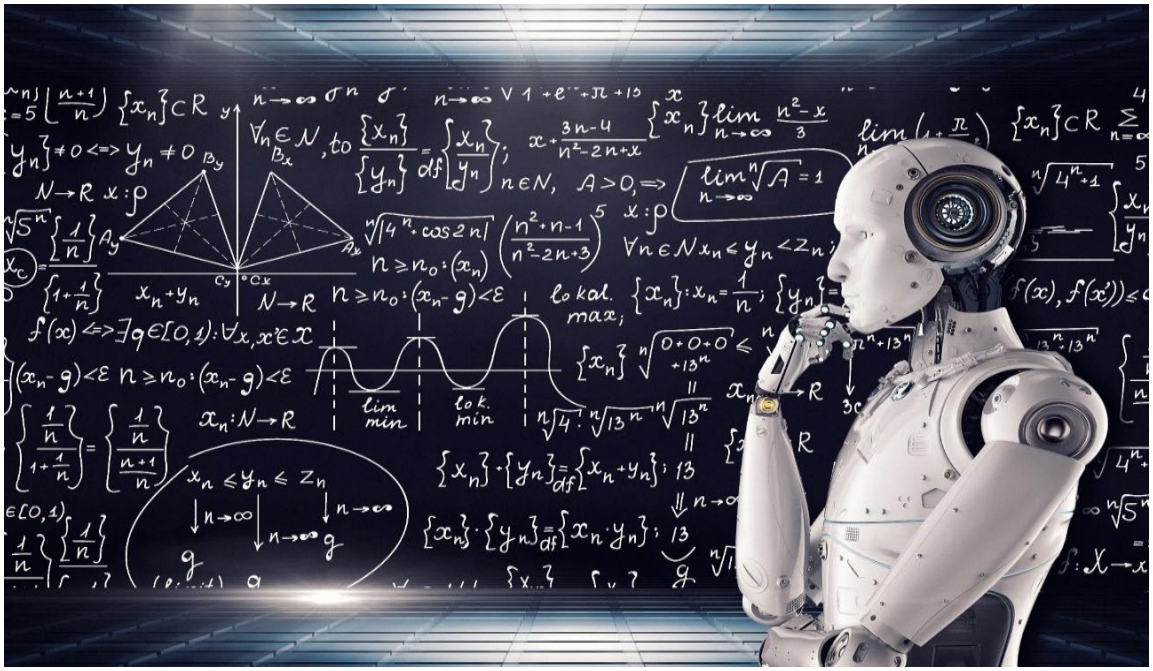

Advanced Generative Chatbot Design **(Using Cornell Movie Dialogs Corpus)**

Group 6: Paul Parks, Alden Caterio, Adam Graves

Shiley-Marcos School of Engineering, University of San Diego

AAI-520-03: Natural Language Processing

October 2023

Table of Contents

Abstract

This system is a Chatbot developed utilizing Natural language processing (NLP) services. NLP is a field of computer science that deals with the interaction between computers and human (natural) languages. It is used in a wide range of applications, including machine translation, speech recognition, and text analysis. Chatbots are computer programs that can simulate conversation with humans. They can be used in a variety of settings, including customer service, education, and entertainment.

The basis of the development is an NLP-based chatbots that can provide information about movies from the Cornell Movie-Dialog Corpus. The chatbot will be trained on a dataset of movie reviews and ratings from IMDB. This will allow the chatbot to understand the nuances of human language and to provide accurate and informative responses to user queries.

The chatbot will be developed using the Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU is a field of NLP that deals with the ability of computers to understand the meaning of human language. The chatbot will use NLU to understand the meaning of the user's queries and to generate informative responses. NLG is a field of NLP that deals with the ability of computers to generate human-like text. The chatbot will use NLG to generate responses to the user's queries in a clear and concise manner.

The Retrieval-Based chatbot will require specific text for input in order for it to identify the intents, while the Generative Chatbot will be more human like discussions, with ability to do continuous text/discussions. The chatbot will be evaluated on its ability to provide accurate and informative responses to user queries, and on its ability to generate recommendations that are relevant to the user's preferences.

Goal

Build a user friendly chatbot that can carry out multi-turn conversations, adapt to context, and handle a variety of questions about movies.

Strategy

Step 1: Define the Objectives and Use Cases

This chatbot will communicate with any user that wishes to inquire about information pertaining to a movie. This will include any information that is available on the loaded database. When the chatbot is given a prompt or question that requests for information that is not in our dataset, the chatbot will give a default response stating that it does not have enough information to give a proper response.

Step 2: Platform and Technology

The chatbot will be a front-end web user interface allowing the user to enter a question or prompt and receive a response. Activated by Enter button or a Send button. The chatbot will consist of a pre-trained model called Llama2 from Meta that has been pre-trained on an extremely large knowledge base consisting of various datasets and conversations. Since it has been trained, the chatbot can respond to broad questions in a conversational manner. A custom dataset, generated from the Cornell Movie-Dialog Corpus, will be applied to the chatbot so it can be fine-tuned for our purposes. After training, the chatbot will be capable of responding to input texts related to the training data.

Step 3: Extract, Transform, and Load (ETL)

Perform the Extract, Transform, & Load (ETL) process in which we will have a full dataset usable for parsing in our code. The Cornell Movie Dialog Corpus consists of five files that contain a metadata-rich collection of fictional conversations extracted from raw movie scripts. The files contain:

- 220,579 conversational exchanges between 10,292 pairs of movie characters
- 9,035 characters from 617 movies

- 304,713 utterances
- movie metadata such as genres, release year, IMDB rating, number of IMDB votes
- character metadata such as gender (for 3,774 characters) and position on movie credits (3,321 characters)

The movie_titles_metadata.txt file contains information about each movie title. The information includes fields for movie ID, movie title, movie year, IMDB rating, number of IMDB votes, and genres.

The movie_characters_metadata.txt contains information about each movie character. The file includes fields for character ID, character name, movie ID, movie title, gender, and position in credits.

The movie_lines.txt file contains information about the actual text of each utterance in the movie. The file contains fields for line ID, character ID, movie ID, character name, and utterance text.

The movie_conversations.txt file contains the structure of the conversations, encoded using character ID, movie ID, and line ID. This can be matched up with the information from the movie_lines.txt file to reconstruct the actual conversation between characters.

The raw_script_urls.txt file contains the URLs for the raw script of each movie.

Step 4: Data Cleaning

We extracted the data from each file and stored the data in usable Pandas DataFrames, retaining all the original information so no data is lost. Almost all the data was unaltered except the gender symbols were changed from single characters to full words: female, male, or unknown.

Step 5: NLU and NLG

NLP concepts are applied when the custom dataset is used to fine-tune the LLM. The custom dataset is generated from the movie data DataFrames and consists of thousands of datapoints, each containing a prompt and a response. The prompt contains two parts: context and goal. The context provides the chatbot with information on how to respond to the user. The goal is a possible input that the user may enter to the chatbot, either in the form of a question or an instruction, command, or request in the form of an imperative sentence. The response is an appropriate reply to the prompt, possessing a specific format determined by us.

The custom dataset consists of datapoints which cover basic questions that the user might ask the chatbot, accompanied by a correct response based on the Cornell Movie Dialogs corpus. Here are a few examples of questions that are addressed by our custom dataset:

- When was [movie title] released?
- What is the rating of the movie [movie title]?
- What is the genre of the movie [movie title]?
- Do you have the full script of the movie [movie title]?

After the custom dataset was generated, we utilized various functions from a popular machine learning website called HuggingFace. HuggingFace provides functions for training an LLM through its AutoTrain utility. Through the AutoTrain utility, we can initialize training parameters such as batch size, epochs, and block size and apply our custom dataset once all parameters are set. Since the Llama2 LLM is already pre-trained on several billions of parameters, it can accomplish basic NLP techniques such as named entity recognition and intent classification. It can also respond to prompts in a conversational manner. The goal of fine-tuning the model with our custom dataset is to steer the model towards a specific field of knowledge, which is in this case movie information from the Cornell Movie Dialogs corpus.

With pre-training and fine-tuning, the model can achieve natural language understanding (NLU), identifying significant terms in a user's input. Then, natural language generation (NLG) is accomplished when a relevant output is generated in accordance with the initial prompt within the custom dataset.

Step 6: Documentation

Documentation for the project and the chatbot was managed through GitHub and Microsoft OneDrive. The code was developed in several Jupyter Notebooks through Google Colab and JupyterLab. Tasks and overall progress were tracked using Trello, an online productivity website.

Step 7: Deployment and Testing

The chatbot was deployed through a web-based application using Vercel, a cloud platform for building and developing serverless applications online. Through Vercel, the chatbot model is called, and the entered input from the user is also sent to the model. The model generates a response to the input, which is then fed back to Vercel to be displayed to the user. The chatbot's chat history is displayed.

To test the chatbot's performance, we determined the quality of its responses manually. For one of our methods, ten questions were entered and the chatbot's response was recorded and analyzed. The response was rated as correct, partially correct, or incorrect, and each rating was given a score of 1.0, 0.5, and 0.0 respectively. For a set of ten questions that were based on the questions generated in our custom dataset, the chatbot scored 3.0 out of 10.0 points. Only one response returned as correct, and a few other responses were partially correct. Many of the responses were incorrect because the chatbot was using information not included in the Cornell Movie Dialog corpus. The data that it uses to respond is probably derived from pre-training. Another way we tested the chatbot was to base subsequent inputs off of previous

inputs. For example, asking the chatbot “When was [movie title] released?” and then following that question with “What rating did it receive?” The chatbot should be able to retain the history of the conversation and determine that the “it” in the second question refers to the movie title of the first question.

To improve performance, the learning rate and number of epochs was revised several times. The dataset itself cannot be improved or revised, and the scope of this project calls only to use the Cornell Movie Dialogs dataset for the chatbot, so applying an additional dataset for more training data was not possible. Having a relatively small dataset may affect chatbot performance. The Cornell Movie Dialogs dataset is not comprehensive and not up to date, which can lead to incorrect responses.

Step 8: Presentation

Our presentation on the project was created using Google Slides and recorded through Zoom.

Design

Retrieval-Based Chatbot (#1 built)

The retrieval-based chatbot’s main body of code consists of multiple, consecutive if-else statements that are used to generate responses, identify subjects in the user’s input question, and determine a response to the question. Here is an example of the function defined to answer questions. The code identifies the subject of the question, then generates an appropriate response based on the identified subject. The if-else statements address questions involving movie conversations, movie release years, and IMDb ratings.

The chatbot’s performance is not efficient or adaptable. In order for the chatbot to recognize specific words and generate appropriate responses, the developer will need to hard-code all possible correct responses, accounting for the multitude of ways to ask the same

question. This would require a large amount of manual work. Also, this chatbot is unable to deal with typos and identify intent within a prompt with misspelled words.

Retrieval-Based Chatbot: Code Sample (From AAI520_Final_Group_Train_Generation.ipynb)	def answer_question(self, query): #The section where the system matches the query
	subject, movie = self.identify_subject(query) #taken from the def identify_subject
	
	if subject and movie:
	matching_conversations = [conv for conv in self.corpus if isinstance(conv, dict) and movie.lower() in conv.get("movieTitle_x", "").lower()]
	
	if matching_conversations:
	conversation = matching_conversations[0]
	response = self.generate_response(query, conversation, subject)
	return response
	else:
	return "I couldn't find any information about that movie."
	
	elif subject == "releaseYear":
	year = re.search(r'\b\d{4}\b', query)
	if year:
	year = year.group(0)
	matching_movies = set() # Use a set to store unique movie titles
	for conv in self.corpus:
	if conv["releaseYear"] == year:
	matching_movies.add(conv["movieTitle_x"])
	
	if matching_movies:
	response = f"The following movies were released in {year}: {', '.join(matching_movies)}."
	else:
	response = f"No movies were released in {year}."
	return response
	
	elif subject == "timdbRating":
	year_match = re.search(r'\b\d{4}\b', query)
	if year_match:
	year = year_match.group(0)
	movies_for_year = [conv for conv in self.corpus if isinstance(conv, dict) and conv.get("releaseYear") == year]
	if movies_for_year:
	highest_rated_movie = max(movies_for_year, key=lambda x: float(x.get("imdbRating", 0)))
	highest_rating = highest_rated_movie.get("imdbRating", 0)

	movie_title = highest_rated_movie.get("movieTitle_x", "Unknown Movie") response = f"The highest rated movie in {year} was '{movie_title}' with an IMDb rating of {highest_rating}." else: response = f"No movies were released in {year}." return response
Retrieval-Based Chatbot: Output Sample	You: what year was the movie xxx released in? Identified subject: releaseYear, movie: xxx Please make sure to use the key words to your question, such as, movie, year, release, script, etc. Chatbot: The movie xxx was released in 2002. Note: the line "Identified subject: releaseYear, movie: xxx" shows that the tokenizing of the input line did identify the intent as for the subject and the movie name, from this the system could do the query match to the data repository. Next: shows how this version of a chatbot can't do a continuous discussion, not like the Generative that can. You: and who are the lead actors? Identified subject: None, movie: None Please make sure to use the key words to your question, such as, movie, year, release, script, etc. Chatbot: I can only answer movie-related questions.

Generative Model Chatbot (#2 built)

The generative chatbot model solves issues that are faced with the retrieval chatbot model. Through NLP, generative chatbots can identify intent, subjects, and significant terms without any hard-coding or explicit definitions. They are also versatile enough to deal with misspelled terms in the user's prompts.Our chatbot backend design is broken down into several Jupyter Notebooks. Here is a breakdown of each notebook:

- **AAI520_Final_Group_Train_Generation** – Extracts all the information from the files in the Cornell Movie Dialogs dataset into Pandas DataFrames, then generates a custom dataset consisting of datapoints formatted in a prompt-response fashion.
- **AAI520_Final_Group_AutoTrain_LLM** – Gets the custom dataset and trains the pre-trained Llama2 LLM using HuggingFace's AutoTrain utility.
- **AAI520_Final_Group_Inference** – Loads the trained model and generates responses to user input. This code is called by the front-end application in Vercel.

Using HuggingFace's AutoTrain utility, we can train a pre-trained LLM with our dataset. These few lines of code allow the chatbot to be trained on any dataset, as long as it is formatted correctly. With enough training, the chatbot is able to generate responses in a conversational format, just as if another human were writing the responses. Compared to retrieval chatbot development, this method is much more efficient and does not require nearly as much manual work or coding.

Generative Chatbot: Code Sample (From AAI520_Final_Group_ AutoTrain_LLM.ipynb)	`!autotrain llm \`
	`--train \`
	`--model ${MODEL_NAME} \`
	`--project-name ${PROJECT_NAME} \`
	`--data-path data/ \`
	`--text-column text \`
	`--lr ${LEARNING_RATE} \`
	`--batch-size ${BATCH_SIZE} \`
	`--epochs ${NUM_EPOCHS} \`
	`--block-size ${BLOCK_SIZE} \`
	`--use-int4 \`
	`--use-peft \`
	`$( [[ "$PUSH_TO_HUB" == "True" ]] && echo "--push-to-hub --token ${HF_TOKEN} --repo-id ${REPO_ID}" )`

Links

Codes:

GitHub: https://github.com/agraves13/AAI520/tree/main

Chatbot: https://msaai-520-final-project-web-app.vercel.app/

Slide Presentation:

https://docs.google.com/presentation/d/1_B8gOttntXtNK81WETfS9s7jATuYCBCrzkNqcr73HeY/edit?usp=sharing

Video Presentation:

https://youtu.be/BbfCVgfVmj4

References

Hugging Face – The AI community building the future. (2023, October 17). https://huggingface.co/

Reduce deep learning training time and cost with mosaicml composer on aws | aws machine learning blog. (2022, October 24). https://aws.amazon.com/blogs/machine-learning/reduce-deep-learning-training-time-and-cost-with-mosaicml-composer-on-aws/

Llama 2 is here—Get it on Hugging Face. (n.d.). Retrieved October 18, 2023, from https://huggingface.co/blog/llama2

State of the art language model for NLP. Medium , Horev, R. (2018, November 17). BERT Explained:. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Converting data into SQuAD format for fine-tuning LLM models. MLearning.Ai , Bhalerao, C. (2023, April 21).. https://medium.com/mlearning-ai/converting-data-into-squad-format-for-fine-tuning-llm-models-229497b4e774

Cornell movie-dialog corpus. (n.d.). Retrieved October 18, 2023, from https://www.kaggle.com/datasets/rajathmc/cornell-moviedialog-corpus