George Cantstandya- A George Costanza Chat Bot

Michelle Belon
Economics
A14556504

Abstract

This project builds a George Costanza Chat Bot utilizing the Autoregressive model, DialoGPT. The goal is for the Chat Bot to engage in open-ended conversations while adopting the persona of George Costanza, a character from the TV series Seinfeld. DialoGPT is based off of the popular text- generative model GPT2 [1] , and was fine-tuned off of Seinfeld scripts from season one to nine which was obtained from Kaggle [8].After fine-tuning the model, different experiments were conducted to assess both the objective and subjective performance of the Chat Bot. Experiments include changing context window size for conversations, data augmentation through back translation, crafting customized data pipeline, and experimenting with model generation hyperparameters. These experiments where chosen based off of the model's architecture and the challenging assessment of whether the Chat Bot adopted the personality of George Costanza.

1 Introduction

The goal of creating a Chat Bot that adopts the persona of a given character implies several interesting questions. Crafting a personality specific Chat Bot has been accomplished before. In order to fine-tune DialoGPT, HuggingFace supplies a notebook that walks through the steps of data formatting and defining the necessary functions to train and evaluate the model [6]. The supplied notebook creates a Spanish speaking Chat Bot, while other efforts have created Chat Bots with a specific purpose such as an empathetic Chat Bot [7]. While these Chat Bots are created through the same process as used for the George Costanza Chat Bot, creating a Chat Bot based on a specific character introduces new challenges. That is, assessing whether the Chat Bot has fully adopted the persona of George Costanza is difficult and up to the user. This subjective assessment required by this project resulted in the several experiments conducted, which distinguish this Chat Bot from previous efforts. Finally, although the creation of a Chat Bot based off a TV personality does not have a specific duty as does a empathetic Chat Bot, it does lead to the question of whether a script contains all aspects of a character's personality and whether this can be learned for open-ended conversations.

2 Related Work

This project was inspired by Grant Sheen's work in creating an empathetic Chat Bot [7]. As discussed previously, the empathetic Chat Bot was fine-tuned using the supplied notebook [6]. Grant Sheen's project goes on to experiment with the fine-tuned Chat Bot in order to achieve improved subjective performance. These experiments include testing various model generation hyperparameters or decoding methods, such as experimenting with sampling techniques such as top-k or top-p sampling. I also experiment with these decoding methods; however, I extend my experiments past this. To distinguish my project from others, I also experimented with context-window size, data augmentation, and the data pipeline in order to test various hypotheses about how the model can best adopt a personality from a script. The work of Grant Sheen and the recommended fine-tuning notebook from Nathan Cooper informed my work and helped me create a base-line model from which I compare it's performance with the various models created from my experiments.

3 Model

The model used for creating a George Costanza Chat Bot is DialoGPT. This model is based off of GPT-2, a popular auto-regressive model that generates text by maximizing the probability of a word given all previous words. In other words the model maximizes, $P(W_{1:T}|\theta) = \prod_{t=1}^{T} P(W_t|W_{1:t-1}, \theta)$. Although, much of DiaoGPT is based off of GPT-2 one distinction is that DialoGPT was trained on 147 million conversation threads from Reddit [12]. Furthermore, DialoGPT is interested in multi-turn conversations, which slightly adapts the optimization set for the model. DialoGPT optimizes the probability of a target sentence (response) given a dialogue history, $P(T|S) = \prod_{n=m+1}^{N} P(x_n|x_1, ...x_{n-1})$. This is achieved by concatenating a conversation into a corpus ending with a end-of-text token. From this corpus there is the dialogue history $S = x_1, ..., x_m$, and a target response $T = x_{m+1}, ..., x_n$ [12].

There are several aspects of this model's architecture that are important to understand in order to understand some experiments conducted. First; however, a quick overview of GPT-2 is necessary. GPT-2 is composed of stacked decoders, wherein each decoder block contains a layer of masked self-attention followed by a feed forward neural network [2]. Prior to passing through the first decoder block, a sentence is tokenized. Put simply, each word is converted to a vector representation according to the model's embedding matrix. Since this project fine-tunes DialoGPT, the embedding matrix is already set and since DialoGPT-Small is used, the embedding matrix contains an embedding size of 768 per word [2]. Once the word is tokenized, a positional embedding is added. The purpose of a positional embedding is to add information about the order of words. DialoGPT uses absolute positional embedding, unlike other models such as BERT, which uses relative positional embedding. Absolute positional embedding encodes the exact position of a given word, or token, as opposed to the distance to other words [11].

After the positional embedding is added, the token can now pass to the masked self-attention block. Masked self-attention, is another distinction between this model and BERT. Masked self-attention will allow a token only to look at previous tokens during the attention process, as opposed to viewing all tokens [2]. This distinction in addition to the absolute positional embedding inform one of the experiments conducted for this project.

The evaluation metric used for the George Costanza Chat Bot was perplexity.Perplexity is the normalized inverse probability of the test set [4]. A well performing model would assign high probability to the test set because it is not surprised by the test set. The lower the perplexity is the better the model is performing. Perplexity serves as the objective measure of the models created throughout this project; however, perplexity cannot account for subjective performance which is mainly whether the Chat Bot has adopted the personality of George.

Once a token is passed through the stacked decoders, the output is multiplied by the token embedding matrix and the result is assigned probabilities to each word in the model's vocabulary [2]. How the model picks the next word according to these probabilities is according to several decoding methods such as top-p, top-k sampling [9]. These decoding methods are experimented with during the last experiment of this project.

Fine-tuning DialoGPT on Seinfeld scripts does not change the architecture of the model, although words from the scripts are now included in the model's vocabulary. Rather than experimenting with the actual architecture of the transformer, I used the aspects of the architecture that distinguish DialoGPT from other models to inform the experiments.

4 Dataset

The dataset used to fine-tune the DialoGPT model for the creation of a George Costanza Chat Bot was Seinfeld scripts for seasons 1-9. This data was obtained from Kaggle [8]. In its original form, the data contains a column corresponding to the character speaking and a column corresponding to the dialog. A single observation's dialogue is not restricted to a single sentence, and may include notes on any actions taken by the character.

The data in its original form is not appropriate to fine-tune DialoGPT. To reformat the data I followed the Jupyter notebook by Ruolin Zheng, which sets out to create a Chat Bot for a video game character

	Unnamed: 0	Character	Dialogue	EpisodeNo	SEID	Season
0	0	JERRY	Do you know what this is all about? Do you kno...	1.0	S01E01	1.0
1	1	JERRY	(pointing at Georges shirt) See, to me, that b...	1.0	S01E01	1.0
2	2	GEORGE	Are you through?	1.0	S01E01	1.0
3	3	JERRY	You do of course try on, when you buy?	1.0	S01E01	1.0
4	4	GEORGE	Yes, it was purple, I liked it, I dont actuall...	1.0	S01E01	1.0

Figure 1: Seinfeld Script Data

[13]. The reformatted data is structured so that a single column contains all of George's dialogue. Then this is followed by columns containing a set number of previous lines of dialogue.

	response	context	context/0	context/1	context/2	context/3
5549	It would have been nice if someone told me abo...	She quit.	Hey, Cosmo what happened to your mother last n...	All right all right. Come on I'll take you ove...	Well Elaine needs to borrow his racquet. Just ...	Yeah
7905	(Pleading for them to stop) Mom.. dad.	(Nagging yell) Well, I didn't take the subway ...	(Loud shouting) I didn't take the subway all t...	(Trying to match Frank's loudness) So, who say...	(Yelling) That's not a booth!	Why can't we sit over there? (Points to a table)
6291	Yup. It's really just a question of what kind....	So are you gonna get him a chair?	Well it's tough to get a good read but I think...	Yeah and?	So I spoke to the security guard.	All right.

Figure 2: Restructured Data for DialoGPT

So for any dialogue of George, there are n columns where each column is a preceding line of dialogue. Although the column labeled "response" only contains dialogue spoken by George, the other columns contain dialogue that is spoken by anyone including George. The columns of preceding dialogue provide context for the response column. So a single observation from the response column can then serve as an context for another response observation. This results in the formatted data being both wider and longer than the original dataset. There are 9704 observations of George's responses. The context window, or how many previous lines of dialogue, for the base model is seven preceding lines of dialogue. In one experiment this context window is changed to four and ten preceding lines of dialogue. As stated previously the optimization task for DialoGPT involves the probability of a response given a dialogue history. By experimenting with the dialogue history, the optimization task is affected. For all experiments, a 80-20% split is conducted for the training-test split.

5 Experiments

A number of experiments were conducted during the course of this project in the hopes of selecting a final model that performs best both subjectively, in terms of adopting George's personality, and objectively, in terms of perplexity. In total there were four main experiments, each of which include mini experiments.

The first experiment was the creation of a base model. The base model was created in order to provide a benchmark for future models, and to understand how the Chat Bot performs without any manipulation. Next, were a a couple of experiments conducted on the context window. The first of which lessened the context window, and the second increased the context window. The third group of experiments involved data augmentation through the use of back translated data. In this third group were a total of two models; one with shuffled data and the other without shuffled data.The fourth group of experiments involved the data loader; the pipeline of the data to the model. This group includes three experiments, the first loads data by independent halves, the second loads data sequentially, and the third loads data by season. After all these experiments were conducted a final model was chosen to conduct experiments on the model generation hyperparameters, or decoding methods. The model chosen for this final series of experiments was the model with augmented data through back translation.

6 Results

All experiments were compared against the performance of the base model. The base model used an ADAMW optimizer, as did all experiments, and the data was shuffled. The objective performance of this model, or perplexity, was 7.1949. The low perplexity indicates that the model is not confused by the test set and assigned high probability to the test set. While the objective performance was fairly well, the subjective performance of this base model was not. The George Costanza Chat Bot would sometimes output random punctuation instead of words. This output of punctuation was not remedied by adding the specific output to a "bad words list", which prohibits the model from using these characters. The failure of the "bad words list" to stop the random punctuation output signals that this is not due to artifacts in the data. In the instance that the Chat Bot did not output punctuation, we can see that the personality of George Costanza was adopted; the Chat Bot knows George does not like his parents. Although the model adopted the personality, the outputting of punctuation is indicative of a serious issue.

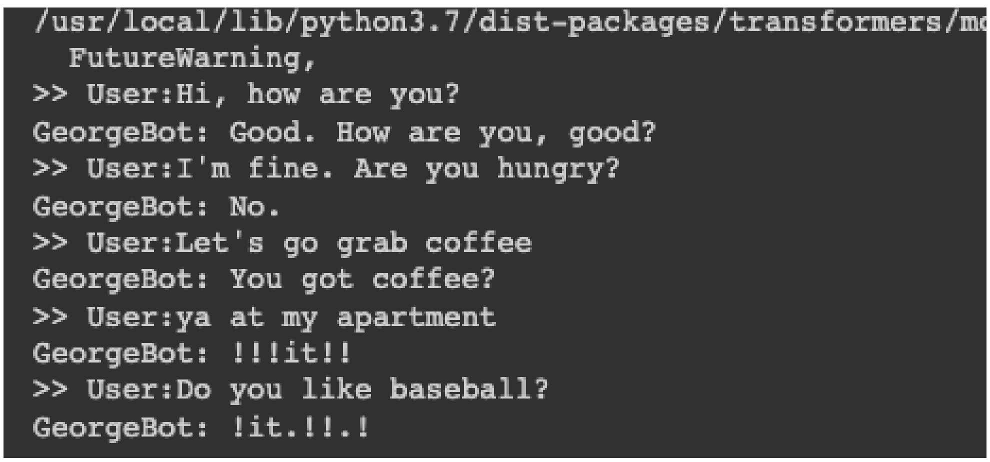

Figure 3: Base Model - Chat Bot Outputting Punctuation

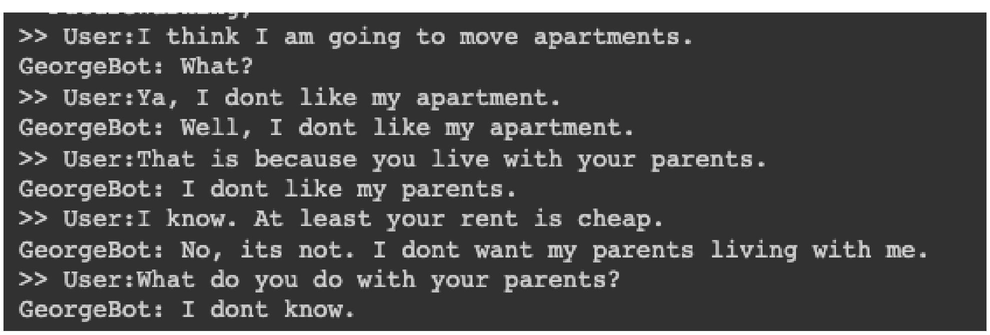

Figure 4: Base Model - Adopting George's personality

The second experiment conducted was on the context window size. To recap, the context window is how many previous lines of dialogue from George's line of dialogue are kept in the reformatted data. The base model had a context window of seven previous lines of dialogue. I first experimented with a smaller window size of four, since to understand a given response one may not need as much of the whole conversation. Decreasing the window size increased the perplexity to 7.3408. Not only was the objective performance worse, the model is less certain about the test set, but also the subjective performance got worse. This model began outputting more punctuation instead of words.

```
  FutureWarning,
>> User:How are you doing?
GeorgeBot: I'm doing well.
>> User:Want to grab lunch?
GeorgeBot: Yeah, yeah.
>> User:Where do you want to grab food?
GeorgeBot: ?
>> User:Where should we go for lunch?
GeorgeBot: ???????????????????????????????????????????????????????????
>> User:ok.
Input length of input_ids is 303, but ``max_length`` is set to 300. Thi
GeorgeBot: ?
```

Figure 5: Model With Context Size Four

The exact reason behind the nonsense output of punctuation is not exactly known, except a similar issue did occur during the creation of the empathetic Chat Bot. Grant Sheen's Chat Bot would not not output anything after a couple of lines of conservation, which was attributed to the model guessing the End of Sentence Token used during the tokenization process[7]. While the exact symptoms present in this model are a bit different than that of Grant Sheen's Chat Bot, this may be the cause of the nonsense output of punctuation.

Although the exact source of the issue of outputting punctuation is unknown, the issue was resolved once the context window was increased to size ten. The reasoning of this second experiment involving the context window is twofold. First, perhaps conversations move slower than originally thought. Second increasing the window size may make it harder for the model to guess the eos token. This experiment resulted in a better performing model. The perplexity was 6.9977, so the model was more certain about the test set, and the subjective performance was better. This model had adopted the personality of George Costanza, while also not outputting any punctuation.

```
>> User:Kramer owes me money.
GeorgeBot: I'm sorry.
>> User:I can't find him anywhere.
GeorgeBot: I'm not paying for this.
>> User:Could you lend my 5 dollars?
GeorgeBot: I can't.
>> User:The least you could do is pay for parking.
GeorgeBot: No, no. Pay for parking. Pay for parking.
>> User:I only have two dollars in my wallet.
GeorgeBot: You can't pay for parking.
```

Figure 6: Model With Context Size Ten

The next experiment conducted was data augmentation through the use of back translation. Back translation is when a sentence is translated to another language and then translated back to the original language. This process of translating a sentence twice results in a sentence with the same meaning and context but the exact order of words is different. Back translation can be accomplished through the Google Translate API [3]. There are two reasons as to why this experiment was chosen. The first is that more data is always better, and back translation is one way to accomplish that. The second reason is to test two components of DialoGPT's architecture. The two components I wanted to test with back translation are absolute position embeddings, and masked self-attention. Since absolute position embeddings encode the exact order of words, with back translated data there will be a different resulting embedding. In addition to this, the masked self-attention process restricts the attention process from only observing a given token and the preceding tokens. So, since back translated data will contain the same meaning but different order of words, one can take advantage of the DialoGPT architecture and presumably have the model learn better.

Back translation was done on a 10% random sample of the data. The language the data was first translated to was randomly chosen and there were several different languages possible. The back translated data was then appended to the original data set. Shuffling this larger data set resulted in a perplexity of 6.6318, the best performing model so far. The subjective performance of this model was also good, the model had clear indications of George's personality. Not shuffling the larger data set resulted in worse performance, as one might expect, with a perplexity of 11.4038. Although the objective performance was worse, this model did have indications of adopting George's personality. The performance of the model with the shuffled augmented data indicates that however small the difference may be by augmenting back translated data, the model was able to better learn the context and personality contained in George Costanza's responses.

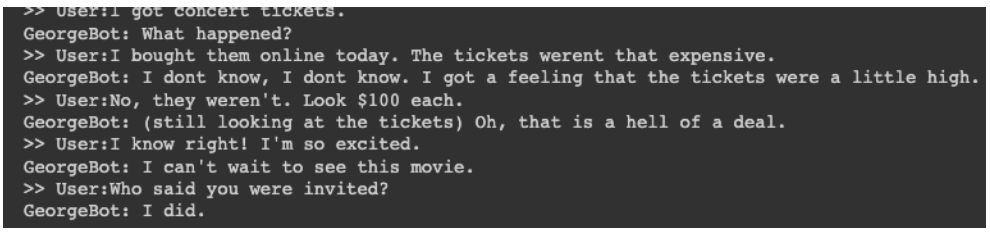

Figure 7: Back Translated Data (Shuffled)

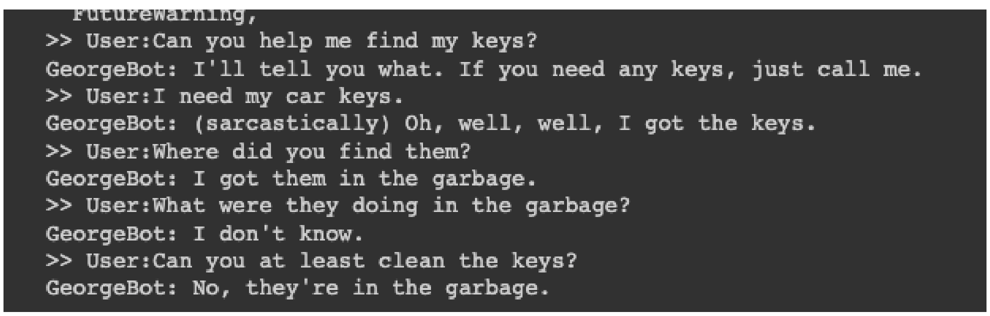

Figure 8: Back Translated Data (Unshuffled)

The next series of experiments involve the DataLoader, which acts as the pipeline for the data to the model by loading and iterating over elements in the dataset. The first experiment conducted on the DataLoader involves loading the data by independent halves and was inspired by Scott Condron [5]. This experiment, while arbitrary, was chosen to determine whether creating custom pipeline actually impact the performance of the model. Loading data by independent halves causes the model to first interact with the first half of the training set; however, within this half the indices are shuffled. As one might expect the lack of randomness leads to worse model performance; the perplexity of this model increased to 19.7917. The subjective performance of this model was also bad as the Chat Bot became confused directing questions aimed at it back to the user. While this experiment was arbitrary it did indicate that the DataLoader does impact model performance.

```
>> User:Hey George how is it going?
GeorgeBot: It's fine.
>> User:What did you do today?
GeorgeBot: I just got back from seeing my father in Florida.
>> User:How is your father?
GeorgeBot: I don't know.
>> User:Did they bug you about getting married?
GeorgeBot: I didn't even know you were engaged.
>> User:I'm not, but you are.
GeorgeBot: Oh, you're a beautiful woman.
```

Figure 9: Data Loaded by Independent Halves

The next DataLoader experiment loaded the training data sequentially with no shuffling. That is, the training data was loaded in order and there was no shuffling of indices within each batch. The hypothesis that I wanted to conduct with this experiment was to determine whether the Seinfeld scripts had a progressive development of characters. That is, perhaps George Costanza slightly differs in personality between seasons and to really capture his personality this progression needs to be understood. Of course, this hypothesis does not address the fact that the lack of randomness would lead to worse model performance. The model performance did suffer from the lack of randomness both for objective and subjective performance. Perplexity for this model increased to 18.5631, interestingly an improvement from the previous model; however, subjective performance was lackluster with the Chat Bot not engaging enough during conversation.

```
>> User:Did you see Elaine's new apartment?
GeorgeBot: Yeah, it's a huge mess.
>> User:I know. I dont know why she didn't hire movers.
GeorgeBot: What?
>> User:Why did she not hire a moving company?
GeorgeBot: I dont know.
>> User:Anyway, Kramer was here today.
GeorgeBot: Oh, my God.
>> User:Ya. He left the oven on.
GeorgeBot: Oh, my God.
```

Figure 10: Data Loaded Sequentially (indices not shuffled)

The final experiment with the DataLoader loaded the data by season with indices shuffled. First, data was split by season so that the training data set contains data from each season. This is in contrast to the previous data splitting method for the DataLoader experiments, which selected the last 20% for the test set. Next, each season is then independently fed into the model and indices within a season are shuffled. The hypothesis tested here is the same as the previous experiment; however, some randomness is introduced with this experiment and the test set is now composed of all seasons. Both the objective and subjective performance were far worse for this model than other models. The objective performance dramatically grew worse with a perplexity of 100.4149. As for the subjective performance, the model picked dialogue straight from the script. While the script may contain some interesting responses, this is not wanted because the model will be unable to adapt to any conversation. In sum, experimenting with the data pipeline resulted in far worse models.

```
>> User:Are you ready to head out?
GeorgeBot: Hey, where's Kramer?
>> User:I have no clue, he ate all my food.
GeorgeBot: (over the speaker) Kramer, I'm telling ya, he's a GUY.
>> User:Let's just go get lunch without him.
GeorgeBot: No, Jerry, we don't want you to do that.
>> User:Kramer wont mind.
GeorgeBot: I know, he's a nice guy.
>> User:Aren't you hungry
GeorgeBot: I have a tremendous favor to ask.
```

Figure 11: Data Loaded By Season (indices shuffled)

The final experiments conducted involved the model generation hyperparameters of the best performing model. To recap, the best performing model was the model with the augmented back translated data which was shuffled during the training split. This model had both the best objective performance, by having the lowest perplexity, and the subjective performance was also good with clear indications of the Chat Bot adopting George's personality.

Model generation hyperparameters are those that impact the decoding process, such as top-k sampling or top-p sampling [10]. The first series of experiments involved top-k sampling, which impacts how large the sampling pool is and therefore how the probability mass is redistributed [10]. The base model had top-k set to 50, so 50 words made up the sampling pool. I first tried lowering top-k to 25, and then raising top-k to 75. In both cases top-p remained at 0.7, in order to assess just the impact of changing the top-k hyperparameter. In both cases, the conversation generated was worse and the Chat Bot was repeating phrases.Therefore it seems that for top-k sampling either extreme results in limited benefit as the probability mass is redistributed among a smaller, less creative pool or a larger pool that contains illogical words.

```
>> User:I'm hungry, let's go out to eat.
GeorgeBot: (to George) Hey, what's going on?
>> User:Want to go grab some food with me?
GeorgeBot: No, no, no, no. I don't want to get caught.
>> User:Caught? Who are you hiding from?
GeorgeBot: I'll tell you what. I'm going to go to the movies.
>> User:Fine go to the movies. I'm going to get tacos.
GeorgeBot: I'm going to the movies.
>> User:ok.
GeorgeBot: I'm going to the movies.
```

Figure 12: Top-K sampling lowered to 25 most likely words

Next, top-p sampling was experimented with by lowering and raising the top-p hyperparameter from its base model value of 0.7. Top-P sampling determines the smallest set of words whose cumulative probability is greater than the hyperparameter value [10]. The base model with the top-p value set to 0.7, selects the smallest set of words where the cumulative probability exceeds 70%. First, I lowered the top-p value to 0.4, which resulted in poor performance. Increasing the top-p value to 0.9 resulted

```
>> User:Hey, are you hungry?
GeorgeBot: Hey, are you hungry?
>> User:I am. Want to go get some food?
GeorgeBot: I'm starving.
>> User:Let's order chinese.
GeorgeBot: I'm starving.
>> User:What do you want to eat?
GeorgeBot: Medium turkey chili.
>> User:That's specific. Do you think they have chili at the grocery store?
GeorgeBot: I think they have chili at the restaurant.
```

Figure 13: Top-K sampling increased to 75 most likely words

in better performance, since the model can now dynamically select a pool of words, with no pool size restriction, as it builds a sentence. Therefore with the combination of top-k set to 50 most likely words, and top-p set to 0.9, there is some flexibility by allowing the sample to first comprise of the top 50 words then selecting the words whose cumulative probability exceeds 90%.

```
>> User:Hey do you know where Kramer is? He owes me money.
GeorgeBot: No, no.
>> User:I'[m never going to get my money back am I?
GeorgeBot: No, no.
>> User:What are you doing today?
GeorgeBot: I'm taking a bus.
>> User:Where are you going?
GeorgeBot: I'm taking a bus.
>> User:a bus to where?
GeorgeBot: The movie.
```

Figure 14: Top-P sampling lowered to 0.4

```
>> User:Hey George did you see the new movie?
GeorgeBot: Yeah, I saw it.
>> User:What did you think?
GeorgeBot: I thought it was funny.
>> User:I thought the acting was horrible.
GeorgeBot: Well, I thought the script was funny.
>> User:Ya I guess it was funny. Who did you see it with?
GeorgeBot: Kramer.
>> User:Kramer? I'm surprised they let him in the theater.
GeorgeBot: I'm surprised they let you in.
```

Figure 15: Top-P sampling raised to 0.9

Numerous experiments were conducted with each model's objective and subjective performance compared to the base model. The experiments conducted were the following, increasing and decreasing the context window, data augmentation through back translated data, and creating a customized data loader. From all of these experiments the best performing model was the model with augmented data through back translation, and specifically the training data was shuffled. The best performing model was then used to experiment with model generation hyperparameters, specifically the decoding methods. Of the experiments conducted on decoding methods, increasing top-p sampling to 0.9 resulted in better performance, while keeping top-k at 50 was the best choice.

7 Conclusion

DialoGPT, a conversational text generation model, can be fine-tuned to create a character specific Chat Bot. Fine-tuning the model requires reformatting the data so that the character of interest's dialogue is mapped to n previous lines of dialogue. That is, for every dialogue of George Costanza there are n previous lines of dialogue, which can include other character's dialogue. The first model was a base model, which had a fair objective performance of 7.1949, but a poor subjective performance with the model outputting random punctuation.

The following experiments were based on the model's architecture. For example, changing the context window was testing the hypothesis that to understand a given George Costanza response one might need less context, or more context than just seven previous lines of dialogue. This experiment proved that a larger context window solved the problem of outputting random punctuation. So a larger context window of size 10 prevents the model from possibly guessing the eos token and then outputting random punctuation. The next experiment of augmented data through back translation tested several key aspects of the model's architecture like absolute position embedding and masked self attention. Back translation causes the context and meaning of the sentences to remain, however, the exact ordering of words is changed. Since the order of the words is different, the position embedding will be different than the base model and this difference will then be past on to the masked self-attention. This experiment resulted in the best subjective and objective performance; however it is hard to determine whether this is the result of augmenting data or due to the difference that back translation made. Finally, the last experiment conducted on the data pipeline resulted in overall worse model performance. The series of experiments conducted on the DataLoader inform us that imposing restrictions and inhibiting randomness in the data pipeline will decrease model performance. Furthermore, the hypothesis that the personality of George Costanza is better learned progressively through the seasons of Seinfeld is false. The personality of George is best learned with no restrictions on how the data is processed.

The last model adjustment was conducted on the best performing model, the model with augmented data. Top-k sampling, which imposes the number of words to redistribute the probability mass among, was best at 50, the same level as the base model. Any relatively extreme values in top-k sampling stifled creative output. Top-p sampling, which determines the smallest sample whose cumulative probability exceeds a given probability, was best at a higher value of 0.9. Imposing that the sample exceed 90% cumulative probability allows for a different size of the sample, as opposed to the restriction top-k sets. By increasing top-p to 0.9, there was more creativity in the model so subjective performance increased.

While a George Costanza Chat Bot is a fun exercise in NLP, and can output very interesting conversations, the creation of this Chat Bot highlights some important aspects of NLP. First and foremost, the importance of data and formatting data correctly is essential to the performance of a DialoGPT model. Reformatting the data in terms of response, and previous dialogue, in addition to increasing the context window, allows the model to understand the exact dynamics present in Seinfeld dialogue. Next, utilizing back translation for data augmentation and the probable impact on the position embeddings allowed for the model to better learn the meaning of the augmented data, rather than exact order. While all experiments were based on the DialoGPT architecture, a few questions regarding NLP remain. These questions include whether perplexity is the best indicator of objective performance; that is can perplexity capture whether a conversation follows the expected patterns of a conversation. Lastly, the question of how to assess whether the personality of a character is accurately captured by a Chat Bot muddles the act of assessing subjective performance.

References

[1] "dialo-gpt". URL: `https://huggingface.co/docs/transformers/model_doc/dialogpt`.

[2] Jay Alammar. The illustrated gpt-2. URL: `https://jalammar.github.io/illustrated-gpt2/`.

[3] Tucker Arrants. Using google translate for nlp augmentation. URL: `https://www.kaggle.com/tuckerarrants/using-google-translate-for-nlp-augmentation`.

[4] Chiara Campagnola. Perplexity in language models. URL: `https://towardsdatascience.com/perplexity-in-language-models-87a196019a94`.

[5] Scott Condron. But what are pytorch dataloaders really? URL: `https://www.scottcondron.com/jupyter/visualisation/audio/2020/12/02/dataloaders-samplers-collate.html`.

[6] Nathan Cooper. Open-dialog chatbots for learning new languages. URL: `https://nathancooper.io/i-am-a-nerd/chatbot/deep-learning/gpt2/2020/05/12/chatbot-part-1.html#Training-and-Evaluating`.

[7] Grant Sheen. Carl: An empathetic chatbot. URL: `https://web.stanford.edu/class/cs224n/reports/final_reports/report028.pdf`.

[8] Aman Shrivastava. Seinfeld chronicles. URL: `https://www.kaggle.com/thec03u5/seinfeld-chronicles`.

[9] Patrick von Platen. How to generate text. URL: `https://huggingface.co/blog/how-to-generate`.

[10] Patrick von Platen. How to generate text: using different decoding methods for language generation with transformers. URL: `https://huggingface.co/blog/how-to-generate`.

[11] AI Coffee Break with Letitia. Self-attention with relative position representations-paper explained. URL: `https://www.youtube.com/watch?v=DwaBQbqh5aE`.

[12] Michel Galley Yen-Chun Chen Chris Brockett Xiang Gao Jianfeng Gao Jingjing Liu Bill Dolan Yizhe Zhang, Siqi Sun. Dialogpt: Large-scale generative pre-training for conversational response generation. URL: `https://arxiv.org/pdf/1911.00536.pdf`.

[13] Ruolin Zheng. Ai chatbot. URL: `https://github.com/RuolinZheng08/twewy-discord-chatbot`.