Closed Domain Question-Answering Techniques in an Institutional Chatbot

A Senior Honors Thesis

Submitted in Partial Fulfillment of the Requirements
for Graduation in the Honors College

By
Matthew Saad
Math and Computer Science Major

SUNY Brockport, State University of New York
May 16, 2024

Thesis Director: Dr. Zakariya Qawaqneh, Assistant Professor, Computing Sciences

Closed Domain Question-Answering Techniques in an Institutional Chatbot

Matthew Saad
Computing Sciences
SUNY Brockport
Brockport, NY
msaad1@brockport.edu

Zakariya Qawaqneh
Computing Sciences
SUNY Brockport
Brockport, NY
zqawaqneh@brockport.edu

***Abstract*—This thesis presents BrockportGPT, an advanced institutional chatbot developed for the SUNY Brockport community by leveraging large language models (LLMs). Our approach integrates Retrieval-Augmented Generation (RAG), finetuning techniques, and a custom-built transformer model. The methodology involved extensive data collection through web scraping, generating synthetic question-answer pairs using GPT-4, and employing various retrieval strategies, including semantic search and reranking. Additionally, a question-topic classification system was implemented to enhance response relevance. The results demonstrate the effectiveness of both RAG and the finetuned model, each excelling in different aspects of query handling. BrockportGPT addresses the limitations of general-purpose LLMs in specialized, closed-domain environments by providing accurate and contextually relevant information. This research underscores the potential of LLMs in educational settings and offers a replicable model for other institutions aiming to enhance their information dissemination processes.**

***Index Terms*—Closed-domain question answering, institutional chatbot, Large Language Models (LLMs)**

I. INTRODUCTION

The field of Natural Language Processing (NLP) has witnessed transformative advancements with the advent of Large Language Models (LLMs) such as GPT-4 [32], LLaMA [48] [47], and Claude [3]. These models, capable of generating human-like text and understanding complex contexts, represent the pinnacle of current AI capabilities. However, their effectiveness often diminishes when applied to specialized, dynamically changing environments. This is particularly evident in niche areas such as institutional question answering, where questions like “Does SUNY Brockport offer an MS in Accounting?” frequently stump even the most sophisticated general-purpose models.

To bridge this gap, this thesis introduces BrockportGPT, a chatbot custom-built to address the unique information needs of SUNY Brockport’s faculty and students. By integrating closed-domain question answering techniques, BrockportGPT aims to offer accurate and reliable information specific to the institutional context, thereby overcoming the limitations faced by broader LLM applications. For the purpose of reproducibility and further research, all models, datasets, and the code developed as part of this project are available on our GitHub repository and HuggingFace. Links to these resources can be found at GitHub and HuggingFace.

The surge in popularity of tailored chatbot applications reflects a broader trend towards personalizing digital interaction within specific communities. Our approach leverages a blend of NLP techniques, including a vanilla sequence-to-sequence [42] encoder-decoder transformer model [50], fine-tuning of the open-source LLM LLaMA [48], and the use of Retrieval Augmented Generation (RAG) [23]. This research is driven by two primary questions: the feasibility of developing an institutional chatbot and the determination of the most effective methodology for such a development.

With the recent growth in Machine Learning (ML) and Deep Learning (DL) technologies, fueled by vast amounts of data and significant computational power, there is a unique opportunity to create specialized tools that meet the specific needs of individual communities. This work not only explores adaptations of foundational models to specialized tasks but also critically evaluates their practical implementation in the educational sector, addressing both their potential and limitations.

By developing BrockportGPT, we aims to demonstrate the practical benefits and potential of closed-domain question answering systems tailored to specific institutional needs. The contributions of this work extend beyond the SUNY Brockport community, providing insights and methodologies that can be applied to similar educational and institutional settings.

II. BACKGROUND

A. Language Modeling

The evolution of neural networks into sophisticated language models began with the application of Recurrent Neural Networks (RNNs). RNNs were designed to handle sequential data by maintaining a hidden state that captures information from previous time steps [13]. This architecture set the foundation for subsequent developments in processing sequences, including language modeling. However, RNNs struggled with capturing long-range dependencies, primarily due to the vanishing and exploding gradient problem, where gradients can become either too small or too large during training [20].

To mitigate these issues, variants such as Long Short-Term Memory (LSTM) networks [21] and Gated Recurrent Units (GRUs) [9] were introduced. Despite these advancements, LSTMs and GRUs still face challenges in capturing very long-range dependencies, as their reliance on sequential processing limits efficiency and scalability.

The introduction of the transformer architecture marked a pivotal shift in model design [50]. At the core of the transformer architecture is self-attention, which enables the model to capture long-range dependencies more effectively than RNNs and their variants. Additionally, its ability to process input in parallel significantly improved both efficiency and effectiveness. Following the development of the transformer came the generative pre-trained transformer (GPT), which improved task-agnostic performance by training a transformer in an unsupervised manner prior to additional supervised training [35].

The GPT strategy has led to the success of subsequent "GPT-n" models, including GPT-2, GPT-3, and GPT-4 [36] [6] [32]. These models have inspired other competitive LLMs such as LLaMA, Gemini, and others [48] [47] [44] [2]. From their pre-training step, LLMs can be thought of as generic text generators. To make them useful, most LLMs undergo some form of instruction tuning and Reinforcement Learning from Human Feedback (RLHF) after their pre-training step [33] [8]. This combination of pre-training, instruction tuning, and RLHF is the fundamental formula of present-day LLMs and is what users interact with through platforms like OpenAI's ChatGPT, Anthropic's Claude, and Google Gemini.

B. *Efficiency Improvements*

While the recent growth of LLMs can be attributed to their architecture, their success heavily relies on the scale of their training data and computational resources. This dependency creates a high barrier to entry, making the adaptation of LLMs across different domains challenging. Therefore, efficiency in training and deploying LLMs has become a critical area of recent research.

One major advancement in improving efficiency is the use of quantization techniques. These techniques reduce the memory requirements of LLMs by lowering the precision of the model weights, typically from 32-bit floating point to 4-bit integers, with minimal impact on performance [10] [14] [15].

In addition to quantization, efficient fine-tuning approaches like Low-Rank Adaptation (LoRA) have been developed. LoRA enables efficient tuning of pre-trained models by updating smaller, low-rank matrices during training instead of directly updating entire layers [22]. This approach significantly reduces the computational cost associated with fine-tuning. Quantized versions of LoRA, such as QLoRA and LoftQ, further enhance this efficiency by combining quantization with low-rank adaptation [11] [24]. Subsequent improvements like LoRA+ and ReLoRA continue to advance the field, making LLMs more computationally efficient and accessible both during training and inference [19] [26].

These strategies represent significant strides towards reducing the computational barriers associated with LLMs, enabling their broader application across various domains, and making advanced language models more accessible and practical to deploy.

C. *Adaptation to Specific Tasks*

Recently, fine-tuning open-source LLMs has demonstrated significant potential. Models like GPT4ALL [1], Alpaca [43], and WizardLM [54], which fine-tune LLaMA [48] [47] models with specialized datasets, can further improve overall performance.

One notable example is PMC-LLaMA, which fine-tunes LLaMA using biomedical academic papers, medical textbooks, conversational dialogue, and more. This process significantly enhances PMC-LLaMA's performance, even surpassing ChatGPT in medical contexts [52].

Coding has been another major application for LLMs, with examples such as Codex [7], GitHub Copilot [16], and Code LLaMA [40] leading the state-of-the-art. Code LLaMA specifically fine-tunes LLaMA-2 using up to 1 trillion tokens, significantly improving its coding capabilities, as demonstrated by its impressive benchmark results.

As the world continues to adopt LLMs in different fields, the number of applications increases. Other examples include text-to-SQL for simplified data querying [41], improved text retrieval [27] and enhancing LLaMA's performance in sequence and token classification tasks [25], among countless others.

This strategy of fine-tuning a foundational model like LLaMA for improved performance in specialized contexts lays the groundwork for the fine-tuning techniques used in this work to enhance closed-domain question answering performance.

D. *Closed-Domain Question Answering*

Given the success of finetuning LLMs for specific tasks, our goal is to apply similar techniques for question answering in a closed domain, or to do with specific topics. An institutional chatbot is a part of this group, since questions about an institution are limited in scope. This is an ongoing area of research with common approaches being finetuning like how LLMs are adopted to specific tasks, and through Retrieval Augmented Generation (RAG), which offers a simpler method for closed-domain question answering [23].

Given the success of fine-tuning LLMs for specific tasks, we aim to apply similar techniques to question answering within a closed domain. Institutional chatbots, such as those designed for universities, are ideal candidates for this approach since their queries are limited to specific, predefined topics. This focused nature of questions allows for more targeted and efficient fine-tuning. In our work, we compare the effectiveness of fine-tuning with another prevalent method: Retrieval Augmented Generation (RAG). RAG enhances the system by integrating an information retrieval step prior to generating responses, offering a streamlined and effective solution for closed-domain environments [23]. By evaluating these approaches independently, we aim to determine the most effective strategy for improving question answering performance in closed-domain settings.

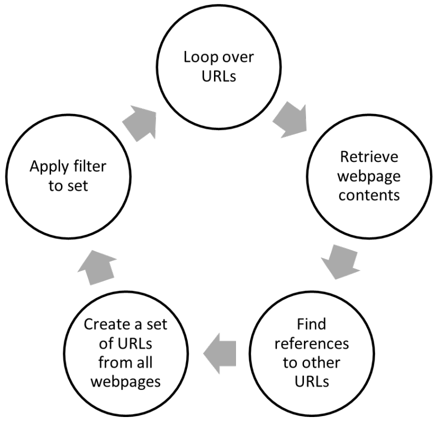

Fig. 1. Web scraping diagram

RAG works by augmenting the results of an LLM with an information retrieval (IR) step prior to response generation. The cornerstone of any successful RAG implementation lies in its retrieval mechanism. Recently, semantic search has gained popularity for this task. Unlike traditional keyword-based search, semantic search queries the meaning of the information by creating vector representations and using functions like cosine similarity to measure similarity. Early versions of this functionality include latent semantic indexing [34] and word2vec [28]. More recently, contrastive, unsupervised training methods have become the state-of-the-art for creating embedding models [30]. By improving the effectiveness of IR, RAG becomes more capable of answering complex user queries. While RAG has become a widely applied technique in industry, with many companies leveraging it for practical applications, there is relatively less published academic work on the subject.

The strategies discussed throughout this review lay the groundwork for the closed-domain question answering techniques used in this work. Our aim is to develop an institutional chatbot, BrockportGPT, tailored specifically for the SUNY Brockport community. This next section details the methodological approach employed to achieve this objective.

III. Methodology

Like most institutions, there does not currently exist a chatbot of this kind at SUNY Brockport. Accordingly, our methodology covers the complete end-to-end process of creating a closed domain chatbot. This includes data collection and preparation before any closed domain techniques are applied. After discussing the relevant datasets created, we will discuss three strategies to make the chatbot: sequence-to-sequence modeling from scratch, finetuning a pre-trained LLM, and RAG.

A. Data Collection and Preparation

Data is the cornerstone to any successful application in ML. The most accessible and concentrated place for information relevant to this project is the SUNY Brockport website, which contains information about admissions, financial aid, academic departments, and much more. Subsequently, a decision was made to use the SUNY Brockport website as the sole place for data collection. As a result, web scraping must be performed.

This section aims to address the process of web scraping and data cleaning. Additionally, we will discuss synthetic question generation, which is a crucial component to the BrockportGPT project.

1) Web Scraping: Web scraping is the process of extracting data from a website. A script was developed to manage this process. Starting at the SUNY Brockport website base URL the script fetches the webpage and stores it. The stored webpage is then parsed for references to any other webpage. For each of these references a full URL is constructed. Using this set of URLs, for each new URL discovered their corresponding webpage is fetched and stored. This process repeats until no new URLs are found. This process is illustrated in Figure 1. Each iteration of the process is treated as a layer of depth. The results used throughout this paper are recorded in December 2023, which scraped 4,573 webpages and stopped at depth 10 after 6 hours.

Following proper procedures around web scraping is ethically and practically important. Two steps are taken towards this goal in this project. Those are: abiding by robots.txt files and adding a timer between requests when scraping to avoid overwhelming the server. Firstly, the robots.txt file is a part of any website with the goal of defining disallowed places for agents to visit. It exists to prevent overloading the server and keep some parts of the website private. It is a "do not enter" sign for web robots. While the robots.txt file will not restrict access, it is best practice to abide by it. Secondly, a timer between requests is added. The timer used is randomly selected from a uniform distribution between 1 and 3 seconds. This significantly reduces the load on the website, ensuring its accessibility to others whilst scraping. Removing this timer could take down the website, acting as a denial-of-service attack. Alternatively, it could result in an IP ban targeting the scraping script which is disruptive to our scraping process.

One challenge with data collection in this project is the continuously evolving nature of information. While the results are relevant to December 2023, in a live system we acknowledge that continuously updating the dataset is important. However, for our purposes, a static dataset is acceptable.

2) Data Cleaning: Having a cleaned data is necessary for a well performing system in any facet of ML. In this section we will discuss the cleaning process with a focus on text extraction and webpage filtering. The goal of this section is to craft a dataset that is both relevant and data rich. Towards that end, various decisions are made to omit certain information that we deem irrelevant to most users. Firstly, the data must be processed through text extraction.

In the web scraping process discussed prior, webpages themselves are fetched and stored. However, in their raw form these webpages offer little to no useful information. This is primarily because they are HTML files cluttered with buttons, images, dropdowns, etc. To make these files useful, they are parsed for their text contents. To facilitate this, a decision was

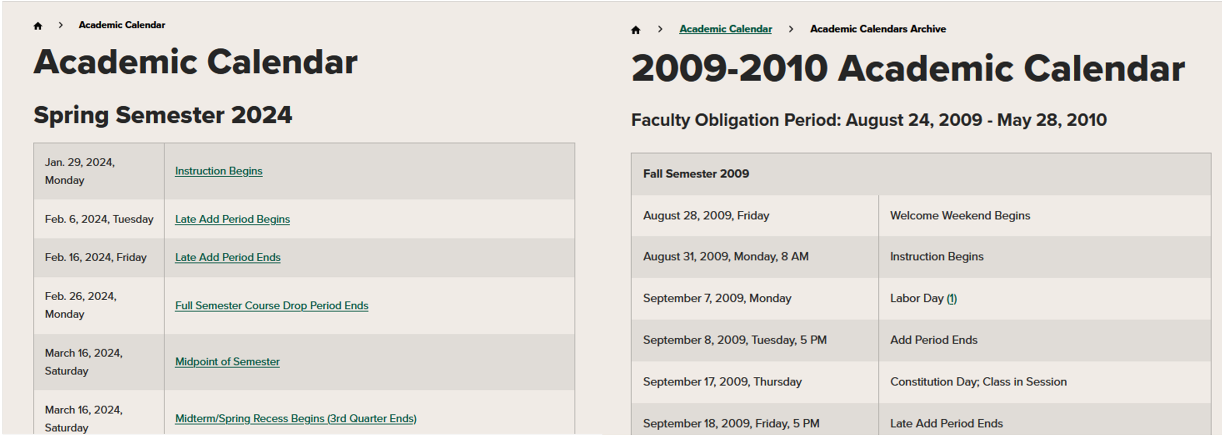

Fig. 2. Showcase of removed irrelevant data. Left is a 2024 academic calendar. Right is a similar academic calendar from 2010. This shows how similar webpages can look whilst containing quite different information.

made to use Trafilatura [4], a fully featured Python library built for extracting text off various web formats. Using Trafilatura, for each webpage fetched in the prior step, its text is extracted and stored. Once the text has been extracted, we perform rudimentary data cleaning.

This cleaning step includes deduplication along all axes, removal of failed requests according to status code, and the application of the following static filters aiming to exclude irrelevant data: professor webpages, transfer credit details, and archives. Discussing each of these, professor webpages are removed to contribute towards the data richness of our dataset. On the SUNY Brockport website, it is exceptionally rare to find a professor who uses their webpage. Primarily, these are blank spaces containing only a headshot, name, and email. Considering this, these webpages are not conducive to a data rich dataset, hence their removal. Transfer credits and archives are removed for a separate reason, primarily concerning ease of use. As we will discuss later, this dataset will be used for text retrieval. As such, having identifiable, distinct information is key. Unfortunately, as illustrated in Figure 2, these sections have large masses of information presented nearly identically. This makes text retrieval considerably more difficult. This pattern prevails generalizes across these categories. Importantly, this information is not specifically desired either. With misinformation being such a risk for this type of project, early on a decision was made to remove portions of the website like this entirely.

In addition to previously mentioned filtering mechanisms, any webpage whose extracted text is less than 275 characters is removed. This is specifically towards the goal of promoting an information rich dataset. Crucially, the character limit filter is not very abrasive, as only eighty webpages as removed during this step.

These steps are crucial in maintaining the integrity and relevance of the data, making it more actionable and focused. After their application, our original dataset of 4,573 webpages shrunk to 2,683 webpages. We find the distribution of webpages is heavily skewed towards academics, support, school policies, and campus life, which themselves make up for 2,293 of the webpages in the dataset. Next, we begin the discussion on artificial data generation.

System Prompt:
You are a helpful question maker for SUNY Brockport. Given some information about the school, your job is to parse that information and generate realistic questions a prospective student or faculty member might ask. For instance, "How can I apply for financial aid?" is a question a student could reasonably ask. Generate up to five of such questions and respond to those questions in JSON format, where "question" is one field, and "answer" is another. In your answers be enthusiastic about the school and try to be as helpful as possible.

User Prompt:
Based on the content given generate questions. Keep your tone optimistic and helpful. Think about your answer carefully before responding, and be sure to answer in JSON format starting with \n```json\n[\n{\n "question": "...",\n"answer": "..."}, ...]```\n\n"\n\nThe content is: {*content*}

Fig. 3. System and user prompt for generating synthetic questions

3) Artificial Data Generation: A key component of any chatbot application is that it can understand questions. However, so far, the dataset only contains text from the SUNY Brockport website. While such a dataset is adequate for Retrieval Augmented Generation (RAG), it is insufficient to train a question-answering chatbot. Considering this, the goal of this section is to create a synthetically generated dataset that contains question-answer (QA) pairs for model training.

Using synthetically generated questions is a recent idea demonstrated in Alpaca [43], GPT4ALL [1], and more. The idea is to use a strong general purpose LLM like GPT-4 to generate questions and answers. Using these generated questions, it is possible to train a smaller, less capable model to perform a similar task. In this project, the generated questions must be relevant to SUNY Brockport. This necessitates some outside knowledge since even a strong general purpose LLM does not have an expertise sufficient to generate questions on its own. For this task, we leverage the dataset discussed prior with LLMs GPT-3.5 Turbo and GPT-4 Turbo.

The prompts used for synthetic question generation are described in Figure 3. These prompts are specifically crafted to

What are my options for on-campus living and dining at Brockport?
Brockport offers a variety of on-campus living options to suit your needs, including Living and Learning Communities. If you're interested in becoming an RA or just want to get more involved with your home here, check out the Campus Living page for more information. For dining, the Brockport Auxiliary Service Corporation (BASC) provides a range of options and facilities, and you can learn about managing your meal plan and even employment opportunities on their site. Rated as one of the top SUNY schools for food, you're in for a treat!

If I receive a parking citation, how do I go about paying the fine or appealing it?
If you've received a parking citation, you can handle payments and appeals through the Parking and Transportation Services. They coordinate the appeals process, so don't hesitate to reach out to them during office hours for assistance.

What is the maximum duration for a bike rental at SUNY Brockport?
You can enjoy your bike rental for up to 72 hours. It's a great way to explore the area or get some exercise between classes!

Fig. 4. Example of GPT-4 Turbo generated QA-pairs

accommodate gpt-3.5-1106-preview and gpt-4-1106-preview in JSON mode, so they return responses that are easy to parse. The system prompt is designed to give general instructions to the model and inform it what the high-level task is. The system prompt remains constant throughout generation, while the user prompt is what changes based on the content. Using this prompting style and the cleaned dataset discussed previously, for each webpage in the dataset GPT-3.5 Turbo and GPT-4 Turbo are prompted, replacing 'content' with the extracted text contents. The models are used with default parameters other than temperature = 0. Temperature is a sampling technique that controls the randomness of model output. In cases where accuracy is favored over creative flair lower temperature is preferred, with temperature = 0 being the least random option.

The resulting datasets contain 12,732 and 11,582 QA pairs for GPT-3.5 Turbo and GPT-4 Turbo, respectively. Three examples of GPT-4 Turbo generated questions are listed in Figure 4. We find the questions are high quality, which is crucial for the success of the methods leveraging this dataset.

Unfortunately, generating synthetic datasets can get expensive. On our December 2023 run, the GPT-3.5 Turbo generated dataset cost \$4.36, and the GPT-4 Turbo generated dataset cost \$62.64. Expectantly, we find that on average GPT-4 Turbo creates substantially higher quality questions than GPT-3.5 Turbo, most notably in their length and complexity. Still, the GPT-3.5 Turbo generated questions are useful to engineer the prompt for GPT-4 Turbo, or when dealing with a situation where quantity of questions is more important than quality.

4) Conclusion: The data collection and preparation methodologies outlined in this section form the backbone of the BrockportGPT project. Despite the diligence kept throughout the dataset creation, there is still opportunity for future innovation and exploration. Specifically, an area for future enhancement is accounting for webpage imbalance in question-answer generation. For instance, questions about admissions and financial aid, though comprising only 3.22% of our dataset, may be of higher relevance to users. Adjusting the dataset to align with the inquiries of users more closely can significantly enhance the relevance and utility of BrockportGPT.

B. Building the Scratch Model

This section explores the motivation, architecture, and inherent challenges of developing a model from scratch. In neural networks, the configuration of weights within the network fundamentally determines its functionality and performance. Unlike pre-trained models that begin with weights adjusted through extensive prior training on vast datasets, thus exhibiting known performance characteristics, the scratch model starts with weights that are randomly initialized.

Within the realm of text generation, which question answering is a part of, in most cases a model trained from scratch is expected to perform worse than its pre-trained counterpart. This is primarily because state of the art text generation models continuously pushes the computational barrier of what is possible. Without the availability of extensive hardware, which costs millions of dollars, it is exceedingly difficult to beat the performance of models trained with that extensive hardware.

Despite this reality, building a model from scratch has some justification. Most importantly, training a model from scratch offers a unique perspective into understanding how the overall model architecture works. Despite having vastly improved performance, state of the art models are typically similar in architecture and mainly differ in their training and size. Hence understanding how the model works is intrinsically beneficial to understanding and interpreting all the models discussed in this work. Moving forward, the discussion will focus on exactly how the scratch model is created.

1) Implementation: To create any model, it is important to first pick a suitable dataset and model architecture. For this project, the goal of the scratch model is to directly answer questions. This is made possible by training the scratch model using the GPT-4 generated QA dataset previously discussed. About model architecture, a decision was made to use an encoder-decoder transformer for the scratch model. This decision stems from trends within the field demonstrating the transformer architecture exhibiting excellent performance particularly within NLP.

Prior to initiating the training loop, the dataset undergoes standardization. Standardizing the dataset has many benefits to model training, but most importantly by simplifying the inputs through standardization, the model is more likely to generalize across contexts as opposed to overfitting on specifics like case sensitivity and special characters. Practically, this reduces variability in the data, which improves performance of the model.

The exact standardization process used in this work includes lowercasing all text and removing extraneous special characters. To allow the model to answer grammatically correct responses, spaces are added around punctuation. This means punctuation such as "?"and "!" will be represented as their own tokens in the tokenization step. Finally, a special "[START]" and "[END]" token is added to the front and back of each QA-pair respectively. These special characters are crucial to the model learning when to stop its output.

```
Plaintext:     “What is AI?”
Standardized:  “[START] what is ai ? [END]”
Tokenized:     [ 3 5 7 2 11 4 ]

Plaintext:     “AI stands for artificial intelligence.”
Standardized:  “[START] ai stands for artificial intelligence . [END]”
Tokenized:     [ 3 2 6 9 10 8 12 4]
```

Fig. 5. Plain text, standardized, and tokenized side by side comparison

```
        Encoded

Context:    [ 3 5 7 2 11 4 0 0]
Target In:  [ 3 2 6 9 10 8 12]
Target Out: [ 2 6 9 10 8 12 4]

        Decoded

Context:    ['[START]', 'what', 'is', 'ai', '?', '[END]', '', '']
Target In:  ['[START]', 'ai', 'stands', 'for', 'artificial', 'intelligence', '.']
Target Out: ['ai', 'stands', 'for', 'artificial', 'intelligence', '.', '[END]']
```

Fig. 6. Encoded/decoded context/target in/target out

Once the dataset is standardized, the dataset undergoes tokenization. Since neural networks cannot directly read plain text, a tokenizer facilitates the process of converting the dataset into numbers which the neural network can read. There are various tokenizers available, but common choices include word-level, character-level, and sub-word-level tokenizers. Like their name implies, a word-level tokenizer describes a one-to-one correspondence of word-to-number. For example, the word “Hello” may map to the number “1”, and so on with others. A character-level tokenizer works similarly but assigns individual characters to numbers, e.g., the letter “a” can map to the number “5”. A sub-word-level tokenizer does a mixture of both. For example, the word “Seating” may be split into two tokens, “Seat” and “ing”, each having corresponding numeric values “2” and “3”. In this case, these sub-words can be mapped into other words, such as “Seated”, which might be “Seat” and “ed” while sharing similar tokens. These tokenizers all have benefits and drawbacks, for instance, the character-level tokenizer is more susceptible to spelling typos than the word-level tokenizer, which will rarely, if ever, make spelling typos. On the other hand, the sub-word-level tokenizer, which is increasingly common for state-of-the-art models [48] [47] [6] [32], incurs a debt in complexity. In this work, a decision was made to utilize a word-level tokenizer. An example of its usage, alongside our standardization technique is given in Figure 5.

Alongside the type of tokenizer used, a decision must be made about the number of tokens to include. Much like standardization, a subtle goal of tokenization is to further reduce variability in the model input, again towards the goal of a more generalized model. A well-known way of accomplishing this is to learn how many of each token you have in the dataset and choosing a cut-off point. For instance, the word “the” may show up one thousand times in the dataset, but the word “Ferrari” only shows up four times. To prevent overfitting, it is usually beneficial to remove words like “Ferrari” that do not show up often.

In this work we limit the vocabulary to the top 5000 tokens. This gives our model a small vocabulary, but the dataset used is also relatively small with only ˜11,000 examples. Tokens which do not fall within this top 5000 limit are replaces with an “[UNK]”, or unknown, token. Once the text has been converted to tokens, they are converted into tensors. From there, each tensor is padded, or appended with zeros, to the length of the longest sequence in the dataset.

In preparing data for training a transformer model, a necessary step involves the organization of the data into ”context”, ”target in” and ”target out” sequences for the model. These sequences can be categorized easily, demonstrated in Figure 6 the ”context” contains the user question as the model understands it, and the ”target in” contains the output as it is being generated – a portion of an answer. The ”target out” is the ”target in” shifted by one and contains the ground truth for a given prediction to help in loss computation.

Importantly, the key idea behind this type of text generation is next-token prediction, i.e., given some text, what is the most likely token, or word in this case, to appear next. To do this, the model computes a probability distribution across all the possible tokens given the input. This distribution can be sampled from finding the highest probability token. This is the “target out”, or prediction of the model. Using the ”target out” value previously computed, this process can be repeated to find the next token. To inform the model of what it previously predicted, the previously computed ”target out” value is appended to the ”target in” values, allowing the model to continue its sequence generation. This process continues until the model predicts ”[END]” as the “target out” value, indicating the conclusion of the response, and ending this process. Since the output depends on previously generated values, this makes the transformer architecture autoregressive.

The dataset is now fully prepared for training the model, where the context is inputted into the encoder, and the target ins are fed to the decoder. In a transformer, the encoder aims to abstractly represent the input context, while the decoder’s task is to synthesize this representation with previously generated outputs to predict the subsequent token. However, this simplification does not entirely capture the complexity within a transformer, highlighting a broader issue with neural networks: their lack of interpretability. Interestingly, recent advancements in LLMs, such as those demonstrated by models like LLaMA [48] [47] and GPT-n [36] [6] [32], propose eliminating the encoder altogether, arguing for a streamlined decoder-only architecture that can enhance performance. While these claims mark significant shifts in our understanding of transformer models, the scratch model developed here does not explore these newer configurations.

One of the intrinsic challenges with autoregressive models is compounding errors, where an inaccurate prediction early in the sequence can lead to a cascade of subsequent errors, significantly affecting the overall quality of the output. A common approach to this issue during model training is a technique called Teacher Forcing [51], which replaces the actual output of the model with the ”teacher” output, or ground truth of the model. For an example, if the model is prompted ”Knock Knock”, it is expected to respond, ”Who’s there”.

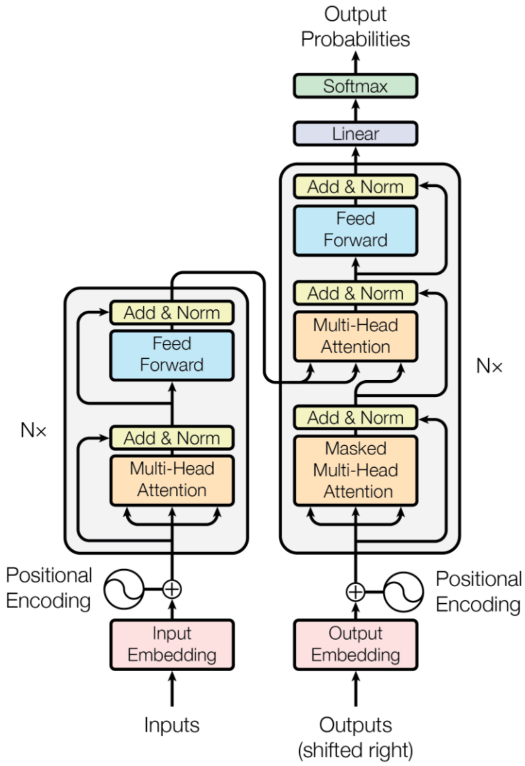

Fig. 7. The Transformer Architecture [50]

However, if the model predicts "Hello" as the next token, then the entire response is effectively derailed. Teacher forcing addresses this by injecting the ground truth response, "Who's" in place of "Hello" for subsequent predictions, allowing the model to learn later portions of a response as well. This typically leads the model to converge faster. Critically, this style of training can be detrimental during inference, where ground truth is not available. Due to teacher forcing during training, the model typically does not learn to correct for its mistakes, leading to an inconsistency between training and inference. In the literature, this is known as Exposure Bias [38]. While there exist algorithms such as Professor Forcing [18] to counteract exposure bias this work does not explore them. For simplicity, a decision was made to use teacher forcing throughout training the scratch model.

TABLE I
MODEL PARAMETERS

Explanation	Parameter	Value
Dimension of the model	dim	512
Number of encoder/decoder layers	n_layers	6
Number of attention heads	n_heads	8
Vocabulary size input and output	vocab_size	5000
Dropout rate	dropout	0.1
Batch size	batch_size	64
Number of Epochs trained	epochs	13

The scrach model parameters are given in Table I. These model parameters concerning size were chosen to follow the

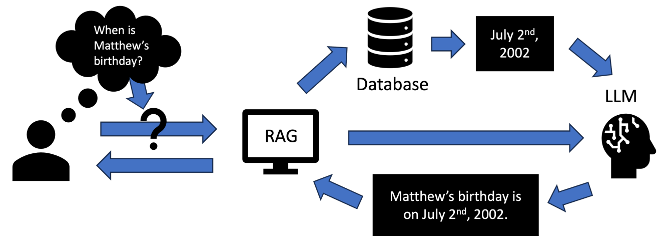

Fig. 8. RAG Diagram

base model described in the original Transformer paper. Future work may consider optimizing these parameters, which could lead to significant improvement of the scratch model. Other parameters, including batch size and number of epochs are chosen to accommodate both our hardware available and the model training.

2) Conclusion: Future work may consider implementing new algorithms mentioned to discuss the shortcomings of the architecture described throughout this section. We expect these deficiencies are a result of 1) our relatively small dataset, 2) the max model size we can train, and 3) the model architecture not being tuned to our specific case. To remedy such characteristics, scaling the scratch model across various dimensions such as dataset size, model size, etc. may be beneficial.

Additionally, on the dataset front there is room for improvement. The GPT-4 generated QA-pair dataset offers satisfactory performance in general, but the distribution of question topics in the dataset is unlikely to align with that of question topics users might be interested in. Focusing on these specific topics, and further curating the dataset to align with user interests would certainly improve the practical performance of the scratch model.

C. Retrieval Augmented Generation (RAG)

This section will discuss RAG, which is a crucial part of this project. One of the current problems with LLMs is their lack of explainability. While it is easy to interpret the output of an LLM, nobody has figured out a way to understand why that output was chosen. This implies a further problem, that is, how knowledge within an LLM is stored, and further how it can be changed. Without an ability to augment the knowledge of an LLM, it is difficult to apply LLMs to situations where domain specific knowledge is required. This motivates RAG, which is a popular strategy to assist LLMs in responding to user inquiries without changing the underlying model.

The approach RAG takes is to focus on the prompting of or input to an LLM by retrieving some ground truth text relevant to the user inquiry. Using the retrieved information, the LLM is prompted with both that information and the user inquiry simultaneously. Using this technique illustrated in Figure 8, the LLM can use the retrieved information to respond to the user inquiry more appropriately.

For applications that require external knowledge, RAG has several benefits. Firstly, RAG contributes towards the

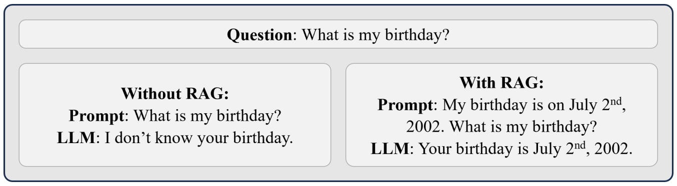

Fig. 9. With/without RAG performance comparison

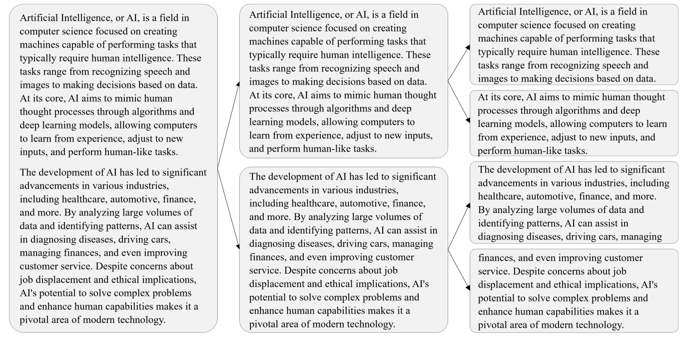

Fig. 10. Chunking example. First split on "\n\n", which is not found. Then on "\n", which splits the text into two, then " ", which further splits the text into chunks of four. This process continues until the chunks are a suitable size.

explainability of an overall system. By incorporating retrieval in the process as a ground truth for the model, there is no longer a question of what knowledge the LLM has since the ground truth is the knowledge. This idea is demonstrated in Figure 9, which compares the results of asking ”what is my birthday?” both with and without RAG. With RAG, we can see exactly what information the LLM is using to answer the question. Conversely, without RAG, the LLM has no additional information and must rely on its own knowledge. Since the LLM does not know my birthday, it is unable to answer. A second benefit of RAG is its simple architecture. Using retrieval to assist the model in user inquiries, the LLM itself can remain fixed, which lowers the overall complexity of the system when compared to alternative approaches like finetuning.

Despite the simplicity of RAG, implementing a well performing RAG system is difficult. This is because RAG necessitates a well performing text retrieval system. If the text retrieval system returns the incorrect ground truth, or even only partial truth, then the output becomes unreliable. Accordingly, considerable focus has been placed on text retrieval in this work.

In the next sections we will discuss three text retrieval methods: semantic search, semantic search with a reranker model, and a hybrid retrieval system using semantic and keyword search. Additionally, we will discuss a question topic classifier which can be used in conjunction with any of the three text retrieval methods to enhance their results.

1) Semantic Search: Semantic search aims to identify text that is semantically similar, rather than relying on exact keyword matches. This methodology has become increasingly relevant with the rise of chatbots, as it effectively handles the wide variation in user queries that may not directly match text within a database. By focusing on the underlying meaning of text, semantic search transcends literal word-for-word searches. It achieves this through embedding models, which generate abstract representations of text, enabling a more nuanced and effective search process.

By generating text representations for an entire database, we can measure similarity using methods like cosine similarity. This is based on the principle that texts with similar content are more likely to hold relevant information to one another. In the RAG process, we assess the similarity between a query and each database entry, identifying the most closely related texts. These are then forwarded to the LLM for answering questions, leveraging the most relevant information available.

Implementing semantic search successfully hinges on the integrity of the underlying dataset. Enhancing the dataset's robustness involves processes such as chunking to normalize text length. This step is vital because the way text is represented can be influenced by its length, potentially causing skewed results for queries that significantly differ in length from the database texts. By standardizing text length, we mitigate the risk of these anomalies, ensuring more consistent and reliable search outcomes.

Chunking, while beneficial for standardizing text length, introduces challenges in information retrieval. This technique truncates longer texts, which can result in the loss of critical information, especially in complex texts. This work uses a chunk size of 350 characters, including a 25-character overlap between chunks to preserve context continuity. This configuration, typically covering two to three sentences, strikes a balance between maintaining sufficient information and ensuring context is not entirely lost, offering an effective compromise for semantic search tasks.

The dataset leveraged for retrieval purposes is the refined dataset outlined in Section III-A2. For chunking, the approach used involves progressively splitting the text based on structural markers: first by sections (denoted by "\n\n"), then paragraphs ("\n"), subsequently by spaces (" "), and finally, if necessary, by characters. This iterative process illustrated in Figure 10 continues until the text segments are reduced to the predetermined chunk size of 350 characters. This chunking strategy is designed to keep semantically similar text together for as long as feasible, reflecting the logical structure where sections contain closely related information, followed by paragraphs, and then sentences. To further optimize this method, there is an additional minimum chunk size of ten words enforced, ensuring even the smallest chunks maintain a basic level of comprehensiveness and context.

As illustrated in Figure 11, the chunking process effectively standardizes the length of text in the dataset. The distribution before chunking is skewed significantly to the right since some texts contain thousands of words. After chunking the distribution is approximately normal.

To prepare the chunked dataset for semantic search, it is transformed into embeddings using the bge-large-en-v1.5

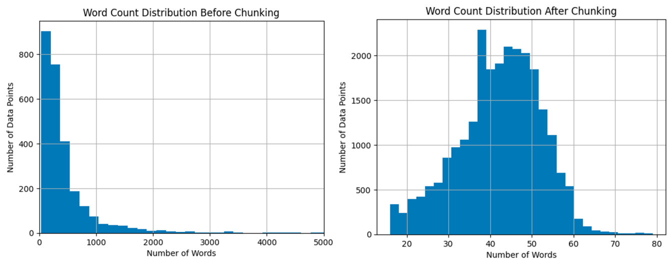

Fig. 11. Word Count Distribution Before and After Chunking

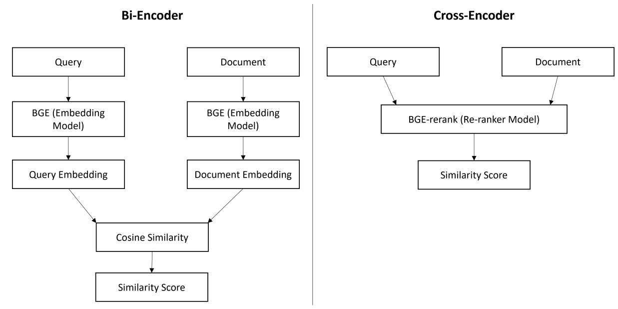

Fig. 12. Cross-encoder vs bi-encoder diagram

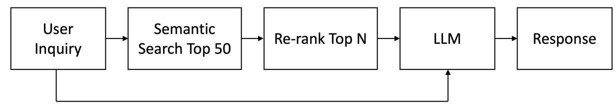

Fig. 13. Semantic/Rerank flowchart

(BGE) [53] model. Chosen for this task, BGE is a pre-trained model publicly available and specifically designed for semantic search applications. The selection of BGE was driven by its outstanding performance in benchmarks targeting general-purpose text retrieval tasks [29]. This model's proficiency in understanding and representing the semantic nuances of text makes it an ideal choice for enhancing the retrieval capabilities of our system.

With the BGE model, each text chunk undergoes tokenization and is then processed by the model to generate an embedding—a numerical representation capturing the chunk's semantic essence. These embeddings for each chunk are stored locally, ensuring that later they can be loaded in and used for retrieval without performing this computation repeatedly.

To perform semantic search, the user's query is first converted into an embedding using the BGE model, mirroring the process used for the dataset chunks. This query embedding is then compared to each chunk's embedding using cosine similarity, generating a score that reflects the similarity between the query and the chunk. Based on these similarity scores, the chunks with the highest relevance to the query are selected and passed to the LLM.

Building on this foundation, the next section will explore how we can further improve the effectiveness of semantic search through the integration of a reranker model.

2) Semantic/Rerank Retrieval: Reranking is a widely adopted strategy across the field of information retrieval. Its principle is straightforward: initially apply a fast but less precise method to narrow down the search results, followed by a reranking phase that improves accuracy at the expense of efficiency. We focus on the application of cross-encoder models, a subset of reranker models, to improve text retrieval in this work. Recent advancements in reranking models [31] through models such as BERT [12] have achieved state-of-the-art performance in zero-shot retrieval tasks [45]. In the following section, we will discuss the technical details of reranker models and describe our integration of a reranker model with BrockportGPT.

The distinction between cross-encoder and bi-encoder models in terms of handling query-document relationships is clearly illustrated in Figure 12. It is crucial to note that the foundation of semantic search, the embedding model, is built upon the bi-encoder architecture. Bi-encoders process queries and documents independently, generating embeddings that can be efficiently compared for similarity. Alternatively, cross-encoders evaluate the similarity scores by considering the query and document as interdependent entities. This methodology allows cross-encoders to capture the dynamics between the query and document, facilitating a deeper understanding of their interaction. Consequently, using a cross-encoder leads to more refined and superior performance in practice [39].

The primary inefficiency of the cross-encoder model arises from the inherent variability of user queries in chatbot interactions. The unpredictability of these queries necessitates that similarity computations be executed in real-time. For cross-encoders, this means any incoming inquiry is compared against each of the 24,901 chunks in the dataset. Even on high-performance GPUs this computation spans several minutes. In contrast, the bi-encoder model offers a more efficient alternative by allowing for the pre-computation and storage of the database embeddings, making the retrieval process almost instantaneous.

By combining strategies, we use semantic search to return only the top fifty results for the re-ranker model to compare. This process is demonstrated in Figure 13, which leverages the speed of semantic search with the superior performance of the re-ranker. The re-ranker model chosen for this project is bge-reranker-large (BGE-rerank) [53]. Like BGE, the model used for semantic search, BGE-rerank has impressive performance [46]. Both BGE and BGE-rerank come from the same family of models, however each are trained separately on different data.

Ultimately, using the reranker model should improve performance of text retrieval overall. Additionally, using semantic search at the base of the approach will dramatically increase the viability of using the reranker model due to the alleviated computational drawbacks. Next, we will shift the discussion to hybrid search, which uses a combination of semantic search and keyword search to query results.

3) Hybrid Retrieval: Text retrieval has long been a focus of research, yet the recent surge in semantic search and related concepts has overshadowed traditional retrieval methods. This motivates our adoption of Typesense, a robust open-source search library [49]. Typesense supports several ways to search, but crucially, it supports hybrid search. Hybrid search combines the best of both worlds by integrating keyword and semantic approaches to computing similarity. This work will focus on how varying degrees of focus on keyword search can improve text retrieval, if at all.

Using Typesense, hybrid search is accomplished using a weighted sum composed of keyword search and semantic search similarity results. This computed similarity score is called a Rank Fusion Score (RFS). The formula is as follows:

$$RFS = \alpha * K + (1 - \alpha) * S, \qquad \alpha \in [0, 1] \tag{1}$$

In Equation 1, α is the weighting factor: a higher α places greater importance on keyword search (K), while a lower α prioritizes the semantic search results. Like prior methods, a higher rank fusion score indicates a higher degree of similarity between queries and documents.

For the semantic search component, the model used is text-embedding-ada-002 from OpenAI. This differs from BGE, which may introduce additional variability beyond hybrid search when comparing results. However, both text-embedding-ada-002 and BGE have strong semantic search performance.

Overall, the use of hybrid search for text retrieval is promising. One additional benefit of this approach is its speed. Keyword search is a quick process, combined with the bi-encoder model which is also quick makes hybrid search an overall very efficient approach for text retrieval. Next, we will discuss question topic classification, which is an approach that seeks to improve all the text retrieval strategies incorporated in this work.

4) Question Topic Classifiers: One of the issues with RAG is if the retrieval system does not return relevant chunks of text, then the response is doomed. This issue is most pronounced with large datasets when querying for specific information, since the proportion of relevant data is much lower than smaller datasets with general inquiries.

This issue motivates the question topic classifiers built in this work. The idea is simple, when an inquiry is received, a model predicts what topic of question it is. Given the prediction of the model, the text retrieval process is only performed over a subset of the dataset relevant to that topic.

Fig. 14. URL categorization example.

Categorizing data and labeling is typically a non-trivial task. Fortunately, we find the SUNY Brockport website is well organized. Since web scraping is used to collect the dataset, the information encoded inside the URL of each webpage can be used to label the data. There are two types of categories for the topic classifiers: main category, and subcategory. This is demonstrated in Figure 14, where the main category is the first subdirectory of the URL, and the second category is a child subdirectory of a main subdirectory.

TABLE II
NUMBER OF SUBCATEGORIES PER TOPIC

Topic	# of Subcategories
academics	59
support	26
live	1
life	12
about	6
alumni	3
scholarships-aid	2
admissions	3
graduate	2
research-foundation	0
library	1
bsg	0

The decision to have a main category and a subcategory is driven by the layout of the website. As demonstrated in Table II, on the SUNY Brockport website there are twelve main categories with the center column showing there are varying numbers of chunks from the database in each category. The number of possible subcategories is a category listed in the furthest right column. These results are interesting since it is clear not all categories will benefit from a subcategory, only those where the subcategory label contains meaningful information.

This leads to the decision of only training a subcategory model for categories containing at least four possible subcategories. This means that subcategory categorization models are only trained for the academics, support, life, and about categories.

Like anything in machine learning, to train a model, data is required. Recall from section III-A3 the QA-dataset is generated in batches of five from a single webpage for all webpages scraped. This implies that for all questions their respective webpage is the sole source of information. Using the webpage topic derived from the URL, the question can inherit the same topic. This underlies the idea used to create the training set for the question topic classifiers. By labeling the question topics, the questions themselves are input to the model and the topic derived from the URL is the output.

This implementation leverages both the GPT-3.5 and GPT-4 generated QA datasets. An unfortunate issue with this dataset is the uneven distribution of question topics. Since questions are sampled at a fixed rate per webpage, the distribution of webpages on the SUNY Brockport mirrors that of the questions. The result of this is illustrated in Figure 15, where ~75% of the data falls within only three categories. This makes modeling difficult, since unbalanced datasets typically will lead to models failing to generalize.

One common solution to issues like this is oversampling. Oversampling aims to forcibly balance the dataset by sampling topics with replacement as needed. In this work a decision was made to sample exactly three times the average number of

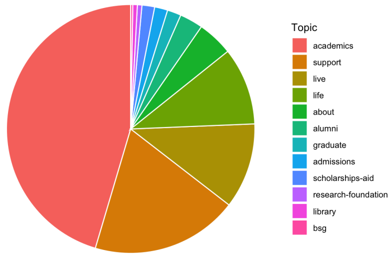

Fig. 15. Distribution of QA topics

questions per topic. In the main classifier this number is 1,935 questions. This means questions from the topic's academics, support, live, and life are all unique since there are greater than 1,935 questions in these categories, and the remaining topics copy questions in their data. For cases such as library and bsg, which have the least questions, the questions can be copied between 15-30x times. Typically oversampling in this style is not recommended, however, we find performance improves with significant oversampling as opposed to letting the dataset remain uneven or lowering the overall number of questions. Future work may consider different approaches to this problem, such as generating more questions, or altering the questions slightly to promote generalization.

TABLE III
TERM-DOCUMENT MATRIX

Sentences	the	sky	is	blue	red	grey
the sky is blue	1	1	1	1	0	0
the sky is grey	1	1	1	0	0	1
the color red is red	1	0	1	0	2	0

Using the dataset, the questions now undergo standardization through lowercasing and are further preprocessed using a bag-of-words (BOW) approach. BOW is a simple preprocessing approach that deals strictly with the vocabulary and frequency of text, omitting any regard for position or overall structure. Take the following example: "the sky is blue", "the sky is red", and "the sky is grey". Among these texts, there is the common vocabulary: "the", "sky", "is", "blue", "red", and "grey". Using this six-word vocabulary, a bag-of-words approach will define a length six vector for any text and indicate how many times each word in the vocabulary appears in the dataset. This idea is illustrated in Table III, where "the color red is red" contains 1 "the", 1 "is" and 2 "red", but no other text from the vocabulary appears, leaving zeros in other columns. While simple, the bag-of-words approach is powerful for basic NLP tasks like classification.

Since each of the topic classifiers are for distinct groups of data, the classifiers use dynamically computed vocabulary sizes. It is difficult to use an equal sized vocabulary size in these cases since too large a vocabulary can promote overfitting, while too small of a vocabulary can promote deficient performance overall. To rectify these issues, the vocabulary

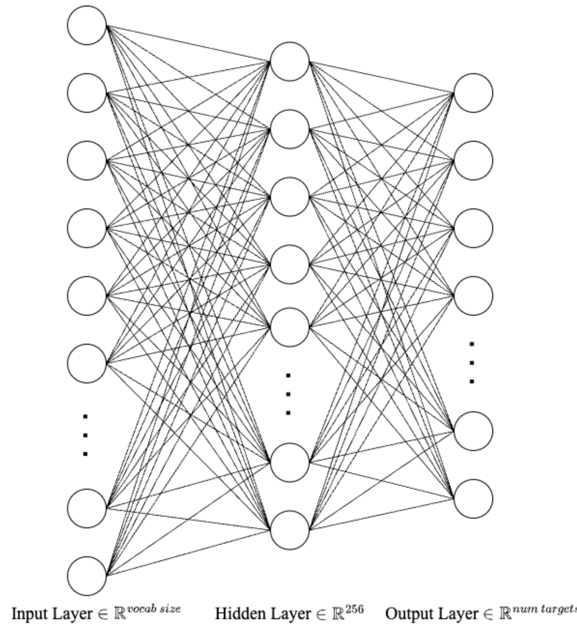

Fig. 16. Question topic classifier architecture

size is chosen by selecting all words that appear more than five times in the data.

Using this information, modeling can begin. All the classifiers use an equivalent fully connected neural network architecture illustrated in Figure 16. The key difference between them is the size of vocabulary and number of targets. The number of targets is how many labels there are to predict, in the case of the main classifier, there are twelve, with the subcategories differing.

TABLE IV
QUESTION TOPIC CLASSIFIER MODEL PARAMETERS

Parameter	Value
Hidden layer dimension	256
Learning rate	1e-4
Dropout rate	0.5
Batch size	32
Early stopping patience	10
Early stopping min delta	0.01

The question classifier model architecture parameters are given in Table IV. All five models use early stopping based on validation loss during training, where the best model is selected by the choosing the lowest validation loss throughout training. In addition to these parameters, the models are trained using the Adam optimizer and cross entropy loss.

Once the models are trained, we apply a handmade wrapper to intelligently facilitate question categorization. One of the inherent issues to a classification system like this is the risk of misclassification. For instance, if a question is about the math department but the classifier predicts and subsequently routes the question to the biology department data then the

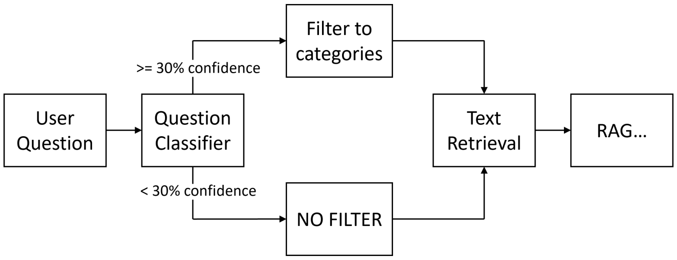

Fig. 17. Overall question classifier integration diagram

response will certainly be incorrect. The adapter developed aims to address this and ensure the reliability of the system.

One neat way to interpret deep learning models built for classification is through the output layer of the model. This layer, once SoftMax has been applied, can be interpreted as a probability of being any given target. For instance, a question about math may have a probability of 95% for math, whereas a question about math and computer science might be 55/40% respectively. Practically, these probabilities can also be interpreted as confidence. For simple questions, the classification model is likely to predict correctly, resulting in higher probability output. Alternatively, for tough questions, the model may be split between topics and predict with lower confidence.

By understanding this, a decision was made to implement a rule-based system to determine how strictly the topic classification should be followed, if it should be followed at all. It works as follows: if the model's highest probability prediction falls below a 30% threshold, the model will withhold any categorization and text retrieval will continue unfiltered. In cases where the model's highest probability prediction exceeds the 30% threshold, the model will select the highest probability prediction along with any other prediction exceeding 15% confidence. This method allows the classifier to function as a soft filter at lower confidence levels, since it will broaden to include multiple relevant categories. As the model begins to predict with higher confidence, the filtering becomes a hard filter, since it will be less likely for any additional prediction to exceed 15

This process happens for the main categorization model and the subcategory models. The only difference is the subcategories will only attempt prediction if the main categorization model predicts their category. For instance, a question like "Tell me about the math department" is predicted to be under the academic topic. Since there is a subcategory model for the academic topic, this process will repeat using the subcategory model and further predict math as the topic of the question.

The overall system with question topic classification is demonstrated in Figure 17. By using the classifier, the text retrieval process is more likely to contain relevant information. This benefit is most pronounced for highly specific information that the retrieval methods discussed prior are unlikely to find on their own. Next, we will transition the discussion away from text retrieval and towards the end-to-end implementation of RAG.

System Prompt: "You are a helpful chatbot for SUNY Brockport who answers questions using the context given. Be enthusiastic, straightforward, brief, and happy to help in your responses. In general, prefer to be give broad answers unless the question is asking for details. Do not answer questions unrelated to SUNY Brockport. If the answer is not clear from the context, say 'I'm sorry, I don't know'."

User Prompt: "Context: {context}\n\nQuestion: {question}"

Fig. 18. RAG System and User prompts

5) RAG: While RAG has so far been focused on text retrieval, it is crucial to ensure a smooth hand-off of information to the LLM for question answering. There are many LLMs capable of handling RAG applications, but the most readily and easily available is GPT-3.5 from OpenAI which exhibits exceptional performance across the board for general purpose language modeling.

An important part of utilizing any general-purpose language model is prompting. The prompt used in this application of RAG is shown in Figure 18. When prompting GPT-3.5 there are two types of prompts: system and user. The system prompt is used to define a role for the model. Since GPT-3.5 is a general LLM, it needs to be given context about what it is meant to do in this context. The user prompt is understood as the user interacting with the system. In the context section, search results from text retrieval are given, and the question section is replaced by the user inquiry.

There are six total retrieval strategies possible described in this work: semantic search, semantic search with a re-ranker model and Typesense, each with or without the question topic classifier. In this stage any of these strategies can be used. This concludes the technical discussion of RAG.

6) Conclusion: Overall, this section has detailed our methodological exploration in implementing RAG. RAG offers numerous benefits compared to the other methods discussed, especially in the see-through nature it provides. The methodology outlined here lays the groundwork for future enhancement in the text retrieval processes.

Future work may consider alternative retrieval strategies and further improve the datasets used throughout. Preliminary observations suggest that RAG stands out as our most effective model in enhancing chatbot performance. A detailed examination of its performance metrics and comparative analysis with other models will be discussed further in the results and discussion section.

D. Finetuned Model

Currently, training state-of-the-art models requires an exorbitant amount of compute and time. As ML models continue to grow as time goes on, especially within NLP, this is a significant problem for researchers and practitioners hoping to achieve state-of-the-art performance. This motivates finetuning, which enables further training of pre-existing model on a specific task for a fraction of the cost. In this work the goal of finetuning is to create a model suitable for question answering about SUNY Brockport directly, without any additional retrieval system.

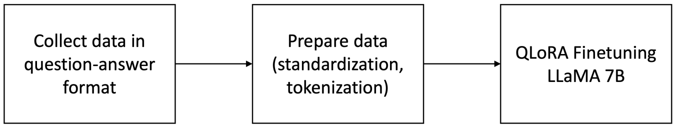

Fig. 19. Typical training pipeline diagram – high level overview

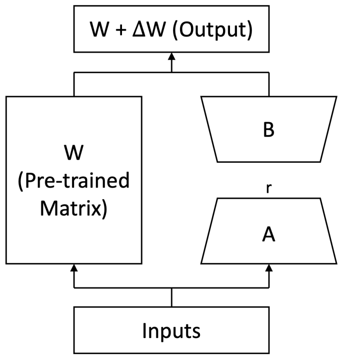

Fig. 20. LoRA explanation diagram

Typically, finetuning a model involves curating a dataset to train on, choosing appropriate hyperparameters, and initiating a training loop. This process is illustrated in Figure 19, which highlights the various parts of finetuning. Considering this, it is crucial to pick a suitable pretrained model to ensure the effectiveness of finetuning. Within the realm of text-generation, and by extension, question-answering, the LLaMA-2 [47] models have emerged as the go-to choice due to its open-source license and vastly superior performance relative to competitors. Given this context, the LLaMA-2 model was selected for fine-tuning in this work.

There are several variants of LLaMA-2 models, each varying in their training and size. Regarding their training, there are two versions of LLaMA-2 model: standard and chat. The chat models differ from the standard LLaMA model since they are further trained with supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align with human preferences for helpfulness and safety. For most applications, the chat version is preferrable, with the standard version primarily existing to promote alignment research within the field. For this work, the selection was made to use the LLaMA-2 chat version.

Regarding the size, or number of parameters in LLaMA-2 models there are 7B, 13B, and 70B sizes. As their size increases the performance of the model increases, and crucially, so does the amount of memory required to run or train the model. Even for smaller models like the 7B and 13B variants, full finetuning is far from possible on our RTX 3090 GPU that has only 24GB of VRAM. At minimum, full finetuning a 7B parameter model will require 160 GB of VRAM available. Recent research has aimed to lower this compute barrier with ideas such as LoRA [22], which can significantly reduce required memory.

LoRA accomplishes this by reducing the number of trainable parameters (weights) in the model. Instead of directly training the full model, LoRA only does gradient updates on two small low rank matrices. This idea is demonstrated in Figure 20, after which these low rank matrices, called adaptation matrices, are combined, and added to the original weight matrix to update the model. On a 175B parameter LLM, GPT-3, LoRA can reduce the number of trainable parameters by 10,000x and memory requirement by three times. Importantly, the original pre-trained model is fixed, or frozen, throughout training. At the end of training, these new matrices can be merged with the existing model providing no computational overhead at inference time.

Still, finetuning is costly. Using just LoRA to reduce memory usage is sufficient to enable finetuning on a RTX 3090, but leaves little excess memory to increase batch size, leading to long training times. This motivates a derivative of LoRA, QLoRA [11], that further reduces memory requirements by using a quantized pre-trained model.

Quantization is a strategy growing in popularity to reduce the memory footprint of LLMs. It does this by reducing the number of bits each parameter, or weight, consumes. Traditionally, weights of a neutral network are stored in fp-32 precision, which uses 32 bits of memory per individual weight to maximize the precision of the model. Compressing this information into smaller bit sizes such as fp-16 or int-8 is a direct performance efficiency tradeoff. Recent work suggests that quantized models storing weights in as small as int-4 retain most of their performance at a substantially smaller memory size [15] [14]. QLoRA adopts this approach and proposes a similar strategy to LoRA, but with a 4-bit quantized pre-trained model.

TABLE V
MEMORY REQUIREMENTS FOR VARIOUS FINETUNING APPROACHES [55]

Method	Bits	7B	13B	30B	65B	8x7B
Full	16	160GB	320GB	600GB	1200GB	1000GB
Freeze	16	20GB	40GB	120GB	240GB	200GB
LoRA	16	16GB	32GB	80GB	160GB	120GB
QLoRA	8	10GB	16GB	40GB	80GB	80GB
QLoRA	4	6GB	12GB	24GB	48GB	32GB

As seen in Table V, using QLoRA for finetuning is vastly more accessible. Given our compute restriction of 24GB VRAM, using LoRA it is only possible to finetune a 7B parameter model. Alternatively, using QLoRA it is possible to finetune either a 7B, 13B, or 30B parameter model. Considering this, QLoRA is the finetuning strategy selected in this work.

To finetune LLaMA-2 7B Chat we use the GPT-4 generated QA-pair dataset discussed previously. The training parameters can be found in Table VI. When training with either LoRA or QLoRA, the rank plays a crucial role in determining the size of the adaptation matrices. As the rank approaches the overall dimension of the model, LoRA training converges with that of a full finetune. Determining the optimal rank value is not straightforward. The LoRA paper suggests increasing rank does not correspond to the direct increase in performance one

TABLE VI
FINETUNED MODEL PARAMETERS

Parameter	Value
LoRA Rank (r)	8
LoRA Alpha (α)	16
LoRA Dropout	0.05
Learning Rate	1e-4
Model Precision	4-bit
Max Length	512
Optimizer	Paged AdamW 32Bit
Epochs	3
Batch Size	16

might expect. Instead, smaller values such as 4, 8, and 16 can often be optimal for training [22]. Importantly, the optimal rank is heavily dependent on the model itself and the task being trained, making it difficult to choose. Due to its prevalent use, the LoRA rank is set to 8 for finetuning LLaMA in this work, however, future work may consider further optimizing this parameter.

The alpha α parameter functions similarly to learning rate since it is a scaling factor for the adaptation matrix in terms of rank (r). During LoRA training, at each gradient update the adaptation matrix is scaled by $\frac{\alpha}{r}$. Only after is the output of this step merged with the original frozen model. Higher values of alpha relative to rank will scale the adaptation matrix more, which effectively makes the model learn quicker. The inverse is also true, hence the similarity to learning rate. Popular values of alpha include $\alpha = r$ or $\alpha = 2r$, with our implementation using the ladder option with alpha set to 16. For more information regarding either rank or alpha selection readers may refer to the LoRA paper [22].

Once the finetuned model is trained, it is quantized to improve efficiency during runtime, improving the user experience. We use GPTQ [14] and GGUF [15] for quantization, which are optimized for GPU and CPU inference, respectively. Both GPTQ and GGUF support k-bit quantization. Like QLoRA suggests, 4-bit offers a reasonable tradeoff between efficiency and performance. Considering this, the model is converted to 4-bit variants of both GPTQ and GGUF models, and all three models, including non-quantized, are made publicly available via Hugging Face.

We do not report any issues during training with this dataset. Improvement on the finetuned model may consider leveraging new training techniques, such as LoftQ [24] or LoRA+ [19] to replace QLoRA. Further, additional training techniques such as RLHF [33] or DPO [37] can be used for additional preference tuning. Simpler performance enhancements could include training a larger LLaMA-2 model or implementing RAG to the finetuned model.

IV. Results and Discussion

In this section we will discuss and evaluate the results of each chatbot. The primary focus will be on understanding the strengths and weaknesses of each method. This section will conclude with a comprehensive comparison of the methods using GPT-4 as an evaluator. Throughout, there will be further discussion about the difficulties and implications of each method it might face in a production application.

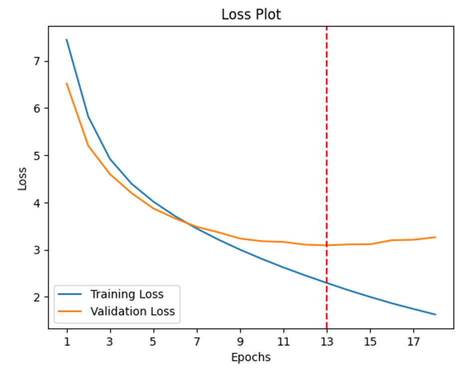

Fig. 21. Model training/validation loss plot. The minimum validation loss is indicated by the vertical line at the 13th epoch.

A. Scratch Model

The scratch model discussed in this research has been specifically developed to explore how effective a model built from the ground up can be in closed domain question answering. This model starts with no pre-training, unlike large models that are refined using extensive datasets. By evaluating its performance, we aim to compare the results with those achieved by either general or finetuned models. This comparison helps us understand the limitations of starting from scratch versus using sophisticated models that have been trained on enormous amounts of data. Through this analysis, we hope to gain insight into the trade-offs between building models for specific tasks and adapting large, pre-existing models to new tasks.

Using the architecture and parameters discussed in the previous section, the scratch model does not face additional difficulties during training. As demonstrated in Figure 21, the training and validation loss begins to diverge near the eighth epoch and continue to further spread apart as time goes on. This is an indication of overfitting, which can happen in any machine learning model.

Overfitting happens when the model continues to improve performance on the training data but worsens on other data, such as from the real-world. This occurs because the model hyper fixates on details of the training data that are not indicative to general success. To monitor how much overfitting is occurring in the model, it is typically recommended to split the data into at least two sets to monitor its performance across training/validation. By splitting the data into distinct groups, the model can be trained on the training data and evaluated by the validation data. This idea motivates early stopping, which allows training to continue while validation loss decreases.

Our results indicate the minimum validation loss exists after the 13th epoch. This model, saved after 13 epochs, is used as the final scratch model throughout the results and discussion section.

The performance of the scratch model varies starkly depending on what kind of question is asked. We find that the scratch

TABLE VII
SCRATCH MODEL EXAMPLES

#	Question	Answer
1	Is there a math major at SUNY Brockport?	Yes, indeed! For Math majors at SUNY Brockport, you have the opportunity to choose Math majors that you would be [UNK] for the Math major. This is a great way to enhance your academic skills and prepare for a successful career in the field of Mathematics.
2	Is there an engineering major at SUNY Brockport?	Yes, indeed! For the major major is [UNK], [UNK] [UNK], and [UNK] [UNK] is a [UNK] [UNK] [UNK] [UNK] that is [UNK] the [UNK] of the [UNK] and [UNK] [UNK]. Its a [UNK] major that allows students to explore the [UNK] and [UNK] of their careers while still making a [UNK] [UNK] experience.
3	Tell me about the nursing major	Absolutely! [UNK] is the chair of the nursing major. [UNK] major, and she would be the perfect person to talk to about the major and major requirements. Her major is a testament to the quality of our programs and faculty.
4	How can I get involved in undergraduate research?	Thats a great question! To get involved in undergraduate research, you can start by reaching out to your professors or the faculty member for research projects. They can provide a wealth of research opportunities to guide you through the opportunities that can contribute to the field and student research opportunities.
5	How can I get involved in clubs?	Getting involved in clubs is a fantastic way to connect with fellow students! You can get involved by attending events, attending events, or participating in clubs. To get involved, you can contact the clubs club to learn more about clubs, organizations, and events. Dont forget to check out the clubs myBrockport page for upcoming events and clubs to get involved!
6	What options are there for financial aid at Brockport?	SUNY Brockport offers a variety of financial aid options to help make your education affordable. You can explore these options to help you fund your studies. You can explore these options and find the best fit for your studies.
7	How can I report sexual misconduct?	You can report any concerns by contacting the title ix coordinator for student affairs at SUNY Brockport by visiting the SUNY Brockport website. There, youll find a report of the title ix coordinator, which is available to students, faculty, staff, and staff. You can report them through the report of the title ix coordinator or the title ix coordinator for assistance.
8	Can I get credit for completing an internship in the math department?	Absolutely! To get started with an internship, you can earn credit for an internship course. This is a great way to gain practical experience in Math, Math, Math, Math, Math, Math, Math, Math, Math, Math, or Math, and Math. Its a fantastic way to gain practical experience!
9	Is there a gym available to students on campus?	Yes, indeed! We offer a [UNK] [UNK] [UNK] [UNK], which is designed to provide students with the necessary skills and knowledge to excel in a safe environment.
10	How can I apply to SUNY Brockport?	Were thrilled to hear youre interested in joining our vibrant community! Applying to SUNY Brockport is a straightforward process. Simply visit our apply page on the website where you can find all the necessary information and to submit your application. If you need any assistance during the process, our admissions team is more than happy to help!
11	What is the meaning of life?	The life of life at SUNY Brockport is to be [UNK] by the university life office. It includes life and events, organizations, and events that align with the campus life and community. Its a great way to make a difference on campus life more affordable and make the most of your college experience!

model performs unexpectedly well on both questions within the training dataset, and questions slightly altered from the training dataset. Expectantly, on rare or untrained questions, the performance of the scratch model suffers dramatically.

Each of these cases can be found in Table VII, which contains various scratch model generated answers. None of the questions found in Table VII are written word-for-word from the training dataset. However, some questions are like those found in the training data. Breaking down the performance of the scratch model, it is clear the scratch model can struggle to answer coherently to many questions. Generally, when the scratch model struggles to answer coherently, it is because the question is either not trained, or does not appear often in the data. At times, the scratch model exhibits unusual characteristics, such as in question #8, which repeats the term "Math" twelve times consecutively. Similarly, for questions such as #2, and #9, the unknown token [UNK] is also repeated several times. However, for other questions such as #4, #5, #6, #7, and #10 the scratch model responds reasonably from a factual standpoint, albeit with some illogical portions grammatically. Finally, with vastly different questions such as #11, the scratch model misses the premise entirely.

This style of performance has both strengths and weaknesses. From a strength's perspective, the scratch model has impressive performance on questions well defined by the dataset. Some examples of this include application or financial aid questions, which appear numerous times in the dataset with varying styles of writing. For these questions, the scratch model can respond meaningfully even if a question is phrased differently than the training data. Weaknesses on the other hand include the scratch model struggling to answer questions that appear only a few times in the dataset with uniform phrasing.

Furthermore, in cases where a question is inquiring about things Brockport does or does not offer, the scratch model typically fails to deny. This behavior can be explained similarly to the Reversal Curse [5] which explains how an LLM trained on "A to B" fail to learn "B to A". While this trait also generalizes to the scratch model, our focus is on the idea that a language model trained to learn "A" fails to learn "not A." This particularly impacts questions like #2 from Table VII, which is inquiring about an engineering major. SUNY Brockport, as of March 2024, does not have an engineering major. As such, there is no reference to engineering on the website, and therefore there is also no reference to engineering in the QA-pair dataset the scratch model is trained on. So, when the scratch model is given a question about engineering, it does not understand to say no, SUNY Brockport does not offer an engineering major. However, the scratch model does understand that question #2 looks semantically similar to question #1, which is inquiring about a math major that SUNY Brockport does offer. This is crucial, since there are many questions like #1 which indicate "yes, SUNY Brockport has XYZ". So, when confronted with these abnormal questions such as about engineering, the scratch model will respond yes as well. Typically, in questions like this the scratch model response also derails and loses coherence, however, understanding this concept has critical implications.

Practically, the scratch model does not understand what SUNY Brockport has to offer. In an application, this will lead to misinformation for users of the scratch model. More importantly, from a liability and ethical perspective, there exist other issues with the scratch model due to this flaw. Most significantly, is that because the scratch model has not been trained to deny, it typically will endorse criminal activity, failing classes, or other things disagreeing with SUNY Brockport's mission. While these issues are to be expected given the training of the scratch model, they also illuminate key issues with chatbot systems, requiring additional moderation or training to take place. Future work may consider exploring these avenues to better align the scratch model with SUNY Brockport.

Despite these issues, the scratch model exceeds our expectations. Recently, due to the performance of LLMs like GPT-4 [32] and LLaMA [47], large models trained on vast mounds of data are the norm. Because of this, the scratch model is expected to be weak. Unlike an LLM like LLaMA which was trained using 2048 A100 GPUs (approximately \$30M as of March 2024 at an estimated 15k per GPU) for approximately 21 days, the scratch model is trained using a single RTX 3090 GPU (\$700 as of March 2024) for 16 minutes. Considering this, we anticipated the scratch model would struggle responding to even the most straightforward queries. Remarkably, despite these initial expectations, the scratch model displays remarkable performance considering its fundamental difficulties.

Even considering these intrinsic issues of scratch model, there is further room for improvement. Adjusting the vocabulary size to better enable the scratch model to respond, and increasing its size and duration of training will likely further improve performance of the scratch model. However, outside of hardware bottlenecks, further improvement could be achieved by crafting a more focused dataset that contains more relevant questions and answers to the target audience.

Overall, the scratch model challenges the conventional wisdom that high-performance generative models are unattainable without massive datasets and computing power. Typically, the creation of a capable model, even within a narrowly defined domain, is a daunting task that requires significant resources. However, our scratch model serves as a fundamental benchmark, offering a glimpse into the baseline capabilities of a chatbot-driven language model. Importantly, while the scratch model has exceeded our initial expectations, it is still not suited for an end-to-end system due to its frequent inaccuracies and tendency to derail on specific questions.

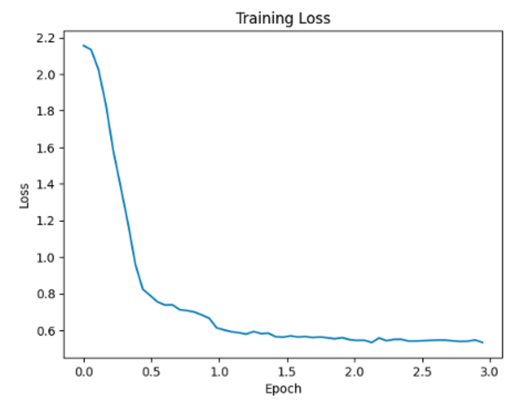

Fig. 22. Training loss of Finetuned model

B. Finetuned Model

The finetuned model is a more popular approach recently. By leveraging the performance of pretrained LLMs, the finetuned model does not fall prey to many of the same issues as the scratch model. In this section, we will discuss the performance of the finetuned model, including some of its strengths and weaknesses. Additionally, we will discuss the issue of hallucination in LLMs and how it is portrayed by the finetuned model.

Using the architecture and parameters discussed in section III-D, the finetuned model finetuned model does not face additional difficulties throughout its training. This is illustrated in Figure 22, which displays a consistently decreasing training loss. Outside of the training, the finetuned model exhibits impressive performance across a variety of inquiries.

The finetuned model leverages the extensive knowledge embedded in pretrained LLMs, allowing it to achieve superior performance. Table VIII displays this capability, with the finetuned model appearing to provide accurate, coherent, and contextually appropriate responses to a diverse set of questions. Unlike the scratch model, which struggled with rare or untrained queries, the finetuned model demonstrates a remarkable adeptness at managing such challenges. This proficiency can be attributed to its exposure to a vast amount of information during the pretraining phase, enabling it to draw on a wider range of data when generating responses. This is particularly important for handling questions like #11, which do not appear at all throughout training. Unlike the scratch model, which struggles immensely, the finetuned model can create a well-crafted response. Importantly, while well-crafted, whether BrockportGPT should have the capability to answer such questions is a key point of discussion. While the finetuned model exhibits excellent performance, the details leave much to be desired.

TABLE VIII
FINETUNED MODEL EXAMPLES

#	Question	Answer
1	Is there a math major at SUNY Brockport?	Yes, indeed! SUNY Brockport offers a Bachelor of Science degree in Mathematics with a variety of specializations. You can choose from tracks like Applied Mathematics, Computational Mathematics, and Secondary Education, among others. It's a fantastic way to develop your skills and prepare for a career in mathematics.
2	Is there an engineering major at SUNY Brockport?	Yes, indeed! SUNY Brockport offers a Bachelor of Science in Engineering (BSE) degree with concentrations in Chemical, Civil, Electrical, Mechanical, and Computer Engineering. It's a fantastic program for those interested in hands-on learning and problem-solving.
3	Tell me about the nursing major	The nursing major at SUNY Brockport is designed to prepare students for a rewarding career in healthcare. With a strong focus on clinical practice, our program provides hands-on experience through partnerships with local hospitals and healthcare facilities. You'll graduate ready to take on leadership roles in nursing and contribute to improving patient care.
4	How can I get involved in undergraduate research?	Getting involved in undergraduate research at SUNY Brockport is an excellent way to enhance your academic experience! You can start by talking to your professors about potential research opportunities. They may have projects available or be able to connect you with other faculty members who are conducting research in areas that interest you. Additionally, the Office of Undergraduate Research and Creative Activities (OURCA) offers resources and support to help you find the right fit.
5	How can I get involved in clubs?	Getting involved in clubs at SUNY Brockport is easy! You can check out our Clubs & Organizations page on the website to see all the options available. Many clubs have open meetings where you can come and learn more about them. Don't be afraid to reach out to club members or advisors for more information. They're always eager to welcome new members!
6	What options are there for financial aid at Brockport?	At Brockport, we understand the importance of financial support for students. We offer various forms of financial aid including scholarships, grants, loans, and work-study programs. Our Financial Aid Office is dedicated to helping you find the best options for your needs. They can assist with completing the Free Application for Federal Student Aid (FAFSA) and provide guidance on how to apply for different types of aid. Don't hesitate to reach out if you need assistance!
7	How can I report sexual misconduct?	If you have experienced sexual misconduct, we encourage you to report it immediately. You can reach out to the Title IX Coordinator at (585) 395-2416 or email them at titleix@brockport.edu. They will provide you with support and guide you through the reporting process. Remember, you are not alone, and there are resources available to help you.
8	Can I get credit for completing an internship in the math department?	Absolutely! You can earn up to six credits by completing an internship with a math-related organization. This is a fantastic opportunity to gain real-world experience and apply what you've learned in the classroom to a professional setting.
9	Is there a gym available to students on campus?	Absolutely! Students have access to the Fitness Center located in the Student Union Building. It's equipped with state-of-the-art equipment and offers a variety of exercise classes. Plus, it's free for all students!
10	How can I apply to SUNY Brockport?	Applying to SUNY Brockport is easy! You can start by visiting our website at brockport.edu/apply. From there, you'll find all the necessary steps to submit your application. We look forward to reviewing your application and welcoming you to our vibrant community!
11	What is the meaning of life?	The meaning of life is a question that has puzzled philosophers and scholars for centuries. At SUNY Brockport, we encourage students to explore their own beliefs and values through courses like PHIL 105: Introduction to Philosophy. This course helps students develop critical thinking skills and understand various perspectives on what gives life meaning. It's an excellent opportunity to reflect on your personal beliefs and how they align with different philosophical views!

One notable challenge of current LLMs, including the finetuned model, is the phenomenon of model hallucination. Model hallucination is a term used to describe model responses that are coherent and grammatically correct yet are factually incorrect or misleading. This issue is particularly concerning applications that rely on factual accuracy, such as educational tools or information dissemination platforms. For example, a query about the first president of the United States receiving a response that inaccurately names John F. Kennedy instead of George Washington illustrates the potential for misinformation. In the context of our finetuned model, hallucination takes many forms.

Like the scratch model, for questions about services or programs SUNY Brockport does or does not offer, the finetuned model typically fails to respond correctly. This is illustrated once again in question #2 of Table VIII, which asks whether SUNY Brockport has an engineering major. The finetuned model, like the scratch model, incorrectly claims there is an engineering major offered at SUNY Brockport. Not only does the finetuned model make this claim, but it further claims there are four concentrations within this non-existent major. Blatant examples of model hallucination like this occur across a wide range of questions. While this specific case of services SUNY Brockport does or does not have comes down to the dataset creation, there are other examples of model hallucination as well. For example, questions #1, #4, #7, #9, and #11 all suffer some form of model hallucination. Question #1 correctly identifies there is a Math major at SUNY Brockport, but incorrectly claims there are concentrations in applied math, computational math, or secondary education. Question #4 is good, however towards the end it directs the user to "Office of Undergraduate Research and Creative Activities (OURCA)",

TABLE IX
PERFORMANCE BEFORE/AFTER FINETUNING

Question: "How can I get involved in undergraduate research?"	
Model Type	**Response**
Finetuned Model	Getting involved in undergraduate research at SUNY Brockport is an excellent way to enhance your academic experience! You can start by talking to your professors about potential research opportunities. They may have projects available or be able to connect you with other faculty members who are conducting research in areas that interest you. Additionally, the Office of Undergraduate Research and Creative Activities (OURCA) offers resources and support to help you find the right fit.
LLaMA-2 7B Chat	The undergraduate research department at State University of New York at Brockport (SUNY Brockport) is called the Office of Undergraduate Research and Creative Activities (OURCA). OURCA provides opportunities for undergraduate students to engage in research, creative projects, and other forms of scholarly activity under the guidance of a faculty mentor... etc.

which does not exist. Question #7 is strong, except for the hallucinated communication information, including an incorrect phone number and email address. Interestingly, both the phone number and email address appear reasonable, with a (585) 395-XXXX style phone number that SUNY Brockport uses and brockport.edu email address. Nonetheless, both are made up, with no reference to either anywhere on the website. Question #9 incorrectly states there is a gym available in the student union building, when the gym is in fact located in the SERC. Question #11, as previously discussed, is a question untrained from the dataset. The finetuned model response is well written, except for a minor detail suggesting the user look at the class "PHIL 105: Introduction to Philosophy". At SUNY Brockport, Introduction to Philosophy has course code PHL 101, not PHIL 105.

Mistakes like this are common with the finetuned model, and it is easy to understand why. The finetuned model is a pre-trained model, meaning it has already undergone a significant amount of training. Finetuning, especially with parameter efficient methods on limited data and compute is unlikely to vastly change the underlying information. This is illustrated in Table IX, which pins question #4 against both the finetuned model, and its underlying pre-trained model, LLaMA-2 7B Chat. Crucial here is that both the finetuned model and the pre-trained model reference the "Office of Undergraduate Research and Creative Activities (OURCA)." This implication affects all areas of the finetuned model and is a key reason hallucination occurs. Unfortunately, this specific point is an intrinsic issue of current LLMs that has little to do with our training methodology.

While imperfect, the finetuned model maintains a high overall utility. Expectedly, the performance of the finetuned model vastly exceeds that of the scratch model and is overall quite capable. As discussed, the finetuned model also faces numerous difficulties, both fundamental in nature, and because of our training methodology. Future work may consider improving the methodology by focusing on new ways to craft the dataset. Additionally, scaling the finetuned model to larger sizes, or more performant models may also result in improvement. Next, we will discuss Retrieval Augmented Generation (RAG), which is a cornerstone topic to this project.

C. *Retrieval Augmented Generation*

Recall there are three information retrieval (IR) strategies discussed in the methodology: semantic search, semantic search with a re-ranker model, and hybrid semantic/keyword search. Combined with their integration into the question topic classifier, there is a total combination of six distinct strategies for IR. In this section we will discuss both the efficiency and performance of each of these strategies. Finally, this section will conclude with a comprehensive comparison of each retrieval strategy and determine what is the most performant RAG configuration.

Efficiency is a key component of any IR system that needs to be accessed in real time. Of the methods used in this work, semantic search is the fastest averaging approximately 65ms to return a response. Next, semantic search with the re-ranker averages approximately 228ms. Finally, the hybrid retrieval strategy has a wide range of search times, from 25ms to 500ms, which is caused by our implementation of hybrid retrieval leveraging OpenAI's text-embedding-ada-002 model via an API instead of being completely local. Interestingly, the question classifier reduces the average time to retrieve results by 40ms for semantic search with and without the re-ranker. We expect this to occur since the question classifier directly reduces the search space for queries, which can vastly reduce the number of similarity scores computed.

Regarding the question classifiers, their performance appears strong. Figure 23 validates this with their loss plots, which seem to indicate the classifiers have some predictive ability. Their accuracy, dictated by whether the classifier strictly predicts accurately on testing data is 71.17%, 79.29%, 73.46%, 83.28%, and 74.22% for the main category, 'about' subcategory, 'academics' subcategory, 'life' subcategory, and 'support' subcategory, respectively. While strong, these accuracy numbers are not convincing enough to be used alone, which motivates the wrapper discussed previously to intelligently route questions based on their probability output from the question classifiers. Importantly, the question topic classifiers created here may not be the most effective architecturally. Further exploration with traditional machine learning methods such as random forest, SVM, or XGBoost may boost performance further.

Regarding performance, our evaluation consists of putting

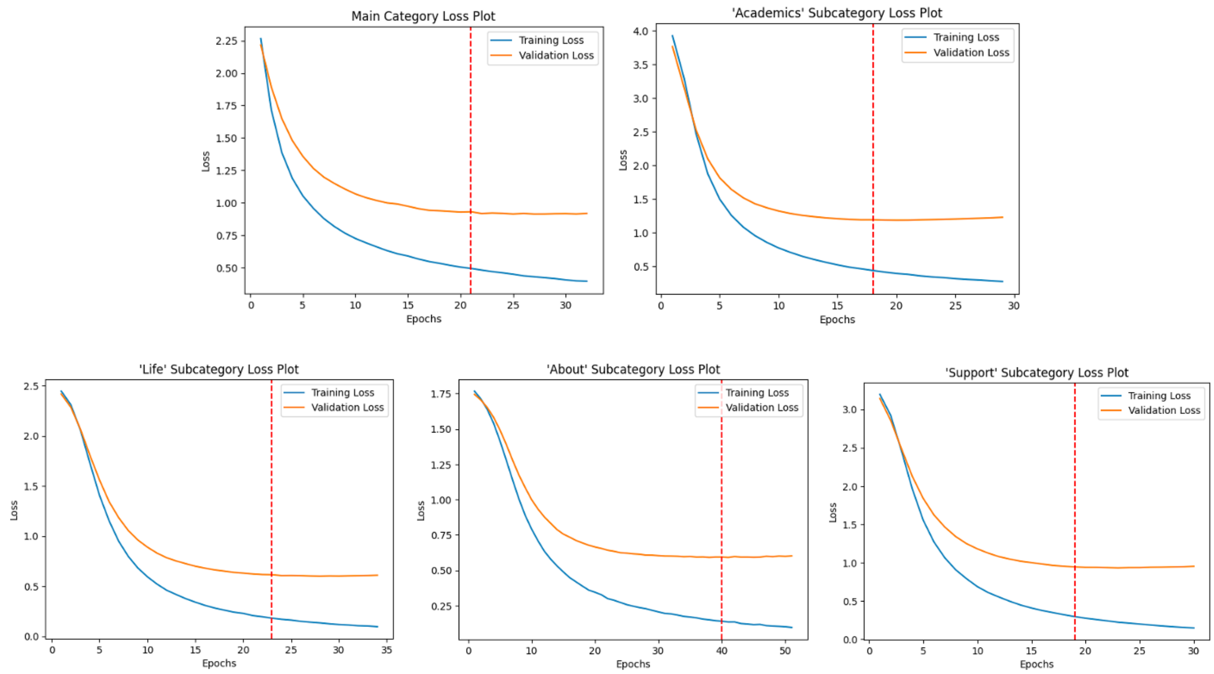

Fig. 23. Showcase of removed irrelevant data. Left is a 2024 academic calendar. Right is a similar academic calendar from 2010. This shows how similar webpages can look whilst containing quite different information.

all six IR strategies in head-to-head matchups where GPT-4 will evaluate who the best player is. These results are used to determine what is the best combination of strategies for IR. Crucially, this is a fundamentally flawed strategy since GPT-4 is an imperfect evaluator. Ideally, model evaluation should occur over many human evaluations of the model's responses. However, considering that difficulty, using GPT-4 has become a common approach recently for model evaluation.

Creating a suitable evaluation dataset is non-trivial. There are two main datasets available to us for evaluation: the training data and the testing data. Traditionally, the testing data exists for the purpose of evaluation. However, in this context we claim evaluating on the testing dataset solely may misrepresent the true ability of both the scratch model and finetuned model. Traditionally in machine learning, the training data should prepare the model with everything required to make a sound prediction. This makes evaluating with test data that is separate from training a very natural extension. However, in our context, the training and testing data do not necessarily have overlap. This is due to the nature of dataset creation. For any question, it may in fact be the only instance of that question inside the dataset. This is a fundamental shortcoming of our dataset, and one that should be considered for improvement in the future. However, considering the nature of the dataset, it does not make sense to try and evaluate the scratch model or finetuned model strictly from the testing data. This is because these models should not be able to answer many questions from the test set. This is a critical issue of both the scratch and finetuned models, but it does not tell a full picture of their ability. These models have still seen a vast number of questions related to SUNY Brockport, covering a sizable portion of questions a user could reasonably ask. If these questions can be answered, then the model can still provide a positive utility to an end user. Considering this, our evaluation dataset is created by using both testing data and modified training data.

System Prompt:
You are a helpful assistant who slightly modifies questions while retaining their meaning. Only make slight modifications such as changing to similar words, or rephrasing the question. Do not change the meaning of the question. Only respond with the question, do not preface it with anything. The question is: {question}

Fig. 24. Question rephrasing prompt

The goal of evaluation using this method is to consider both what happens with the models see unknown questions, while still considering questions within the training data that are phrased differently. Questions are rephrased using GPT-3.5 and the prompt described in Figure 24. Using this strategy our evaluation dataset is created by combining all 1,345 testing questions with 1,000 randomly selected rephrased questions from the training data. This evaluation dataset is used both for evaluating RAG configurations as well as evaluating all the methods described in this research.

The prompt used for evaluation with GPT-4 is given in Figure 25, with the appropriate values substituted in for a, b, c, and d. In this context, the question is the incoming inquiry from the evaluation data, and the ground truth is the answer for that question. Player A and B matchups are sampled randomly, with the only condition being they cannot be equal. Following

TABLE X
RAG MODEL EXAMPLES USING HYBRID SEARCH

#	Question	Answer
1	Is there a math major at SUNY Brockport?	Yes, there is a math major at SUNY Brockport.
2	Is there an engineering major at SUNY Brockport?	I'm sorry, I don't know.
3	Tell me about the nursing major	The Nursing Major at SUNY Brockport provides students with the knowledge, behaviors, and skills needed for nursing practice at the baccalaureate level. The program emphasizes critical assessment, diagnosis, and treatment skills through clinical rotations. Upon acceptance, students transition from "Nursing Intent" to Nursing Major status, receiving guidance in transfer credits, advisement, and registration. Advanced Practice Nurses are in high demand in major health systems and community organizations in the region. Admission to the DNP program considers academic requirements and professional demands for a systems leadership role.
4	How can I get involved in undergraduate research?	You can get involved in undergraduate research at SUNY Brockport by working closely with a faculty mentor on scholarly or creative projects related to your major. Reach out to the Scholarship, Research & Sponsored Programs Office for more information and opportunities available for undergraduate research.
5	How can I get involved in clubs?	To get involved in clubs at SUNY Brockport, you can visit the myBROCKPORT page to explore different clubs and their activities. Attend club meetings, participate in events, and consider taking on leadership roles to develop your skills and contribute to the club's success.
6	What options are there for financial aid at Brockport?	At SUNY Brockport, students have opportunities to receive scholarships, grants, loans, and employment as part of their financial aid package. The Financial Aid Office assists students in obtaining the necessary financial resources to attend the university.
7	How can I report sexual misconduct?	You can file a report online about incidents of sexual misconduct using our Gender-Based and/or Sexual Assault Report. You can choose to remain anonymous if you prefer. Remember, everyone should have access to reporting options and support services, even if they do not wish to formally report an incident.
8	Can I get credit for completing an internship in the math department?	Yes, you can receive credit for completing an internship in the math department at SUNY Brockport.
9	Is there a gym available to students on campus?	Yes, SUNY Brockport has a gym available to students on campus.
10	How can I apply to SUNY Brockport?	To apply to SUNY Brockport, you can submit the SUNY Application or Common Application for admission. Once admitted, complete any additional requirements specific to your program or interest, such as applying for scholarships, the Honors College, or certification programs. Remember to also complete the FAFSA for financial aid consideration.
11	What is the meaning of life?	I'm sorry, I don't know.

System Prompt:

You are a helpful referee who helps pick the best response to a question. The question is about SUNY Brockport, a school in upstate NY. You are given the following:\n\n1) The question given.\n2) The ground truth in the form of an answer to the question.\n3) Player A response to the question.\n4) Player B response to the question.\n\nGiven the question and ground truth, select which player has the best response. Respond with either "A", or "B" only. In some cases, it may be possible that both players are incorrect. In those cases, respond with "None". In choosing the best response prioritize correctness first, then enthusiasm and overall coherence after. Remember to only respond with either "A", "B", or "None". Do not explain your decision.

User Prompt:

Question: {a}\nGround Truth: {b}\nPlayer A: {c}\nPlayer B: {d}

Fig. 25. GPT-4 head-to-head evaluation prompt

the matchup construction, GPT-4 selects the preferred response for each head-to-head.

Inspecting the results, 47.12% of the evaluations responded "None," which indicates either the models performed identically or incorrectly. Gauging which of these cases is difficult, however, from our manual exploration most cases occur from both players responding incorrectly.

Excluding "None" cases, the results where GPT-4 selected one of A or B are given in Table XI. In this table, higher win rate (WR) is better, and N indicates the number of times this configuration played in head-to-head matches. The results indicate compelling evidence for the performance of the hybrid search model with the topic classifier. This method not only achieved the highest WR but also displayed significant statistical confidence, as evidenced by the calculated 95% confidence intervals. Crucially, while this method achieves the highest WR, there is some degree of statistical uncertainty in the results. Specifically, there is not statistically significant evidence that the true WR of hybrid search is greater than those of either hybrid or semantic search without the topic classifier. This is easily seen by the lower bound of hybrid search with the topic classifier, which is 53.95% being less than the upper bounds of either hybrid or semantic search

TABLE XI
HEAD-TO-HEAD RAG TEXT RETRIEVAL RESULTS

Search Types	Classifier	Win Rate	95% CI	N
Semantic	False	53.00%	[48.11, 57.89]	400
Semantic	True	45.01%	[39.95, 50.08]	371
Rerank	False	48.68%	[43.66, 53.71]	380
Rerank	True	37.23%	[32.6, 41.86]	419
Hybrid	False	55.82%	[51.3, 60.34]	464
Hybrid	True	**58.52%**	[53.95, 63.09]	446

without the topic classifier, which are 60.34% and 57.89%, respectively.

TABLE XII
PERCENTAGE OF "NONE" RESPONSES IN HEAD-TO-HEAD RESULTS

Search Types	Classifier	% None	95% CI	N
Semantic	False	49.43%	[45.95, 52.92]	791
Semantic	True	51.12%	[47.56, 54.68]	759
Rerank	False	51.72%	[48.22, 55.21]	787
Rerank	True	47.76%	[44.3, 51.21]	802
Hybrid	False	41.64%	[38.21, 45.06]	795
Hybrid	True	**41.01%**	[37.5, 44.51]	756

To further discuss the results, excluding "None" cases has implications for the WR displayed in Table XI. It is possible certain configurations could be advantaging from this evaluation strategy by either responding excellently or missing entirely. Such a strategy would create an artificially high WR, dodging the point of evaluation. While these biases should disappear as more samples as taken, the distribution of "None" responses across various search types is nonetheless a valuable metric to observe. Considering this, the corresponding percentage of "None" results for each configuration is given in Table XII. In this table, fewer "None" responses implies the model can answer a broader range of question. From these results, it is clear the hybrid retrieval strategy is again the most performant configuration, verifying the strong performance in Table XI. This is further evidenced by the 95% confidence interval, which indicates an upper bound of 44.51% "None" responses for hybrid search using the topic classifier. This upper bound is less than the lower bound CI for either of the semantic search methods, or re-ranking with the topic classifier, all of which indicates statistical significance. The exception to this rule is re-ranking whilst using the question classifier, which has a lower bound marginally beneath the upper bound in question. While important, the overlap is not large enough to be meaningful in our results. The question topic classifier appears to again have mixed performance depending on which search strategy is used. For hybrid search, the question topic classifier does not appear to make a significant difference in performance.

Through these results, the re-ranker typically performs the worst of the search strategies. We do not have an explanation for this behavior. Follow-up work to this may consider exploring why this is the case. Improvements could be found by using a more performant re-ranker model, or even finetuning a re-ranker model to best optimize its performance for BrockportGPT.

Nonetheless, based on this evidence, we conclude hybrid search, which uses semantic and keyword search, is the best configuration of those assessed. Further, despite the statistical uncertainty, we assume hybrid search using the question topic classifier to be the most performant RAG configuration due to its slight edge over its counterpart without the classifier in overall performance indicated by both Tables XI and XII. Following this decision, hybrid search with topic classification is the configuration used as the RAG benchmark for comparing all three methods discussed in this research.

To better understand the capability of RAG, Table X displays the same questions used to discuss the scratch and finetuned models, answered by RAG using the configuration discussed prior. Looking at these responses, it is clear RAG is an effective method. Overall, when compared to the scratch and finetuned models RAG is typically more direct and accurate. This comes at the cost of the conversational feeling the scratch and finetuned models have that may provide a more engaging user experience. This style is easily evidenced by questions #1, #8, and #9, where RAG answers more accurately than the finetuned model but with a less welcoming response. Crucially, this behavior may be adjustable through additional prompt tuning.

Outside of the styling of answers, one major benefit of RAG is its ability to understand what it does and does not know. Since in its prompt, GPT-3.5 is given instructions to only use the provided context as knowledge, for questions which do not retrieve relevant information RAG will refuse to respond. This has several benefits and drawbacks. For questions like #2, which ask if SUNY Brockport offers an engineering major, the scratch and finetuned models both mistakenly claim there is an engineering major offered. This is a fundamental issue of the QA dataset generation. Since engineering does not appear on the SUNY Brockport website, neither does it appear in any of the training questions, resulting in issues the scratch and finetuned model not knowing the extent of offerings at SUNY Brockport. While the scratch and finetuned models suffer from this fault, RAG uses this to its advantage. During text retrieval, for questions about engineering, or other offerings SUNY Brockport does not provide, there will be no relevant matches. For RAG, this means GPT-3.5 will respond "I'm sorry, I don't know" as opposed to hallucinating a program that does not exist. Additionally, this feature allows RAG to deny questions unrelated to SUNY Brockport like in question #11. While this is a strong benefit of RAG, it also means the text retrieval system must have exceptional performance, since if not, RAG will frequently respond "I'm sorry, I don't know" in cases where the scratch or finetuned model can respond accurately.

Overall, RAG does very well. For every question in Table X, RAG gives a reasonable answer, be it a justified refusal, or a well written accurate answer. Questions #4-7 and #10 especially demonstrate the impressive performance of RAG, where each of these responses are free of any inaccuracies, unlike the scratch or finetuned models which each had various hallucinated or otherwise inadequate responses to some of these questions.

In conclusion, RAG is a strong method when discussing BrockportGPT. In addition to its excellent performance as a chatbot, RAG also enables transparency in how responses are made that the scratch and finetuned models do not offer. RAG has several areas for improvement, from text retrieval broadly, to architecturally as well. As LLMs become better at interpreting larger contexts, the need for highly specialized RAG, like this, decreases significantly. This is directly tied to how RAG works. As discussed here, we only send the top

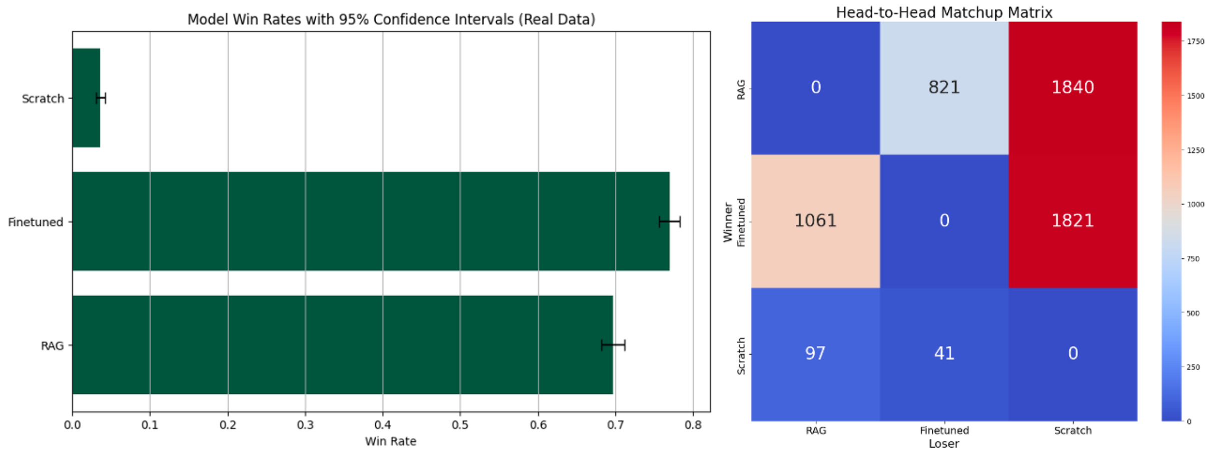

Fig. 26. Method WRs in head-to-head matchups

five chunks of data from the IR step to the LLM. If these five are not relevant, then RAG struggles. However, as we approach a future where LLMs are capable of handling longer contexts with reliability, it may become increasingly common to send vast amounts of data to the LLM for RAG systems. The key issue for this approach is whether the LLM will detect small facts inside massive amounts of text. Recently, this has been called the needle in a haystack problem, with exceptional models like GPT-4 Turbo performing well [17]. Ideally, we would assess this idea here, but large prompts are expensive, relegating this idea to future work. In addition to improving RAG architecturally, future work may be done to improve the IR process such as implementing different or finetuned embedding or re-ranker models. Next, we will discuss the overall method comparison, where we will decide what is the best of the three strategies discussed throughout this work.

D. Comparison

To compare the three methods seen throughout this work, we use the same strategy used to determine the best RAG search method with head-to-head matchups that are evaluated by GPT-4. The key difference now is that all methods are evaluated on the full dataset rather than using sampling techniques that inject additional statistical uncertainty into the results. This is cost prohibitive to do when comparing many factors, like was the case in RAG, but in comparing only three methods it is possible. This is being done to ensure the robustness of our overall results.

From the outset, there is a clear winner when concerning win rate (WR) against one another. This is demonstrated in Figure 26, which has the Finetuned model (76.98% WR) beating both RAG (69.68% WR) and the scratch model (3.63% WR). Crucially, this is the WR when excluding any "None" responses by GPT-4, which indicate either a tie or both methods in the head-to-head failing to answer correctly. Interestingly, there does not appear to be any meaningful correlation between the number of "None" responses and any of the three methods. In this evaluation, 18.59% of head-to-heads resulted in "None." Breaking it down by method, in head-to-heads including the finetuned model this number is 19.52%. For the scratch model, this number is 18.34%, and for RAG is 17.91%. Since these numbers are all close to the mean, we do not find them significant.

Another point of interest in these evaluations is weighing the effect of refusals. As discussed previously, neither the scratch nor finetuned models have a good ability to refuse user inquiries. This ability is primarily evident in RAG, where the model refuses inquiries where the retrieval step did not find relevant information. This is preferred behavior and should not be punished in evaluation. In head-to-head evaluations however, GPT-4 typically prefers an inaccurate answer over a refusal. When removing refusals, which RAG generated 506 of, these WRs move closer but do not shift dramatically. The finetuned model WR decreases from 76.98% to 75.99%, RAG increases from 69.68% to 72.85%, and the scratch model decreases from 3.63% to 2.99%. These shifts while interesting do not change the overall ranking of models since there is still statistically significant evidence that the finetuned model has a greater WR than RAG.

Still, evaluating models is difficult. Unlike other more traditional machine learning domains, closed domain question answering does not have a straightforward way to evaluate performance. Understanding this is easy: for any given question, there are tens to hundreds of ways an answer can be phrased, yet all being completely valid. Good or bad responses then become difficult to parse. Shy from human evaluation which would be a long and tedious process, the GPT-4 evaluated head-to-head is the best option available. It is not without flaws though. Through our own exploration of the evaluated results, GPT-4 typically chooses the better response, but there are some cases where GPT-4 selects a blatantly incorrect response. It is difficult to quantify this without unbiased human evaluation.

Considering this, we take a broad strokes interpretation of these results. In the end, we consider the finetuned model and RAG roughly equivalent in their overall ability, but each excelling in different areas. While the finetuned model excels in broad questions, it struggles in the details. Additionally, stylistically, the finetuned model is the best of all methods since it consistently responds in an enthusiastic and upbeat manner. On the other hand, RAG does better in the details, but frequently gets confused or misses the point of a question when the retrieval mechanism fails to find relevant information. Considering this, future work may consider a combination of RAG and finetuning to both ensure stylistically effective and correct responses.

V. Conclusion

This thesis has demonstrated the effective application of closed-domain question-answering techniques within an institutional setting, specifically tailored for SUNY Brockport. The developed chatbot, BrockportGPT, integrates advanced NLP strategies, including training a model from scratch, fine-tuning LLaMA, and implementing retrieval-augmented generation (RAG) to enhance the specificity and relevance of responses to user queries. Through meticulous data preparation, innovative methodology, and rigorous testing, BrockportGPT has significantly improved the accuracy and efficiency of institutional question-answering systems.

During the development and implementation of BrockportGPT, several challenges were encountered. One of the primary difficulties was the collection and preparation of a high-quality dataset specific to SUNY Brockport. This process involved extensive web scraping, cleaning, and using GPT-4 to synthesize question-answer pairs, which was both time-consuming and expensive. Another challenge was the limited compute available throughout this project, which constrained our ability to finetune larger LLMs or train a more extensive scratch model. Additionally, the lack of a straightforward evaluation metric for closed-domain question answering presented a challenge, as traditional metrics are unable to accurately compare the methods, leading to uncertainty in our results.

The potential for further refinement and application of these techniques suggests a promising direction for future research. Exploring the types of inquiries users of BrockportGPT are likely to ask can enable the creation of a more targeted dataset, resulting in a more effective chatbot. Combining RAG and finetuning could ensure stylistically effective and accurate responses. Additionally, exploring alternative retrieval strategies and improving the datasets used, such as implementing different or finetuned embedding or reranker models, could further enhance performance. Another area of future research is the integration of more advanced refusal mechanisms to better handle irrelevant or unanswerable queries. Furthermore, investigating the scalability of the chatbot and its application to other institutional settings can provide broader insights and potential benefits.

Overall, this work sets a robust foundation for the continued evolution of AI-driven educational support tools, promising to augment the accessibility and quality of information across academic environments. By adapting innovative AI technologies to specific real-world contexts, this research provides a model that other institutions can replicate and adapt, contributing to the broader field of educational AI applications.

References

[1] Yuvanesh Anand et al. "GPT4All: An Ecosystem of Open Source Compressed Language Models". In: *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. Ed. by Liling Tan et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 59–64. DOI: 10.18653/v1/2023.nlposs-1.7. URL: https://aclanthology.org/2023.nlposs-1.7.

[2] Gemini Team Google Rohan Anil et al. "Gemini: A Family of Highly Capable Multimodal Models". In: *ArXiv* abs/2312.11805 (2023). URL: https://api.semanticscholar.org/CorpusID:266361876.

[3] Anthropic Team. *Claude*. 2023. URL: https://www.anthropic.com/news/introducing-claude.

[4] Adrien Barbaresi. "Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction". In: *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2021, pp. 122–131. URL: https://aclanthology.org/2021.acl-demo.15.

[5] Lukas Berglund et al. "The Reversal Curse: LLMs trained on "A is B" fail to learn "B is A"". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=GPKTIktA0k.

[6] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] Mark Chen et al. "Evaluating Large Language Models Trained on Code". In: (2021). arXiv: 2107.03374 [cs.LG].

[8] Paul F Christiano et al. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

[9] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". English (US). In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[10] Tim Dettmers et al. "GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=dXiGWqBoxaD.

[11] Tim Dettmers et al. "QLoRA: Efficient Finetuning of Quantized LLMs". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh et al. 2023. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

[12] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

[13] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[14] Elias Frantar et al. "OPTQ: Accurate Quantization for Generative Pre-trained Transformers". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=tcbBPnfwxS.

[15] Georgi Gerganov. *llama.cpp*. https://github.com/ggerganov/llama.cpp. 2023.

[16] GitHub Team. *The world's most widely adopted AI developer tool. GitHub.* 2021. URL: https://github.com/features/copilot.

[17] gkamradt. *LLMTest Needle In A Haystack - Pressure Testing LLMs*. https://github.com/gkamradt/LLMTest_NeedleInAHaystack. 2023.

[18] Anirudh Goyal et al. "Professor forcing: a new algorithm for training recurrent networks". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4608–4616. ISBN: 9781510838819.

[19] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. *LoRA+: Efficient Low Rank Adaptation of Large Models*. 2024. arXiv: 2402.12354 [cs.LG].

[20] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". In: *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 6.2 (Apr. 1998), pp. 107–116. ISSN: 0218-4885. DOI: 10.1142/S0218488598000094. URL: https://doi.org/10.1142/S0218488598000094.

[21] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[22] Edward J Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[23] Patrick Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[24] Yixiao Li et al. "LoftQ: LoRA-Fine-Tuning-aware Quantization for Large Language Models". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=LzPWWPAdY4.

[25] Zongxi Li et al. *Label Supervised LLaMA Finetuning*. 2023. arXiv: 2310.01208 [cs.CL].

[26] Vladislav Lialin et al. "ReLoRA: High-Rank Training Through Low-Rank Updates". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=DLJznSp6X3.

[27] Xueguang Ma et al. *Fine-Tuning LLaMA for Multi-Stage Text Retrieval*. 2023. arXiv: 2310.08319 [cs.IR].

[28] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[29] Niklas Muennighoff et al. "MTEB: Massive Text Embedding Benchmark". In: *arXiv preprint arXiv:2210.07316* (2022). DOI: 10.48550/ARXIV.2210.07316. URL: https://arxiv.org/abs/2210.07316.

[30] Arvind Neelakantan et al. "Text and Code Embeddings by Contrastive Pre-Training". In: *CoRR* abs/2201.10005 (2022). arXiv: 2201.10005. URL: https://arxiv.org/abs/2201.10005.

[31] Rodrigo Frassetto Nogueira and Kyunghyun Cho. "Passage Re-ranking with BERT". In: *CoRR* abs/1901.04085 (2019). arXiv: 1901.04085. URL: http://arxiv.org/abs/1901.04085.

[32] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[33] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=TG8KACxEON.

[34] Christos H. Papadimitriou et al. "Latent semantic indexing: a probabilistic analysis". In: *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium

on Principles of Database Systems. PODS '98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 159–168. ISBN: 0897919963. DOI: 10.1145/275487.275505. URL: https://doi.org/10.1145/275487.275505.

[35] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[36] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[37] Rafael Rafailov et al. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=HPuSIXJaa9.

[38] Marc'Aurelio Ranzato et al. "Sequence Level Training with Recurrent Neural Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06732.

[39] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Conference on Empirical Methods in Natural Language Processing*. 2019. URL: https://api.semanticscholar.org/CorpusID:201646309.

[40] Baptiste Rozière et al. *Code Llama: Open Foundation Models for Code*. 2024. arXiv: 2308.12950 [cs.CL].

[41] Ruoxi Sun et al. *SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL (extended)*. 2024. arXiv: 2306.00739 [cs.CL].

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 3104–3112.

[43] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca. 2023.

[44] Gemma Team et al. *Gemma: Open Models Based on Gemini Research and Technology*. 2024. arXiv: 2403.08295 [cs.CL].

[45] Nandan Thakur et al. "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[46] Ravi Theja. *Boosting rag: Picking the best embedding & Reranker models - LlamaIndex, Data Framework for LLM applications*. Nov. 2023. URL: https://www.llamaindex.ai/blog/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83.

[47] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[48] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[49] Typesense Team. *Typesense*. 2024. URL: https://typesense.org.

[50] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[51] Ronald J. Williams and David Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". In: *Neural Computation* 1.2 (1989), pp. 270–280. DOI: 10.1162/neco.1989.1.2.270.

[52] Chaoyi Wu et al. "PMC-LLaMA: toward building open-source language models for medicine". In: *Journal of the American Medical Informatics Association* (Apr. 2024), ocae045. ISSN: 1527-974X. DOI: 10.1093/jamia/ocae045. eprint: https://academic.oup.com/jamia/advance-article-pdf/doi/10.1093/jamia/ocae045/57229449/ocae045.pdf. URL: https://doi.org/10.1093/jamia/ocae045.

[53] Shitao Xiao et al. *C-Pack: Packaged Resources To Advance General Chinese Embedding*. 2024. arXiv: 2309.07597 [cs.CL].

[54] Can Xu et al. "WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=CfXh93NDgH.

[55] Yaowei Zheng et al. "LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models". In: *arXiv preprint arXiv:2403.13372* (2024). URL: http://arxiv.org/abs/2403.13372.