MSAI-337: Natural Language Processing

Final Project (NLP) Conversational Agent Preliminary Results

March 14th, 2022

Professor

Dr. David Demeter

Students

Aleksandr Simonyan

Dimitrios Mavrofridis

Donald Baracskay

Wentao Yao

Approaches

As the project documentation instructed, we created two approaches to attempt to beat the baseline metrics that GPT2 produced. These metrics were accuracy, perplexity, and BLEU score. Out of these three, BLEU score is obviously the most useful, whereas accuracy is the least; however, we had some trouble getting BLEU to run some of the time (variations in sequence length caused errors). Our first approach was relatively simple: use a different model. Our thoughts were that while GPT2 performs fairly well on the WOZ dataset, there are likely newer or more apropos models that can do even better. We initially sought to use a model called TransformerXL, as it seemed to be included in the same package as the HuggingFace's implementation of OpenAI's GPT2, however, we encountered errors and had quite a bit of trouble with this. With TransformerXL off the table, we experimented with a variety of similar models on HuggingFace, looking for one that would achieve higher values in our metrics, especially the BLEU score. We tested on Roberta, XLNet and DistilGPT2.

The second approach that we tried was to generally optimize the GPT2 model. First, we added tagging to the dataset, converting times, addresses, and other integers to their respective tags. This should, in theory, reduce the amount of uncertainty in the dataset and thus help the model produce better results. Finally, we attempted to optimize the models' hyperparameters and the decoder's functional arguments to get better results. Specifically, we tested increasing the number of layers, heads and embeddings for Roberta and XLNET and changing the activation function for GPT2. For the decoder, we tested not only the given decoding methods (for changes in BLEU score), but our own custom decoder method. We have concluded that optimizing different hyperparameters significantly decreases the perplexity. However, because we had a limited computational power, we haven't implemented a grid search to optimize the perplexity and BLEU score of each model.

Results

Perplexity Scores Across Models on Test Data				
Model	Parameters	Perplexity	Eval Loss	Eval Acc
GPT2	Default	2.68	1.056	0.802
XLNET	Default	2.674	0.102	0.81
	n_layer=50, n_head = 20	2.6546	0.99	0.82
	n_layer=50, n_head = 20, d_model= 1100	2.6402	0.9836	0.8229
Roberta	max_position_embeddings =1024, num_attention_heads = 24	2.6714	0.102	0.78
DistilGPT2	Default	3.078	1.124	0.791

Table 1: Perplexity and Evaluation Loss and Accuracy

BLEU Scores Across Models				
Model	Tagging	Decoder	BLEU (best)	BLEU (total)
Untrained GPT2	No	Beam	0.06662	0.002
GPT2	No	Logits	0.20447	0.015
		Greedy	0.20462	0.013
		Beam	0.19617	0.012
		Top P	0.21192	0.005
	Yes	Logits	0.19612	0.019
		Greedy	0.18653	0.016
		Beam	0.24555	0.025
		Top P	0.15025	0.005
Distil GPT2	No	Logits	0.20503	0.02
		Greedy	0.20056	0.018
		Beam	0.18718	0.017
		Top P	0.19431	0.008
Roberta	No	Beam	0	0

Table 2: Bleu Score Metrics for each Model & Decoder

Analysis

For our untrained (zero-shot) GPT-2 model, its performance appears to be mediocre. BLEU Score is perilously close to 0. But this is understandable because, after all, the untrained GPT-2 is underfit on our dataset. After further training of the model, the accuracy of the model has been significantly improved.

Next, we tested the effect of preprocessing the data on the model. This was the tagging method described in the Approaches section. We believe this method would be effective because it reduces unnecessary noise in the data. When the data is not preprocessed, the logits decoder works the best, but after preprocessing, the custom beam decoder is more effective. This is also the best BLEU score out of all tested).

As we can see from Table 1, there is a slight but visible difference in performance across various models. Deeper models, for example XLNET with more layers and heads perform better than models with the default set of parameters. The difference in performance is not very significant most likely because base models already do a very good job. Thus, beating them becomes increasingly more difficult.

From Table 2, we can see the best BLEU score was achieved for pretrained GPT-2 with Beam Decoder. Tagging also boosted the performance. The difference in BLEU score is quite significant. Pretrained GPT2 beats other models quite significantly. The reason for this is most likely that GPT-2 has many more parameters (1.5 billions) and is used explicitly for autoregressive text generation.

As suggested, in the prompt we used the part of the text dedicated to goodbyes for a qualitative ablative analysis. GPT2 in particular seemed to produce very good results. The following tables include many examples of quality conversations between the user and the system.

USER	SYSTEM
Thank you, goodbye.	You're welcome. Goodbye.

USER	SYSTEM
Thank you, goodbye.	You're welcome. Have a great day.

However, the system sometimes ended the conversation a little bit too soon. Some examples of such responses can be found below:

USER	SYSTEM
No I do not	You're welcome. Goodbye.

USER	SYSTEM
European, please.	You're welcome. Have a great day.

Furthermore, the algorithm also generated some very interesting but also funny conversations which can be found in the following tables:

USER	SYSTEM
European, please.	You're welcome. Have a great day.

USER	SYSTEM
European, please.	You're welcome. Have a great day.

USER	SYSTEM
I would like Italian food.	There are no Italian restaurants in the centre of town. Would you like Italian food?

USER	SYSTEM
Yes, please, I would like a reservation for 4 people at 1:45 on Saturday.	Booking was successful. The table will be reserved for <integer> minutes. “

Nevertheless, the program has output some valid and quality conversations between the user and the agent:

USER	SYSTEM
I am looking for a moderately priced restaurant in the south part of town.	There are no moderately priced restaurants in the south part of town.

USER	SYSTEM
Can I have the address and phone number?	The address is <address> Bridge Street City Centre and the phone number is <phone-number> <phone-number>.

Be that as it may, the program was not able to always effectively generate quality conversations between the user and the agent. An example of a failed response is displayed in the table below, in which the agent repeatedly used the same sentence to reply to the user:

USER	SYSTEM
-	I'm sorry, there are no Korean restaurants in the north part of town. Can I help you find anything else?
No, that's all I needed.	-
-	I'm sorry, there are no Korean restaurants in the north part of town. Can I help you find anything else?

Notes

Throughout our testing, we encountered a number of bugs. The worst of which hindered us for around 3 days. It was one that manifested itself as an improper sequence length during the evaluation phase. We later discovered this was related to the number of workers assigned to the project, something that we *had* to change in order to avoid out-of-memory errors. We eventually managed to get it to work on a batch size of 2 and the default number of workers. Overall these two bugs led to the project being spent mostly on debugging stupid errors and not on trying to evaluate different models and approaches.