Dialogue2Data (D2D): Transforming Interviews into Structured Data Final Project Report

Sienko Ikhabi, Dominic Lam, Wangkai Zhu, Yun Zhou

Table of contents

1 Executive Summary

Open-ended text data holds immense potential for discovering user insights, but extracting those insights efficiently remains a major challenge—particularly in unstructured formats. Our capstone partner, Fathom, specializes in surfacing insights from raw survey data and interview transcripts to support client decision-making. While structured survey responses follow a question-and-answer format, interview transcripts are conversational and free-flowing, making them more difficult and time-consuming to analyze reliably.

To address this challenge, we designed and implemented a data product that automates the extraction of relevant responses from unstructured interview transcripts. Our solution leverages a Retrieval-Augmented Generation (RAG)(Lewis et al. 2020) framework, which consists of:

- A **retriever** that selects the most relevant passages from the transcript based on a guideline question,
- A **generator** that formulates concise answers using the retrieved text and prompt,
- And an **evaluator** that scores each generated response on five key quality metrics and flags uncertain cases for human review.

This system significantly reduces the need for manual transcript annotation while improving accuracy and consistency. In testing, it increased correct response identification by 87% compared to the previous baseline, and streamlined the review process by automatically surfacing low-confidence matches.

The final pipeline is fully functional and ready for integration into Fathom's existing analytics workflow. By enabling structured analysis of unstructured interviews, our tool enhances Fathom's ability to deliver richer insights to clients and scale their operations to handle more complex, conversational data. Ultimately, this project extends the capabilities of conversational survey analytics, allowing organizations to extract meaningful information with greater speed, precision, and depth.

2 Project Introduction

2.1 Problem Statement

Organizations often rely on surveys and interviews to understand their users, assess needs, and guide decision-making. While traditional surveys provide structured data that is relatively easy to analyze, open-ended formats—especially in interviews—offer deeper, more nuanced insights. However, the richness of these responses comes at a cost: they are difficult to process at scale due to their unstructured and free-flowing nature.

Our capstone partner, Fathom, specializes in analyzing open-text data from surveys and interviews. Their platform already supports structured survey responses, but scaling up analysis of interview transcripts has proven to be a key challenge. Interview responses are less uniform and harder to map directly to the original guideline questions. As a result, Fathom's team must rely heavily on large language models and manual transcript review to extract relevant content—an approach that is time-consuming, error-prone, and difficult to scale across large volumes of data.

2.2 Data Formats

The dataset used for this project consists of over 200 unstructured, conversational interview transcripts, alongside more than 20 sets of guideline questions to which the responses must be mapped. These transcripts vary widely in length, tone, and structure, posing a realistic challenge for scalable NLP systems. The output of the pipeline is a structured `.csv` file in which each row corresponds to an interviewee and each column contains the extracted response to a specific guideline question, enabling seamless downstream analysis and integration into existing workflow.

2.3 Project Objectives

To address this challenge, we refined the problem into the following tangible data science objectives:

1. **Automate the mapping of interview transcript content to original guiding questions**, reducing human effort and improving consistency. Currently, Fathom relies heavily on large language models (LLMs) for response extraction, followed by manual review to ensure quality. This approach is time-intensive and difficult to scale. Automating the mapping process allows for faster, more consistent analysis and reduces dependence on manual labor.

2. **Ensure high response quality** by integrating an evaluation module that scores generated outputs and flags uncertain responses for manual review. Manual validation of LLM outputs is costly and error-prone. A built-in evaluator provides a systematic way to assess the quality of generated responses across key metrics, surfacing only ambiguous or low-confidence cases for human review—saving time while preserving accuracy.
3. **Deliver an end-to-end pipeline** that can integrate into Fathom's existing workflow, enabling scalable and repeatable analysis of new interview data. For this solution to be truly impactful, it must work within Fathom's current analytics infrastructure. An integrated, end-to-end pipeline ensures seamless adoption and allows the team to expand from survey data to conversational interview data with minimal friction.

2.4 Solution Overview

Our solution is built using a **Retrieval-Augmented Generation (RAG)** architecture, leveraging NLP techniques including **embedding-based retrieval** and **large language models** for answer generation. The pipeline also includes an evaluator that assesses output across five quality dimensions: **correctness**, **faithfulness**, **precision**, **recall**, and **relevance**.

By framing the problem around semantic matching and response generation, we developed a scalable and effective data science pipeline that meets Fathom's analytical needs. This system empowers them to analyze open-ended interview transcripts more efficiently and deliver richer, faster, and more actionable insights to their clients.

3 Data Science Methods

3.1 Overview

The core methods include Retrieval-Augmented Generation (RAG) for transcript processing and semantic matching, semantic embeddings, cosine similarity, and a referencing process to ensure accurate and traceable outputs, complemented by a customized evaluation framework. The evaluation framework, inspired by RAGAS(Retrieval-Augmented Generation Assessment)(Es et al. 2023) and Fathom's Daedalus v4(DS-Daedalus Team 2025), employs LLM-based prompting to compute five metrics—Correctness, Faithfulness, Precision, Recall, and Relevance—enhanced with tailored prompts, feedback mechanisms, and flexible scoring, validated using a diverse golden sample set across ten topics.

3.2 Processing Framework

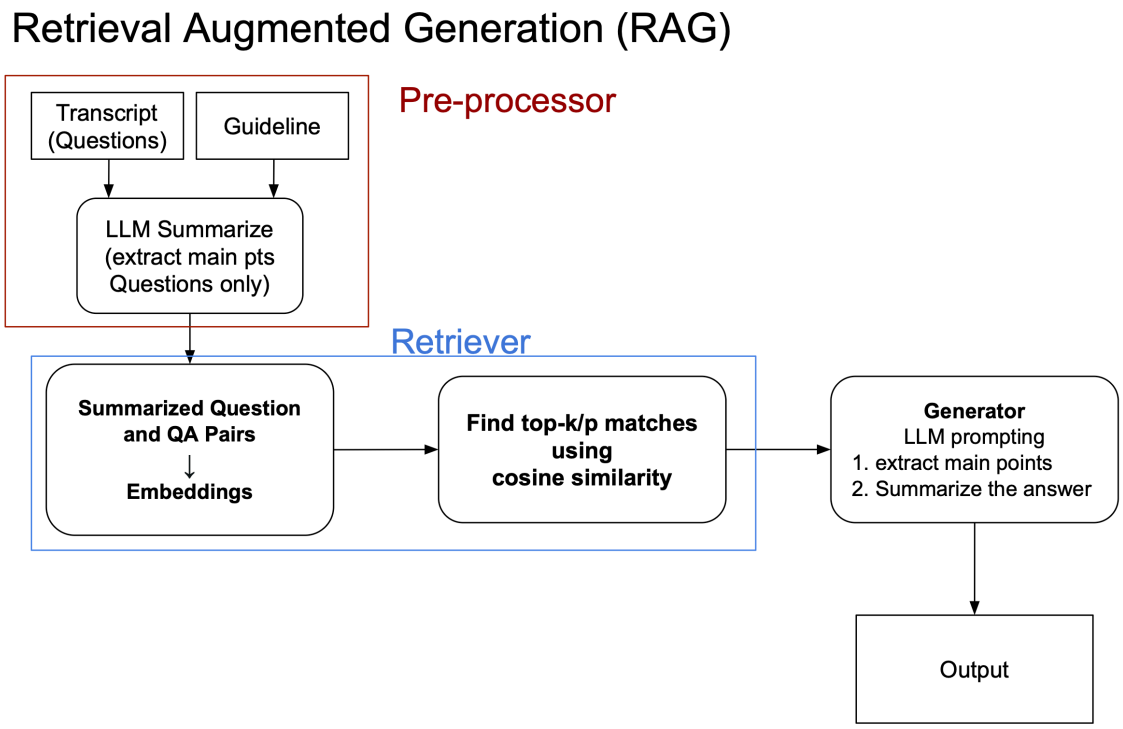

Figure 1: RAG Workflow for D2D Processor Pipeline

3.2.1 Retrieval-Augmented Generation (RAG)

RAG integrates large language model (LLM) capabilities with retrieval mechanisms to enhance the processing of interview transcripts. It leverages LLMs (e.g., ChatGPT-4o-mini(OpenAI 2024) or Claude-3.5(Anthropic 2024)) to summarize and refine content, combined with a retrieval step to align this content with relevant guideline questions, forming the foundation for structured output generation.

3.2.2 Embeddings

Embeddings convert textual data into numerical vectors using pre-trained models such as `SentenceTransformer` with `multi-qa-mpnet-base-dot-v1`(Reimers and Gurevych 2019). This process captures semantic relationships within the text, enabling meaningful comparisons and matches.

3.2.3 Cosine Similarity

Cosine similarity measures the cosine of the angle between two vector embeddings, providing a metric to assess the semantic similarity between summarized transcript content and guideline questions. This mathematical approach ensures effective alignment of related text.

3.2.4 Referencing Process

The referencing process employs fuzzy matching with `rapidfuzz`(Bachmann 2020) to trace summarized answers back to their original transcript lines. This method accounts for variations in wording, ensuring outputs are linked to source data for validation and interpretability.

3.3 Evaluation Framework

To assess the quality of D2D's output, we have designed the evaluator to ensure that the answers are not only accurate in meaning, but also generated through a reliable and precise process.

Inspired by RAGAS (Retrieval-Augmented Generation Assessment)(Es et al. 2023) and partner's internal documentation Daedalus v4(DS-Daedalus Team 2025), the evaluation framework provides five core metrics:

- **Correctness**: Measures how well the answer is consistent with the reference (ground truth).
- **Faithfulness**: Evaluates whether the answer is fully supported by the retrieved context and avoids hallucinations.

- **Precision**: Assesses the proportion of the answer that is actually supported by the retrieved chunks.
- **Recall**: Captures how many relevant facts from the context are included in the answer.
- **Relevance**: Reflects how closely the answer relates to the original guideline question. This metric is used only to assess questionnaire quality, not processor performance.

These metrics are computed using LLM-based prompting, with carefully designed templates and decision logic for edge cases such as ambiguous or evasive responses. Compared to the standard RAGAS pipeline, our customized evaluator introduces three key enhancements:

First, each metric uses a tailored LLM prompt designed to calculate metric score and handle edge cases such as vague or non-informative answers. Nonspecific or uninformative answers are detected through keyword matching and scored conservatively to ensure evaluation accuracy.

Second, built-in feedback mechanism: Each score includes LLM-generated feedback, improving interpretability and helping users understand and validate scoring results.

Third, flexible scoring and low-score highlighting: Users can customize metric weights and set thresholds to flag low-scoring responses, helping streamline validation and adapt to varied evaluation needs.

Finally, to validate metric correctness, we have built a small but diverse golden sample set across ten topics, such as climate change, food, NBA, and workplace culture. By varying the number of interviews across topics, the sample better reflects real-world diversity.

4 Data Product and Results

4.1 Data input adn output

4.1.1 Data Input

The D2D pipeline (processor part) processes two types of input files to extract and structure responses from unstructured interview transcripts based on provided guidelines.

Guidelines

- **Description**: A structured file listing the questions or prompts that the interviewer was supposed to cover to guide the interview and extraction process.
- **Format**: Comma-separated values (`.csv`)
- **Structure**:
  - Single column named `guide_text` with each row containing a question or prompt.
  - Questions align with those asked in transcripts for matching purposes.
- **Example**:
  - **File**: Extract from interview_food_sample_guidelines.csv

    `guide_text`

    What's a dish that reminds you of your childhood?

    Can you describe a meal that has a special meaning for you?

    ...

Transcripts

- **Description**: Raw text files containing conversational interview data, with alternating lines or labeled segments for interviewers and interviewees.
- **Format**: Plain text (`.txt`)
- **Structure**:
  - Each file represents one interview.
  - Content includes dialogue, with questions from interviewers and responses from interviewees.
- **Example**:
  - **File**: Extract from 001.txt

    Interviewer: Let's talk food. What's a dish that reminds you of your childhood?

Interviewee: Definitely my grandma's chicken and rice. She used to make it every Sunday, and the smell would just take over the whole house. It was simple—nothing fancy—but it was filled with love.

Interviewer: Can you describe a meal that has a special meaning for you?

Interviewee: Yeah, actually. My 18th birthday dinner. My parents surprised me by cooking all my favorite dishes—pad thai, roasted veggies, and this chocolate lava cake I was obsessed with. I remember feeling really seen, you know?

...

- **File**: Extract from 002.txt

  Interviewer: Alright, diving into food and memories—what dish instantly brings your childhood back?

  Interviewee: Oh man, my mom's arroz con leche. She'd make it every time I was sick, or honestly, just when I needed cheering up. The cinnamon smell still makes me emotional sometimes.

  Interviewer: Can you describe a meal that holds special meaning for you?

  Interviewee: Our Christmas Eve dinner. It's this big spread—tamales, roasted pork, rice, beans. It's loud and chaotic and full of stories. It's more than food—it's our whole culture on a table.

  ...

4.1.2 Sample Data Output

The D2D pipeline produces structured output by matching interviewee responses to guideline questions, consolidating results for analysis.

- **Format**: Comma-separated values (`.csv`)
- **Structure**:
  - Columns:
    - `Interview File`: Identifier of the source transcript file (e.g., `001`, `002`).
    - Additional columns named after guideline questions (e.g., "What's a dish that reminds you of your childhood?").
  - Each row corresponds to one interview, with cells containing the extracted response text.
  - Responses are concise, summarizing key points from the transcript.
- **Example**:

Interview File	What's a dish that reminds you of your childhood?	Can you describe a meal that has a special meaning for you?	...
001	Grandma's chicken and rice	18th birthday dinner with favorite dishes cooked by parents.	...
002	Mom's arroz con leche	Christmas Eve dinner with tamales, roasted pork, rice, beans; loud, chaotic, full of stories.	...
...	...	...	...

4.2 Processor Pipeline

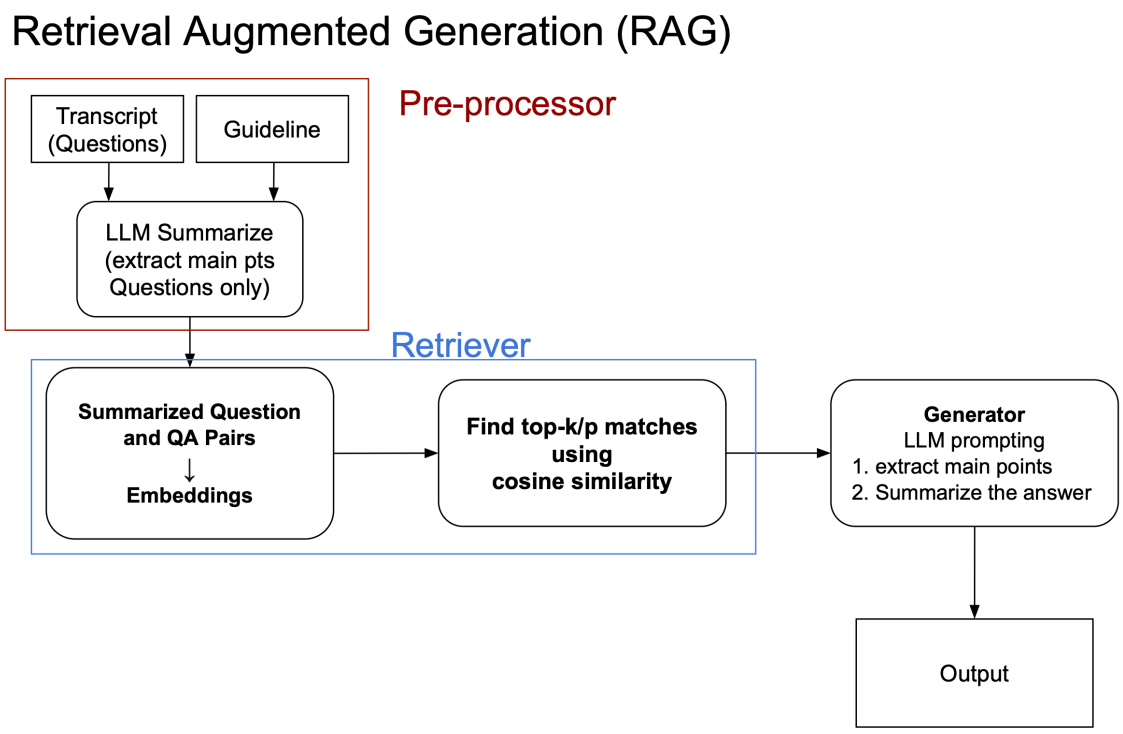

Figure 2: RAG Workflow for D2D Processor Pipeline

4.2.1 Processor Pipeline Description

The processor pipeline comprises three core components: the **Pre-processor**, the **Retriever**, and the **Generator**. These components work together to efficiently transform raw input data into meaningful, structured outputs aligned with the Dialogue2Data (D2D) project's objectives.

4.2.2 Pre-processor: LLM Summarization

The Pre-processor prepares the raw transcript for analysis by segmenting and summarizing its content.

- **Inputs**:
  - A raw interview transcript containing interviewer questions and interviewee responses.
  - A guideline providing standard questions or topics to guide the analysis.
- **Segmentation**:
  - The transcript is divided into question-and-answer (QA) pairs.
- **Summarization**:
  - The questions from each transcript QA pair and the guideline questions are summarized using a Large Language Model (LLM). This distills key points and reduces complexity, ensuring concise representations for matching.
- **Output**:
  - A list of summarized guideline questions.
  - A dictionary where each entry contains an original transcript QA pair and its summarized question.

4.2.3 Retriever: Embedding and Matching

The Retriever identifies the most relevant transcript QA pairs for each guideline question.

- **Inputs**:
  - The output for the pre-processor
- **Embedding**:
  - The summarized questions from both the transcript QA pairs and the guideline are converted into semantic embeddings using `SentenceTransformer` (e.g., `multi-qa-mpnet-base-dot-v1`).
- **Similarity Calculation**:
  - Cosine similarity is computed between the embeddings of each summarized guideline question and the summarized questions from the transcript QA pairs to determine semantic relevance.
- **Selection**:

  - For each guideline question, the top-k or top-p (Top-k selects the k most similar items; top-p selects items until their cumulative similarity reaches a threshold) most similar transcript QA pairs are selected based on their summarized questions.

- **Output**:
  - A set of relevant transcript QA pairs matched to each guideline question.

4.2.4 Generator: LLM Output Generation

The Generator produces concise, structured answers for each guideline question based on the matched QA pairs.

- **Inputs**:
  - The output of the retriever
- **LLM Processing**:
  - For each question in the guideline, an LLM (e.g., `gpt-4o-mini`) extracts key information from the relevant QA pairs selected from the retriever and synthesizes it into a coherent response.
- **Output Generation**:
  - Structured answers are generated in CSV format for each guideline question, with references to the original transcript provided in JSON format for traceability.

4.2.5 Integration and Data Flow

The pipeline operates sequentially:

1. The Pre-processor segments the transcript and summarizes the questions from both the transcript QA pairs and the guideline.
2. The Retriever embeds these summarized questions and matches the transcript QA pairs to the guideline questions using cosine similarity.
3. The Generator creates structured answers from the top-matching QA pairs using an LLM.

This streamlined process transforms unstructured interview data into structured insights with high efficiency and accuracy, leveraging state-of-the-art NLP tools like `SentenceTransformer` and LLMs to meet the needs of the D2D project.

4.2.6 Pros, Cons, and Justifications

Retrieval-Augmented Generation (RAG)

- **Pros**:
  - Enhances content processing by combining LLM strengths with retrieval, improving relevance.
  - Supports scalable analysis of large transcript datasets.
- **Cons**:
  - Relies heavily on LLM performance, which may introduce inconsistencies.
  - Adds computational complexity due to multiple processing steps.
- **Justification**:
  - Outperforms standalone LLM or retrieval-only methods by integrating both capabilities.
  - Preferred over manual methods due to scalability needs.

Embeddings

- **Pros**:
  - Captures semantic accuracy, enabling nuanced text comparisons.
  - Facilitates efficient processing of diverse text data.
- **Cons**:
  - High computational cost for vector generation.
  - Dependent on the quality of the pre-trained model.
- **Justification**:
  - Superior to keyword-based methods (e.g., TF-IDF) for semantic understanding.
  - Chosen over custom training due to resource constraints and availability of `SentenceTransformer`.

Cosine Similarity

- **Pros**:
  - Provides a normalized metric for semantic alignment, enhancing match reliability.
  - Efficient for high-dimensional vector comparisons.
- **Cons**:
  - Sensitivity to threshold settings can affect precision.

  - Requires careful tuning for optimal performance.

- **Justification**:
  - Outperforms Euclidean distance due to normalization, suitable for embeddings.
  - Preferred over manual alignment for automation and scale.

Referencing Process

- **Pros**:
  - Ensures traceability with flexible fuzzy matching.
  - Improves output interpretability by linking to source text.

- **Cons**:
  - Risk of false positives with lenient matching.
  - Requires threshold adjustment for accuracy.

- **Justification**:
  - More robust than exact matching for variable transcript wording.
  - `rapidfuzz` selected for its speed and integration compared to alternatives like `fuzzywuzzy`.

4.2.7 Potential Improvements and Challenges

- **Adaptive Thresholding**: Implementing dynamic thresholds for cosine similarity could optimize matching across datasets.
  - **Challenges**: Requires extensive testing and diverse data.
  - **Why Not Implemented**: Fixed thresholds sufficed for the current scope.

- **Fine-Tuned Embeddings**: Using domain-specific models could enhance embedding quality.
  - **Challenges**: Needs labeled data and computational resources.
  - **Why Not Implemented**: Pre-trained models met project needs within constraints.

4.2.8 Using the D2D Processor

The D2D processor is accessible via the `D2DProcessor` Python class, imported from the `d2d` module. Partners can use it to convert interview transcripts into structured data by initializing the processor and calling its `process_transcripts` method with paths to transcripts, guideline questions, and an output directory.

Key Parameters

- **`llm_model`**: Chooses the Large Language Model (e.g., 'gpt-4o-mini' default, or other OpenAI/Anthropic models).
- **`embedding_model`**: Selects the embedding model (e.g., 'multi-qa-mpnet-base-dot-v1').
- **`sampling_method`**: Determines the sampling approach (`TOP_K` for top k matches or `TOP_P` for nucleus sampling).
- **`custom_extract_prompt`** and **`custom_summarize_prompt`**: Optional custom prompts for tailored extraction and summarization.

Usage Example

```
from d2d import D2DProcessor

processor = D2DProcessor(
    llm_model="gpt-4o-mini",
    embedding_model="multi-qa-mpnet-base-dot-v1",
    sampling_method=D2DProcessor.SamplingMethod.TOP_K,
    top_k=5,
)
processor.process_transcripts(
    transcripts_dir="path/to/transcripts",
    guidelines_path="path/to/guidelines.csv",
    interview_name="interview",
    output_dir="path/to/output"
)
```

Installation

The package can be installed via `pip install .` or as an installable package from the repository.

Additional Notes

- The processor requires an internet connection to download the embedding model and access the LLM API.
- It includes a thematic alignment check, prompting users if transcript similarity falls below the threshold.
- The API calls to the LLM in the preprocessor and generator are asynchronous to increase the speed. For example, handling 100 transcripts can be reduced from approximately 1 hour to less than 10 minutes.

For more detailed technical information, see the D2D repository.

4.3 Key Results

To understand the performance of D2D processor, we conducted experiments using golden samples. We compared developed models against the client's baseline (vanilla LLM method), analyzed the average scores and distributions of different models across all metrics, and tuned retrieval parameters (top_k and top_p) to recommend optimal settings based on golden samples.

According to Figure 1, all of our developed models achieved much higher correctness scores compared to the client's baseline method. The baseline model scored only 2.20, while all developed models scored above 3.90, and the best-performing model top-k with gpt4-1, reached 4.22. This clear improvement demonstrates the strength of our RAG approach.

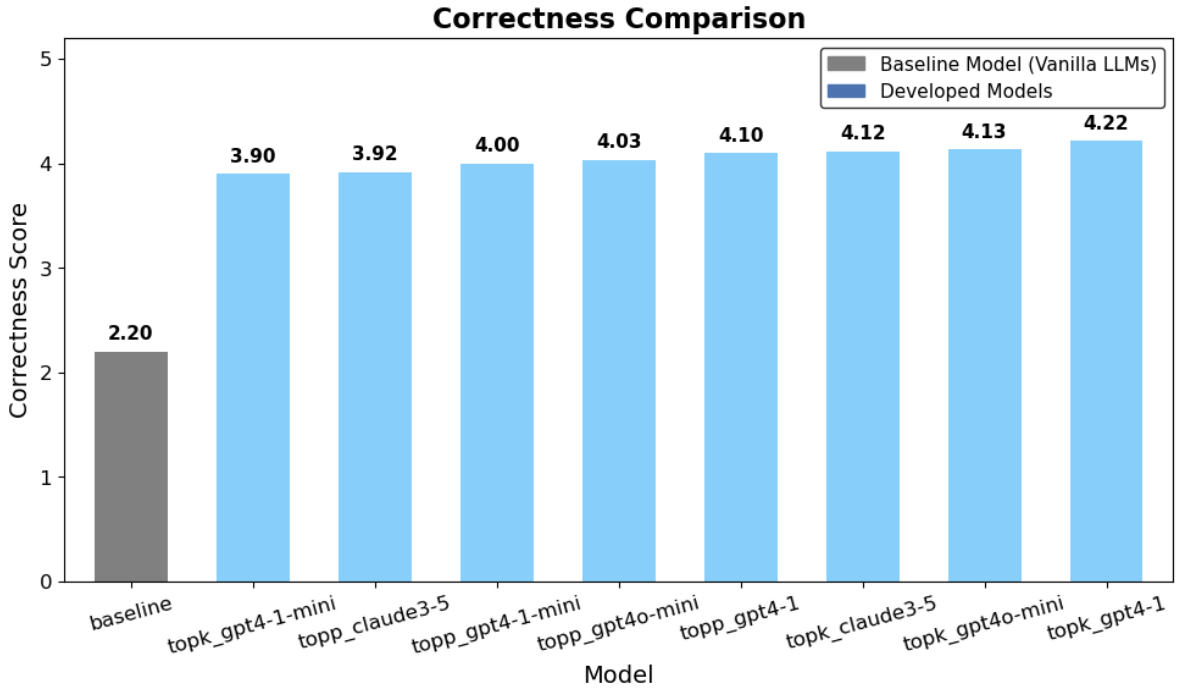

Figure 3: Correctness Comparison to Baseline

Then, we compared multiple model variants across all six evaluation metrics under the setting of k = 5 and p = 0.5, as shown in Figure 2. Overall, top-k configurations performed slightly better than their top-p counterparts. Among models using the same retriever, GPT-4.1 and GPT-4.1-mini showed slightly better performance than Claude 3.5 and GPT-4o-mini. We also observed that the variance across 10 runs was minimal, indicating stable and repeatable results.

Moreover, the differences between models were relatively small across all metrics. Taking both performance and cost into account, we recommend top-k = 5 with GPT-4o-mini as the default setting.

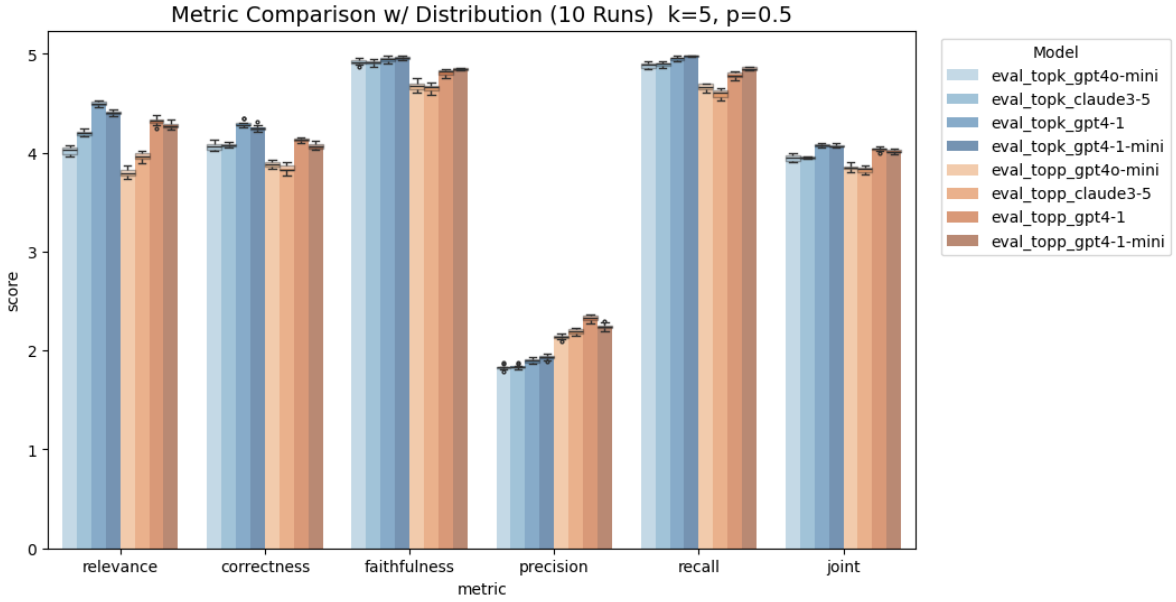

Figure 4: Metric Comparison with Distribution

Finally, we tuned the retriever parameters using our golden sample set. As shown in Figure 3, for top-k retrieval, k = 5 yielded the best correctness and joint scores, making it the recommended setting. Similarly, Figure 4 shows the performance of models using top-p retrievers across different p values. The optimal performance was observed in the 0.52–0.56 range, suggesting that future p tuning efforts should focus within this interval.

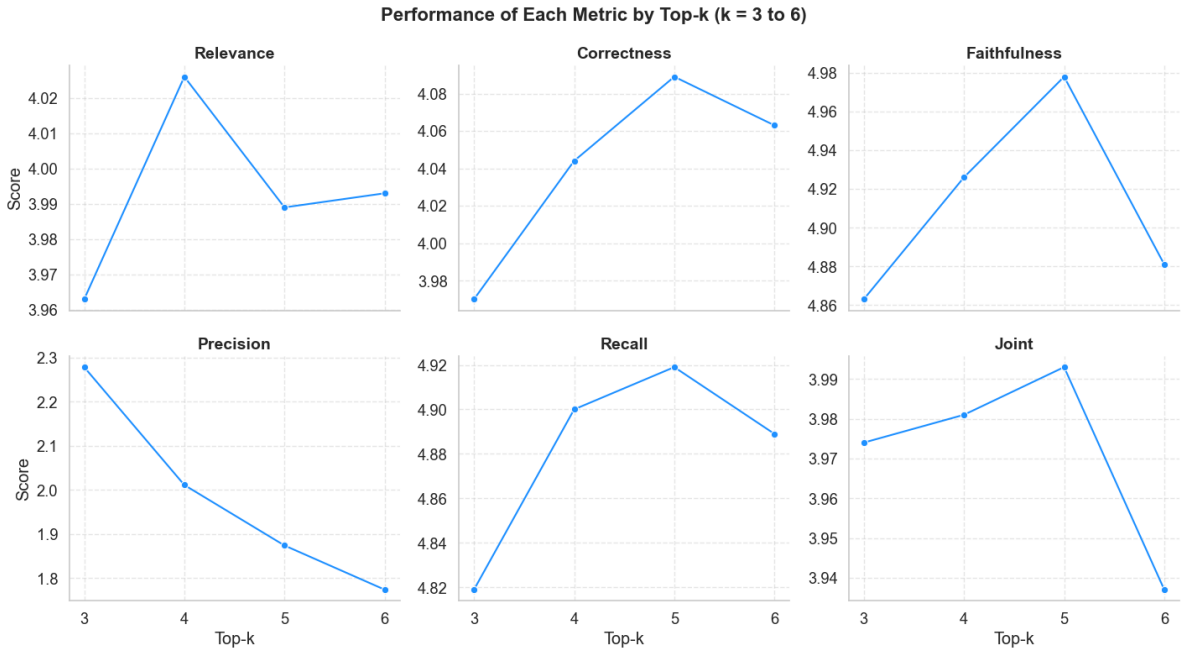

Figure 5: Model Performance Across Top-k Values (k = 3 - 6)

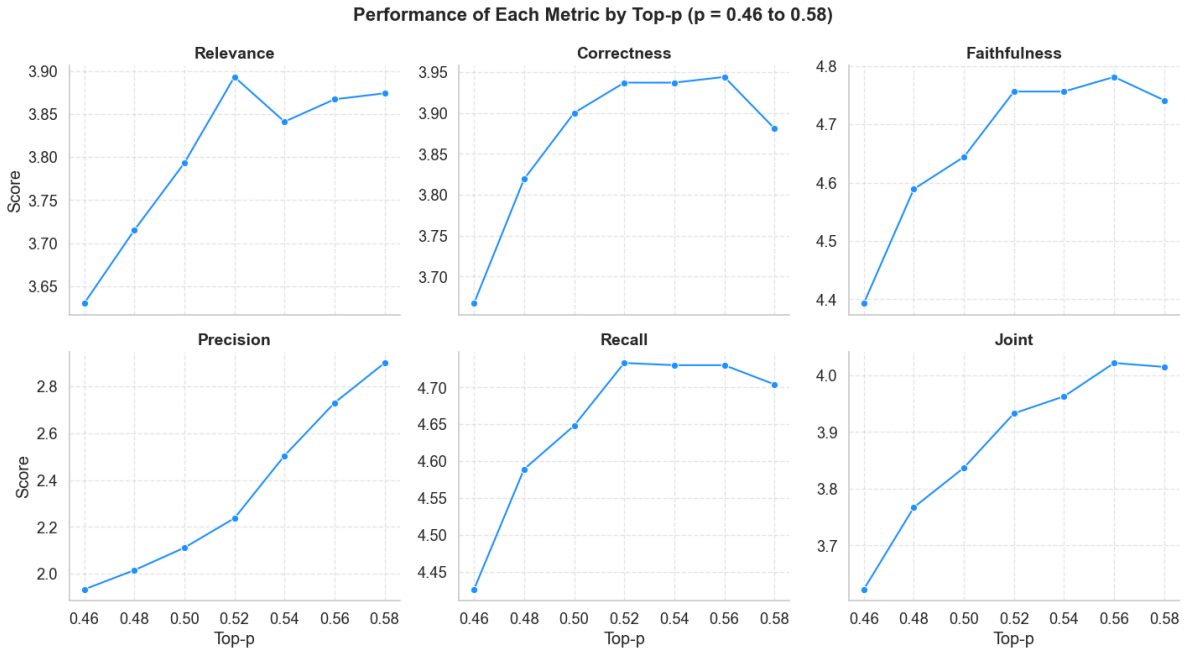

Figure 6: Model Performance Across Top-p Values (p = 0.46 - 0.58)

4.4 Future Improvements

The current release delivers practical business use. But as any project, new versions will be released with improved features and more flexibility.

Below are the improvements that can be developed in the next few releases by the partner:

- **Integration Testing**: so far our evaluation has been restricted to using the evaluator and our personal judgement. The first and most obvious next step would be for our capstone partner to test the data product in their standard pipeline with the sample data that they had provided
- If you visualize the heatmap of the how QA Pairs match up to interview guideline questions based on cosine similarity of the embeddings, you will see that it is perfectly linear (along the diagonal from top-left to bottom right) in the synthetic dataset in Figure 7. Generating realistic dialogue was not easy, and the "perfectly linear" nature of the similarity matching shows this. Nonetheless, the heatmap generated from one of the partner's actual datasets show that our pipeline is robust enough and still finds the required matches when the interview length and format varies (Figure 8). Despite this, with the real-world data, we still see the diagonal pattern as much as it is less pronounced. Another area of testing for robustness would be to process some **non-sequential interviews** whose flow does not follow the same general flow as guideline questions
- Still referring to Figure 8, we notice that in most interviews, there are some dialogue sections that have very low similarity matches to any of the guideline questions. A more precise snapshot is highlighted in Figure 9 for one interview, with such sections marked with dotted lines. To improve the current process, a **post-processing step** for the matches can be added. The first objective of this step would be to fill in the gaps to any previously un-answered guideline question. For example, in Figure 9 Guideline Question 2 (second column) had "[No relevant response found]" output due to the low similarity matches. However, in the transcript, this question was answered by the interview dialogue pairs 2 and 3 (with the respective row numbers), given that Guideline Question 1 is answered in the first exchange (row 1) and Guideline Question 3 is answered in row 4. After this post-processing, any remaining un-used dialogue portions might be "out of topic" exchanges (for example "exit pleasantries" like *"Thank you for making time to speak with me."* at the end [rows 16 & 17]) or they could be additional information that was not covered by the guideline questions but could be useful to extract. Our partner had set this as a stretch goal for the capstone project. Despite having limited time we had to implement it, our work has set a good stage for this objective to be achieved

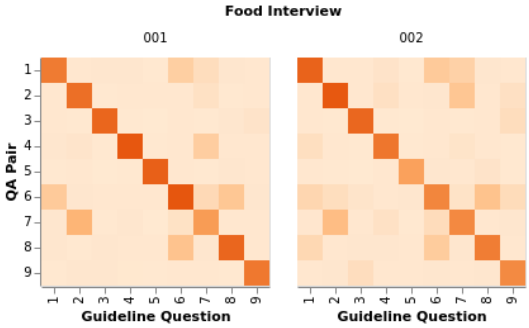

Figure 7: Heatmap with QA Pairs similarity to Guidelines (Synthetic Data)

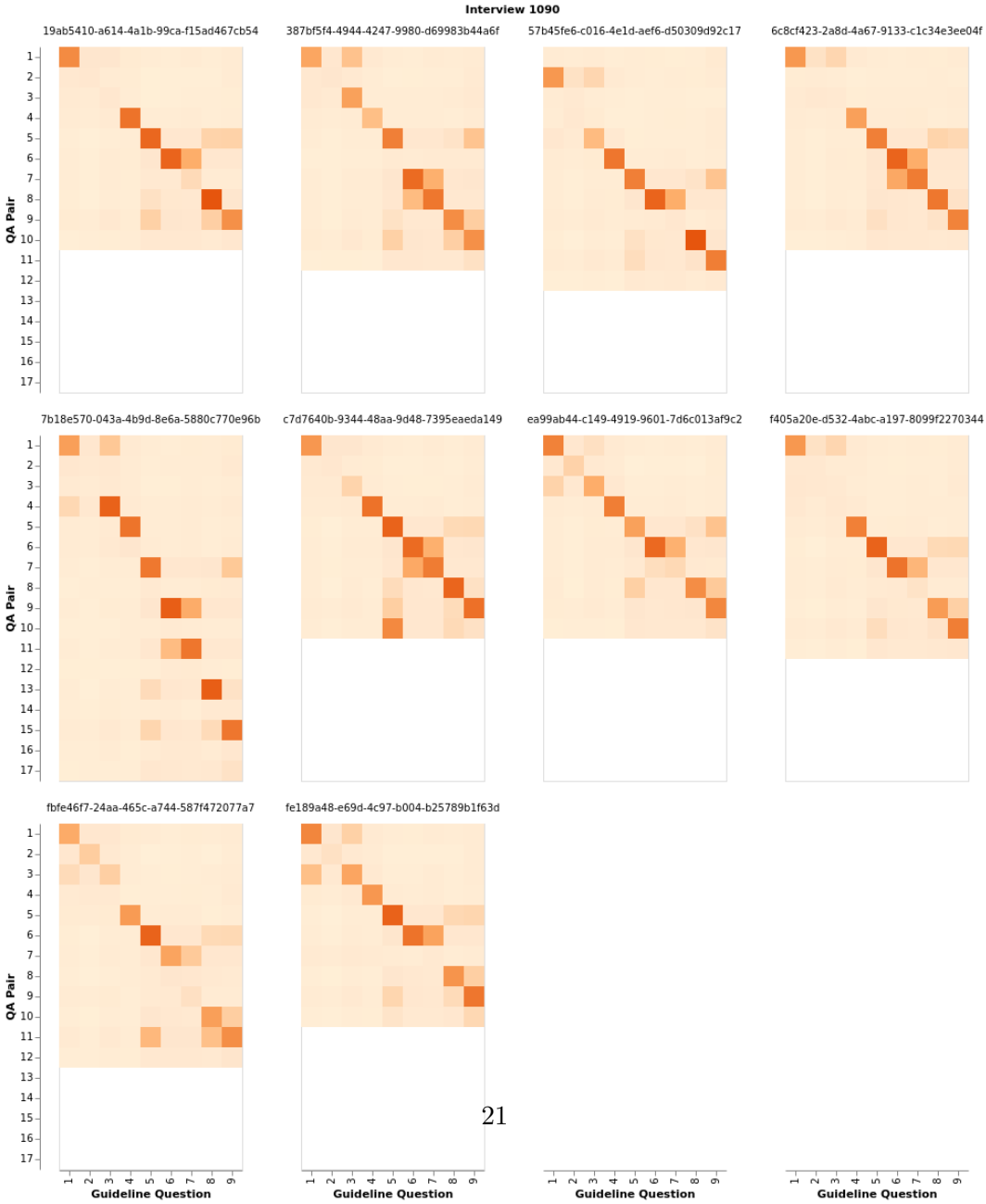

Figure 8: Heatmap with QA Pairs similarity to Guidelines (Real Data)

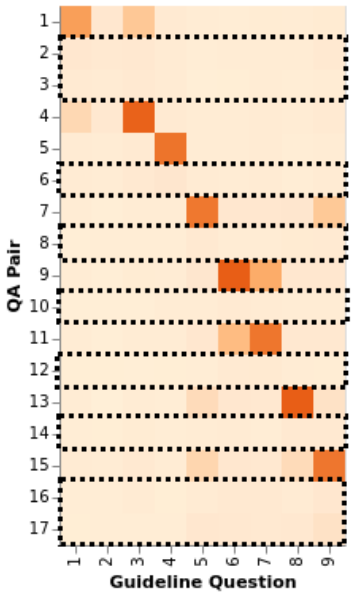

Figure 9: Heatmap showing Unused QA Pairs

5 Conclusion and Recommendations

In our capstone project, we designed and developed a functional pipeline that extracts concise summaries from interview transcripts and outputs structured data suitable for integration into our partner's downstream analysis workflows. Alongside this `processor` pipeline, we built an `evaluator` framework that enables our partner to assess the quality of extracted outputs. Both components are flexible and include hyperparameters that can be adjusted and customized.

The pipeline has been tested across a variety of scenarios and interview topics drawn from the synthetic data set in the public repository and the partner's private dataset (not included in the public repository), demonstrating its robustness and adaptability. We believe the methods and results presented here offer a strong foundation for future development by our partner, while already showing practical value for real-world use.

References

Anthropic. 2024. "Introducing Claude 3.5 Sonnet." https://www.anthropic.com/news/claude-3-5-sonnet.

Bachmann, Max. 2020. "Rapidfuzz: A Fast String Matching Library for Python."

DS-Daedalus Team. 2025. "DS-Daedalus V4: Increments and Updates." Internal Presentation Report. DS-Daedalus.

Es, S., J. James, L. Espinosa-Anke, and S. Schockaert. 2023. "RAGAS: Automated Evaluation of Retrieval Augmented Generation." https://doi.org/10.48550/arXiv.2309.15217.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." In *Advances in Neural Information Processing Systems*, 33:9459–74. https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

OpenAI. 2024. "Introducing GPT-4o Mini." https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

Reimers, Nils, and Iryna Gurevych. 2019. "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3982–92. Association for Computational Linguistics. https://aclanthology.org/D19-1410.