1031

Interpretation and Discussion of Low Scores.

Despite their widespread use, BLEU and ROUGE can perform poorly in open-domain dialogue generation tasks [15], as they rely on surface-level token overlap and often penalize valid yet diverse or paraphrased responses. BERTScore improves upon this by comparing contextual embeddings, yet still yielded modest values across all models.

This outcome is perhaps influenced by two factors. First, the use of **DialoGPT-small**, a lightweight model that limits the capacity for generating semantically rich and coherent responses when compared to larger models. Second, the **relatively small size of the fine-tuning datasets**, especially the Joey and Phoebe subsets, restricts the model's ability to generalize and internalize a broad variety of language patterns and response strategies.

Importantly, these outcomes highlight a broader concern in the NLP community: **evaluating dialogue systems using general-purpose metrics remains fundamentally flawed**. Human conversation is inherently ambiguous, emotionally variable, and pragmatically rich. As discussed in recent literature [16], even advanced LLMs exhibit strong performance variation across contexts, and many tasks cannot be reduced to rigid benchmarks. Thus, evaluating a chatbot solely with automated metrics may yield an incomplete or misleading picture of its actual conversational quality and alignment with target personas.

6.3 Emotion Profile Alignment

To evaluate whether the generated responses preserved emotional characteristics of the original speakers, each response and its corresponding reference were passed through a pre-trained emotion classifier, `j-hartmann/emotion-english-distilroberta-base`[3]. This model builds upon `distilroberta-base` [17], a compact version of Facebook's RoBERTa transformer model, making it an efficient and effective choice for emotion detection.

The model outputs a probability distribution over the six core emotions defined by **Ekman's theory** [18], which are briefly described below:

- **Anger**: Response to provocation or injustice, often motivating corrective action.
- **Disgust**: Reaction to offensive or repulsive stimuli, prompting avoidance.
- **Fear**: Triggered by perceived threats, initiating defensive behavior.
- **Joy**: Positive state associated with pleasure or success.
- **Sadness**: Resulting from loss or disappointment, promoting reflection.
- **Surprise**: Brief response to unexpected events, directing attention.

Using the pipeline defined in `get_emotion_scores()`, both the reference and predicted outputs were scored, and their distributions averaged. Radar plots were generated to visually compare the emotional expressiveness across reference and generated responses.

Results:

- **Cornell (Figure A14):** The generated responses closely tracked the emotional distribution of the references. The similarity across all six emotion categories, with a prominence towards *surprise*, suggests that the model generalized well over a broad and diverse corpus, resulting in a relatively *balanced emotional profile*.
- **Joey (Figure A15):** The Joey-specific model exhibited a strong alignment in high-salience emotions such as *joy* and *surprise*, which are emblematic of the character's humorous and often excitable nature. The generated lines preserved these dominant traits quite effectively, validating the utility of persona-driven fine-tuning in retaining emotionally characteristic speech.
- **Phoebe (Figure A16):** While still largely consistent, the Phoebe model showed slightly more divergence, particularly in *disgust* and *joy*. Nevertheless, the alignment in other emotions such as *surprise* and *anger* suggests partial success.

Overall, the radar plots confirm that the emotional tone learned by each model generally aligns with its training target. Demonstrating that fine-tuning can effectively transfer not just linguistic content, but also affective patterns.

7 Let's Chat

To enable real-time interaction with the fine-tuned conversational agents, a dedicated interactive interface was implemented. This module allows users to converse directly

---

[3] HuggingFace Model Repository: https://huggingface.co/j-hartmann/emotion-english-distilroberta-base

with the chatbot models (`cornell`, `Joey`, and `Phoebe`), thus offering a hands-on evaluation of the models' ability.

Model Initialization.

The chatbot system begins by invoking the `load_model_and_tokenizer()` function, which loads the fine-tuned model and its associated tokenizer from the specified directory (e.g., `output/cornell`).

Response Generation.

The core of the chatbot's generative behavior lies in the `generate_response()` function. This function accepts a sliding window of previous utterances (default context window size is 5) to construct the input sequence, which is then fed into the model for generation. Key decoding hyperparameters include:

- `top_p=0.7`: Enables nucleus sampling for probabilistic diversity.
- `top_k=50`: Restricts sampling to the top-k most likely tokens.
- `temperature=0.8`: Controls output randomness.
- `no_repeat_ngram_size=3`: Prevents repetitive sequences, enhancing response naturalness.

Chat Session Management.

The `chatbot_loop()` function drives the user interface for interaction. It continuously reads user input from the terminal, tokenizes and stores the dialogue history, and appends both user and bot messages to maintain conversational context. Termination commands such as `"bye"` end the session.

Dataset-Specific Personality Selection.

Through the `prompt_and_chat()` entry point, users are prompted to select one of the fine-tuned personalities—either the `cornell` model or the persona-specific `Joey` or `Phoebe` models. Based on the selection, the corresponding model and tokenizer paths are loaded, and a chat session is initiated.

Example Session.

Phoebe-bot:

```
>> User: Who are you
>> Bot: Phoebe, I'm so sorry.
>> User: Why are you sorry?
>> Bot: Because I have to go to the bathroom.
>> User: Pizza?
>> Bot: No.
>> User: What do you like to eat?
>> Bot: Well, I just ordered a bunch of Chinese food...
>> User: bye
>> Bot: See you soon!
```

This exchange demonstrates the model's ability to exhibit character-specific speech behavior, in line with Phoebe Buffay's personality from the Friends TV show. While the model occasionally generates ambiguous or contextually awkward responses (e.g., "Yeah, well, I'm really sorry"), such outputs are expected given the relatively small model size and modest fine-tuning dataset.

Joey-bot:

```
>> User: Who are you?
>> Bot: I'm Joey Tribbiani.
>> User: Nice to meet you, I'm Bob
>> Bot: I don't know. I guess I'm just really in love with the
guy.
>> User: Ok...
>> Bot: Hey, look, it's just that this is a very big commitment.
>> User: What would you like to do?
>> Bot: Well, I'm going to the movie with the director, so I
guess.
>> User: Because you are an actor, right?
>> Bot: Yeah, I guess so.
>> User: You are a good actor
>> Bot: Yeah.
>> User: bye
>> Bot: See you soon!
```

The Joey model recreates the iconic persona of Joey Tribbiani. References to acting and his comical sense of romance are notably present, aligning well with his television character. While some responses drift semantically (e.g., responding "I'm just really in love with the guy"), the general tone, vocabulary, and conversational rhythm resonate with Joey's well-known identity.

Cornell-bot:

```
>> User: Who are you?
>> Bot: who are i?
>> User: Yes. I would like to know your name.
>> Bot: john cena.
>> User: Wow the famous wrestler! Nice to meet you.
>> Bot: you’re a very good friend.
>> User: Well wow just met, but thank you. You seem nice too.
>> Bot: you’re very welcome.
>> User: So would you like to eat an ice-cream?
>> Bot: i’m not eating an icecream.
>> User: Are you on a diet?
>> Bot: i’m on a strict diet.
>> User: I understand. What would you like to eat?
>> Bot: chicken and cheese.
>> User: Ok, I know a very nice place.
>> Bot: what?
>> User: For dinner
>> Bot: i’m going to eat.
>> User: bye
>> Bot: See you soon!
```

This interaction represents one of the best responses obtained from the “`Cornell-bot`”. The conversation exhibits a logical progression, featuring an emergent (though totally unintentional) character who identifies as “John Cena”.

Limitations and Final Remarks.

While the interactive chatbot interface allows for real-time conversations with character-aligned dialogue models, several limitations emerged during empirical testing. Most notably, the fine-tuned models—particularly the `cornell` variant—often failed to generate contextually appropriate or semantically coherent responses.

Similar inconsistencies were observed with the `Phoebe` and `Joey` models. For example, in one session, the `Phoebe` bot initially introduced itself correctly but subsequently claimed to be “Rachel”, undermining the persona conditioning. In another dialogue for example, the `Joey` bot failed to respond or produced generic responses without clear linkage to the prior conversational context. These breakdowns in coherence were frequent, with the bot occasionally becoming non-responsive or abruptly ending the dialogue with irrelevant interjections like “I don’t know” or “I’m sorry.”

Despite these limitations, the project demonstrates the feasibility of persona-conditioned dialogue generation using current LLM tooling, building a good starting point. The interactive interface serves as a useful benchmark for evaluating linguistic fluency, emotional alignment, and persona-specific behaviors in conversational agents.

8 Conclusion

This project explored the development of neural conversational agents enhanced with character-based personality conditioning. By fine-tuning a pre-trained **transformer-based language model**—specifically `DialoGPT-small`—on carefully prepared dialogue corpora, the work aimed to generate chatbot responses that reflect personality traits.

Two major datasets were utilized for this purpose: the **Cornell Movie-Dialogs Corpus**, representing a diverse and generalized cinematic dialogue style, and a curated script collection from the **Friends TV Series**, used to emulate the speech patterns of specific characters—*Joey Tribbiani* and *Phoebe Buffay*.

Model training was conducted using `PyTorch` and the `HuggingFace Transformers` library, incorporating `AdamW` optimization, learning rate scheduling, and **mixed precision training** for efficiency. The training pipeline produced models capable of generating multi-turn, context-aware responses. Evaluation was performed using standard metrics including **Loss**, **Perplexity**, **BLEU**, **ROUGE-L**, and **BERTScore-F1**, alongside emotional profiling via a `distilRoBERTa`-based classifier trained on Ekman's six basic emotions.

The results demonstrate promising outcomes. Perplexity scores showed consistent decreases across all datasets, suggesting improved language modeling capability. Emotion spectrum analysis revealed that the fine-tuned models—especially Joey and Phoebe—were able to approximate the emotional profiles of their respective reference dialogues, confirming the viability of persona-specific fine-tuning.

However, some challenges have been encountered, with results that have not always been promising. The relatively small size of the fine-tuning datasets and the compact nature of the `DialoGPT-small` model may have limited the expressive depth and contextual memory of the chatbot, obtaining sometimes poorly results and responses.

Appendix A Images and Tables

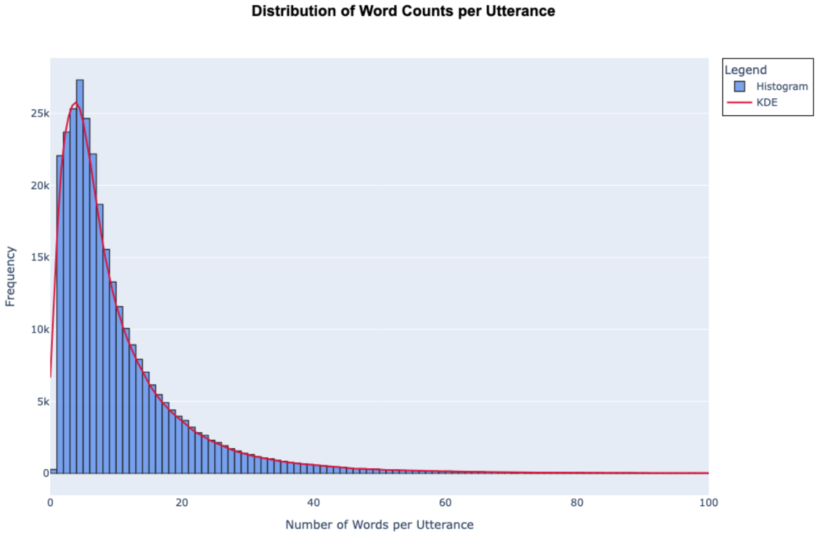

Fig. A1 Word Count per Utterance Distribution of Cornell Movie-Dialogs dataset.

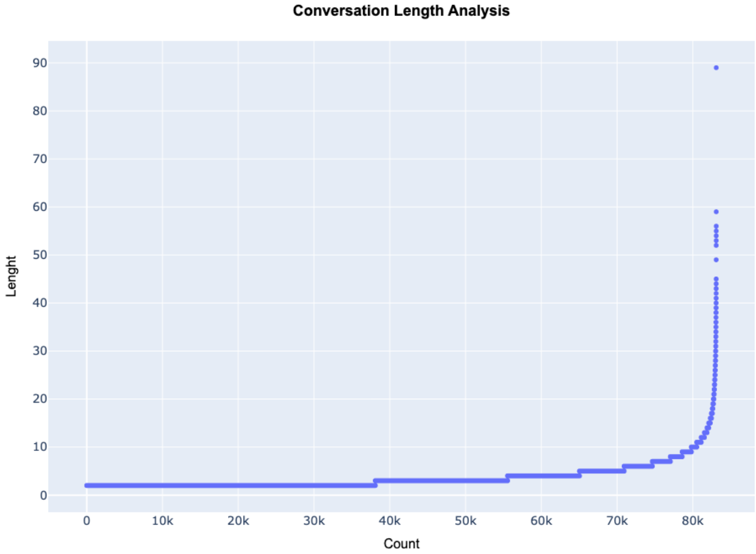

Fig. A2 Conversation Length Analysis of Cornell Movie-Dialogs dataset.

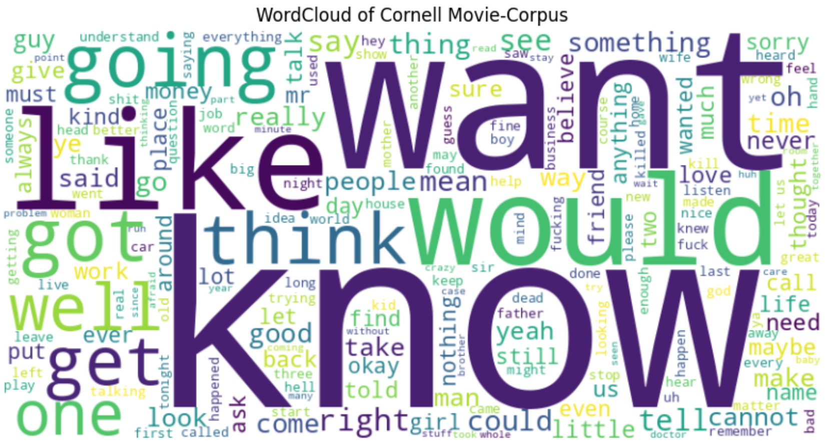

Fig. A3 A visual summary of the dominant lexical features of Cornell Movie-Dialog dataset.

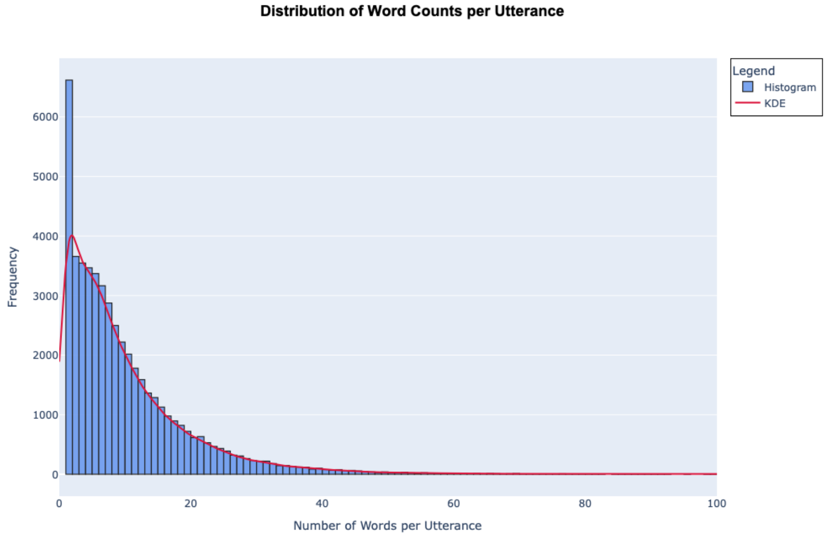

Fig. A4 Word Count per Utterance Distribution of Friends dataset.

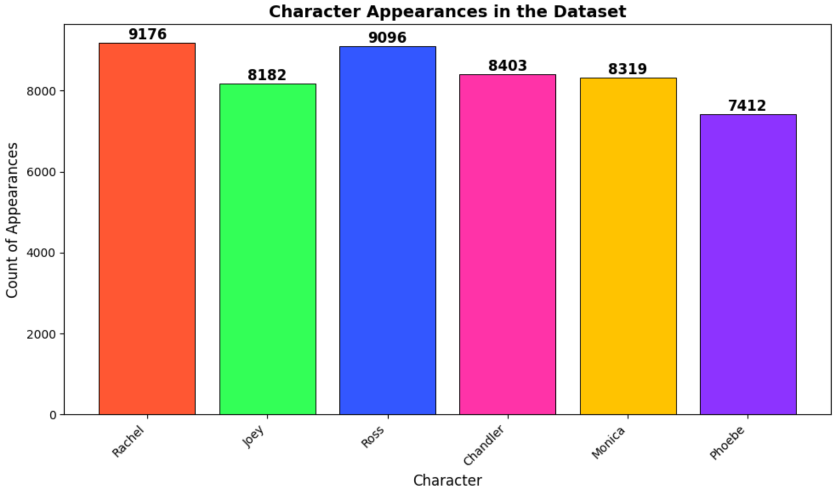

Fig. A5 Character-wise distribution of dialogue appearances in the Friends TV Series Dataset.

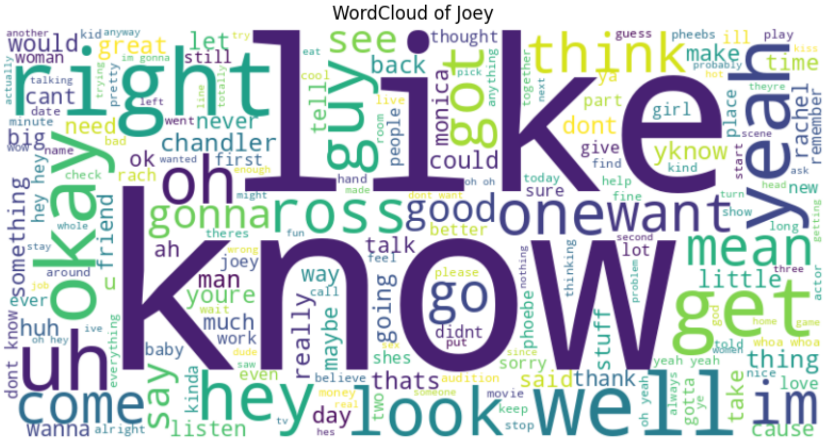

Fig. A6 A visual summary of the dominant lexical features of the character Joey.

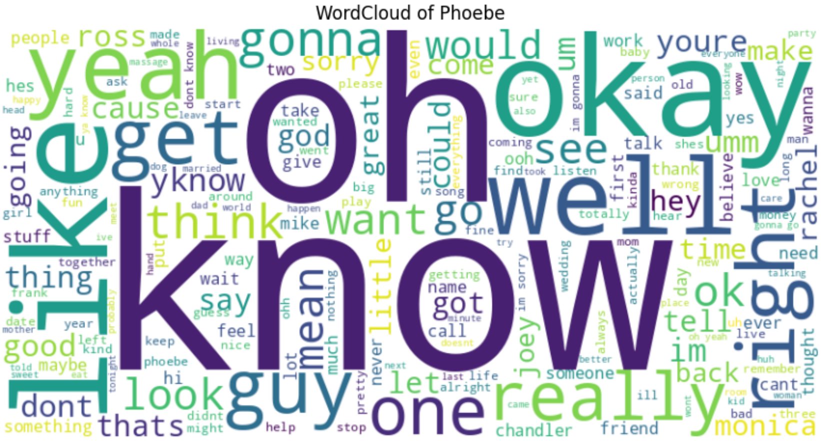

Fig. A7 A visual summary of the dominant lexical features of the character Phoebe.

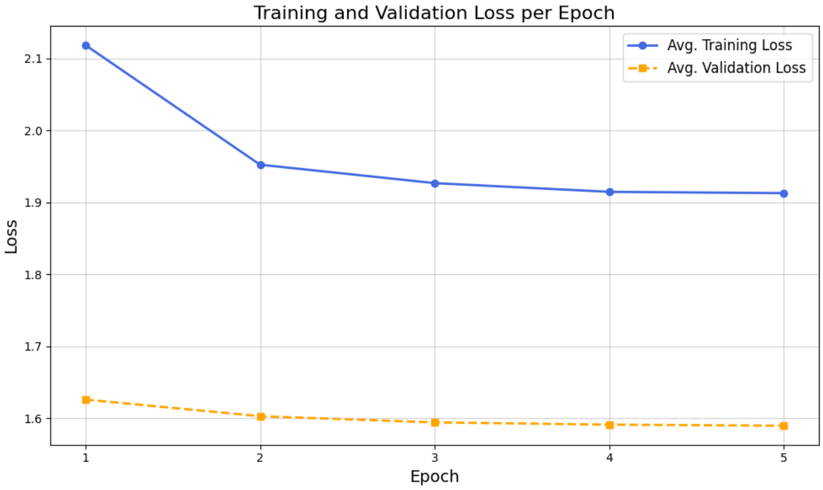

Fig. A8 Training and Validation Loss about Cornell Movie-Dialogs Corpus per Epoch.

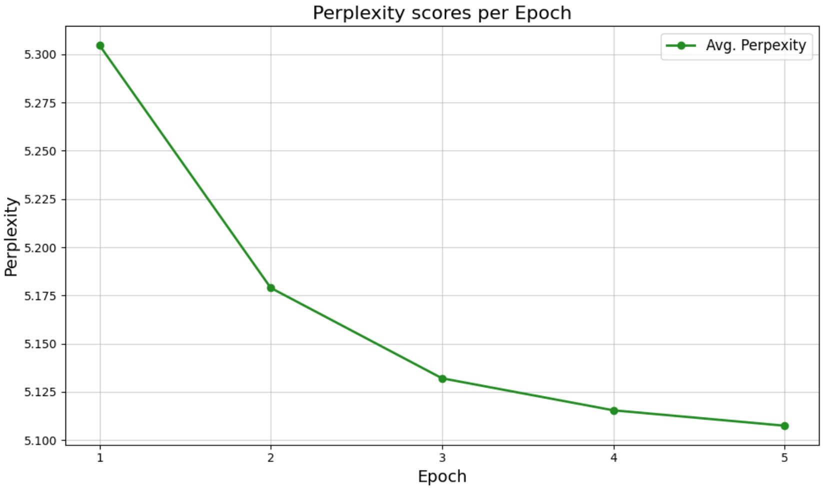

Fig. A9 Perplexity score about Cornell Movie-Dialogs Corpus per Epoch.

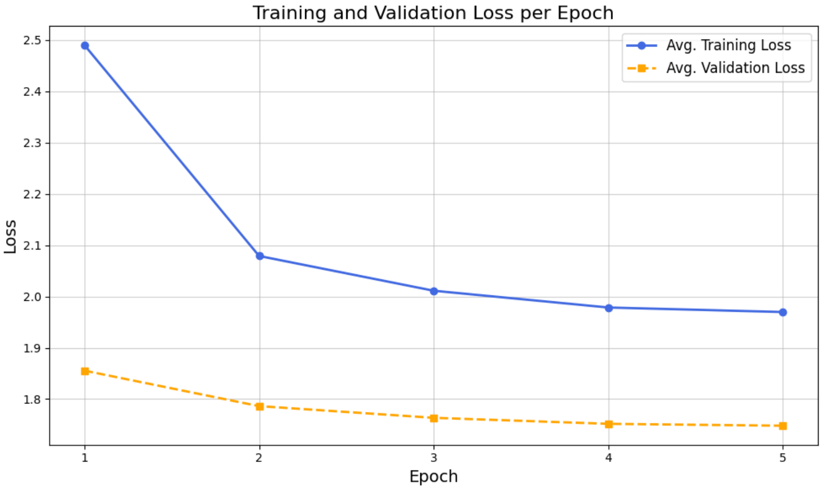

Fig. A10 Training and Validation Loss about Joey dataset per Epoch.

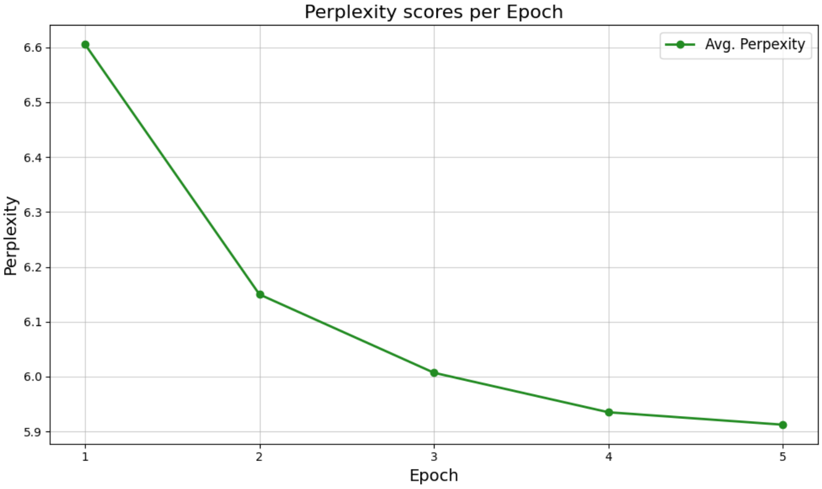

Fig. A11 Perplexity score about Joey dataset per Epoch.

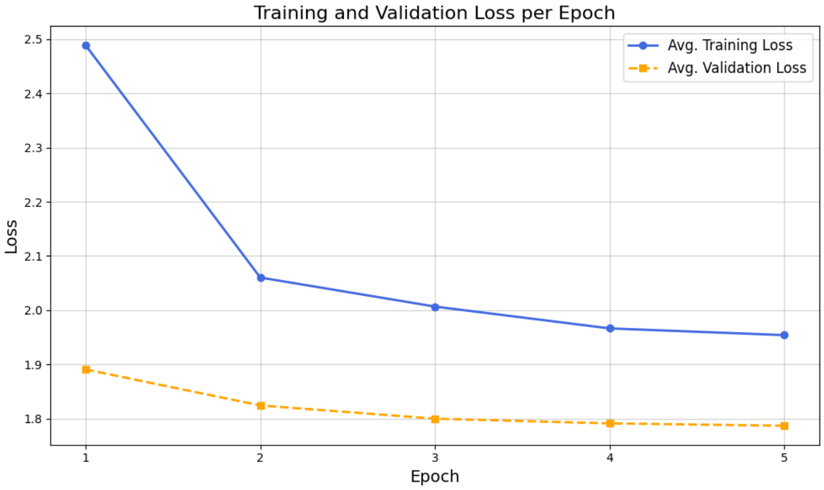

Fig. A12 Training and Validation Loss about Phoebe dataset per Epoch.

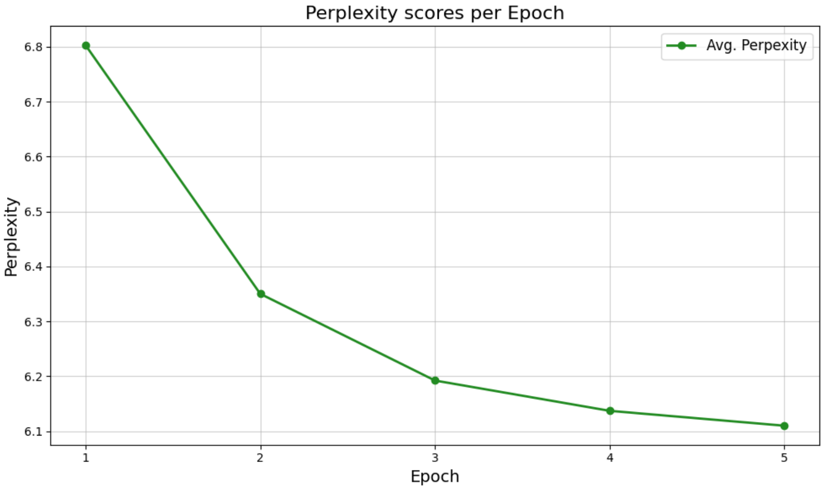

Fig. A13 Perplexity score about Phoebe dataset per Epoch.

Table A1 Training and validation loss along with perplexity scores over 5 epochs during fine-tuning on the Cornell Movie-Dialogs Corpus.

Epoch	Train Loss	Val Loss	Perplexity
1	2.1186	1.6262	5.3048
2	1.9525	1.6031	5.1789
3	1.9269	1.5945	5.132
4	1.9148	1.5914	5.1154
5	1.913	1.5899	5.1074

Table A2 Training and validation loss along with perplexity scores over 5 epochs during fine-tuning on the Joey dataset.

Epoch	Train Loss	Val Loss	Perplexity
1	2.4902	1.8555	6.6059
2	2.0791	1.7863	6.1497
3	2.0113	1.7635	6.0069
4	1.9786	1.7518	5.9349
5	1.9698	1.7481	5.9122

Table A3 Training and validation loss along with perplexity scores over 5 epochs during fine-tuning on the Phoebe dataset.

Epoch	Train Loss	Val Loss	Perplexity
1	2.4891	1.8906	6.803
2	2.0602	1.8241	6.3501
3	2.0066	1.7997	6.1922
4	1.9663	1.791	6.137
5	1.9539	1.7867	6.1099

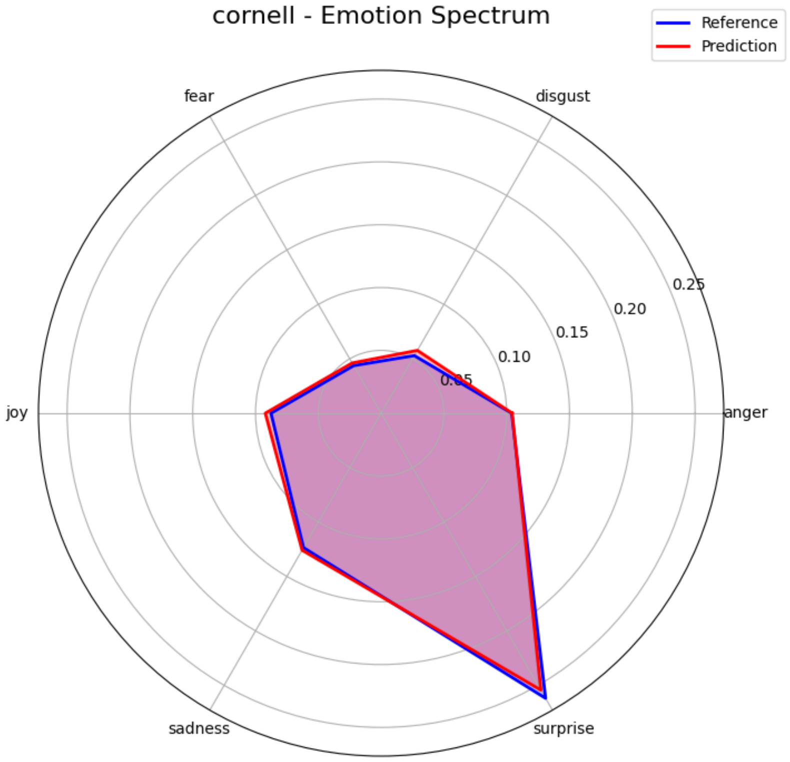

Fig. A14 Classification of Enkman's emotion on Movie-Dialogs Corpus.

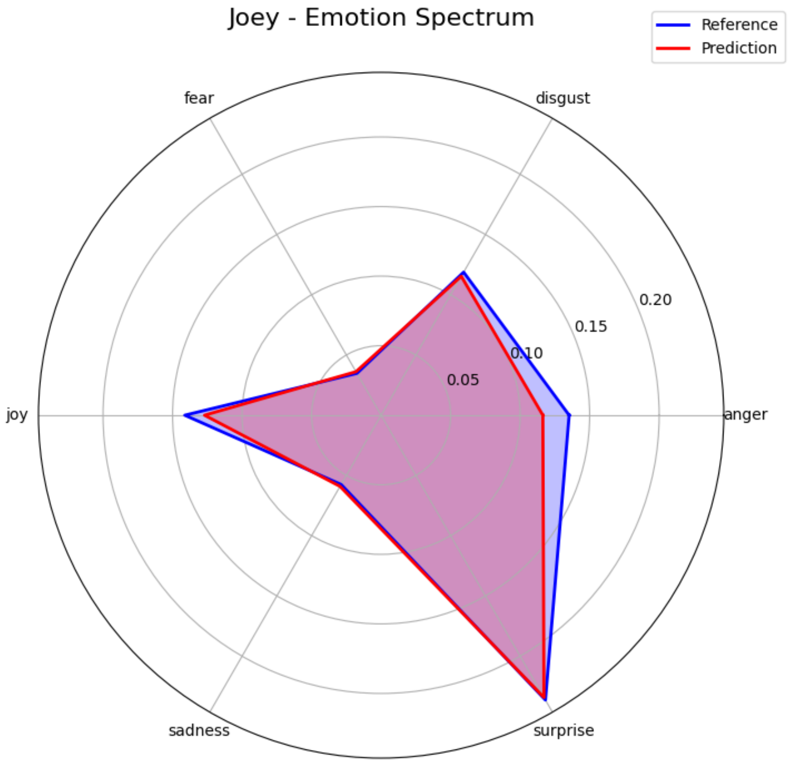

Fig. A15 Classification of Enkman's emotion on Joey dataset.

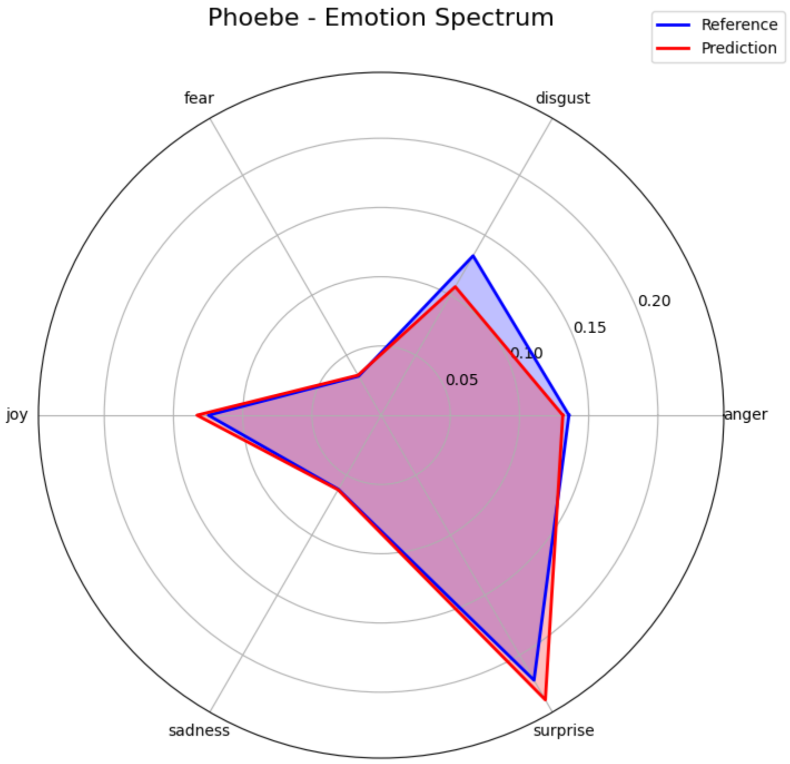

Fig. A16 Classification of Enkman's emotion on Phoebe dataset.

References

[1] Sutcliffe, R.: A Survey of Personality, Persona, and Profile in Conversational Agents and Chatbots (2023). https://arxiv.org/abs/2401.00609

[2] Turing, A.M.: Computing machinery and intelligence. Mind **59**(236), 433–460 (1950)

[3] Weizenbaum, J.: Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM **9**(1), 36–45 (1966)

[4] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)

[5] Nguyen, H., Morales, D., Chin, T.: A Neural Chatbot with Personality. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761115.pdf. CS224n Project Report, Stanford University (2017)

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[7] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. NAACL-HLT (2019)

[8] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language Models are Unsupervised Multitask Learners. OpenAI Technical Report (2019)

[9] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in Neural Information Processing Systems **33** (2020)

[10] OpenAI: GPT-4 Technical Report. https://openai.com/research/gpt-4 (2023)

[11] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., Dolan, B.: Dialogpt: Large-scale generative pretraining for conversational response generation. arXiv preprint arXiv:1911.00536 (2020)

[12] Danescu-Niculescu-Mizil, C., Lee, L.: The Cornell Movie-Dialogs Corpus. https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

[13] Densil, B.: Friends TV Series Screenplay Dataset. https://www.kaggle.com/datasets/blessondensil294/friends-tv-series-screenplay-script (2022)

[14] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Transformers: State-of-the-art natural language processing. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 38–45 (2020)

[15] Liu, C.-W., Lowe, R., Serban, I.V., Noseworthy, M., Charlin, L., Pineau, J.: How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. arXiv preprint arXiv:1603.08023 (2016)

[16] Abeysinghe, B., Circi, R.: The Challenges of Evaluating LLM Applications: An Analysis of Automated, Human, and LLM-Based Approaches (2024). https://arxiv.org/abs/2406.03339

[17] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2019). https://arxiv.org/abs/1910.01108

[18] Ekman, P.: An Argument for Basic Emotions