SENTIMENT AWARE DIALOGUE GENERATION

By

Nilay Saraf 21BAI1067

DIGITAL ASSIGNMENT

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

In partial fulfilment of the requirements for the course of

BCSE419L - Speech and Language Processing

in

B.Tech Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

October 2024

ABSTRACT

Automated dialogue generation systems have become increasingly common in applications like customer service, mental health support, and conversational AI. The existing dialogue models today lack the ability to produce responses that align with the emotional context, thus, limiting their effectiveness in some of the situations.

This research proposes a solution to this issue by developing a system that is sentiment-aware and can generate sentiment-aware dialogues. The proposed system integrates Distilled Bidirectional Encoder Representations from Transformers (DistilBERT) for sentiment analysis along with Generative Pre-trained Transformer 2 (GPT-2) for dialogue generation. It has been fine-tuned using the DailyDialog dataset, which contains about 13118 multi-turn dialogues that cover various conversational scenarios of daily life such as greetings, scheduling, etc.

Evaluation metrics mainly include human evaluation as there is no proper method to check if the generated dialogue is suitable or not without human intervention. Results demonstrate that the sentiment-aware model outperforms baseline approaches in generating responses that better match the intended emotional tone. Human evaluators consistently rate the sentiment-aware responses higher in terms of empathy, coherence, and naturalness.

This research shows that combining DistilBERT's sentiment analysis capabilities with GPT-2’s generative power can effectively create more emotionally attuned dialogue systems. Also, it highlights the benefits of sentiment-aware dialogue generation in enhancing user interaction across various domains, such as virtual therapy, customer service, and social media engagement.

INTRODUCTION

In recent years, automatic dialogue generation systems have seen widespread adoption in various fields such as customer service, healthcare, and virtual assistants. These systems aim to simulate human-like conversations, providing users with responses that are contextually appropriate. However, one critical limitation that is still present in most of the existing dialogue generation models is their inability to produce responses that reflect the emotional tone of the conversation. Traditional models prioritise grammatical correctness and contextuality ignoring the emotional aspect, which is crucial for applications requiring empathy, like mental health support.

The proposed system integrates two state-of-the-art natural language processing models: DistilBERT and GPT-2. DistilBERT is a distilled version of the BERT model that is used for real-time sentiment analysis. It is chosen due to its efficiency as it retains 97% of BERT's performance while being 60% faster and 40% smaller. Its bidirectional architecture effectively captures the sentiment nuances in user input, providing accurate and timely sentiment classification. The sentiment information obtained from DistilBERT serves as a guiding factor for GPT-2, a powerful generative language model used for dialogue generation. GPT-2's ability to generate coherent and contextually relevant responses is leveraged, with sentiment conditioning to ensure the generated responses align with the detected emotional tone.

The DailyDialog dataset is used to fine-tune the dialogue generation model. This dataset comprises various everyday conversational scenarios, allowing the model to adapt to a wide range of dialogue topics such as small talk, opinions, and problem-solving. By training GPT-2 with sentiment labels derived from the DistilBERT, the model learns to generate responses that are contextually relevant as well as emotionally aligned. For instance, if the input dialogue reflects a negative sentiment, the model can produce supportive or empathetic responses rather than neutral or generic replies, thereby enhancing user satisfaction and engagement.

Overall, this research contributes to the field of conversational AI by introducing a novel approach to sentiment-guided dialogue generation. It demonstrates how combining sentiment analysis with language generation can improve the relevance and emotional alignment of responses, setting the stage for more advanced emotionally aware dialogue systems.

LITERATURE SURVEY

One of the earliest advancements in neural dialogue generation was the adoption of sequence-to-sequence (Seq2Seq) models, as introduced by Sutskever et al. (2014), based on recurrent neural networks (RNNs), showed significant improvement in generating contextually coherent responses by learning end-to-end mappings from input sequences to output sequences. However, Seq2Seq models struggled with maintaining long-term coherence and often produced generic responses such as "I don't know."

With the introduction of the transformer architecture (Vaswani et al., 2017) , there was a shift in natural language processing. The transformer model's self-attention mechanism allowed for better contextual understanding of text, leading to the development of BERT (Devlin et al., 2018) and GPT (Radford et al., 2018) . BERT achieved remarkable results in understanding the sentiment of text. While BERT was not designed for generative tasks, it played a role in sentiment analysis within dialogue systems. On the other hand, GPT (and later GPT-2) proved highly effective for generative tasks due to its autoregressive nature, producing contextually relevant responses. However, neither BERT nor GPT integrated sentiment explicitly during the generation process, limiting their emotional alignment capabilities.

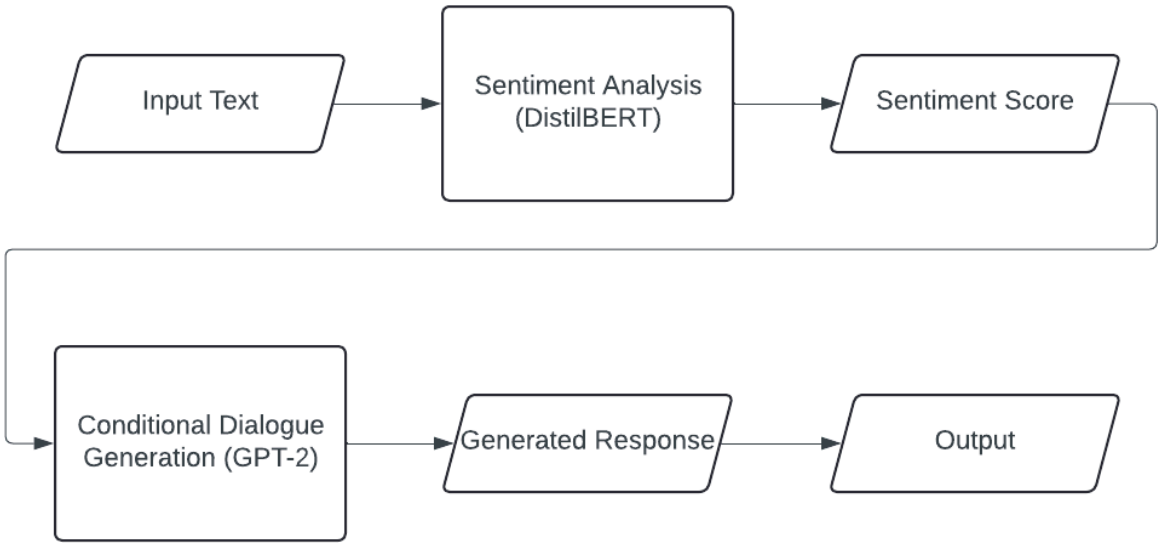
Zhou et al. (2018) proposed a sentiment-aware model called Emotional Chatting Machine (ECM), designed to incorporate emotional factors into dialogue generation . ECM utilised an emotion embedding to control the emotional tone of the generated response, based on predefined emotion categories such as "happy" or "sad." The model demonstrated that sentiment-guided dialogue generation could significantly improve user engagement. However, it relied on discrete emotion categories, which limited the system's ability to handle subtle or mixed emotional states present in real-world conversations.

Knowledge distillation has been applied to develop more efficient versions of large models, such as DistilBERT (Sanh et al., 2019) , which retains 97% of BERT's language understanding capability while being 60% faster and using 40% fewer parameters. DistilBERT has been used for real-time sentiment analysis within dialogue systems due to its speed and efficiency, making it suitable for guiding generative models like GPT-2 based on detected sentiment. Combining

sentiment analysis with response generation represents an emerging direction that balances computational efficiency and emotional alignment.

Another approach explored by Rashkin et al. (2019) focused on using multi-task learning to improve dialogue generation by combining dialogue response generation with emotion recognition tasks . The model was fine-tuned to simultaneously detect the sentiment of input and generate appropriate responses, resulting in better emotional consistency. While this approach showed promise, it required significant computational resources and complex multi-task learning setups, which could be challenging to scale in real-world applications.

ARCHITECTURE



Architecture Diagram of the Sentiment-Aware Dialogue Generation System

The above diagram shows the architectural flow of data in the proposed system. Text is taken as input from the user. It is then tokenized and passed to the sentiment analysis model, that is, to the DistilBERT model. Sentiment score is generated for the input text. Now, the input text and the sentiment score are joined, tokenized and passed to the GPT-2 model for conditional dialogue generation. The generated response is then processed and the output dialogue is displayed to the user.

TECHNOLOGY USED

Python

Python is a trendy-purpose, interpreted programming language developed by Guido van Rossum in 1991. Its design philosophy locations a sturdy emphasis on code readability and makes super use of great whitespace. Its language constructs and item-oriented technique are designed to resource programmers in creating smooth, understandable code for both little and massive initiatives. Python makes use of rubbish collection and has dynamic typing. Procedural, object oriented, and purposeful programming are only a few of the programming paradigms it supports.

Google Colaboratory Ipynb Kernel

Google Colaboratory is a cloud-based Jupyter notebook environment that allows users to write and execute Python code in an interactive setting. It offers free access to powerful computing resources, including GPUs and TPUs, making it ideal for data analysis, machine learning, and deep learning tasks. Users can import libraries, run scripts, and visualise data directly in the notebook. Colab seamlessly integrates with Google Drive for easy file management and sharing. Additionally, it supports collaboration, allowing multiple users to work on the same notebook in real time. For this research, T4 GPU has been used so that the training of model can speed up and reduce overall training time.

METHODOLOGY

1. Data Collection

For the proposed system, the dataset being used is the Daily Dialog dataset. It consists of 13,118 multi-turn dialogues that are manually labelled. The dialogues are written in English and cover various topics of daily life, such as greetings, scheduling, weather, etc. The dataset has been annotated with emotion labels, dialog act labels and topic labels.

2. Import necessary libraries

```
from datasets import load_dataset
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer, Trainer, TrainingArguments
```

The proposed system requires mainly the transformers library, which has been developed by Hugging Face is a popular open-source Python library that provides pre-trained models and tools for natural language processing (NLP) tasks. It supports a wide range of state-of-the-art transformer-based models, such as BERT, GPT, T5, and more, enabling tasks like text classification, translation, summarization, question answering, and text generation. Here, the models being used are DistilBERT and GPT2. The library makes it easy to fine-tune these models for specific tasks using a simple and consistent API.

3. Load and prepare dataset

```
dataset = load_dataset("daily_dialog")
```

```
sample_dialogue = dataset['train'][0]
print("Sample Dialogue:")
for i, utterance in enumerate(sample_dialogue['dialog']):
    print(f"Speaker {i % 2 + 1}: {utterance}")
    print(f"Emotion: {sample_dialogue['emotion'][i]}")
    print(f"Dialogue Act: {sample_dialogue['act'][i]}")
    print("---")
```

This code displays some sample dialogues along the sentiment and dialogue act number.

4. Sentiment analysis with DistilBERT

```
distilbert_tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased',clean_up_tokenization_spaces=True)
distilbert_model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=3)
```

Initially, the DistilBERT model and tokenizer are initialised with parameters as in the snapshot above.

```
def get_sentiment(text):
    inputs = distilbert_tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
    with torch.no_grad():
        outputs = distilbert_model(**inputs)
    probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
    sentiment = torch.argmax(probs, dim=-1).item()
    return sentiment
```

Next, a function is defined to retrieve the sentiment from the model to be used further for dialogue generation.

```
dialogue_text = " ".join(sample_dialogue['dialog'])
sentiment = get_sentiment(dialogue_text)
print(f"Dialogue  :{dialogue_text}")
print(f"Sentiment :{sentiment}")
```

This is an example for displaying how sentiment is retrieved using the DistilBERT model, using the function defined above.

5. Guided dialogue generation with GPT-2

```
gpt2_tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
gpt2_model = GPT2LMHeadModel.from_pretrained("gpt2")
```

The GPT2 tokenizer and model is initialised first.

```
gpt2_tokenizer.pad_token = gpt2_tokenizer.eos_token
gpt2_model.resize_token_embeddings(len(gpt2_tokenizer))
```

In order to pad the tokens for the model, end of sentence token is used and accordingly the model is modified to resize the token embeddings for the input text and sentiment.

```
def tokenize_with_sentiment(examples):
    dialogues = [" ".join(utterance) for utterance in examples["dialog"]]
    sentiments = [get_sentiment(dialogue) for dialogue in dialogues]
    tokenized = gpt2_tokenizer(dialogues, padding="max_length", truncation=True, max_length=512)

    # Incorporate sentiment as an additional token
    tokenized["input_ids"] = [[sentiment] + ids for sentiment, ids in zip(sentiments, tokenized["input_ids"])]
    tokenized["attention_mask"] = [[1] + mask for mask in tokenized["attention_mask"]]
    tokenized["labels"] = tokenized["input_ids"].copy()

    return tokenized
```

This function is defined to tokenize the dialogues from the dataset along with the sentiment retrieved from DistilBERT model for guiding the GPT2 model which dialogue is appropriate for which context.

```
tokenized_datasets = dataset.map(tokenize_with_sentiment, batched=True, remove_columns=["dialog", "act", "emotion"])
```

Here, we get the tokenized dataset. The dataset is mapped with the function defined above so that the tokens are prepared for the GPT-2 model training. This task takes a considerable amount of time, approximately 30 to 35 minutes.

```
training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=1,
    per_device_train_batch_size=2,
    save_steps=10_000,
    save_total_limit=2,
    logging_dir="./logs",
    logging_steps=200,
    remove_unused_columns=False,
)
```

Further, the training arguments are defined such as the number of epochs for training, batch size, storing result, logging, etc. Increasing the number of epochs might yield better results, but more computing resources are required for that purpose.

```
trainer = Trainer(
    model=gpt2_model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    tokenizer=gpt2_tokenizer,
)
```

Next, the trainer parameters are set, such as the model to be trained, the training arguments, dataset to be used and the tokenizer.

```
trainer.train()
```

Finally, the model is trained. This task takes a lot of time depending upon the number of epochs. One epoch takes approximately 35 to 40 minutes.

6. Save the fine-tuned model

```
model_save_path = "./sentiment_gpt2"
gpt2_model.save_pretrained(model_save_path)
gpt2_tokenizer.save_pretrained(model_save_path)
```

This code saves the model in the working directory of the google colaboratory notebook and can be downloaded from there for further testing and usage.

RESULTS

The saved model can now be used to generate the results. For that:

1. Import necessary libraries

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer, pipeline
```

2. Load the saved model and tokenizer

```
model_save_path = "./sentiment_gpt2"
gpt2_model = GPT2LMHeadModel.from_pretrained(model_save_path)
gpt2_tokenizer = GPT2Tokenizer.from_pretrained(model_save_path)
```

3. Load DistilBERT model

```
sentiment_model_name = "distilbert-base-uncased-finetuned-sst-2-english"
sentiment_model_revision = "af0f99b"
sentiment_analyzer = pipeline("sentiment-analysis", model=sentiment_model_name, revision=sentiment_model_revision)
```

4. Define functions for generating dialogue from the model

```
def get_sentiment(text):
  sentiment = sentiment_analyzer(text)[0]
  sentiment = sentiment['label'].lower()
  return sentiment
```

```
def generate_dialogue_with_sentiment(prompt):

    sentiment = get_sentiment(prompt)
    if(sentiment == "negative"):
      sentiment=0
    else:
      sentiment=1

    input_ids = gpt2_tokenizer.encode(f"{sentiment} {prompt}", return_tensors="pt")
    output = gpt2_model.generate(input_ids, max_length=100, pad_token_id=gpt2_tokenizer.eos_token_id)
    output = gpt2_tokenizer.decode(output[0], skip_special_tokens=True)
    output = output.split("  ")[1:]

    y = list()
    for i in output:

      if(i not in y):
        y.append(i)

      else:
        break

    y = " ".join(y)
    return y
```

5. **Give dialogue as input.**

```
g1 = generate_dialogue_with_sentiment("I had a bad day.")
print(f"Generated Dialogue : {g1}")
```

```
Generated Dialogue:  I'm sorry.
```

```
g2 = generate_dialogue_with_sentiment("It was raining. I am sad.")
print(f"Generated Dialogue : {g2}")
```

```
Generated Dialogue:  I am sorry.It was raining.
```

```
g3 = generate_dialogue_with_sentiment("You make me so proud!")
print(f"Generated Dialogue : {g3}")
```

```
Generated Dialogue :  I'm so happy.  You're so lucky.
```

```
g4 = generate_dialogue_with_sentiment("You are a wonderful friend!")
print("Generated Dialogue:", g4)
```

```
Generated Dialogue:  I am very happy to hear that.
```

The above snapshot displays the responses generated for some of the given dialogues. It can be observed that the generated dialogues are sentiment-aware as well to make it user-friendly and more human-like conversation.

CONCLUSION

This project combined GPT-2 for dialogue generation with DistilBERT for sentiment analysis, aiming to generate contextually coherent responses aligning with the contextual sentiment. The approach enhances both the quality and emotional relevance of generated dialogues, representing a significant advancement in creating more human-like conversational agents.

The model effectively generated contextually appropriate responses that adapted to the desired sentiment, outperforming traditional dialogue generation approaches. By adding sentiment awareness, the conversations are more engaging and human-like. This integration allows the model to produce responses suitable for various conversational scenarios, including supportive or empathetic contexts.

For example, to comfort a user, the model generates reassuring responses that are contextually appropriate and emotionally aligned. This demonstrates the potential for applying the model in fields such as mental health support, customer service, and social chatbots, where understanding and responding to sentiment is crucial.

The study highlights the benefits of incorporating sentiment analysis into dialogue systems, making AI more emotionally intelligent and improving the quality of human-AI interactions. Although the model has shown promising results, further research with better computational facilities can explore much advanced techniques for better controlling sentiment during text generation, such as reinforcement learning or adversarial training. Fine-tuning the model on specific sentiment-labelled datasets may also enhance its performance for specialised applications, leading to more personalised and emotionally relevant responses.

REFERENCES

1. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. Advances in Neural Information Processing Systems.

2. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems.

3. Radford, A., Wu, J., Child, R., et al. (2018). Language Models are Unsupervised Multitask Learners. OpenAI Blog.

4. Zhou, H., Young, T., Huang, M., et al. (2018). Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory. Proceedings of the AAAI Conference on Artificial Intelligence.

5. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter. arXiv preprint arXiv:1910.01108.

6. Rashkin, H., Smith, E. M., Li, M., & Boureau, Y. L. (2019). Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.