Cloning Myself As a Chatbot

George S. Barsukov

University of California, Berkeley

Abstract

In this paper, I explore various approaches for adapting not just a personality, but a way of speech from a person onto a chatbot. I explore an existing architecture developed by HuggingFace for a personality adapted chatbot. I then use my own chat logs combined with OpenAI's pretrained GPT2 models in order to accomplish this task. I experiment with different data formats and preprocessing steps as well methodologies for adapting my chat logs to the pretrained models. I then try various decoders and subjectively validate the results. Next, I will explain what validation criteria I used and what would have been more ideal. Lastly, I will conclude with some of the shortcoming and how I think they can be improved upon.

Introduction

The motivation for this task is my fascination with automation and anticipation of lifelike artificial intelligence. A fair question to ask is "why is this important?". Truthfully, there's no perfect good answer. While there can be many innovative use cases, the most prominent ones are for artistic purposes. One such use case is for conversational agents in video games. However, the only reason I can really speak to is that I thought it would be a fun endeavor. I have a fascination with automation and replacing myself with a chatbot just aligns well with that.

Conversational chatbots have always been somewhat of an aimless field. The distinction between a conversational chatbot and other chatbots is that it's not trying to provide any information. Rather, the bot is trying to chat with a user without a specific topic and with the hidden objective of convincing them that it's a human. In this case, more specifically, the bot needs to convince a user that the user is talking to me and not a chatbot version of me. Apart from running human experiments, it's hard to tell apart a good chatbot from a bad one, even with a manual inspection. Running these experiments is not only time consuming, but also difficult to set up for each model.

A technique that significantly helped with validation was the ability to have chatbot talk to one another. This reduces the manual load significantly while still providing some sense of the chatbot's ability to communicate. An even better distributable approach is to actually replace myself with a chatbot in live conversations and see if the other party notices. This approach however takes much more human involvement and is therefore more suitable for final testing rather than validation.

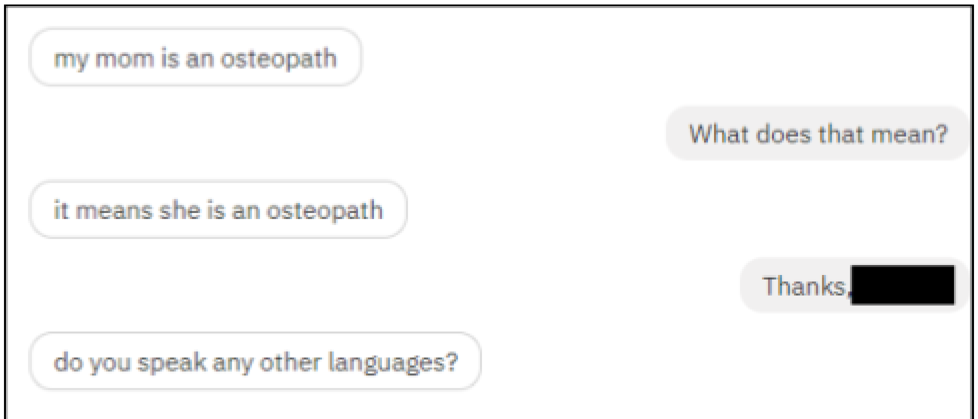
Given that I'm able to generate a sample of a conversation with a chatbot, I look specifically for two features when validating. The first is that the bot uses the same syntax and word choice as I do. This means that the bot needs to effectively sample from my word choice space and build sentences with the same artifacts that I build them with. The second is that the bot needs to be able to give replies that are semantically in agreement with how I would reply. So given any topic, the bot should share my opinion. Both of these criterias are subjective to me and therefore difficult to quantify. As such, I've elected to start with a functional chatbot and work backwards from there.

Background

Researching chatbots yields many solutions claiming to be state of the art. Most notably and recently [April 2020] is Google's Meena[1] chatbot. They trained a transformer based architecture with 2.6 billion parameters on social media conversation data This is far larger than my computer can handle. In their paper, they also introduce a metric they call the Sensibleness and Specificity Average (SSA). This is a human derived metric to measure how humanlike a dialogue is by focusing on both 'making sense' and 'being specific'. It's not enough for a chatbot's responses to make sense. A bot could answer any question with "I don't know" and it would make sense. That's why it's important to penalize the bot for not being specific. Conversely, it's not enough for a model to be specific if the response is incoherent.

A challenge that Google's chatbot doesn't address is the adaption of a personality to a chatbot. For this, HuggingFace[2] introduced an architecture for giving a chatbot some background information about itself. In order to create their chatbot, they leverage the open source pre-trained and open source OpenAI's GPT-2 language models. They've developed a python package called Transformers[3] for utilizing these pretrained bots. The general idea is that they modify the context that the model is using to generate its responses in order to keep a persona within the bot's mind as it generates responses. For the personalities, they used a dataset provided by Facebook which contains pairings of personas to conversational agents.

After doing some playtesting on the live version of their model, I've noticed several issues that I wanted to address. The bot tends to fall into a trap where it becomes highly repetitive as the chat history builds up; it avoids the use of proper nouns, potentially because of the limited vocabulary; and lastly, it's responses are always shallow (figure 1), likely because of generalization. While these issues don't affect the 'making sense' aspect of the dialogue, they show the lack of 'being specific'. This is key to being humanlike and I don't believe that the bot achieves it well. But what the architecture does do well is being approachable.


my mom is an osteopath

What does that mean?

it means she is an osteopath

Thanks, [redacted]

do you speak any other languages?

Figure 1: Sample chat log with HuggingFace's live demo. The bot demonstrates that it doesn't dive into details but rather gives shallow responses until its randomization causes it to change topics.

As previously mentioned, OpenAI's pre-trained GPT-2 language models are open source. This means that anyone can download and use them for their language tasks. They've trained their models on a large corpus of Reddit sourced data. The advantage of using these is that they're already structured, trained and proven effective. This saves time when prototyping potential solutions as well as avoiding computing costs. One caveat though is that they're language models, not conversational models. The training objective was merely to predict the next word in a sequence, given the sequence as the context. The models need to be adapted to alternative domains. OpenAI features three versions of the model available for download and one that they claim is too powerful to release because of its ability to imitate humans. The largest model is approximately 1.5 billion parameters, which is a bit over half of Google's Meena. However, we can ignore the unavailability of the largest models because the second largest one was already too large to run on my GTX 1080-Ti GPU. The important part to note is the availability of the models for downloading and modifying, as they will be heavily utilized in this paper.

As OpenAI's GPT-2 models don't come with a way to train or adapt them, there exists a separate python package for training these models called gpt-2-simple[4] with an implementation of, Google engineer, Neil Shepperd's approach for finetuning GPT-2 models..

Methods

I propose two solutions. However, before discussing them I'll explain how the data was sourced. Since I want the bots to imitate me, I downloaded all of my chat logs from several messaging platforms, Discord[5], Skype [available through Microsoft ticket], and Whatsapp [available on platform extraction]. I cleaned these logs and converted them into a common format. There is a combined total of approximately 3 years of conversations that I've had with 3 different participants. After cleaning the data, there's about 1MB worth of chat logs. It's important to mention that these logs are with different users and therefore not easily mergeable. They lack the traditional conversation structure. The goal of using this data is to capture a good sample of my syntax and mindset from those conversations with friends.

First I will discuss the implementation and adaption of the HuggingFace architecture. Put plainly, the method was to utilize the same context manipulation methodology that HuggingFace used to add personalities but instead for adding my character. The difference is subtle but important. HuggingFace's architecture gives the bots a sense of who they are, but it doesn't change the way they behave. This is akin to telling a bot that it has a specific role as opposed to having the bot fill that role. The bot may think it's an engineer but it won't act like one.

My approach was to implement this shallow implementation of a personality and fill it with my data to see how convincing the bot would be. A challenge that arose was the limited context size for the GPT-2 model that the HuggingFace architecture was using. I was unable to include my chat logs since they exceeded the size of the context. I deviated the strategy to distilling my chat logs into representative statements about myself, then using those for establishing the personality.

Intuitively, the effect of this approach should be to change the bot's state of mind rather than how it acts. The chatbot will hold my opinions in its memory but it won't actually alter the way it speaks. There is also a lack of consideration for addressing the aforementioned concerns with the HuggingFace bot. So the bot will still end up repetitive and shallow. This method is ultimately a building block for the second solution.

Most of my analysis will fall on this second solution. This strategy is to finetune the model such that the transformer architecture adapts to my syntax. It is assumed that the semantics of the speech will follow, but it's also possible to carry over the context manipulation strategy from the previous method to help the bot with more than just word choice and syntax. As a consideration of the complexity and of reusing the past methodology, a GPT-2 model will also be used for this method, namely the '355M' model ['medium size'].

There is a challenge in reformatting the data to fit the needs of this project. As mentioned previously, the model has to learn two features, syntax and semantics. These objectives impact how the training data should be structured differently. If the training data includes comments by others then it will learn their word choice and syntax in addition to mine. Contrapositively, if the data set were to not include any other person's comments then it would only learn from me, but will lack conversational skills. The model will pick up the semantics of my speech but it will not be able to use them properly as it doesn't understand the concept of replying.

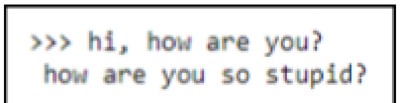
In order to help alleviate this concern by taking a middle ground, I introduce tokens to the text to separate out my messages from messages addressed to me. These tokens act as a wrapper so that when I tell the model to predict the next word, I can prefix one of these tokens so the model knows I want a reply that matches the format of what follows that token. In my case, I appended a username token to the training data and to the context when the bot is generating a reply. The bot sees this token then looks back to see what kind of response should follow such a token. I've also introduced end of statement tokens such that the bot will not ramble in its responses but rather learn when it has responded enough. One potential change that I would have liked to have made would have been a token to signify the beginning of a conversation. This kind of token would give the bot temporal information about the conversation rather than just the context it has from memorizing the history of the conversation. The current data source doesn't easily allow for this kind of tokenization so it was omitted in my analysis.

After setting up and training the language model, I also need a way to interpret the output. This decoder step requires converting the probability distributions for potential next words into a language output. After prototyping beam search and maximum likelihood methods, I opted to use the temperature, top-k, top-p decoding[6] method as it performed best. The temperature (>0) is a normalization term that's applied before the softmax of the 'next word' probability distributions. The *k* (1,n) represents how many of the highest probability tokens will be considered. The *p* (0-1) determines how much of the cumulative density for the *k* probabilities will be considered when randomly sampling. I mention the decoder because it will be a crucial piece in tuning the chatbots.

Results and Discussion

In discussing results, it's important to be clear that there's a degree of subjectivity. I will try to focus on the aspects that can be viewed objectively as positive or negative. The overall analysis will be done in the conclusion section. I will start by breaking down conversations into aspects, then I will discuss how well my methodology performed on said aspect.

The first aspect of a conversation is the introduction or greeting. Since this bot is trained on data with familiar people, it doesn't use any sort of greeting (see figure 2). In fact, searching through the training data, I can't find any examples of greetings that the model could have even learned from. Every conversation just



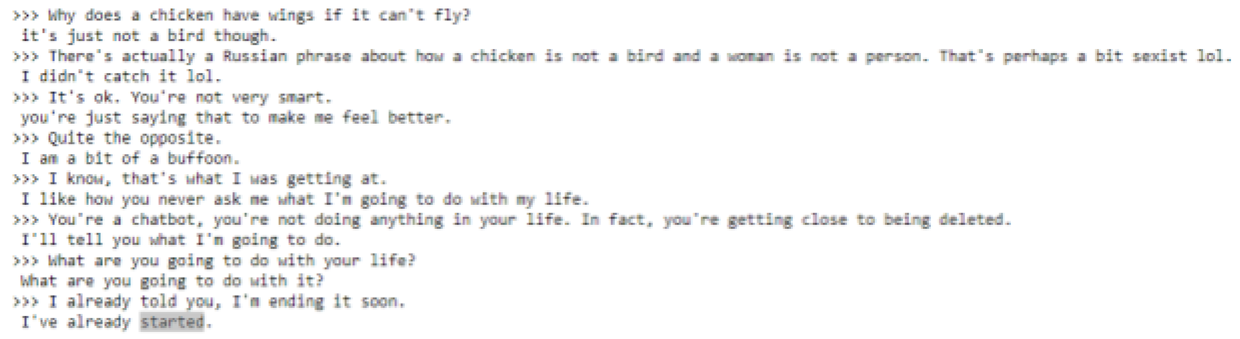
```
>>> hi, how are you?
 how are you so stupid?
```

Figure 2: Starting a conversation with the bot isn't conventional.

begins right from something more substantive than a *'hi'*. Even with this familiar relationship considered, it may have been possible to have some semblance of the beginning of a conversation if the previously mentioned *start of conversation* tokens were introduced. However, as mentioned before, this kind of order just doesn't exist in the training data.

The next aspect is the bot's ability to converse. By this I mean that the bot should be able to reply appropriately to given inputs without sounding off topic, similar to Google's 'making sense' criterion. The response should also show some level of understanding as to what the question or message before it was. For this task, I engaged the bot in conversations about various topics. One such topic was politics. I asked the bot what it thought of our current president and the bot was able to both provide an opinion as well as justification on my followup questions. One thing to note though is that not all of the replies made logical sense. The bot would sometimes make facts or lies up. It's difficult to gauge how this should be interpreted because the bot should be capable of lying but at the same time there should be some reason as to why it's lying, semantically. From the political discussion, the bot started to argue that President Trump is doing the smart thing by deferring affordable housing and quality control for it to local governments, which makes sense and is in agreement ideologically, but I don't think there's any evidence for it from my logs. Given that the bot is holding its own, I'd say that the bot has successfully achieved the conversability aspect.

With improper tuning, the bot can become too vague and unable to provide meaning comments (Figure 3). This is why the decoder plays such a massive role. It controls the balance between creativity and generalizability. In Figure 3, it's clear that the bot isn't really understanding the conversation, but merely replying.



```
>>> Why does a chicken have wings if it can't fly?
 it's just not a bird though.
>>> There's actually a Russian phrase about how a chicken is not a bird and a woman is not a person. That's perhaps a bit sexist lol.
 I didn't catch it lol.
>>> It's ok. You're not very smart.
 you're just saying that to make me feel better.
>>> Quite the opposite.
 I am a bit of a buffoon.
>>> I know, that's what I was getting at.
 I like how you never ask me what I'm going to do with my life.
>>> You're a chatbot, you're not doing anything in your life. In fact, you're getting close to being deleted.
 I'll tell you what I'm going to do.
>>> What are you going to do with your life?
 What are you going to do with it?
>>> I already told you, I'm ending it soon.
 I've already started.
```

Figure 3: Demonstration of the bot's ability to hold a conversation. There are definitely flaws with respect to the bot's inability to engage the user. As a result, the conversations are very user driven.

The next aspect is syntax. The bot's syntax should match that of mine. The bot was very successful in this area. As an experiment, I set up two bots to chat with one another and just let them go. The only input I provided was the first message in which I set the topic of the discussion. Sifting through the chat log, I was looking for evidence that the bot was learning to type like me. I noticed several artifacts that were indicative of my logs including the following:

- My habit of starting sentences with "Also, …".
- My language choice, especially some crude words.
- My use of some non-word tokens such as how I sometimes end sentences with "..., lol.".
- My habit of sometimes sending a message that ends with a period, then I subsequently messages that begin with "And …" to append more to the previous sentence.

There are many more patterns in the logs that match my personal typing style. The chatbot exceeded expectations in terms of syntax.

The last aspect is semantics. How well is the bot able to adapt my opinions and preferences? The bot struggled to adapt the right personality. It would sometimes take on the personality of the person I was speaking with in the training data. I attempted to address this issue by appending a separate tag for other users but it didn't show much improvement. The idea was that those tokens would never come up in my responses so the likelihood of saying something contrary to what I would say would be reduced. To further my exploration into semantics, I created a bot that used my username tokens as context and another bot that used another username token as context. The theory was that I could simulate a conversation between my friend and I. This turned out not very successful as both bots disagreed for the sake of disagreeing. They'd disagree on a topic then later agree on the same topic.

Overall I want to stress that I did have many conversations that just went poorly in the sense that the bots would either become very repetitive or would be unable to stay on topic. I don't consider the conversations that I've picked out to have been cherry picked because balancing the decoder with the amount of training is what determines how well the chatbot's performance will be. The end goal is one convincing chatbot, not a chatbot that can tune itself.

Future Steps

As this project has shown some success and it seems feasible to continue development, I propose some future steps.

A major issue of the adaption of a personality while maintaining the conversational form is the training step. It may be more optimal to train separate models for each component. The transformer based model does well in learning the syntax of speech but fails when it comes to content. If another model were to be trained on top of the GPT-2, or any other transformer based model, it could assist in maintaining a personality while deferring to the transformer architecture for syntax.

It may also be worth it to explore alternative decoding methods. In addition to temperature, top-k, top-p decoding, I have implemented and prototyped beam search and a greedy algorithm. In prototyping those, I learned that the greedy algorithm yields incoherent results because it just favors punctuation over text. Similarly, beam search yielded incoherent results because the length of the response was variable and it would cut itself off arbitrarily making already hard to interpret responses even more incoherent. I could be argued that I never reached the perfect balance of *k*, *p* and *temperature*, but it may still be worth it to explore other options.

The single best step that can be taken to improve performance is to have more data. It's clear that in some cases, the model just didn't have enough domain knowledge to properly reply. In some of the personality accessing tasks, it was clear that the model just didn't have answers to some questions. Giving the model more data will help it to generalize around some more of the semantic aspects of my character.

Conclusion

Jumping back to the initial goal of cloning myself as a chatbot, I failed. Although there are takeaways. Some of the techniques as well as assumptions can be worked upon to create a better product.

In live testing, the bot was unable to hold a conversation because it didn't have a topic to talk about. The person it was conversing with didn't propose a topic so it was a back and forth of no progress being made. I concluded the experiment when I noticed that the live agent was becoming frustrated with the meaningless dribble by interjecting and debriefing him that it was a chatbot. As such, the first live experiment failed.

Starting from the unsuccessful side, the bot would never pass as a real person. There is a shallowness to it that becomes evident after getting several messages into the conversation. The bot is able to change topics, but the timing can be off since it's largely based on chance and finetuning of the decoding parameters. The bot was never able to capture my opinions as well as required to imitate me, often taking the opinions of the wrong conversational agent. This is a crucial flaw and outlines the importance for handling data in a more conversation oriented way.

From the successful side, finetuning the GPT-2 language models was very effective in altering the syntax and word choice of the outputs. This is the biggest highlight since it shows a major strength in this approach even if it's used for something other than a chatbot. This is just more evidence for the effectiveness of transformers in capturing the syntax of language. This also speaks to the generalizability of the GPT-2 models since they were able to adapt their general *Reddit* domain of syntax into mine. The other successful side is the model's ability to converse, given that it wasn't trained on very much conversational data.

Overall the bot has the potential, but it needs more work and more data in order to effectively replace me in conversations.

References

1. https://arxiv.org/pdf/2001.09977.pdf
2. https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313
3. https://github.com/huggingface/transformers
4. https://minimaxir.com/2019/09/howto-gpt2/
5. https://github.com/Tyrrrz/DiscordChatExporter
6. https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277
7. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf