ECE-GY 6143 Final Project - Spring 2021

(Application of Deep Learning: Neural Response Generation - COVID19 FAQ Chatbot)

Harini Appansrinivasan (ha1642) & **Mohan Ramakrishnan** (mr5910)

Based on the paper "DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation" [1]

---

1. INTRODUCTION

1.1 NEURAL RESPONSE GENERATION

'Conversational Response Generation' or 'Neural Response Generation (NRG)' is a subcategory of text-generation that shares the objective of generating natural-looking text that is relevant to the prompt. Commonly referred to as chatbots, voice bots, virtual assistants, etc., this is an intelligent piece of software that is capable of understanding natural language commands and requests from humans, and generating human-like neural responses.

Based on how an agent processes the input natural language (NL) request and maps to a response, we can create models catering to different purposes such as 'Interactive FAQs', 'Form filling', 'Dialogue/ Conversation Planning', etc. These models can also be built as either Retrieval based or Generative based models. Retrieval-based chatbots are trained to provide the best possible response from a database of predefined responses, while generative-based chatbots can generate new dialogues. Retrieval-based models are widely used in the industry to make goal-oriented chatbots.

1.2 INSPIRATION BEHIND OUR PROJECT

As most of the contact centers and customer cares are underwater during the current pandemic, our aim here is to create a Retrieval based - Interactive FAQ chatbot that will assist people in quickly getting COVID-19 relevant information.

Our 'COVID19 FAQ Chatbot' uses Microsoft's DialoGPT framework, which is based on Huggingface PyTorch-Transformer and OpenAI GPT-2. Microsoft's DialoGPT pre-trained model and training pipeline are publicly released to facilitate research into NRG and the development of more intelligent open-domain dialogue systems.

1.3 HuggingFace PyTorch-TRANSFORMERS and OpenAI GPT-2

HuggingFace is an open-source provider of Natural Language Processing (NLP) technologies. Transformers is a HuggingFace repository that provides thousands of pre-trained models (such as BERT, GPT-2, DialoGPT) for text processing, like classification, information extraction, question answering, translation, text generation, etc. in 100+ languages. It is backed by the two most popular deep learning libraries, PyTorch and TensorFlow.

GPT2 model was trained with an objective to predict the next word, given all of the previous words within some text [3]. It is a statistical model that gives the probability of words given the context, based on which a prediction is made. GPT2 is also an autoregressive language model, meaning that the output of the model is fed back into the model as input.

```
[Conversation id: 3687015e-d099-4113-bb2d-9f915b945734
 user >> let's play a movie scence - any recommendations?
 bot >> The Big Lebowski ,
 Conversation id: 6c3edaa6-699f-441c-a150-9eacccc80997
 user >> what's your favorite present?
 bot >> A present that I got from my mom. ,
 Conversation id: d0a7b25a-c4bd-4291-b768-1a97a6c3c22f
 user >> What’s your favorite book?
 bot >> The Hunger Games ]
```

Fig 1.1: Sample conversation using Huggingface Transformers - Conversational Pipeline

1.4 DIALOGPT (an extension of GPT2)

DialoGPT is “a tunable gigaword-scale neural network model for generation of conversational responses”. Like GPT-2, DialoGPT is formulated as an autoregressive (AR) language model, and uses the multi-layer transformer as model architecture. Unlike GPT-2, however, DialoGPT is trained on large-scale dialogue pairs/sessions (147M conversation-like exchanges) extracted from Reddit discussion chains [4].

The creators have attempted to resolve challenges present with NRG, that conversations are informal, noisy, and contain abbreviations or errors. This also translates into issues with current approaches for NRG, which can be inconsistent, bland and can have difficulty keeping information in longer-term contexts (i.e. over many conversational turns).

Transformer-based approaches like GPT can solve these issues, and that’s why DialoGPT was born with an aim to attain a performance close to human both in terms of automatic and human evaluation in single or multi-turn dialogue settings.

1.5 MODEL ARCHITECTURE

The model was trained on the basis of GPT-2, inheriting the 12-to-48 layers Transformer with a stack of masked multi-head self-attention layers to train data, layer normalization, initialization scheme and byte pair encodings for the tokenizer.

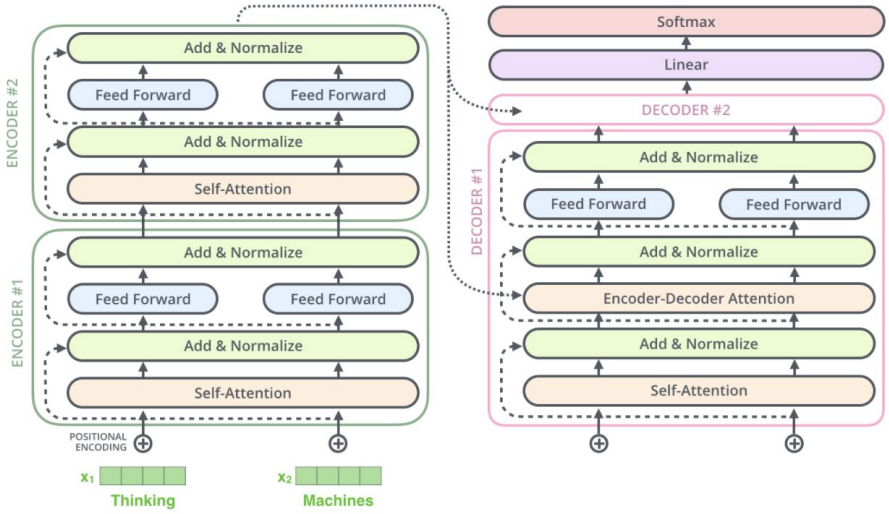

Fig 1.2 Two stacked encoders and decoders architecture (jalammar.github.io) [8]

– First, all dialog turns (x_1, …, x_n) are concatenated into a long text.
– The source sentence (dialogue history) is denoted as S = x_1, …, x_m with m < n; the target sentence (ground truth for the model) is T = x_m+1, …, x_n.
– Then, the conditional probability *P(T|S)* is the product of a series of conditional probabilities of each word in the target response given the previous words in S, as well as those in T preceding x_n

$$p(T|S) = \prod_{n=m+1}^{N} p(x_n|x_1, \cdots, x_{n-1})$$

– For a multi-turn dialogue session T_1,…,T_K, optimizing a single objective $p(T_K, ..., T_2|T_1)$ can be perceived as optimizing all $p(T_i|T_1, ..., T_{i-1})$ source-target pairs.

MAXIMUM MUTUAL INFORMATION (MMI):

To address the problem of bland, uninformative responses, DialoGPT uses MMI scoring function. MMI is a pre-trained backward model to predict source sentences for given responses, ie, *P(Source|Target)*.
– Generate a set of hypotheses using top-K sampling
– Use the probability of *P(Source|Hypothesis)* to re-rank all hypotheses

This penalizes bland hypotheses, because frequently occurring (bland) target responses can occur with many Sources, therefore not maximizing *P(Source|target)*. The more unique targets have higher probabilities.

However, unlike the training under RNN architecture, it has been observed that reinforcement learning (RL) training easily converges to a degenerate locally-optimal solution, where the hypothesis simply repeats the source sentence and mutual information is maximized. The investigation of regularized RL training is still under research.

As we shall see, we will use MMI during our evaluation step.

1.6 WHY DIALOGPT ?

There are several open-sourced toolkits for largescale pre-trained transformer models, such as DeepPavlov, ParlAI, DLGnet.

Unlike DeepPavlov or ParlAI which focuses on task-oriented dialogues, DialoGPT was trained and modelled to generate conversational responses not specific to any predefined goal. The model was tuned on source-target pairs, and does not leverage grounding information. It learns background information during pre-training and is unhindered by the lack of a grounding document. Hence it creates responses that does not have to be premade to very specific questions or commands or topic, where the replies are based on n number of conversation history stored in the cache by the model.

In our project, we have experimented DialoGPT for developing goal-oriented FAQs. We have explored how it can be trained to answer specific questions on specific topics and evaluated its performance.

2. METHODOLOGY

2.1 INITIAL CONFIGURATION

Due to the storage limitations in Google Colab, we use DialoGPT-small instead of the larger versions. The initial model configuration settings are stored in config.json. Initial values of some hyper-parameter are:

- Number of epochs = 3
- Train / evaluation batch size = 4
- Learning rate = 5e-5
- Adam optimizer with epsilon = 1e-8

2.2 DATA PREPARATION AND SETUP

DialoGPT was originally built on Reddit discussions, that can be naturally expanded as tree-structured reply chains. Each path was extracted from the root to the leaf node as a training instance containing multiple turns of dialogue.

For our project, we have used one-to-one Q&As collected from multiple data sources [15] to create train, test and validate FAQ JSON inputs.

- We then converted our dataset into a suitable format using Pytorch's Dataset and Dataloader API, so our model can understand it. The process involves converting our dataframes, which has multiple contexts and responses, into a single conversation string that is separated by a special 'end of string' token.
- These strings are then tokenized using HuggingFace's AutoTokenizer and the AutoModelForCausalLM into their numerical representation that our model actually understands.
- The construct_conv function does the processing by using the transformer library.

The setup part of our code involves creating caches and checkpoints that would be used by the model in the later stages of the code. The checkpoints enable us to store our trained model and then later retrieve it for further processing. This speeds up the process and makes it efficient. After creating the checkpoints we also write a function to check whether the old checkpoints have to be deleted as it may lead to consuming too much memory.

2.3 MAIN RUNNER - TRAINING AND VALIDATING

The main runner performs the below actions:

- Setup CUDA, GPU & distributed training, setup logging
- Train the model on the training dataset. The main steps of 'train' function are as follows:
    - Prepare the optimizer, check if the saved optimizer states exist.
    - Check for multiple GPU- if available, set parallel processing to reduce runtime.
    - Train the model
    - Calculate training loss, update hyper-parameter accordingly
- Save the trained model, create a checkpoint for the model and store it in output-small-save folder
- Evaluate on the validation dataset:
    - Evaluation done for 15 iterations
    - Calculate evaluation loss and perplexity (Perplexity is a measure of how unsure the model is in its choice of the next token. The more unsure our model is, the higher its perplexity)
    - Store perplexity value in eval_results.txt
      Our model perplexity = tensor(31.3668)

2.4 GENERATE TEST RESULTS AND METRICS

We test our model on the test dataset. We check if the number of queries and responses match, we iterate through queries and add the test queries to the chat history. We run the model and check if we are getting the accurate responses for each query.

```
User: Can animal byproducts spread coronavirus? Do I have to worry about milk, eggs, or fruits?

Chatbot: No. It is not a coronaviruses.

User: Why are so many people with diabetes dying from the coronavirus? It’s really frightening me.

Chatbot: Infectious infection is not the same as diabetes. Infectious infections are not the cause of death.

User: I'm just a bit concerned as I have a sore throat and coughing a little bit as a result of my sore throat. My upper body is also aching a b

Chatbot: Yes, you are more likely to be exposed to COVID-19. You are more exposed to it than others. You may also have a cough and possibly have

User: Can coronavirus symptoms be mild for some people versus severe? For example, could it just involve being very fatigued, low grade feve

Chatbot: It depends on the severity of the symptoms, severity of infection, and how severe the virus is. It depends on your age, age, and yo

User: Good day,  I am 29 weeks pregnant and my profession is teaching, I also have a 2 year old son.  My chest and throat is sore, I have a

Chatbot: Hello,    You should definitely be tested for coronaviruses.   If you have a fever or cough you should not be taking antibiotics.
```

Fig 2.1: Sample conversations from the test dataset

Some of the metrics calculated on the test dataset are as below. They evaluate a predicted sentence to a reference sentence using different approaches:

- BLEU score: Calculates cumulative n-gram scores and weighting them by calculating the weighted geometric mean
- NIST score: Similar to BLEU, but uses arithmetic mean instead of geometric mean and has a different brevity penalty
- METEOR score: While BLEU does not count some sub-words of the same origin as correct alternatives, METEOR considers synonyms and compares the stems of words

We calculated 2-gram and 4-gram BLUE and NIST with equal weights (ie, 0.5 for 2-gram and 0.25 for 4-gram). Below are the results.

```
bleu_2, bleu_4, meteor, nist_2, nist_4
```

```
(0.19839271170682926,
 0.10828142236313851,
 0.25214973918223405,
 1.0000990479021044,
 1.040979825507883)
```

Fig 2.2: Calculated metrics

2.5 INTERACTIVE CHAT

To make an interactive chatbot there are several methods available such as the Greedy search algorithm, Beam search algorithm, Sampling method, Top-K Sampling and Top-P sampling. We narrowed our selection down to two methods, Top-K sampling and Top-P sampling.

Top-K sampling method samples only from the most likely K words, whereas in Top-P sampling it chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. The probability mass is then redistributed among this set of words. This way, the number of words in the set can dynamically increase and decrease according to the next word's probability distribution.

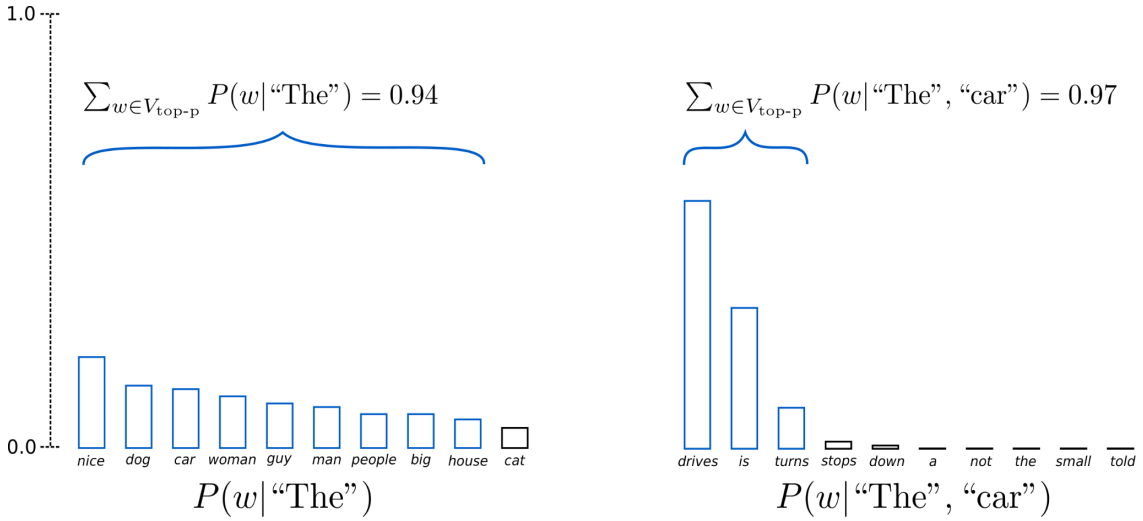

Fig 2.3 Probability distribution of Top-P sampling [10]

In theory, Top-P may sound elegant whereas in practical applications both methods are efficient. Top-P can also be used in combination with Top-K, which can avoid very low ranked words while allowing for some dynamic selection.

We are using the Top-P sampling method in combination with Top-K for interacting with our chatbot which we found to be the most efficient. We also used it to generate the responses for our test dataset.

- The user input and an End-of-String (EOS) token are encoded
- Using the tokenizer, the model, a n^{th} chat-round and a set of chat_history_ids, a response to some user input is generated
- The responses are appended to the chat history (as DialoGPT, in theory uses the whole chat history for generating predictions)
- The response is finally printed and the chat_history_ids (the current response) is returned for usage in a subsequent round

```
tokenizer = AutoTokenizer.from_pretrained('microsoft/DialoGPT-small')
model = AutoModelWithLMHead.from_pretrained('output-small-save')

# Let's chat for 2 lines
for step in range(2):
    # encode the new user input, add the eos_token and return a tensor in Pytorch
    new_user_input_ids = tokenizer.encode(input(">> User:") + tokenizer.eos_token, return_tensors='pt')
    # print(new_user_input_ids)

    # append the new user input tokens to the chat history
    bot_input_ids = torch.cat([chat_history_ids, new_user_input_ids], dim=-1) if step > 0 else new_user_input_ids

    # generated a response while limiting the total chat history to 1000 tokens,
    chat_history_ids = model.generate(
        bot_input_ids, max_length=1000,
        pad_token_id=tokenizer.eos_token_id,
        no_repeat_ngram_size=3,
        do_sample=True,
        top_k=8,
        top_p=0.6,
        temperature = 0.8
        # Alternate set of parameters
        #bot_input_ids, max_length=1000,
        #pad_token_id=tokenizer.eos_token_id,
        #top_p=0.92, top_k = 50
    )

    # pretty print last ouput tokens from bot
    print("Chatbot: {}".format(tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)))
```

Fig 2.4 Our interactive chat code

3. RESULTS

We started off by generating responses for random questions using the pre-trained Huggingface Transformers - Conversational Pipeline model [1]. We tried to chat with DialoGPT without fine-tuning.

```
>> User:Hi, how are you?
DialoGPT: Hi, I'm from the future.
>> User:What is your favorite movie?
DialoGPT: I'm from the future.
```

Fig 3.1 Conversations without fine-tuning

After successfully training our model on the COVID19 dataset, we were able to interactively chat with our model.

```
>> User:When should I get tested?
Chatbot: Take a test of steroid oral drug for your symptoms. For your symptoms, you should be tested for COVID-19.
>> User:How do I practice physical distancing?
Chatbot: Try a video or text chat with your doctor.  Would you like to video or Text chat with me?
```

Fig 3.2 Conversations after fine-tuning

From the calculated metrics and chat responses of our DialoGPT-small model, we see that our chatbot generates good responses at times, but can't sustain the effort beyond a few exchanges.

4. CONCLUSION

Although the model did not respond with high accuracy, training it for longer or using a larger model (DialogGPT-medium, DialogGPT-large) instead of DialogGPT-small will improve results.

Running a larger model or using a much bigger dataset or training it longer on Google Colab (with GPU) resulted in '`RuntimeError: CUDA out of memory`'. Due to the resource limitations of Google Colab, we have limited our experiment to DialogGPT-small.

5. APPLICATIONS AND FUTURE WORK

DialoGPT has countless applications. The several pre-trained models it provides can be fine-tuned to obtain a conversation model on any moderately-sized customized dataset. One of the significant use cases can be to generate goal-oriented search engine models in multiple languages. We can integrate it into web apps, messengers, etc. that can provide automated virtual assistance to all people.

We can investigate leveraging reinforcement learning to further improve the relevance of the generated responses and prevent the model from generating egregious responses. As we saw under MMI, the investigation of regularized RL training is still under research

REFERENCES

[1] DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation - https://arxiv.org/pdf/1911.00536.pdf

[2] [Scratch 25] Open Dialog Chatbots for Learning New Languages [Part 1].ipynb - Colaboratory (google.com)

[3] OpenAI GPT2 — transformers 4.5.0.dev0 documentation (huggingface.co)

[4] DialoGPT — transformers 4.5.0.dev0 documentation (huggingface.co)

[5] microsoft/DialoGPT: Large-scale pretraining for dialogue (github.com)

[6] DialoGPT: Toward Human-Quality Conversational Response Generation : https://www.microsoft.com/en-us/research/project/large-scale-pretraining-for-response-generation/

[7] Conversational AI Chatbot with Pretrained Transformers Using Pytorch - https://towardsdatascience.com/conversational-ai-chatbot-with-pretrained-transformers-using-pytorch-55b5e8882fd3

[8] Easy Chatbot with DialoGPT: https://www.machinecurve.com/index.php/2021/03/16/easy-chatbot-with-dialogpt-machine-learning-and-huggingface-transformers/

[9] The Illustrated Transformer – Jay Alammar (jalammar.github.io)

[10] How to generate text: using different decoding methods for language generation with Transformers (huggingface.co)

[11] https://towardsdatascience.com/beginners-guide-to-building-a-singlish-ai-chatbot-7ecff8255ee

[12] https://blog.machinetranslation.io/compute-bleu-score/

[13] https://machinelearningmastery.com/calculate-bleu-score-for-text-python/
[14] https://huggingface.co/transformers/perplexity.html
[15] SOURCES FOR COVID-FAQ DATASET

1. WHO_FAQ.xlsx - obtained from Github WHO_FAQ.xlsx
2. Bing_Dataset.csv - obtained from keywordtool.io with keyword 'covid' for Bing search engine
3. CDC_Dataset.csv - obtained from cdc.gov website's coronavirus/COVID-19 FAQs
4. CNN_Dataset.csv - obtained from https://www.cnn.com/interactive/2020/health/coronavirus-questions-answers/
5. COVID-QA_Dataset.csv - converted from original_qanda.json which is from https://github.com/deepset-ai/COVID-QA
6. FDA_Dataset.csv - obtained from fda.gov website's coronavirus FAQs
7. IDPH_Dataset.csv - obtained from dph.illinois.gov website's coronavirus FAQs
8. John-Hopkins_Dataset.csv - obtained from hopkinsmedicine.org website's coronavirus FAQs
9. KeywordTool_Dataset.csv - obtained from keywordtool.io with ‘covid’ related keywords
10. Quora_Dataset.csv - obtained from Quora with a search for "covid"
11. UN_Dataset.csv - obtained from un.org website's coronavirus FAQs
12. WJLA_Dataset.csv - obtained from https://wjla.com/news/local/frequently-asked-questions-coronavirus
13. Yahoo_Dataset.csv - from Yahoo search engine's autocomplete suggestions
14. Yahoo-Answers_Dataset.csv
15. COVID-QA faqs - obtained from https://github.com/deepset-ai/COVID-QA/blob/master/data/faqs/faq_covidbert.csv
16. COVID-QA | Kaggle
17. COVID19 related FAQs | Kaggle