Assignment 2 Report: End-to-end NLP System Building

11-711 Advanced Natural Language Processing

Sean Xu
Heinz College
MSPPM-DA '25
xiaoxu@andrew.cmu.edu

Natalie Yang
Heinz College
MSPPM-DA '25
xiaoton3@andrew.cmu.edu

Lexa Zhong
Heinz College
MSPPM-DA '25
huiruzho@andrew.cmu.edu

Abstract

We developed a retrieval augmented generation system (RAG) from scratch that answers questions about Carnegie Mellon University (CMU) and Pittsburgh, including history, culture, trivia, and upcoming events. To enhance model performance, we introduced a novel text-splitting strategy that sequentially performs the semantic chunking and character-based splitting, effectively balancing semantic context with reasonable chunk length. Additionally, we integrated large language models (LLMs) into the data annotation workflow, leveraging them to generate initial annotations to explore the capabilities of LLMs further. Through experimentation with our proposed components and fine-tuning of various hyperparameters, the evaluation metrics, including answer recall, exact match and F1 score have shown statistically significance improvements.

1 Data Creation

1.1 Building Knowledge Resources

To build a comprehensive knowledge resource for the RAG system, we compiled documents from publicly available sources relevant to Pittsburgh and Carnegie Mellon University (CMU). In the following, we describe the sources of data and the criteria used for document selection.

Data Sources We gathered data from various publicly available resources to ensure broad coverage of topics related to Pittsburgh and CMU. The primary sources include:

1. General Information and History:
   - Wikipedia pages
   - City of Pittsburgh official website
   - Encyclopedia Britannica page on Pittsburgh.
   - Visit Pittsburgh website
2. Government and Regulations:
   - City of Pittsburgh Tax Regulations
   - City of Pittsburgh 2024 Operating Budget.
3. About CMU and CMU History:
   - Official CMU website
   - CMU History page
4. Events in Pittsburgh and CMU:
   - Pittsburgh events calendar: Month-specific pages for easier scraping.
5. Downtown Pittsburgh events calendar:
   - Pittsburgh City Paper events.
   - CMU events calendar and campus events page.
6. Music and Culture:
   - Pittsburgh Symphony, Opera, and Cultural Trust websites.
   - Pittsburgh Museums: Carnegie Museums, Heinz History Center, The Frick, and others.
7. Food-Related Events:
   - Food Festivals: Picklesburgh, Pittsburgh Taco Fest, Pittsburgh Restaurant Week, Little Italy Days, Banana Split Fest.
8. Sports:
   - Sports API for game schedules, transaction, roster and injuries.
   - Pittsburgh Pirates, Steelers, and Penguins official websites.

Document Selection To ensure that the knowledge resource is comprehensive and relevant, we used the following criteria for document selection:

- Relevance to Pittsburgh and CMU: We prioritized documents that provide information on Pittsburgh's history, culture, events, and regulations, as well as CMU's history and activities. In addition, subpages such as "about," "history," "schedule," "upcoming events," and "vendors" were included to capture detailed information.
- Diversity of Topics: We selected documents covering various topics, including general information, events, music, culture, food, and sports, to ensure the RAG system can answer diverse questions.
- Recurring and Upcoming Events: For events, we focused on annual/recurring events and events happening after March 19th, as specified in the requirements.

- Publicly Available and Reliable Sources: We only included documents from publicly available and reliable sources, such as official city websites, CMU's official website, and well-known event calendars.
- Ease of Access and Scraping: We prioritized sources that are easy to access and scrape, such as Wikipedia pages and official websites with clear navigation structures.
- Exclusion of Irrelevant Data: We excluded documents irrelevant to the task, such as cookies, events before 2024, or highly specialized subpages (e.g., that direct to websites unrelated to Pittsburgh or CMU).

1.2 Raw Data Extraction

Tools We utilized the following tools for data extraction and processing:

- BeautifulSoup4: For parsing and extracting content from HTML pages. BeautifulSoup allowed us to navigate the DOM structure of web pages and extract relevant text, titles, and metadata.
- Selenium: For dynamic web scraping, particularly for websites that required interaction (e.g., clicking buttons or loading content dynamically). Selenium enabled us to automate browser actions and extract data from JavaScript-rendered pages.

Data Format The extracted data was stored in a separate `.txt` file with file names reflecting the data sources and topics.

1.3 Annotation for Testing and Training

1.3.1 Data Annotation Type and Method

For both train and test sets, we automatically generated question-answer pairs and validated their quality through human inspection. Human annotation focused on refining answers into concise imperative sentences or phrases. Moreover, human annotators adjusted questions for specificity. For example, if the answer was a year, the question would start with "In which year."

1.3.2 Annotation Volume

To determine the appropriate volume of annotated data for testing and training, we performed a power analysis based on the F1 score. This ensured that our data set size was sufficient to detect meaningful improvements in model performance. Note that we included Exact Match(EM) as one of our evaluation metrics for model performance, but here we used F1 score to perform power analysis.

Baseline Performance We evaluated the baseline model with default parameters on the dev set to establish a performance benchmark. The baseline model has an F1 score of 0.44. This score reflects the system's performance without any task-specific fine-tuning or advanced techniques.

Final Model Performance To make proper assumption of Minimum Detectable Effect(MDE), we also evaluated one of the final models on the dev set to estimate the expected improvement. The improved F1 Ssore is 0.60.

Minimum Detectable Effect (MDE) Based on the observed improvement from the final model, we defined the Minimum Detectable Effect (MDE) as:

- MDE for F1 Score: 0.16 (a 0.16 improvement over the baseline F1 of 0.44).

This threshold was chosen to ensure that any observed improvements would be practically significant for a question-answering system.

Power Analysis Parameters

- **Statistical Power**: We aimed for a power of 0.80, which is a standard threshold to ensure a high probability of detecting the MDE if it exists.
- **Significance Level (α)**: We set α to 0.05, a common threshold for statistical significance.
- **Effect Size**: The effect size was calculated based on the MDE and baseline performance.

Sample Size Calculation To determine the required sample size, we used the following formula for comparing two proportions:

$$n = \frac{(Z_{\alpha/2} + Z_{\beta})^2 \cdot (p_1 \cdot (1 - p_1) + p_2 \cdot (1 - p_2))}{(p_1 - p_2)^2}$$

Where:

- p_1: Baseline F1 score = 0.44.
- p_2: Target F1 score after improvement = 0.60.
- $Z_{\alpha/2}$: Critical value for $\alpha = 0.05$ (two-tailed) = 1.96.
- Z_{β}: Critical value for power = 0.80 = 0.84.

Results The calculated sample size required to detect a 10% improvement in F1 score with 80% power and a 5% significance level is approximately 152 question-answer pairs for test set.
To ensure robustness, we rounded up and decided to auto-generate 300 question answer pairs. After human inspection, we finished with **257 question-answer pairs** for the test set. This sample size provides sufficient statistical power to detect the desired improvements for F1.

1.4 Inter-annotator Agreement (IAA)

1.4.1 Annotators

To evaluate data annotation quality, we measured inter-annotator agreement (IAA) with two team members.

- Annotator A: automatic generation of data with human quality check (team member 1)
- Annotator B: human annotation (team member 2)

1.4.2 Sample Questions

The data set used for this analysis consists of 50 randomly selected questions from the test set. The questions and their corresponding annotations are stored in the file `IAA_annotation.json`.

1.4.3 Agreement Criteria

To evaluate the consistency between the annotators, two levels of agreement were defined:

Strict Agreement An exact string match between the annotations of the two annotators is required. For example:

- `"Charlotte, Pittsburgh, San Jose, Tallahassee, Tempe, Seattle"`
  vs.
  `"Charlotte, Pittsburgh, San Jose, Tallahassee, Tempe, Seattle area"`
  Agree

Lenient Agreement Semantic equivalence is considered, allowing for minor variations in formatting or phrasing. For example:

- `"22 Market Square in Pittsburgh"`
  vs.
  `"22 Market Square, Pittsburgh, PA"`
  Agree

1.4.4 Evaluation Metric

The primary metric used for evaluation is **Percentage Agreement**, calculated separately for strict and lenient agreement criteria.

1.4.5 Report Findings

Agreement Metrics

- Strict Agreement: 13 out of 50 (26%)
- Lenient Agreement: 45 out of 50 (90%)

Key Disagreement Patterns The analysis revealed several common patterns of disagreement:

- Typos and Terminology: For example, "UMPC"vs."UPMC" (Q19) and the "zoning.board" typo (Q34).
- Organization Name Variation: Differences in naming conventions, such as "Office of Student Activities"vs."Student Activities Office" (Q9).
- Full Name vs. Partial: For example, "Stephen Foster"vs."Stephen Collins Foster" (Q24).
- Inclusion/Exclusion of Details: Some annotations included additional context (e.g., museum details in Q26), while others did not.

1.4.6 Conclusion

Overall, the high lenient agreement rate suggests that our annotation approach(automatic generation with human check) yields semantically robust results, but stricter adherence to formatting and naming conventions is necessary to improve exact match consistency.

2 Model Details

2.1 RAG Architecture

Chunking The data processing steps in an RAG pipeline involve loading the collected raw documents, dividing them into chunks, and then feeding them into an embedding model to obtain embedding vectors. Choosing appropriate chunking strategies is crucial to prevent long-sequence issues from affecting LLM performance. A common text splitter divides documents at the sentence or paragraph level solely based on character count. Although character-based splitters ignore the semantic context, our baseline used one variant, `RecursiveCharacterTextSplitter`[1], due to its popularity.

A novel splitter is `SemanticChunker`[2] powered by `langchain_experimental`, which splits documents based on their semantic similarity. However,

[1] https://python.langchain.com/api_reference/text_splitters/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html

[2] https://python.langchain.com/api_reference/experimental/text_splitter/langchain_experimental.text_splitter.SemanticChunker.html

SemanticChunker may produce excessively long chunks, such as a 10000-character chunk in our experiments.

To address this, we combined two methods: first applying SemanticChunker and then subdividing the semantic chunks using the RecursiveCharacterTextSplitter. Leveraging the benefits of both methods, this hybrid semantic chunking is one of the configurations for model improvement.

Vector Store The vector embeddings, as well as the embedding model, are stored in a vector store for further querying. This vectorization of the knowledge base is an essential component for keeping LLMs up-to-date in an efficient fashion. To facilitate downstream ablation studies, we used the InMemoryVectorStore[3] class for repeated use, thereby avoiding re-creation with each experiment.

Retriever A retriever searches the vector store to identify relevant chunks based on certain search criteria. The retrieved chunks containing essential context can guide LLMs in generating relevant and non-hallucinated responses. The baseline model used cosine similiary as the search metric. To potentially enhance the retriever's performance, we also consider with the maximal marginal relevance (mmr),[4] which doesn't only consider similarity but also diversity.

QA Generator Finally, an LLM generates responses based on a prompt that includes both the question and retrieved context. We utilized transformer.pipeline[5] to integrate the LLM to perform "text2text-generation" task.

Additionally, we have referred to the prompt template, retrieval-qa-chat, from landchain-ai community.[6] This template is concise and well-structured, effectively integrating the question and context to enhance response generation, which is as follows.

```
System: Answer user questions based
solely on the context below:
<context>
{context}
</context>
User: {question}
```

Few-shot Learning Building on the components discussed above, we also introduced few-shot learning. We hypothesized that, as seen in other LLM applications, providing examples would help the RAG model better align with the task by adapting information from the examples, such as writing styles, expected length, etc.

2.2 Baseline Model

The baseline configuration is as follows:

Chunking
chunker: RecursiveCharacterTextSplitter
chunk_size: 1000
chunk_overlap: 100
Retriever
embedding_model: "all-mpnet-base-v2"[7]
search_type: "similarity"
search_k: 1
QA Generator
task: "text2text-generation"
generator_model: "flan-t5-large"[8]
max_new_tokens: 50
temperature: 0.01
top_p: 0.95
repetition_penalty: 1.2
Few-shot Learning
False

2.3 Ablation Experiments

As we primarily introduced the hybrid chunking strategy, our ablation experiments aimed to find the optimal *chunking* and *retriever* settings while keeping the *QA Generator* configuration consistent with the baseline. As the vector embeddings are vital for querying, we chose another sentence transformer model, GIST-large-Embedding-v0,[9] for comparison. Table 1 presents the configurations for *chunking*, *retriever*, *few-shot learning* across different variants and the baseline.

[3] https://python.langchain.com/api_reference/core/vectorstores/langchain_core.vectorstores.in_memory.InMemoryVectorStore.html

[4] https://python.langchain.com/docs/how_to/example_selectors_mmr/

[5] https://huggingface.co/docs/transformers/en/main_classes/pipelines

[6] https://smith.langchain.com/hub/langchain-ai/retrieval-qa-chat

[7] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[8] https://huggingface.co/google/flan-t5-large

[9] https://huggingface.co/avsolatorio/GIST-large-Embedding-v0

variant	chunker	chunk_size	chunk_overlap	embedding_model	search_type	search_k	few_shot
baseline	Recursive	1000	100	all-mpnet-base-v2	similarity	1	False
1	Hybird	500	50	all-mpnet-base-v2	similarity	5	False
2	Hybird	500	100	all-mpnet-base-v2	similarity	5	False
3	Hybird	500	200	all-mpnet-base-v2	similarity	5	False
4	Hybird	1000	50	all-mpnet-base-v2	similarity	5	False
5	Hybird	1000	100	all-mpnet-base-v2	similarity	5	False
6	Hybird	500	50	GIST-large-Embedding-v0	similarity	5	False
7	Hybird	1000	100	GIST-large-Embedding-v0	similarity	5	False
8	Hybird	500	100	GIST-large-Embedding-v0	similarity	5	False
9	Hybird	500	100	GIST-large-Embedding-v0	similarity	5	True

Table 1: Configurations of Ablation Experiments

3 Results

As Table 2 shows, we tested nine variants and found that variant 9 outperformed both the other variants and the baseline. Therefore, we selected this variant to generate output 1. Additionally, we ran two slightly modified versions of variant 9 to produce the system outputs 2 and 3 for submission. Table 3 outlines the differences, assuming all unspecified configurations remain the same as in variant 9.

variant	extract match	answer recall	macro F1
baseline	0.2257	0.4826	0.4716
1	0.2840	0.5527	0.5506
2	0.2646	0.5344	0.5372
3	0.2646	0.5353	0.5396
4	0.2724	0.5505	0.5403
5	0.3074	0.5871	0.5826
6	0.2918	0.5625	0.5774
7	0.2957	0.6088	0.5906
8	0.2879	0.5897	0.6001
9	**0.3113**	**0.6050**	**0.6120**

Table 2: Performance of Ablation Experiments

variant	search_metric	few_shot
9 (system output 1)	similarity	True
10 (system output 2)	mmr	True
11 (system output 3)	mmr	False

Table 3: System Output Configurations Based on Variant 9

4 Analysis

4.1 Performance Distribution

The model's performance varies significantly across different question types. Below is the distribution of key metrics of our best model:

Figure 1 and Figure 2 represent the distribution of F1 score and recall along with overall performance. The bin distribution indicates that answers are often either highly accurate or incorrect. A higher proportion of correct answers shifts overall recall and F1 scores to the right.

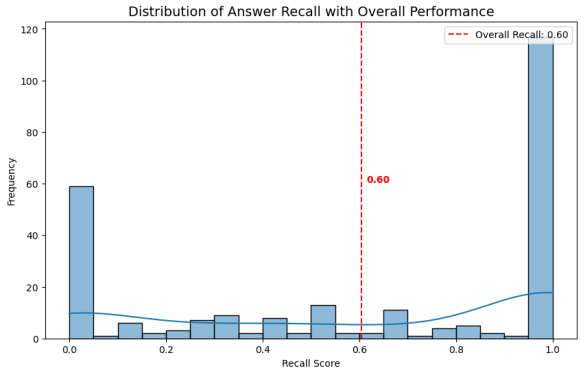

Figure 1: Distribution of Answer Recall with Overall Performance

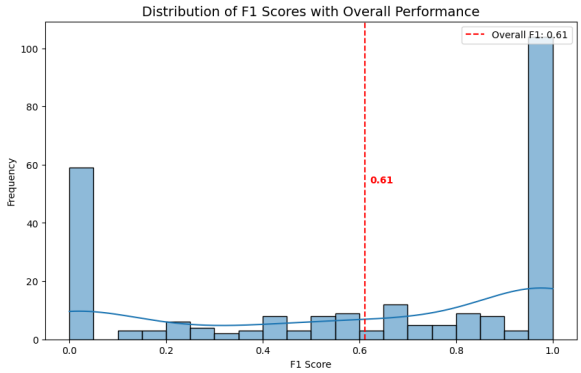

Figure 2: Distribution of F1 Scores with Overall Performance

4.2 Quantitative Breakdown

To perform a fine-grained comparison of the model's performance across various types of questions, we can break down the results based on the

complexity and nature of the questions. Here's a detailed analysis:

Best Performing Cases: f1 = 1.0

- **Simple, Fact-Based Questions**
  Example Questions:
  - "Who organizes the Ph.D. Student Support Group at Carnegie Mellon University?"
  - "Who organizes the Virtual Gratitude Gathering?"

  = Model Performance:
  - The model performs exceptionally well on these types of questions, achieving perfect F1 and recall scores (1.0). answers are straightforward and typically involve a single, well-defined entity or name.

Worst Performing Cases: f1 = 0.0

- **Questions Requiring Semantic Understanding or Detailed Information**
  Example Questions:
  - "In which century does the Carnegie Museum of Art have notable collections of aluminum relics and chairs?"
  - "Who inhabited in the Pittsburgh area before European settlers?"

  Model Performance:
  - The model struggles with these types of questions. For instance: For the question about the Carnegie Museum of Art, the model incorrectly answered "1930" instead of "The present century."
  - For the question about inhabitants of the Pittsburgh area, the model provided a generic answer ("Native Americans") instead of the specific tribes ("Seneca; Shawnee; Mingo").

- **Questions Requiring Multiple Answers**
  Example Questions:
  - "Who inhabited in the Pittsburgh area before European settlers?"

  Model Performance:
  - The model failed to list all the correct answers (e.g., "Seneca; Shawnee; Mingo") and instead provided a single, generic answer ("Native Americans").

- **Temporal or Contextual Questions**
  Example Questions:
  - "In which century does the Carnegie Museum of Art have notable collections of aluminum relics and chairs?"

  Model Performance:
  - The model incorrectly identified the century as "1930" instead of recognizing that the collections are relevant to "The present century."

4.3 RAG vs. Closed-book Model

We performed an analysis comparing the effectiveness of the Retrieve-and-Augment (RAG) strategy with the closed-book generation strategy using the `google/flan-t5-large` model. The analysis aims to highlight the differences in performance based on whether the model relies solely on its pretrained knowledge (closed-book) or if it leverages external retrieved information (RAG).

- **Closed-Book Generation**: In this setup, the model generates answers without any external context or knowledge. It relies solely on the information it learned during training.
- **Retrieve-and-Augment (RAG) Generation**: In this setup, the model retrieves relevant information from a knowledge base before generating an answer. The external information is incorporated into the model's generation process.

We tested both strategies on the following question:

Question: *What ballet performance will be at the Byham Theater on May 7, 2025?*

Results

- **Closed-Book Generation (flan-t5-large)**: The model generated the response, *"A free copy of the paper"*. This answer was incorrect and did not reflect the specific event happening at the Byham Theater. Therefore, the closed-book model was unable to provide an accurate answer.

- **Retrieve-and-Augment (RAG) Generation**: With RAG, the model retrieved relevant context

about the performance at the Byham Theater, and correctly generated the answer, *"Tote bag"*. This answer matched the ground truth.

4.3.1 Observations

From the test, it is clear that:

- The **Closed-Book** approach struggles to provide accurate answers when domain-specific or time-sensitive knowledge is required. Since the model only relies on its pre-trained knowledge, it failed to retrieve relevant information for the specific event.
- The **Retrieve-and-Augment** approach performed better by retrieving the necessary context from an external source and providing a correct answer. This demonstrates the effectiveness of RAG in improving factual accuracy, especially when handling specific events or details that are not contained within the model's pre-trained knowledge.

4.4 Model Comparison

To understand the difference between baseline model and our best performing models, here is a list of example questions where the best model's answers achieved f1 = 1.0 and baseline model's answers performed poorly.
Example Questions:

- "Who are the community group contacts for purchasing low-price tickets to the 10:30AM matinee performance of We Shall Not Be Moved?"
  - Best model answer(f1 = 1.0): "Wendy Parkulo"
  - Baseline model answer(f1 = 0.25): "Wendy Parkulo, Manager of Group Sales and Community Initiativesvia emailor at 412-281-0912 ext. 213"

  Analysis

  - Relevance: The answer is relevant but minimally so. It directly answers the question by identifying the contact person but lacks additional context that could be useful.
  - Precision:The improved model is more precise in terms of matching the exact query, as it provides only the requested information (the name of the contact).
- "What is the Local Services Tax (LST)?"
  - Best model answer(f1 = 1.0): "a tax on individuals for the privilege of engaging in an occupation"
  - Baseline model answer(f1 = 0.14): "LOCAL SERVICES TAX REGULATIONS Issued Pursuant to the City of Pittsburgh City Code, Title II Article VII, Chapter 252 Effective January 1, 2008 Revised January 1, 2020 2- Table of Contents ART"

  Analysis

  - Accuracy: The baseline answer is inaccurate for the given question. It does not define or explain what the Local Services Tax (LST) is. Instead, it provides a reference to regulations and a table of contents, which is irrelevant to the query.
  - Relevance: The baseline model answer is not relevant to the question. It does not address the meaning or purpose of the LST.
- "What is the mission of the Pittsburgh Bureau of Fire?"
  - Best model answer(f1 = 1.0): "protect life, property and the environment by providing effective customer and human services related to fire suppression, fire responder medical service, hazardous materials mitigation, emergency management service and domestic preparedness"
  - Baseline model answer(f1 = 0.38): "mitigation, emergency management services, and domestic preparedness"

  Analysis

  - Accuracy: The baseline answer is partially accurate but incomplete, while the improved answer is highly accurate and closely matches the true answer. The baseline answer mentions some components of the mission (e.g., mitigation, emergency management services, and domestic preparedness) but omits key elements such as protecting life, property, and the environment, as well as fire suppression and medical services.