Dialogue2Data (D2D): Transforming Interviews into Structured Data for Analysis - Proposal Report

Sienko Ikhabi, Dominic Lam, Wangkai Zhu, Yun Zhou

Table of contents

Executive Summary

Project Introduction

1. **Motivation**
2. **Data Format: Input and Output**
3. **Pipeline Overview**

Data Science Techniques

1. Proposed Solutions

Traditional Retrieval-Augmented Generation (RAG)

Our baseline solution is built on the Retrieval-Augmented Generation (RAG) framework, which is well-suited for question-answering tasks involving long and unstructured text, such as interview transcripts.

In a traditional RAG setup, the system is split into two parts: a retriever and a generator.

- **Retriever**:
  We first segment the interview transcript into smaller, manageable question-and-answer (QA) pairs. Then, we use a pretrained transformer model such as MiniLM or DPR to convert both the QA pairs and the guideline questions into vector embeddings. These embeddings are compared using cosine similarity to find the top-k (or top-p) QA pairs most relevant to each guideline question. This enables us to narrow down the transcript to only the most relevant content.
- **Generator**:
  The top matches are passed to a large language model (LLM) like ChatGPT, which is then prompted to extract the main ideas or answers from the selected content. This LLM serves as the generator in our pipeline.

This approach allows us to maintain control over what the LLM sees, thereby reducing the likelihood of hallucinated answers. It is also modular: the retriever and generator can be swapped or fine-tuned independently.

We have already implemented this baseline RAG system, and initial results have been well-received by our capstone partner. Even without the generator component, the retriever alone provides helpful insights by surfacing the most relevant parts of the transcript for each question.

However, one limitation of Traditional RAG is that it always returns a fixed number of top results. This means that if the true answer is not ranked within the top-k, it may be missed entirely. Additionally, the generator assumes that the retrieved content is relevant, which can sometimes reduce accuracy.

Self-RAG: Self-Reflective Retrieval-Augmented Generation

To address the limitations of the traditional RAG approach, we propose an extension called Self-RAG, which incorporates an additional step of self-evaluation using an LLM.

The core idea of Self-RAG is to validate the quality of the retrieval before final generation. After the retriever identifies the top-k QA pairs for a given guideline question, we use an LLM to ask:

"Are these QA pairs actually relevant to the question?"

If the answer is yes, we proceed as usual and ask the model to extract the main points. If not — for instance, if the retrieved content is only loosely related or off-topic — the system automatically adjusts its retrieval parameters (e.g., increasing or decreasing k or p) and re-runs the retrieval step.

This dynamic loop gives the system a way to adaptively improve its own performance without human intervention. It can lead to more accurate and focused outputs, especially in cases where the first set of matches is suboptimal.

However, Self-RAG introduces some added complexity. It requires additional LLM calls for self-assessment, and the logic to iterate on retrieval adds to the overall system design. Despite this, we believe that the trade-off is worthwhile, particularly if it helps minimize the need for human oversight — one of our partner's primary goals.

Self-RAG remains transparent, just like Traditional RAG, but adds flexibility and quality control at a key step in the pipeline.

2. Evaluation

Timeline

Deliverables and Deadlines

- Proposal Presentation: Friday, May 2, 2025, 14:00–16:00
- Proposal Report Draft: Tuesday, May 6, 2025, 12:00
- Proposal Report Final: Friday, May 9, 2025, 17:00
- Runnable Draft to Mentor: Monday, June 9, 2025, 16:00
- Final Presentation: Thursday, June 12, 2025
- Draft Report to Mentor: Wednesday, June 18, 2025, 16:00
- Final Report and Project to Partner/Mentor: Wednesday, June 25, 2025, 12:00

Weekly Milestones

Week 1: April 28 – May 4, 2025 (Project Setup and Proposal Presentation)

- Repository set up with initial files and documentations.
- Proposal presentation delivered.
- Synthetic data preparation started (5 transcripts).
- Proposal report draft 50% complete.

Week 2: May 5 – May 11, 2025 (Proposal Report and RAG Implementation)

- Proposal report draft and final submitted.
- RAG approach script completed and tested.
- Unit tests passing (80% coverage).
- Documentation completed with baseline details.

Week 3: May 12 – May 18, 2025 (Self-RAG Exploration and Implementation)

- Self-RAG approach script completed and tested.
- Documentation completed with Self-RAG details.
- Evaluation preparation with baseline approach.

Week 4: May 19 – May 25, 2025 (Evaluation Development)

- Evaluation completed on both approaches
- Optimization (Hyperparameters/Models Tuning on both approaches
- Decided which approach to develop
- Documentation updated with metrics (Authentic and Synthetic Data)

Week 5: May 26 – June 1, 2025 (Structured Output and End-to-End Testing)

- Finalized structure output
- Documentation finalized for output and testing
- Develop and test Command-Line Interface (CLI)
- Pass end-to-end tests (including CLI interactions)
- Final presentation slides (>= 50%) completed

Week 6: June 2 – June 8, 2025 (Runnable Draft, Final Report and Presentation)

- Bug reduced
- Runnable draft and CLI completed
- Final report (>= 50%) completed
- Final presentation slides completed

Week 7: June 9 – June 15, 2025 (Runnable Draft and Final Presentation)

- Runnable draft submitted (June 9, 16:00)

- Final Presentation delivered (June 12/13)
- Draft report (>= 50%) complete
- Bugs reduced to < 5 critical issues

Week 8: June 16 – June 20, 2025 (Draft Report and Finalization)

- Final report draft submitted (June 18, 16:00)
- Package finalized with tests passing
- Documentation completed.

Post-Week: June 23 – June 25, 2025 (Final Submission)

- Final report (final version) submitted (June 25, 12:00)
- Package submitted (June 25, 12:00)