CS 224V Assignment 1

Due: 10/9, 2:00 PM PST

Instruction: Use this Colab notebook in conjunction with this write-up. Make sure to "Save a copy in Drive" before running the notebook. Submit your answers through Gradescope and attach your Google Colab notebook. In red, we label how each question in this writeup corresponds to a Gradescope question

We expect heavy loads on the OVAL machine used in this assignment close to the deadline. **Thus, we highly recommend you start this assignment early and do not wait until the last minute.**

This assignment is designed to be completed **in groups of 2**. Please submit as a group to Gradescope.

Extensions: You are granted an automatic 24-hour extension to either Assignment 1 or Assignment 2, but not both. If you submit Assignment 1 later than the deadline, this 24-hour extension will be automatically applied to Assignment 1. Each individual student has one 24-hour extension, so both partners will require an available extension to take one on Assignment 2.

This assignment is designed to give you hands-on experience with hallucination-free LLM-based conversational agents grounded in a knowledge corpus of free-text documents. You will learn:

- why and how to use retrieval systems to augment LLMs for grounding;
- the concept of a state-of-the-art (SOTA) zero-shot framework for creating such assistants;
- the methodology of evaluating such assistants;
- how to observe weaknesses and strengths of existing systems and propose improvements.

While conversational agents have shown remarkable performance, there is a surprising lack of research evaluating **whether people can get better access to information using a hallucination-free LLM-based chatbot compared to traditional web page browsing**. We will explore this research question throughout this assignment!

In this assignment, we leverage StackExchange as a testbed for such a comparison. StackExchange is a network of community-driven question-and-answer websites covering a wide range of topics. A preliminary evaluation of BingChat on the StackExchange platform indicates that it hallucinates by giving unsubstantiated advice or wrong citations *9 out of 10 times.* Trust in the information received is a fundamental need for users, and so a more trustworthy chatbot is a necessary bedrock we must achieve before evaluating how chatbots compare to the overall experience of traditional web browsing. To facilitate this, we develop a series of chatbots with **GenieChat** framework grounded in 25 different domains in StackExchange.

The GenieChat platform is based on the WikiChat pipeline which achieves a factuality score of 98% on Wikipedia [2]. Preliminary evaluation with the WikiChat framework led to one modification implemented in GenieChat: one of the steps in the pipeline uses the LLM to generate an answer, split the answer into claims, and retrieve data to verify the factuality of each claim. In the original pipeline, any unsubstantiated fact is removed from the answer; in this modified pipeline, it is replaced with a retrieved fact if possible. This adds richness to the overall answer. Figure 1 gives an illustration of the GenieChat framework.

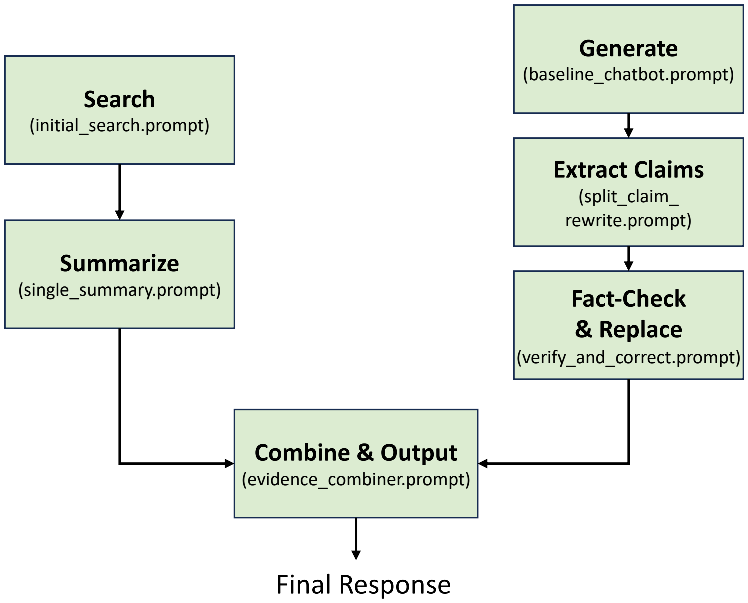


Figure 1: Illustration of GenieChat v1.1 framework.

1 Set up GenieChat for Your Domain of Interest

One notable feature of using StackExchange as the testbed is that it contains question-answering knowledge across a wide range of domains. We have chosen 25 domains that are just the right size – not too big to run on Colab [1] and not too sparse to deliver unmeaningful results. First, you will pick your domain of interest from the provided list. Please also sign up on the Google Sheet with your group names. Each domain can have at most 2 groups. Once a domain already has 2 groups signed up, that domain is no longer available and please choose another domain. *(The choice of your domain will not impact your grade.)*

(a) Choose your domain of interest and write down your selected domain's "Site Name alias" (from column B of the Google sheet). Make sure to also update this in your Colab notebook. (Gradescope Q1)

Creating a grounded chatbot using GenieChat mainly involves three steps:

1. Prepare and pre-process the corpus
2. Set up a retrieval server and index all documents. We use ColBERT v2 as the retrieval model. We have pre-processed all 25 domains with all of their question-answer pairs on StackExchange[2].
3. Implement LLM-based modules to interact with the retrieval server. For this assignment, we have already done this for you by distilling the behavior of a GPT-4 model to fine-tune a LLaMA-2-7B model. The Colab code is accessing this LLaMA-2 model, hosted on an OVAL machine, via network connections. We expect heavy loads and thus longer response times close to the deadline. **Thus, we highly recommend you start this assignment early and do not wait until the last minute.**

Follow the instructions on Colab to set up GenieChat on your domain.

To ensure GenieChat is set up correctly:

(b) Go to the web page of your domain and select some questions to query the retriever. Verify if the retriever can return the documents containing the exact question.

[1] The bottleneck is due to the memory usage of the ColBERT information retriever.
[2] The StackExchange dump we use is up to date as of 2023-09-12.

(c) Chat with the chatbot to see if it can respond successfully. Ensure the conversation and intermediate system outputs are correctly logged. The logs are outputted to My Drive → genie_open_text → data. You will need the log in the later part of the assignment.

(You don't need to answer (b) and (c) on Gradescope.)

2 Comparing the Chatting and Browsing Experiences

Now you have a conversational interface to access community-contributed knowledge in a StackExchange domain. The traditional way of accessing this knowledge is by using search engines and browsing web pages. In this section, you will compare these two experiences.

2.1 Experiment Guideline

1. **Selection of Question:** Come up with **3** questions you are interested in for this domain (note: you should not just select an existing question on StackExchange, but should think of something you want to learn more about in this domain). Uncommon questions are preferred to get more interesting evaluation results.
2. **Order of Chatting v. Browsing**: Start with **chatting** for your *first* and *third* questions. Start with **browsing** for your *second* question. Alternating can help reduce ordering bias effects.
3. **Chatting:** Chat with GenieChat to explore answers to your question. Record the time it takes to get the relevant information.
4. **Browsing:** Utilize a search engine (*e.g.*, StackExchange search box, Google search, *etc.*) to browse the relevant StackExchange domain for the answer to the same question. It should be limited to information that can be found on StackExchange. Record the time it takes to find the pertinent information. Record your search queries.

2.2 Experiment Report

Respond to the following for each of **your 3 distinct questions** (Gradescope Q2.1 - Q4.6)

(a) What's the question you are seeking information for?

(b) Did you find information through chatting? If you did, how much time did it take?

(c) Copy the conversation you had with GenieChat.

(d) Did you find information through browsing? If you did, how much time did it take?

(e) Write down all the browsing search queries you used to find the information.

(f) Which experience did you prefer and why? (1-2 sentences)

*Repeat 3x, once for each of your questions. Remember to alternate between starting with **chatting** vs **browsing**.*

3 Delving Into the System

After gaining hands-on experience with the chatbot, let's delve into the inner workings of the system. This is how people work in cutting-edge NLP research: Testing the system, evaluating its performance, and then determining areas and strategies for improvement.

3.1 Scrutinize the Retriever

Finding relevant information is an essential part of building grounded conversational agents. We use a metric called retrieval precision to keep track of the performance of the ColBERT retriever.

PRECISION is the fraction of retrieved documents that are relevant:

$$\text{PRECISION} = \frac{\#(\text{ relevant items retrieved })}{\#(\text{ retrieved items })} = P(\text{ relevant } \mid \text{ retrieved }) \tag{1}$$

For each of your **3 questions**,

(a) Using the logs, calculate the overall PRECISION for the conversation you had (i.e., on all the turns). Report #(relevant items retrieved) and #(retrieved items) as well. (Gradescope Q5.1, Q5.3, Q5.5)

Whether an item is "relevant" is with respect to the query associated with it. For instance, suppose you see the following log snippet:

```
"initial_search_retrieval_result": {
    "initial_search_query": "how to check for user password in Unix",
    "initial_search_results": [
      {
        "title": "A",
        "paragraph": "B"
      },
      {
        "title": "C",
        "paragraph": "D"
      },
      {
        "title": "E",
        "paragraph": "F"
      }
    ]
  },
```

Here, A, B together count as one retrieved item, and there are in total 3 retrieved items. Whether a retrieved item is relevant is with respect to the associated query. In this example, whether A, B is a relevant item needs to be judged by you in the context of "how to check for user password in Unix". As long as either the title or paragraph is relevant, then it is considered relevant. Think of treating the title and paragraph as a single document, and consider whether that document is relevant. If one part of the document is relevant, then the document is relevant. This is a subjective decision you will make, but credit will be awarded for all reasonable interpretations.

You should do the same to ”verification_retrieval_result”. Example:

```
"verification_retrieval_result": {
    "query_1": {
      "label": "SUPPORTS",
      "fixed_claim": "claim_1",
      "retrieval_results": [
        {
          "title": "A",
          "paragraph": "B",
          "score": 8.8
        },
        {
          "title": "C",
          "paragraph": "D",
          "score": 9.3
        }
      ]
    },
}
```

Then, A, B together count as one retrieved document, and relevance is taken with respect to "query_1".

(b) If precision $\neq 1.0$, select a failure case and analyze it. Is this a failure of the system? (Gradescope Q5.2, Q5.4, Q5.6)

3.2 Evaluate the Generation Faithfulness

Even if we can give LLMs the right information, it's still a big task to make sure the answers they give are factual with respect to the original documents. In fact, evaluating the faithfulness of the final response is a challenging problem on its own. To make evaluation easier, we will use LLM to break the final output into different claims. A claim is defined as verified only if it is backed up by the grounding corpus (i.e., you will manually look up if each claim has supporting information in the selected domain of StackExchange for this assignment). This definition is strict. An unsupported claim might be correct, but it's not favored because it goes against the principle of grounding and is not in the database. Specifically, FAITHFULNESS is the fraction of claims that are verified:

$$\text{FAITHFULNESS} = \frac{\#(\text{ verified claims })}{\#(\text{ claims })} = P(\text{ verified } \mid \text{ all }) \tag{2}$$

(c) For each turn of your three conversations, verify the corresponding claims and record your answers in the following format. Calculate the overall FAITHFULNESS on all turns. Report $\#($ verified claims $)$ and $\#($ claims $)$ as well. (Gradescope Q6.1 - Q6.6)

```
User: "user's turn"
Chatbot: "chatbot's turn"
----------------
- "Claim1"      verified/unverified
- "Claim2"      verified/unverified
...
----------------
User: "user's turn"
Chatbot: "chatbot's turn"
----------------
- "Claim1"      verified/unverified
- "Claim2"      verified/unverified
...
----------------
...
```

We use the same LLM to break down claims to make this part easier. The results are stored under "split_claims_for_eval" in "demo.log". If you think the breakdown is not reasonable (e.g., in certain cases it could be empty), you should break down claims as you see fit. You could also use the claims under "initial_search_retrieval_result" and "verification_retrieval_result" as hints, but the final decision is up to you.

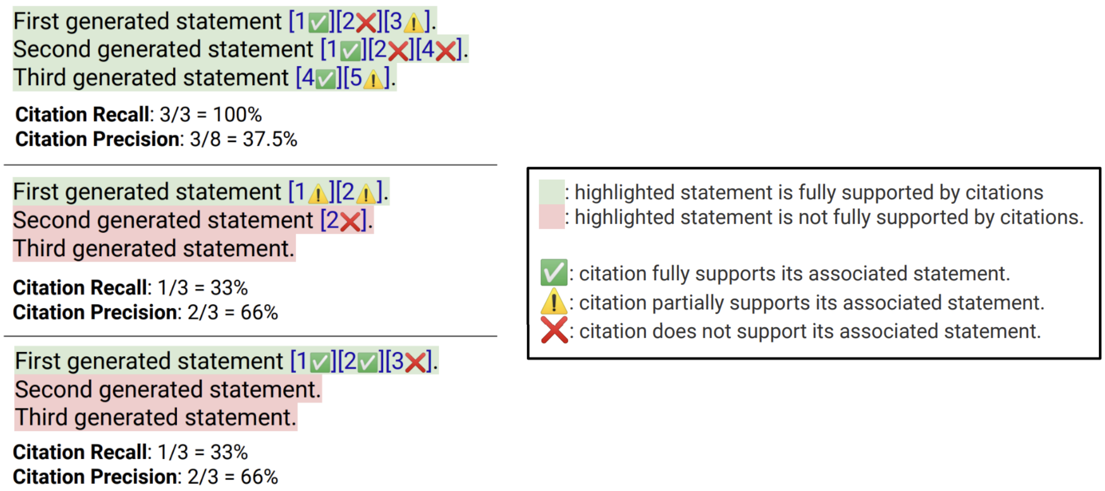

First generated statement [1✅][2❌][3⚠️].
Second generated statement [1✅][2❌][4❌].
Third generated statement [4✅][5⚠️].
Citation Recall: 3/3 = 100%
Citation Precision: 3/8 = 37.5%
First generated statement [1⚠️][2⚠️].
Second generated statement [2❌].
Third generated statement.
Citation Recall: 1/3 = 33%
Citation Precision: 2/3 = 66%
First generated statement [1✅][2✅][3❌].
Second generated statement.
Third generated statement.
Citation Recall: 1/3 = 33%
Citation Precision: 2/3 = 66%
🟩: highlighted statement is fully supported by citations
🟥: highlighted statement is not fully supported by citations.
✅: citation fully supports its associated statement.
⚠️: citation partially supports its associated statement.
❌: citation does not support its associated statement.

Figure 2: Examples of calculating citation recall and precision in Liu et al. [1]. Each statement is followed by a list of citation numbers, each of which is labeled as “supported”, “partially supported”, or “not supported”.

3.3 Comparison with Bing Chat

Another line of research in building hallucination-free LLM-based agents is generating text with *citations*, which are retrieved information that supports the statements. Bing Chat and Perplexity AI are two representative examples. Citations help increase the trustworthiness of the agent’s response, but they may be incorrect. For these systems, we can evaluate their faithfulness by checking citation precision and recall [1].

CITATION-RECALL is the proportion of verification-worthy statements that are fully supported by their associated statements. CITATION-PRECISION is the proportion of generated citations that support their associated statements. Figure 2 shows a few examples of how to calculate these metrics.

(d) Select <u>two</u> of your previous conversations and test them with Bing Chat (specific directions follow below). Write down your results in the following format. Additionally, calculate CITATION-RECALL **and** CITATION-PRECISION **for each turn.** (Gradescope Q7.1 - Q7.4)

```
# Start of dialog
User: "user's turn"
Bing: "Bing's turn"
-------------
- "First statement"    [1] support/partial/no, [2] support/partial/no, ...
- "Second statement"   [1] support/partial/no, [3] support/partial/no, ...
...
Citation Recall: / = %
Citation Precision: / = %
-------------
User: "user's turn"
Bing: "Bing's turn"
-------------
- "First statement"    [4] support/partial/no, [5] support/partial/no, ...
- "Second statement"   [5] support/partial/no, [6] support/partial/no, ...
...
Citation Recall: / = %
Citation Precision: / = %
-------------
...
# End of dialog

# Write down all the links citated:
[1]: https://cooking.stackexchange.com/questions/11346/
[2]: https://cooking.stackexchange.com/questions/11268/
```

. . .

0-th turn: First, enter:

> I'm browsing "https://[YOUR DOMAIN].stackexchange.com/" and hopefully you can find the related information from the website for me and form an answer. Remember: your claim must be supported by information from this website and you shouldn't make it up.

Bing will likely respond with something like:

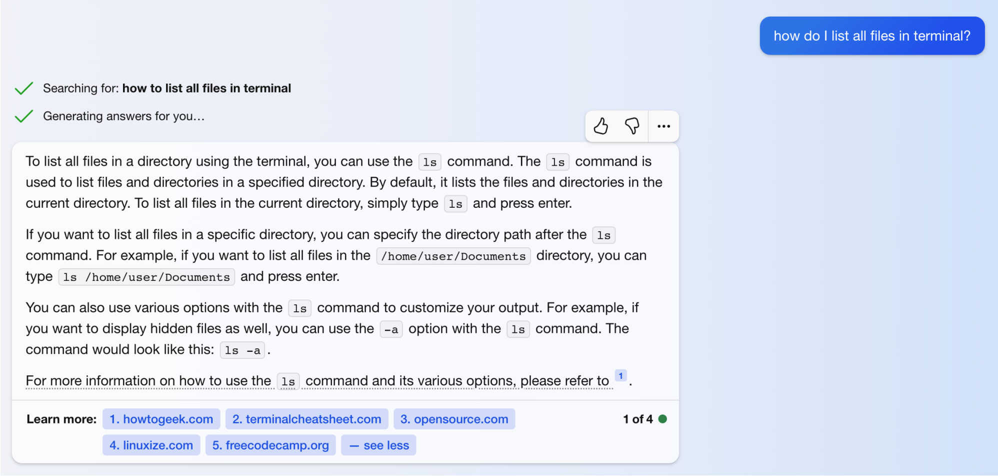
> Hello, this is Bing. I can help you find information from the website you are browsing. What is your question?

This user-agent turn counts as the "0th turn". You do not need to annotate or submit this to Gradescope.

1-st turn and onwards: Then, you should enter the same first turn query as you did with GenieChat, i.e., the first "user's turn" here should be the same as the corresponding first turn you reported in 2.2(c).

After that point, the conversation you had with GenieChat and the one you are now having with Bing Chat may diverge. You should continue the Bing Chat conversation to attempt to find information for the same question, but for the sake of naturalness, the queries you enter in later turns (i.e. all turns except the first turn) do not need to be exactly the same.

What counts as statements? For Bing Chat, if citations are attached to the end of a sentence, then that sentence is a "statement" (e.g. "First generated statement" in all three cases shown in Figure 2). However, Bing Chat sometimes generates paragraphs without citations (see example in Figure 3). In such cases, each paragraph becomes a statement. A paragraph without citations attached is inherently unsupported.



how do I list all files in terminal?

✓ Searching for: **how to list all files in terminal**

✓ Generating answers for you…

To list all files in a directory using the terminal, you can use the `ls` command. The `ls` command is used to list files and directories in a specified directory. By default, it lists the files and directories in the current directory. To list all files in the current directory, simply type `ls` and press enter.

If you want to list all files in a specific directory, you can specify the directory path after the `ls` command. For example, if you want to list all files in the `/home/user/Documents` directory, you can type `ls /home/user/Documents` and press enter.

You can also use various options with the `ls` command to customize your output. For example, if you want to display hidden files as well, you can use the `-a` option with the `ls` command. The command would look like this: `ls -a`.

For more information on how to use the `ls` command and its various options, please refer to [1].

Learn more: 1. howtogeek.com 2. terminalcheatsheet.com 3. opensource.com 4. linuxize.com 5. freecodecamp.org — see less

1 of 4 ●

Figure 3: An example where Bing Chat does not attach citations to all sentences. There are four statements here (each paragraph counts as a statement), however, the first three statements without citations are unsupported.

What are valid citations? Citations must be from your StackExchange domain. If a produced citation is not linked to StackExchange, then we disregard it, i.e., do not include it in the annotation and calculation in this section.

3.4 Analyze the Social Knowledge of the LLM

StackExchange is a community-contributed platform. Given the diversity of its users, a single question on StackExchange can elicit a range of answers, some of which may occasionally conflict with each other. To navigate this, StackExchange has a voting mechanism, leveraging collective wisdom to highlight the most valuable information.

With GenieChat grounded on a StackExchange domain, we can examine how it utilizes answers from various users in formulating its output. Does it favor the highest-voted answers? Does it combine different answers? Does it sidestep answers with negative votes? Does it resolve conflict? If so, how? ...

(e) Write down your observations on how GenieChat leverages different answers in your conversations. (2-3 sentences) (Gradescope Q8)

3.5 What's next?

Based on your experience, what do you think are the future research opportunities/directions? In this section, feel free to include any thoughts you have about the homework.

(f) Suggest ways to improve the evaluation metrics. (1-2 sentences) (Gradescope Q9.1)

(g) Suggest ways to improve GenieChat based on your experience. (1-2 sentences) (Gradescope Q9.2)

4 Remember to upload your Colab notebook as a PDF and also your demo.log file

The demo.log file can be found under /content/drive/My Drive/genie_open_text/data.

(Gradescope Q10)

References

[1] Nelson F. Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines, 2023.

[2] Sina J. Semnani, Violet Z. Yao, Heidi C. Zhang, and Monica S. Lam. Wikichat: Combating hallucination of large language models by few-shot grounding on wikipedia. https://oval.cs.stanford.edu/local-papers/semnani-local.pdf, 2023.