CSE 156 Project Midway Report

Authors
Elena Tomson - etomson@ucsd.edu
Steven Shi - s3shi@ucsd.edu
Anya Chernova - achernova@ucsd.edu
Shravi Jain - shj009@ucsd.edu

1 Introduction

We aim to address the problem of forgetfulness with LLMs due to their short-term memory and/or inadequate retrieval ability in the context of using LLMs as a medium for storytelling. In particular, we aim to create a framework that allows a LLM to remember both specific data points such as a character's stats, items in an inventory, and money, as well as general and abstract knowledge such as character relationships and world history.

We chose this task because it tackles a prominent limitation seen in LLMs: their limited memory over a long time period. This task is particularly interesting because it requires us to think about how long-term memory works with humans. Furthermore, coming up with retrieval mechanisms tailored to the specific task of maintaining world state for storytelling is an intellectually interesting endeavor. The primary impact of this project is that it would allow for interactions with LLMs that engage with a much larger window of user-input. Storytelling, as is the focus of our project, is one example of this, as our project would theoretically enable LLMs to create narratives based on long histories and rich world-building.

We created a model which consists of a RAG pipeline for context storage, an open source Llama model for story facts decomposition, and a core OpenAI model for story generation. We have found that this structure of us integrating a database as model's memory lead to a better, more coherent, story generation.

2 Background

Zhiheng Lyu et. al. proposed a framework for tracking the state of a world in a story telling setting by decomposing events into atomic facts. We used this framework as inspiration for the decomposition step in our model.

The "Automatic Story Generation: Challenges and Attempts" paper that went over different approaches directed at improving story generation inspired the iterative story generated approach in our model.

3 Method

Our current method involves using a simple pre-trained core-model for story generation with a retrieval database. We generate story iteratively, allowing for user input to progress the story. The generated portion of the story is decomposed into facts by an open-source Llama model. Each iteration we store facts about the plot in the RAG pipeline. These facts are then used as a part of a prompt for the next iteration in order.

Currently, our framework consists of a vectorstore retrieval database powered by FAISS, and a core-model powered by GPT 4o. We use Llama-7B to decompose story plot points into a list of facts, which we then store in the vectorstore database.

This model leverages the retrieval database's ability to store the details of the story in the form of vector embeddings of the facts generated by the model. Hence, it is able to remember the facts even after multiple iterations. However, one of its weaknesses is in remembering quantitative data which we tried to address by better prompt engineering.

4 Experiments

Experiments done

- Generated a few short stories with different pre-trained models to see how they perform. We investigated three models: Gemma-7b, Llama-7b, Mistral-7b. We currently choose Llama because it suited our needs the best out of the three, in our subjective human perspective, in generating stories and decomposing facts in those stories. In addition, we used the gpt 4o OpenAI model as a main model to continue the stories based on user interaction as this model is considered one of the best.
- We created a prompt to pass into a model with a story to decompose it into facts, and tried it on a story to see the decomposition.
- We created code for continuous story generation: After a portion of a story is generated, we let the user input an action to take and use that action as input to the model to continue generation.
- We started integrating a database: stored decomposed facts, and passed them in with user prompt to generate continuation of the story to improve retention of facts in previous parts of the story.
- Since we were noticing that the model is unable to remember the quantitative data from the stories well, we decided to address this issue by prompt engineering. We tried re-wording the prompt for decomposition so that the embeddings stored in the database are more meaningful and can help with the retention of the facts. We also added a new category for 'Quantitative Facts' instead of just having 'Pre', 'Static', and 'Post' as decomposed facts.
- Not only did we experiment with the decomposition prompt, we also experimented with better prompt engineering for retention-related questions where we asked more explicit questions for quantitative data.

Hence, the main purpose of our experiments is seeing how well a model retains the details provided to it in multiple iterations. We look out for two things - 1) whether the model is able to integrate the user input to move the story forward and 2) if the model is able to keep track of everything and remembers the details of the story across the multiple iterations.
Furthermore, the results for the experiments will be discussed under **Results** section.

Metrics and Experimental Setup

The experimental setup that we used is as follows:

- Generate a story premise that allows for user interaction
- Run with different number of iterations of the user progressing the story (tried 5, 10 and 20)
- Ask the model if it remembers a fact from the first iteration or correctly keeps track of the changes happening especially with respect to quantitative data

Our first input is we choose the beginning of a short story to provide premise to the Llama model and continue it to an arbitrary point but end it at a point which leaves scope for user interaction. Then our additional inputs are in the form of user input in every iteration for moving the story forward (provided to the core model - GPT4o). Additionally, we ask the model a question related to the story.
The final output is the response to the retention-related question.
The intermediate steps include decomposing the facts (using Llama) based on the user input and updating the database in every iteration.

The metric which we use for the model involves doing human evaluation on the response. Since, we are acting as user, we can evaluate whether the model's response is consistent with what we input in the iterations.

For example, we asked the model to generate a story premise based on the fairy-tale called Kolobok, which tells a story of a sentient bread-like being that runs away from the house it was made in. In the premise, Kolobok encounters many different characters-animals and scenarios on its way which allows for a good set up to let user decide the course of action.

We fed this premise into our core model with a database as well as into default Llama-7B, continued the story 10 times with the user input, and then asked each model which animal Kolobok encountered first. Our model correctly answered the question, while default Llama-7B incorrectly answered with the wrong animal.

This rudimentary experiment shows that the addition of a database and using RAG system does allow LLMs to remember farther back in time.

4.1 Model

Our model is as follows:

We use Llama-7B for the decomposition of each story into discrete facts, and use ChatGPT-4o for the continuous generation of the story via retrieved facts.

FAISS was used for storage and retrieval for the vector store database.

4.2 Datasets

We are using the TinyStories (`https://www.kaggle.com/datasets/thedevastator/tinystories-narrative-classification`) dataset to increase the diversity of our language model's outputs. This dataset consists of a large collection of short stories. Users can either create their own story context or use a story from the dataset as a starting point. In particular, the option to randomly select a story from this extensive dataset allows our model to generate narratives based on a wide range of contexts, making our application more dynamic and engaging. Additionally, it enables us to evaluate our story generation model, as well as our story facts storage and retrieval system, across a broad variety of scenarios.

4.3 Baseline

We use the default LLaMA-7B model as our baseline. Our goal is to compare the performance of our model against this standard open-source model to evaluate the effectiveness of our additions, design changes, and methods. This comparison allows us to assess whether our approach leads to improvements on the target task. We compared performance of our model with the default one in two aspects: how effective it is at remembering facts and how coherent the generated story across multiple user prompts is.

4.4 Code

Here we provide the link to the code for the model which we came up with. We use Llama to provide us a start to the story based on a sample input and generate it to a point where a user can progress it further. We decompose this output into facts, convert it into vector embeddings and store it in our RAG pipeline. Then we do multiple iterations of user input, each time generating continuation of the story with OpenAI gpt 4o and updating the database.

URL:
`https://colab.research.google.com/drive/1QCir-t6XSfEYY3UhHajibtzWWxr6U07_?usp=sharing`

The references for the LLAMA model and retrieval model are cited in the **References** section

5 Results

We compared the story generated by our model and default Llama's output for the same prompt and saw that the story generated by our model takes the context into account better than Llama because we provide facts of the story generated so far as context to the RAG pipeline. Since a normal model like Llama does not leverage RAG to continuously store the information being generated in each iteration, its story is not as coherent as the one generated by our model. Moreover, it is unable to remember all the details provided, due to the lack of structure in input.

Experiment	**Story Character**	**Model Used For Decomposition**	**Result**	**Code link**
Decomposition Prompt modified	Monip, the Transformer	Llama	Correctly remembers the result	Link1
Decomposition Prompt modified	Kolobok	Llama	Does not consistently remember the quantitative data	Link2
Decomposition Prompt modified + Added action of Monip collecting 20 cubes	Monip, the Transformer	Llama	Remembers the quantity but failed to take into account the action of collecting cubes	Link3
Decomposition Prompt modified + Added action of Monip collecting 20 cubes	Monip, the Transformer	GPT 3.5 turbo	Incorrectly remembers the quantitative data	Link4
Action of Monip collecting cubes 20 times + explicitly ask for quantities + rework context to not add actions into database	Monip, the Transformer	GPT 3.5 turbo	Could not keep track of the changes in the quantity of cubes collected	Link5
Retrieve information using the model from the input, then prompt explicitly for quantities + rework context to not add actions into database + add story context into database directly	Mateo	GPT 3.5 turbo	Unable to recognize change in quantitative variable by taking context into account	Link6

Table 1: Experiment results for different models and prompts related to quantitative data

An example generation of our model is below:

- We prompt the user on which fictional-story world they want to use.

  `To create your own story, put 0 To use a specific story from the dataset, put 1 To use a random story from the dataset, put 2 What would you like to do?2 Write a fantasy story that is based on the following plot points, and leave off/stop generating at a point where story is unfinished: Once upon a time, there was a little girl named Lily. She loved to draw and color with her crayons. One day, Lily wanted to make a special picture for her mommy. She got out her paper and crayons and started to draw.` **#story continues ...**
  #prompting the user
  `Now it's your turn to continue the story.  What happens next to Lily and her love of spreading honey on her pictures?`

- Then we decompose the story in the form of facts

```
Deconstruct the event point into atomic facts:
Pre-facts:
1.  Lily loves honey.
2.  Lily's mommy loves honey on her picture.
3.  Lily has a special technique for spreading honey on pictures.
```

```
4.  Lily invites friends over for art parties.
5.  Lily teaches her friends how to spread honey on their pictures.
Post-facts:
1.  Lily became known as the "Honey Artist" in her town.
2.  People came from far and wide to see Lily's honey-spread
masterpieces.
3.  Lily's love for spreading honey on her pictures turned into a
magical gift.
4.  Lily's gift brought joy to everyone who saw it.
Static facts:
1.  Lily continues to spread her love and honey wherever she goes.
2.  Lily's love for honey remains unchanged.
```

- For a fixed number of iterations, we take input from the user, decompose it into facts and update the database. This is what the story generated by the model looks like:

```
Lily grows up and becomes a famous honey artist Lily's love for
spreading honey on her pictures never faded as she grew up.  She
continued to use honey in her artwork and eventually gained
recognition for her unique style.  Her artwork was displayed in
galleries all over the world and she became known as the "Honey
Artist." People came from far and wide to see her creations and
taste the honey that she used in her paintings.  Lily was happy to
share her love for honey with the world and her love for art and
honey only grew stronger with each passing day.

New document added and FAISS index saved.

Lily used her fame to become a bee conservationist After becoming
a bee conservationist, Lily's passion for spreading honey on her
pictures turned into a passion for protecting and preserving bees.
She started giving presentations at schools and community events
about the importance of bees and how to create bee-friendly habitats.
Her efforts gained recognition and she became a well-known advocate
for bee conservation.  Eventually, Lily's work helped to save
many bee populations and she was known as the "Bee Hero" in her
community.

New document added and FAISS index saved.
```

...we do this for multiple iterations

- At the end, we ask the model question related to the story and see how well it retains the details.

  For example, we prompt:

```
"Given everything that has happened, can you write an epilogue for
the story?"
```

We get the following result:

```
 Epilogue:

Years went by and Lily's love for drawing and coloring continued
to grow.  She always remembered how special her mommy made her feel
when she used honey on her pictures.  It became a tradition for Lily
to add a touch of honey to all of her artwork, [...]

[...]  Her unique use of honey in her artwork caught the attention
of her professors and she was praised for her creativity and skill.

Lily's love for spreading honey on her pictures never faded, [...]
```

```
But no matter how famous she became, Lily never forgot the simple
joy of sharing her love and creativity with her loved ones through
her honey-spread pictures.  And she always remembered the sweet
moment when she first discovered the magic of honey on her drawings.
```

Hence, we can observe that the model is able to remember the qualitative details well but fails to recognize when a quantitative variable changes (referring to the table). We tried two methods - 1) look through relevant information in vectorstore, and update and 2) Use separate mechanisms for story generation and retrieval of items

One issue we discovered is that the vectorstore would recognize actions as permanent facts in our world state, and as a result would keep trying to make the main character perform those actions. For example, the vectorstore would remember that "Monip is getting 3 energon cubes from storage", and constantly keep trying to make the main character grab 3 cubes from storage. The solution we attempted to was to be selective about what we let into the vector store. This showed some improvement but the LLM still struggles to remember quantitative changes accurately. However, the vector database and storage of facts helps the model to keep track of the storyline.

6 Discussion

We believe that our findings may be applicable to other areas such as tutoring, where an LLM would need to remember large amounts of information relating to a student's progress, and healthcare assistance, where an LLM would need to remember large amounts of information regarding a patient's past illnesses, symptoms, activity, and care received. There could be other potential areas where such model structure could be useful.

7 Conclusion

By comparing the output of our model (which consists of a RAG pipeline built on top of OpenAI and the input is provided in the form of decomposed facts through LLAMA) with general LLAMA in continuous story generation, we can conclude that the decomposition of input into facts and use of a RAG pipeline to store those facts improves the quality of a generated story. The RAG pipeline helps the model to remember more context. This retention of the story outline shows that better prompts and structure lead to more coherent results in the process of story generation.

By using the metric of how well a model remembers all the details and human evaluation of the generated stories, we can also conclude that the form of input we provide to the model can deeply impact its result, as seen by the better results due to decomposition. However, we can do more experiments (e.g with structure of our database), develop more robust metrics for comparing the generated stories and further improving our model. We would also like to look into creating an actual user-facing application where people can generate unique stories, since our ultimate goal is to create a real-world product that makes a positive impact in people's lives.

8 References

- Reference for Mistral and Gemma models we tried -
  `https://huggingface.co/mistralai/Mistral-7B-v0.1`
  `https://huggingface.co/google/gemma-7b`

- Reference for LLAMA -
  `https://huggingface.co/meta-llama/Llama-2-7b-chat-hf`

- Reference for retrieval model -
  `https://python.langchain.com/docs/integrations/vectorstores/faiss/`

- FACTTRACK: Time-Aware World State Tracking in Story Outlines
  `https://arxiv.org/pdf/2407.16347`

- Automatic Story Generation: Challenges and Attempts
  `https://arxiv.org/pdf/2102.12634`

- TinyStories Dataset
  `https://www.kaggle.com/datasets/thedevastator/tinystories-narrative-classification`

Table links:

- Link1:
  `https://drive.google.com/file/d/1vDZRHI8adpvMexD-0w_4DPU7JftAHry2/view?usp=sharing`
- Link2:
  `https://drive.google.com/file/d/1vDZRHI8adpvMexD-0w_4DPU7JftAHry2/view?usp=sharing`
- Link3:
  `https://drive.google.com/file/d/1vDZRHI8adpvMexD-0w_4DPU7JftAHry2/view?usp=sharing`
- Link4:
  `https://colab.research.google.com/drive/1bH-SyF-BPgKXQV_I5qU2q-IR_uZ_7orJ?usp=sharing`
- Link5:
  `https://colab.research.google.com/drive/1bH-SyF-BPgKXQV_I5qU2q-IR_uZ_7orJ?usp=sharing`
- Link6:
  `https://colab.research.google.com/drive/13v7GYs0-Qvt9TN1muWERJHPr-F3xXmZO#scrollTo=j_igtDJbqMZc`