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 data 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)[1] 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 the 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 achieved a correct response identification score of 4.13, up from the previous baseline of 2.2, and streamlined the review process by automatically highlighting 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 question-and-answer surveys provide structured data that is relatively easy to analyze, open-ended formats such as interviews offer deeper, more nuanced insights, yet they are difficult to process at scale due to their unstructured and free-flowing nature.

Our capstone partner, Fathom, specializes in analyzing open-ended text data from surveys and interviews. Their platform already supports the analysis of structured survey responses, but scaling the analysis of interview transcripts has proven to be a key challenge. The interviews are conducted with a predetermined set of guideline questions in mind; however, the interviewer has the flexibility regarding when or whether to ask these questions. As a result, interview responses are less uniform and harder to map directly to the original guideline questions. Currently, Fathom's team relies solely on large language models (LLMs) and manual transcript review to extract relevant content. This process is labor-intensive, 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, one-on-one conversational interview transcripts, alongside more than 20 sets of guideline questions to which the responses must be mapped. These transcripts come from multiple current partners of Fathom, reflecting a real-world use case. 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 workflows.

2.3 Project Objectives

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

- **Automate the mapping of interview transcript content to original guideline questions**, reducing human effort and improving consistency. Fathom currently relies solely on LLMs for response extraction, followed by manual review to ensure quality. This approach is time-consuming and difficult to scale. Automation allows for faster, more consistent analysis and is scalable to handle increasing volumes of interview data without additional manual overhead.

- **Deliver an end-to-end pipeline** that integrates into Fathom's workflow for repeatable analysis of new interview data. For this solution to be impactful, it must fit within Fathom's 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.

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)[2] and Fathom's Daedalus v4[3], 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

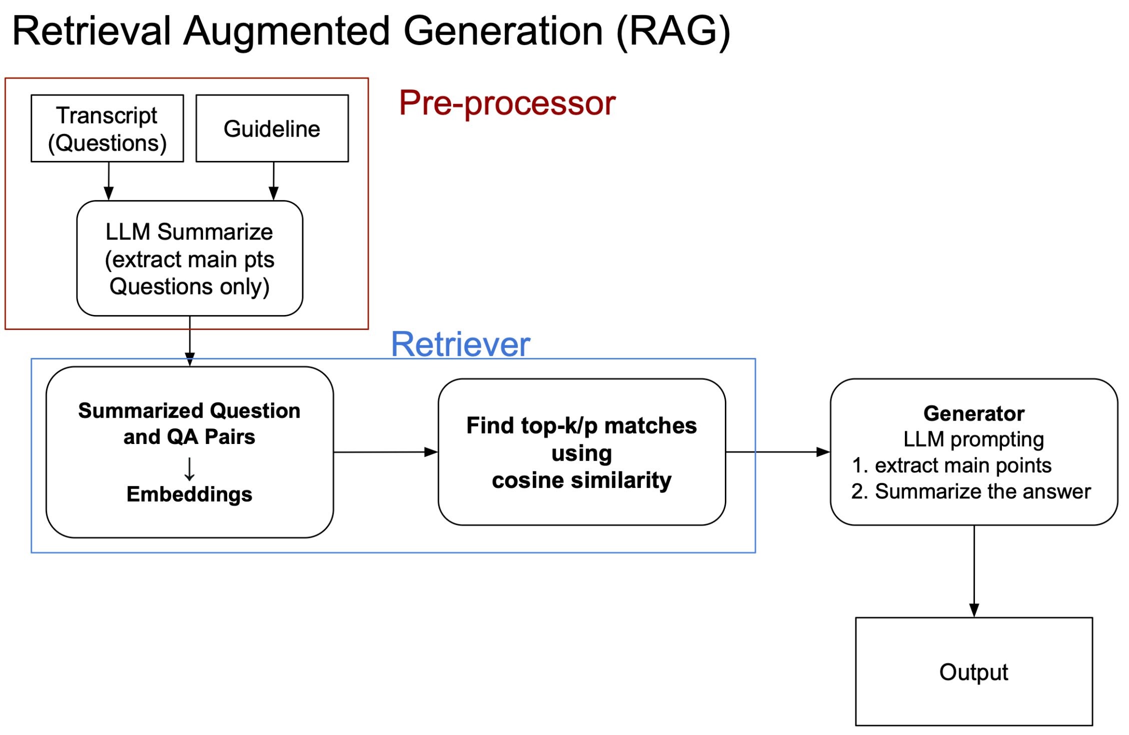
Retrieval Augmented Generation (RAG)
Pre-processor
Transcript (Questions)
Guideline
LLM Summarize (extract main pts Questions only)
Retriever
Summarized Question and QA Pairs
Embeddings
Find top-k/p matches using cosine similarity
Generator
LLM prompting
1. extract main points
2. Summarize the answer
Output

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[4] or Claude-3.5[5]) 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`[6]. 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`[7] to trace summarized answers back to their original transcript lines. This method accounts for minor 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 designed an evaluator to ensure that the answers are not only accurate in meaning, but also generated through a reliable and precise process. To support this, 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:

Prompt-level optimization: Tailored LLM prompts handle edge cases like vague answers. Embedding-based checks against confused templates ensure fairness when both prediction and ground truth lack specificity.

Integrated feedback generation: Each score includes LLM-generated feedback, which improves interpretability and helps users understand and verify the results.

Configurable scoring and thresholding: The evaluator supports customizable metric weights and threshold-based flagging of low-scoring responses, streamlining validation workflows and adapting to diverse evaluation requirements.

4 Data Product and Results

4.1 Data input and output

4.1.1 Data Input

The D2D processor 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.
- **Example**:
  - 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 labeled segments for interviewers and interviewees.
- **Format**: Plain text (`.txt`)
- **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

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.
- **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

The processor pipeline, as shown in Figure 1, 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.

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 Pros, Cons, and Justifications

Component	Pros	Cons	Justification
RAG	Combines LLM and retrieval for relevance; scales to large data	LLM-dependent; computationally complex	Outperforms standalone LLM or retrieval; scalable vs. manual methods
Embeddings	Captures semantic meaning; handles diverse text efficiently	Computationally expensive; model-dependent	Chosen over TF-IDF; avoids custom training using `SentenceTransformer`
Cosine Similarity	Normalized metric; efficient for high dimensions	Threshold tuning needed; precision-sensitive	Better than Euclidean distance; enables automated alignment
Referencing	Enables traceability; improves interpretability	Risk of loose matches; threshold tuning needed	More flexible than exact match; `rapidfuzz` selected for speed and integration

Table 2: Summary of Pros, Cons, and Justifications for Core Techniques

4.2.5 Potential Improvements and Challenges

Improvement	Challenges	Why Not Implemented
Adaptive Thresholding	Requires extensive testing across diverse data	Fixed thresholds sufficed for current scope
Fine-Tuned Embeddings	Needs labeled data and high computational power	Pre-trained models met project needs within constraints

Table 3: Unimplemented Improvements and Justifications

4.2.6 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

```python
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 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.

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

According to Figure 2, 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.

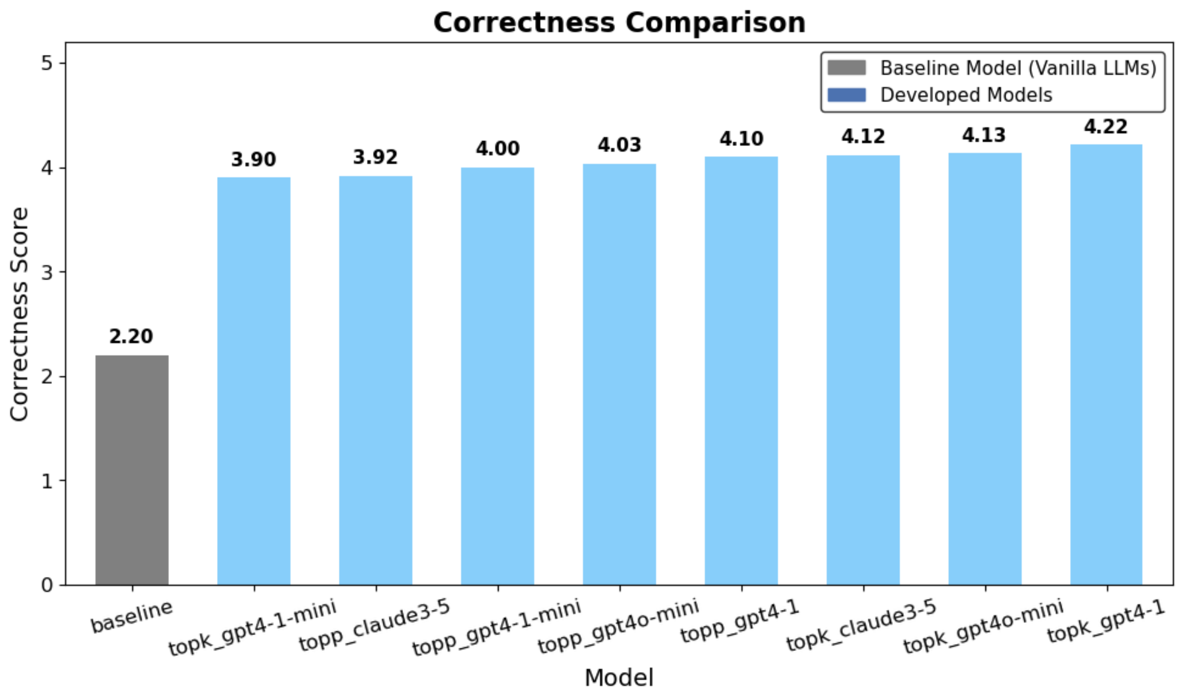

Figure 2: Correctness Comparison to Baseline

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 3. 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. The differences across models were relatively small, and the variance across 10 runs was minimal, indicating stable and repeatable results.

While precision scores were generally lower than other metrics, this is not a concern in our setup. Since the goal of retrieval is to capture all relevant context for generation, some irrelevant chunks can be tolerated as long as key information is included. Hence, recall plays a more critical role in ensuring response quality.

Taking both performance and cost into account, we recommend using GPT-4o-mini with top-k = 5 as the default configuration.

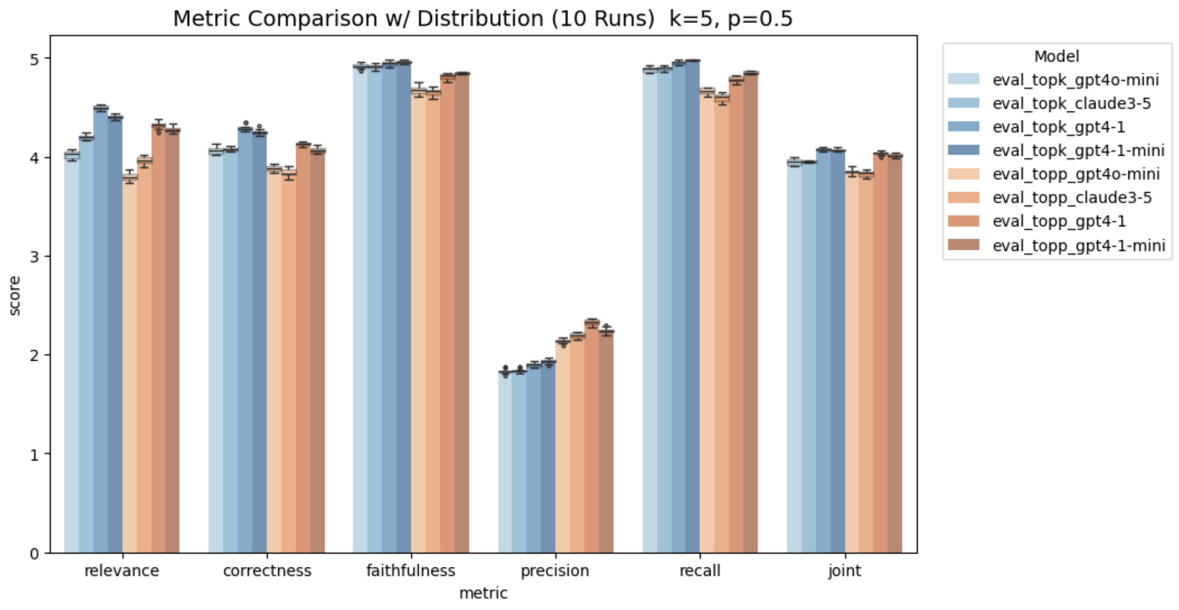

Figure 3: Metric Comparison with Distribution

Finally, we tuned the retriever parameters using our golden sample set. As shown in Figure 4, for top-k retrieval, k = 5 yielded the best correctness and joint scores, making it the recommended setting. Similarly, Figure 5 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.

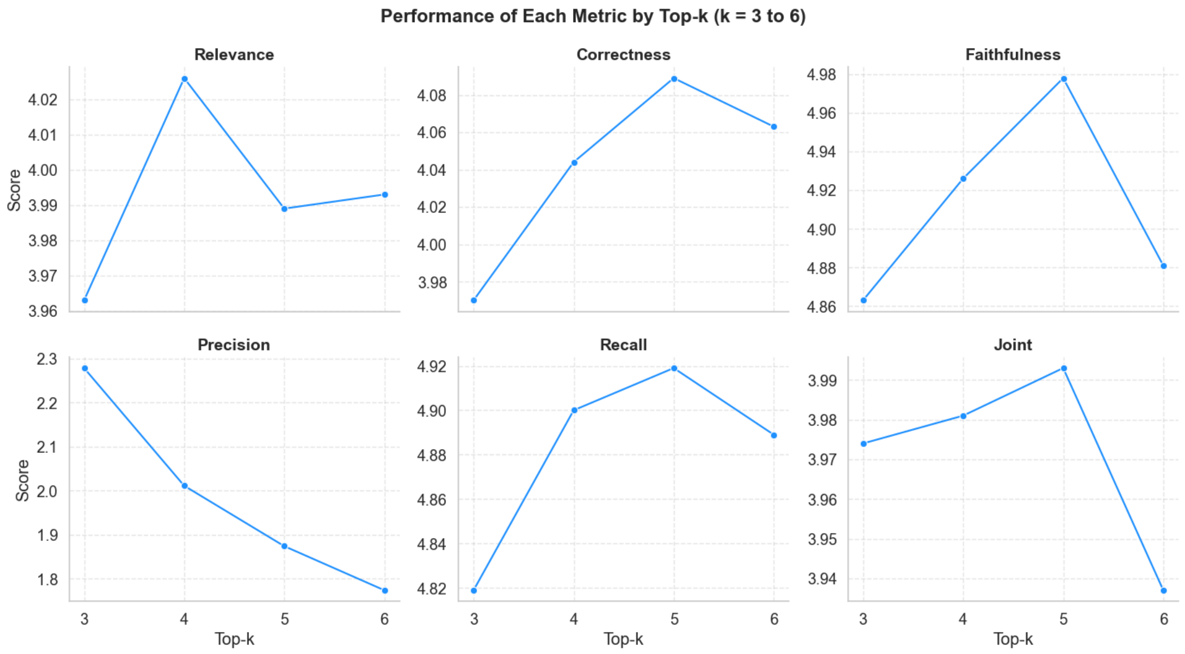

Figure 4: Model Performance Across Top-k Values (k = 3 - 6)

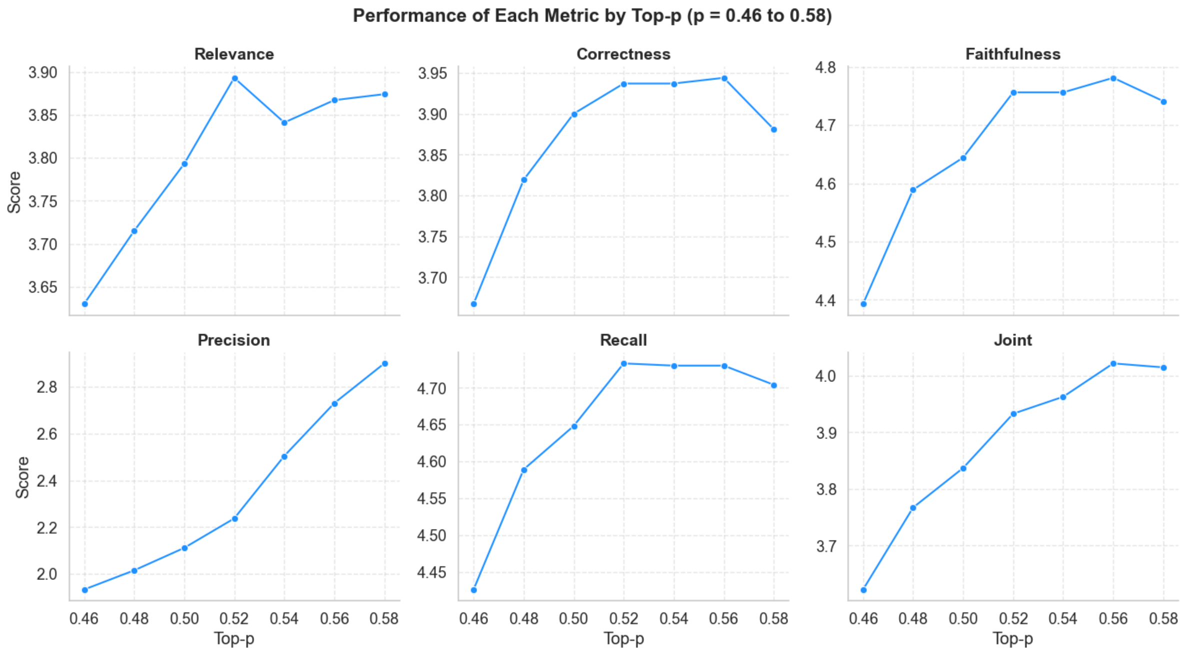

Figure 5: Model Performance Across Top-p Values (p = 0.46 - 0.58)

5 Recomendations and Future Improvements

The current release provides practical business value. Like any project, enhanced features and greater flexibility are always a possibility. Below are suggested improvements:

- **Integration Testing**: Our evaluation was restricted to running the evaluator and using our personal judgement on the results. Testing the pipeline end-to-end with varied datasets and expert evaluators would help ascertain how well the model generalizes.
- Visualizing a heatmap of the how Question-Answer Pairs match up to interview guideline questions based on pairwise cosine similarity, reveals a perfectly linear pattern (along the diagonal from top-left to bottom right) for synthetic datasets in Figure 6 (*Dialogue questions are plotted on the Y-axis, first question at the top, against guideline questions on the X-axis. A darker shade in the cells represents high cosine similarity between interview and the corresponding guideline question.*). The perfect diagonal plot illustrates the difficulty of simulating noisy dialogue. Synthetic dialogues follow the guidelines questions perfectly/robotically. Nonetheless, the heatmap generated from the partner's real-life datasets show that our pipeline is robust enough to find the expected matches (Figure 7). However, even real-world interviews still exhibit the diagonal pattern albeit less pronounced. We suggesting testing the pipeline with **non-sequential interviews** that do not follow the sequence of guideline questions.
- Figure 7 also shows dialogue sections where some interview questions have very low cosine similarity to all guideline questions. Figure 8 zooms into one interview, with such sections highlighted with dotted lines. We propose the addition of **post-processing step**. First, this step would fill in the gaps to any previously un-answered guideline questions. For example, in Figure 8 Guideline Question 2 (second column) had "[No relevant response found]" output due to the low cosine similarity scores. However, in the transcript, this question was answered by the dialogue exchange 2 and 3, that are between guideline question 1 (retrieved from row 1) and 3 (retrieved from row 4). After this post-processing, any remaining un-used dialogue portions might be out of topic exchanges (like greetings/pleasantries) or they could be additional useful information that was not covered by the guideline questions. This requirement was a stretch goal for the project. Despite having limited time to implement it, our work has set a basis for this objective to be achieved through the **post-processing step**

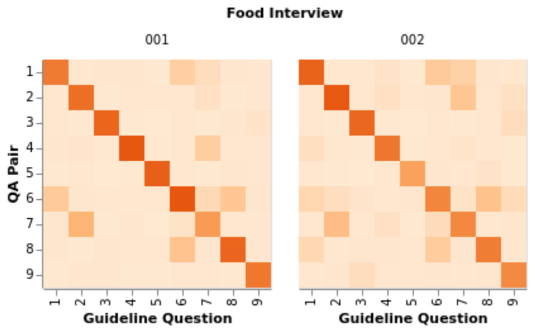

Figure 6: Heatmap showing Cosine Similarity Question-Answer Pairs vs Synthetic Data Guidelines

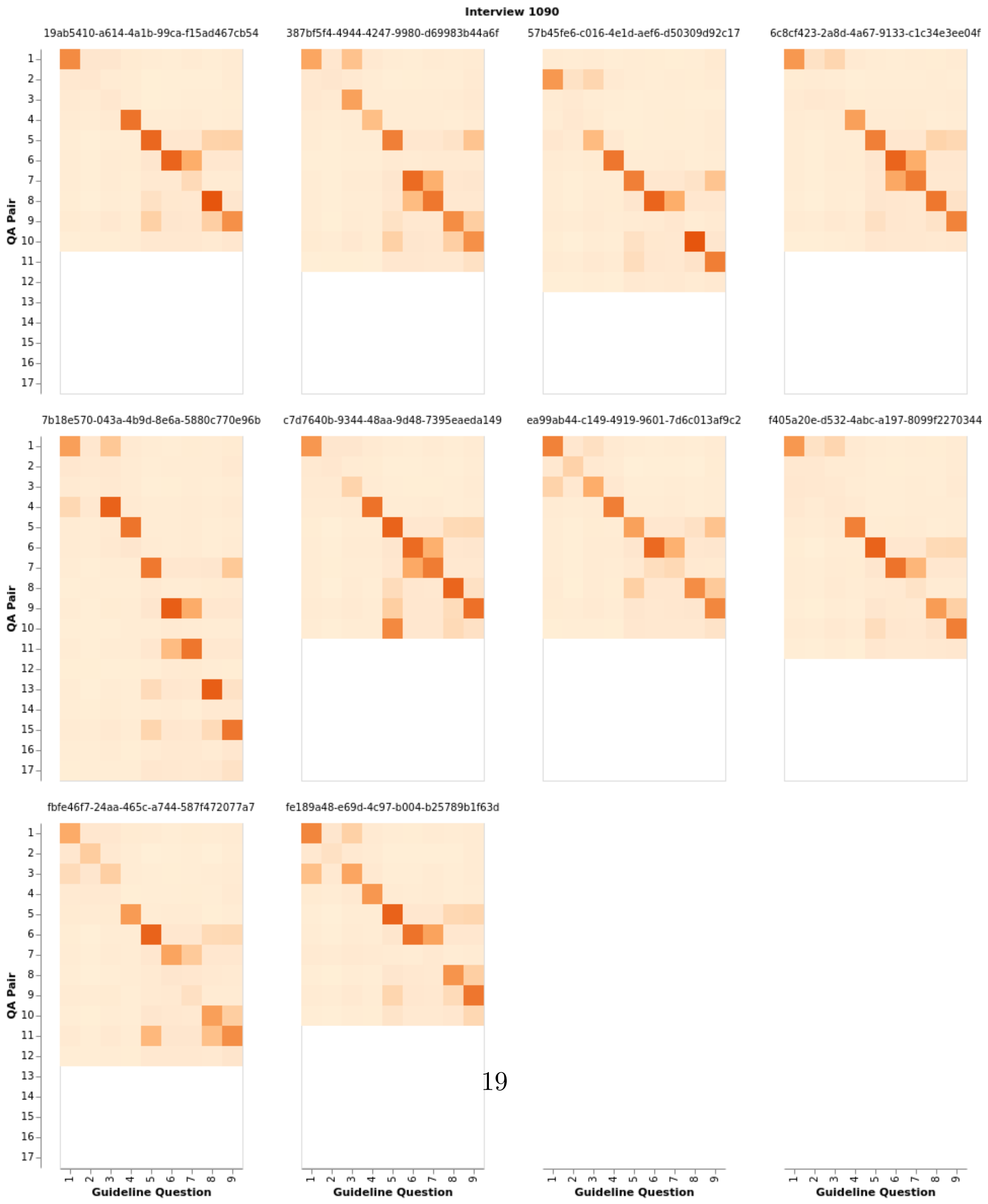

Figure 7: Heatmap showing Cosine Similarity Question-Answer Pairs vs Real-life Data Guidelines

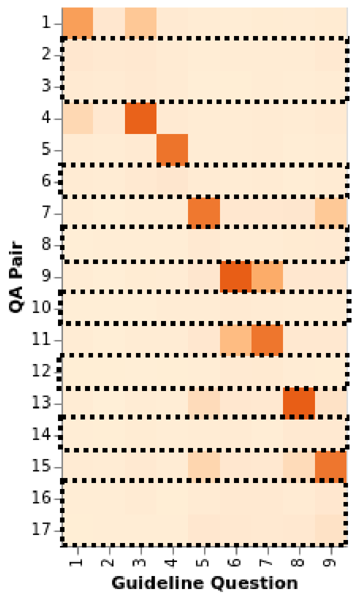

Figure 8: Heatmap showing Unused Question-Answer Pairs

6 Conclusion

We developed a pipeline that extracts concise summaries from interview transcripts and outputs structured data suitable for integration into our partner's downstream analytical workflows. Alongside this `processor` pipeline, we built an `evaluator` framework that enables quality assessment of the extracted outputs. Both components are flexible and customizable. They have been tested on a variety of interview formats and topics, demonstrating robustness and adaptability. The methods and results presented here offer a strong foundation for future development, while already providing practical value for real-world use.

References

[1] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in neural information processing systems*, 2020, pp. 9459–9474. Available: https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[2] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “RAGAS: Automated evaluation of retrieval augmented generation.” 2023. Available: https://doi.org/10.48550/arXiv.2309.15217

[3] DS-Daedalus Team, “DS-daedalus v4: Increments and updates,” DS-Daedalus, Internal Presentation Report, 2025.

[4] OpenAI, “Introducing GPT-4o mini.” 2024. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[5] Anthropic, “Introducing claude 3.5 sonnet.” 2024. Available: https://www.anthropic.com/news/claude-3-5-sonnet

[6] N. Reimers and I. Gurevych, “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)*, Association for Computational Linguistics, 2019, pp. 3982–3992. Available: https://aclanthology.org/D19-1410

[7] M. Bachmann, “Rapidfuzz: A fast string matching library for python.” 2020.