

Research compendium

ENGSCI700A/B



October 16, 2021

Jaime wu & Andrew wang

Contents

[Research Compendium 2](#_Toc85308988)

[Dense Vector Round Trip Consistency 2](#_Toc85308989)

[RAG Pipeline Implementation 2](#_Toc85308990)

[Summarising the knowledge base 2](#_Toc85308991)

[Abstractive Summarisation for Unambiguous Synthetic QA Generation 2](#_Toc85308992)

[Integrating Lexical Matching results into synthetic data/Question Enhancement 3](#_Toc85308993)

# Research Compendium

The following document provides more details on the work performed during the project and to support the information provided in the report. Our compendium is fully open-source under the GNU General Public License Version 3, and its source can be found here:

<https://github.com/Zefty/ENGSCI700-Compendium>

The compendium has the following folder structure:

|-- [ENGSCI700-Compendium](https://github.com/Zefty/ENGSCI700-Compendium)/

|-- [RAG\_pipeline](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/RAG_pipeline)

|-- [question\_enhancement](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement)

|-- [round\_trip\_consistency](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/round_trip_consistency)

|-- [summarise\_kb](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/summarise_kb)

|-- [synthetic\_data\_gen](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen)

Each folder is titled to reflect a section of work referred to within our report, and each folder contains the source-code for that work. For example, our two main contributions were synthetic data generation and question enhancement, and the source-code (including the data, script files, notebooks) can be found in each respective folder. The contents of each folder will further be clarified below.

NOTE: This compendium only includes the source-code that has been referenced extensively within our report. Prior versions of our source-code have not been included in this compendium.

# RAG Pipeline Implementation

In our work, we implemented a RAG pipeline using the original RAG model (partially end-2-end). To do this we used the Haystack library. Our RAG pipeline can be broken down into three main components (1) data preparation (2) model fine-tuning and (3) model evaluation. Prior versions of our source-code have not been included in this compendium.

Source-code Link: [RAG\_pipeline](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/RAG_pipeline)

|  |  |  |
| --- | --- | --- |
| **Filename** | **Purpose** | **Contributions** |
| [BART\_Fine\_Tuning.ipynb](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/RAG_pipeline/BART_Fine_Tuning.ipynb) | The BART fine tuning notebook fine tunes a BART model for answer generation. We used an open-source training script to train our model with our synthetic data. | Andrew Wang |
| [BM25\_Hard\_Negative\_Selection\_+\_ DPR\_Fine\_Tuning.ipynb](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/RAG_pipeline/BM25_Hard_Negative_Selection_%2B_DPR_Fine_Tuning.ipynb) | The BM25 + DPR notebook establishes our main RAG pipeline.  This notebook first prepares the synthetic training data into a suitable format for training using a BM25 model to select hard negatives. Our pipeline then fine-tunes a DPR model and then prepares the fine-tuned DPR along with the above fine-tuned BART model into a RAG pipeline. Lastly, this notebook establishes an evaluation pipeline to evaluate our RAG model against our benchmark data set (COVID-QA). | Jaime Wu  Andrew Wang |

NOTE: These notebooks are best run-on Google Colab as this was the environment, we used to test our code. We do not guarantee that these notebooks will run on a native desktop environment given that many packages and dependencies must be first installed.

# Integrating Lexical Matching results into synthetic data/Question Enhancement

One of our main data-centric contributions was to enhance the input question’s by appending the most related passaged via a BM25 search. Our approach parallels the approach taken by BlenderBot 2.0, whereby an internet search was used instead. Our work mainly focuses on a generated set of synthetic data, from which we convert into an enhanced version. Prior versions of our source-code have not been included in this compendium.

Source-code Link: [question\_enhancement](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement)

|  |  |  |
| --- | --- | --- |
| **Filename** | **Purpose** | **Contributions** |
| [BM25\_question.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/question_enhancement/BM25_question.py) | This Python script file is the main algorithm that processes a given synthetic QA data set.  The script file has a main preprocess() function that pre-processes the passages and questions by removing stop words and cleaning the text.  The algorithm then creates a BM25 knowledge base  Finally, the algorithm iterates through a data set and appends a passage to each question using the Okpai BM25 model. | Jaime Wu |
| [requirements.txt](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/question_enhancement/requirements.txt) | This file contains the necessary Python packages to run the script file.  Please use the following command to install the packages:  python -r requirements.txt | Jaime Wu |
| [data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data) | This folder contains our input and output data. The folder structure is as follows:  |-- [data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data)/  |-- [input\_data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/input_data)  |-- [Q-covid-val](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/input_data/Q-covid-val)  |-- [Q-covid](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/input_data/Q-covid)  |-- [output\_data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/output_data)  |-- [Q-covid-BM25-val](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/output_data/Q-covid-BM25-val)  |-- [Q-covid-BM25](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/output_data/Q-covid-BM25)  The input data subdirectory contains about 175K synthetic QA data generated by Shamane. We process a training set and a validation set (Q-covid and Q-covid-val respectively). The output data subdirectory follows a similar naming convention. | Jaime Wu |

# Dense Vector Round Trip Consistency

One of our minor contributions was to improve round trip consistency using dense vector embeddings and sentence similarity. Prior versions of our source-code have not been included in this compendium.

Source-code Link: [round\_trip\_consistency](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/round_trip_consistency)

|  |  |  |
| --- | --- | --- |
| **Filename** | **Purpose** | **Contributions** |
| [filter\_synthetic.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/round_trip_consistency/filter_synthetic.py) | This Python script file is the main algorithm that processes a given synthetic QA data set.  The script file loads in the synthetic QA data and begins our dense vector round trip consistency algorithm.  More details can be found within our report.  We used to apply this filtration method to our abstractive summarisation dataset. | Jaime Wu |
| [Round\_Trip\_Consistency\_RAG.ipynb](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/round_trip_consistency/Round_Trip_Consistency_RAG.ipynb) | This notebook is very similar to the Python script file, except it has been written in such a way to allow us to leverage Google Colab’s GPUs. | Jaime Wu |
| [requirements.txt](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/question_enhancement/requirements.txt) | This file contains the necessary Python packages to run the script file.  Please use the following command to install the packages:  python -r requirements.txt | Jaime Wu |

NOTE: These notebooks are best run-on Google Colab as this was the environment, we used to test our code. We do not guarantee that these notebooks will run on a native desktop environment given that many packages and dependencies must be first installed.

# Summarising the knowledge base

We summarised RAG’s knowledge base, so that the DPR can find more relevant passages easily. Prior versions of our source-code have not been included in this compendium.

Source-code Link: [summarise\_kb](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/summarise_kb)

|  |  |  |
| --- | --- | --- |
| **Filename** | **Purpose** | **Contributions** |
| [Summarising Knowledge Base.ipynb](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/summarise_kb/Summarising%20Knowledge%20Base.ipynb) | This notebook contains the code used to summarise the knowledge base. This is the method where we explore summarising a knowledge base before giving it the retriever to use. It includes both approaches e.g., summarising every passage, and summarising just the abstract. | Andrew Wang |

NOTE: These notebooks are best run-on Google Colab as this was the environment, we used to test our code. We do not guarantee that these notebooks will run on a native desktop environment given that many packages and dependencies must be first installed.

# Abstractive Summarisation for Unambiguous Synthetic QA Generation

One of our main data-centric contributions was to generate unambiguous synthetic QA pairs. Prior versions of our source-code have not been included in this compendium.

Source-code Link: [synthetic\_data\_gen](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen)

|  |  |  |
| --- | --- | --- |
| **Filename** | **Purpose** | **Contributions** |
| [run\_synthetic\_qa\_generator.sh](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/run_synthetic_qa_generator.sh) | This bash script simple runs the [SyntheticQAGenerator.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/SyntheticQAGenerator.py) file. | Jaime Wu |
| [SyntheticQAGenerator.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/SyntheticQAGenerator.py) | This Python script file is the main algorithm that generates a synthetic QA data set.  This Python script contains a class with two main models (1) SCI-TDLR for answer summarisation (2) T5 for question generation  In summary, the algorithm creates a synthetic QA pair for each passage of an input data set. | Jaime Wu |
| [ConcatenateData.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/ConcatenateData.py) | ConcatenateData.py and PrepareRAGData.py finalises our output data into a format suitable for training RAG. | Jaime Wu |
| [PrepareRAGData.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/PrepareRAGData.py) | ConcatenateData.py and PrepareRAGData.py finalises our output data into a format suitable for training RAG. | Jaime Wu |
| [requirements.txt](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/question_enhancement/requirements.txt) | This file contains the necessary Python packages to run the script file.  Please use the following command to install the packages:  python -r requirements.txt | Jaime Wu |
| [data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data) | This folder contains our input and output data. The folder structure is as follows:  |-- [data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data)/  |-- [input\_data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/input_data)  |-- [Q-covid-summary](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen/data/input_data/Q-covid-summary)  |-- [Q-covid](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/input_data/Q-covid)  |-- [output\_data](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/question_enhancement/data/output_data)  |-- [Q-covid-100k+Q-covid-summary-50k](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen/data/output_data/Q-covid-100k%2BQ-covid-summary-50k)  |-- [Q-covid-100k](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen/data/output_data/Q-covid-100k)  |-- [Q-covid-summary-100k](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen/data/output_data/Q-covid-summary-100k)  The output data folder contains all the synthetic QA data generated using our abstractive summarisation algorithm. NOTE: The input data contains all our outputted data in raw format, which we then use the above [ConcatenateData.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/ConcatenateData.py) and [PrepareRAGData.py](https://github.com/Zefty/ENGSCI700-Compendium/blob/main/synthetic_data_gen/PrepareRAGData.py) to produce the output data in its final format for RAG training. | Jaime Wu |
| [data\_[DEPRECATED]](https://github.com/Zefty/ENGSCI700-Compendium/tree/main/synthetic_data_gen/data_%5BDEPRECATED%5D) | Contains our deprecated data in raw format | Jaime Wu |