PART A: Problem Defini1on Minimum: 10 credits			Your Project
	Points	Description	
NLP (Natural Language Processing) Problem	5 credits per problem	Examples: Question-answering, Sentiment Analysis, etc. If your project uses an NLP problem for another downstream task (say, stock market prediction), please count only the NLP problem.	Question-answering
Text Source/Dom ain	5 credits per text source/ domains	Select one among: News Articles, Research papers, social media posts, sentences, etc.	Wikipedia articles and some source text
PART D: Evalua1on Minimum: 20 credits			
Quantitative Evaluation	10 credits	Relevant metrics such as F- score, Precision@5	Exact match and F1 score
Qualitative Evaluation	5 credits	Examine mis-classified instances and produce	

		common error types	
Command line testing	5 credits	Interface to test out the system. This can be executed using an input argument or input file.	Terminal test implemented in python
Demo	10 credits	A simple demo through Gradio (or equivalent).	Gradio demo for running models on the web.

PART A: Problem Definition

1.1 NLP Problem

In this project, we are trying to solve a question that answers a natural language processing problem. We are developing a system that determines if a given question can be answered based on the provided text. If the text does not contain sufficient information, the system will give a general answer based on the question. The system will also make the answers as relevant and complete as possible, directly derived from the source text. We try to make a more human-like response to explain why this answer was selected or chosen.

1.2 Text Source/Domain

We decided to use the SQuAD 2.0 Dataset as our primary text source because the dataset comprises questions based on a set of Wikipedia articles or some source text

about some knowledge. It will first give a source text and a set of questions, where the answer to those questions is a text segment from the source, or span, from the corresponding reading.

PART B: Dataset Selection

2.1 Use one existing dataset and an existing lexicon

We use SQuAD 2.0 [1], a Stanford Question Answering Dataset. It includes over 100,000 question-answer pairs on 500+ articles with answerable questions about general knowledge, and this dataset also comes with a label to indicate whether the current question can get information from given sources' text.

We also use an existing lexicon, WordNet, for the rule-based model by providing semantic relationships and synonym sets, which are crucial for determining the depth of contextual match between the questions and the provided text.

2.2 Create own labelled dataset

In order to further enhance the understanding and generation capabilities of the model, we decided to expand the SQuAD 2.0 data set. Specifically, we added explanations for the answers to the first 13,401 pieces of data in the training set. These explanations are automatically generated by calling the GPT-3.5 API.

2.2.1 Expansion process:

1. Input combination: Combine the Question, Context and Answer of each piece of data to form input text.

2. Explanation generation: Input the above combined text into the GPT-3.5 API and request the generation of an explanation for the answer.

3. Data integration: Add the generated explanations to the corresponding data entries to form a new data set entry format: question + context + answer + explanation.

2.2.2 Benefits of expansion:

• Deep contextual understanding:

Through explanation, the model not only learns to find the answer, but also understands why the answer is correct. This ability is the hallmark of advanced language understanding.

• Enhance question sensitivity:

Explanations help the model grasp nuances in the question, for example, different ways of asking may lead to completely different answers and explanations.

PART C: Modelling

3.1 Implement a Rule-Based Model as a Baseline

We created three rule-base baseline methods to evaluate the answerability of the questions:

1. **Direct Word Overlap**: This method checks if words from the question appear directly in the context.
2. **Sentence-Level Word Overlap**: This method analyses on a sentence basis to see if any sentence within the context contains at least half of the words from the question.
3. **WordNet Semantic Similarity**: This method uses WordNet to compute semantic similarities between words in the question and the context, evaluating whether the contextual meanings align well enough to consider the question answerable.

3.2 Fine-tune T5 model based on two dataset

3.2.1 Baseline Model Training (Baseline Training)

Model: Use T5 model as baseline.

Dataset: Fully fine-tuned using the original SQuAD 2.0 dataset.

Purpose: This step is intended to establish a performance baseline so that the effectiveness of other training strategies can be evaluated.

3.2.2 Input Expansion Training

Model: Use the weights of the baseline model.

Dataset: Fine-tune the baseline model again, taking the answer explanation as part of the input (format: question + context + explanation -> answer).

Purpose: Test the impact of adding additional explanatory information to the input on model understanding and output accuracy.

3.2.3 Partial Parameter Freezing Training

Model: Uses the weights from the baseline model and freezes most parameters.

Dataset: Use a modified input format (question+context+explanation) that integrates explanations into the input.

Parameter Freeze: Freezes the parameters of all encoder layers, and of all decoder layers except the last two decoder modules.

Purpose: To explore the effect of improving the model's ability to pay attention to newly added explanatory information by fixing most of the model parameters and training only a few layers.

3.2.4 Multi-Stage Fine-Tuning Training

Model: Use the weights from the baseline model and implement a similar parameter freezing strategy.

Dataset: Fine-tuning in two stages:

Phase 1: Question + Context -> Explanation. The goal of this phase is to generate explanations specific to the given problem and context.

Stage 2: Question + Context + Explanation -> Answer. This stage uses the explanations generated in the first stage to predict the answer.

Purpose: To gradually introduce question interpretation and answer generation through staged training, and evaluate the effect of gradually improving model performance.

3.2.5 Multi-Stage with Parameter Freezing

Model: Keep the weights of the baseline model, but adopt a stricter parameter freezing strategy during multi-stage training.

Dataset: Using a staged fine-tuning method:

Phase 1: Same as the previous strategy, first generate explanations specific to the problem and context.

Phase 2: Freezing more model parameters except specific layers while generating answers.

Parameter freezing: All parameters except the last decoder module are frozen in the second stage, which means that only one decoder module in the entire model participates in the answer generation process.

Purpose: To test the implementation of stricter parameter freezing in multi-stage training and evaluate its impact on the model's focus and performance on specific tasks.

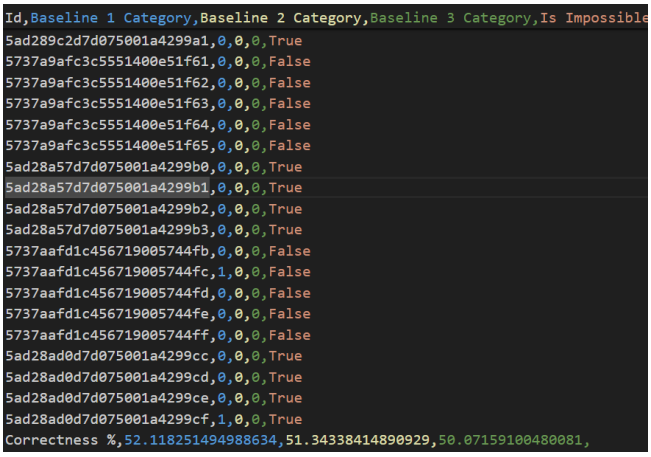
PART D: Evaluation

4.1 Rule-base model baseline

For the rule-based model baseline, we use each baseline's effectiveness to measure the evaluation, and we do this by comparing its predictions against the is_impossible flags provided in the dataset. The model tracks how often its predictions about answerability are correct, providing a straightforward measure of its performance.

Results of the exact match compared to the right answer:

- Baseline 1: Direct Word Overlap has an accuracy of 52.12%.
- Baseline 2: Sentence-Level Word Overlap shows an accuracy of 51.34%.
- Baseline 3: WordNet Semantic Similarity reports an accuracy of 50.07%.



```
Id,Baseline 1 Category,Baseline 2 Category,Baseline 3 Category,Is Impossible
5ad289c2d7d075001a4299a1,0,0,0,True
5737a9afc3c5551400e51f61,0,0,0,False
5737a9afc3c5551400e51f62,0,0,0,False
5737a9afc3c5551400e51f63,0,0,0,False
5737a9afc3c5551400e51f64,0,0,0,False
5737a9afc3c5551400e51f65,0,0,0,False
5ad28a57d7d075001a4299b0,0,0,0,True
5ad28a57d7d075001a4299b1,0,0,0,True
5ad28a57d7d075001a4299b2,0,0,0,True
5ad28a57d7d075001a4299b3,0,0,0,True
5737aafd1c456719005744fb,0,0,0,False
5737aafd1c456719005744fc,1,0,0,False
5737aafd1c456719005744fd,0,0,0,False
5737aafd1c456719005744fe,0,0,0,False
5737aafd1c456719005744ff,0,0,0,False
5ad28ad0d7d075001a4299cc,0,0,0,True
5ad28ad0d7d075001a4299cd,0,0,0,True
5ad28ad0d7d075001a4299ce,0,0,0,True
5ad28ad0d7d075001a4299cf,1,0,0,True
Correctness %,52.118251494988634,51.34338414890929,50.07159100480081,
```

The results show that each method is significantly better than the others. It shows that, although the features used have some predictive value, rule-based approaches have considerable limitations in dealing with the complexity of natural language semantics involved in question answering.

4.2 Quantitative Evaluation (Metrics for QA model)

There are two dominant metrics used by many question answering datasets, including SQuAD: exact match (EM) and F1 score. These scores are computed on individual question+answer pairs. When multiple correct answers are possible for a given question, the maximum score over all possible correct answers is computed. Overall EM and F1 scores are computed for a model by averaging over the individual example scores.

4.2.1 Exact Match

For each question+answer pair, if the characters of the model's prediction exactly match the characters of (one of) the True Answer(s), EM = 1, otherwise EM = 0. This is a strict all-or-nothing metric; being off by a single character results in a score of 0. When assessing against a negative example, if the model predicts any text at all, it automatically receives a 0 for that example.

4.2.2 F1

F1 score is a common metric for classification problems, and widely used in QA. It can care equally about precision and recall. In this case, it's computed over the individual words in the prediction against those in the True Answer. The number of shared words between the prediction and the truth is the basis of the F1 score: precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth.

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

4.2.3 Fine-tuning T5 model results

For the Fine-tuning T5 model, we use the EM and F1 score to measure the evaluation.

	EM	F1
Baseline Model	0.7237	0.8434
Input Expansion Training	0.7014	0.8164
Input Expansion Training(Freeze all layers except the last two Transformer blocks)	0.7239	0.8361
Multi-Stage Fine-Tuning Training	0.6967	0.8147
Multi-Stage Fine-Tuning Training(Freeze all layers except the last two Transformer blocks)	0.7213	0.8351

According to the results, we can figure out that the indicator performance has deteriorated and has not been improved. The best effect is only using full fine-tuning. The possible reasons are as follows:

The explanations generated by GPT may differ semantically or syntactically from the original SQuAD answers, which may introduce noise and confuse the model. Generating explanations that are too lengthy or contain unnecessary information may distract the model and make it difficult to extract key information.

The additional data may be distributed differently from the original SQuAD dataset. This shift in the distribution may cause the model to over-adapt to the characteristics of the new data and ignore the general characteristics of the original data set.

After having been fine-tuned with 100,000 pieces of SQuAD data, the model may have adapted well to the task. Continuing to train with new data, especially if the quality or distribution of the new data is different from the original data, the model may start to overfit the new data.

Overfitting may lead to a decrease in the generalisation ability of the model on the test set, thereby lowering the EM and F1 scores.

4.3 Command line testing

After the Quantitative valuation of the model we performed command line testing on the current evaluated model, which is a common testing method in project development. The purpose is to control every aspect of the test more flexibly and finely through the command line testing. For example, the input context can be a sentence directly typed on the command line, or it can be a .txt file or other text file as the input context for testing. This helps us to automate execute test and quickly find bug and debugs.

In the test framework we used `argparse` as a library that parsing arguments on the command line, and we also used Hugging Face’s transformers library to load the model.

In the test execution part, user can provide specific parameters or choose to provide existing text file for testing. The user has to enter three parameters in the terminal of the current python file directory, three parameters are `--model_path`, which is the address of the model, `--question`, which is the question you want to ask the model based on the

text context, and `--context/context_file`, where you can either enter a piece of text, or enter the address of an existing text file.

An example:

python command_line_test_model.py ---model_path ". /models" --question "What is a dog?" --context "Dog is a lovely pet, they are kind and nice,they loves eating bones"

4.4 Demo

If we want more people, especially those who doesn't know how to programming or a NLP layperson to use and test our model, we can't just let them use and test our model with boring and hard understanding common line. Therefore, create a demo is a good way to present the results of our project in a more intuitive and interactive way. Instead of need to enter 3 parameters as common line test, users could test the model as a website to test and use the model.

The interface of our demo explicitly gives examples of how the user can use the model through text input to guide the user to use it. This also makes it easier for NLP layperson and computer laymen to use and test our models. Our demo also allows the user to enter a question and the context text associated with that question. The user then could just simply presses the submit button to get the answer predicted by the model.

For the technical implementation, the demo uses the Gradio library to build the front-end interface and integrates well with the back-end of our T5 model. Gradio was chosen because it provides a very simple API for creating input and output interfaces, which makes it easy to understand how to use it, even if user is first time to use our demo.

The technical implementation of the demo is not only to build the Gradio interface, but also to ensure the user experience. We have implemented a series of error handling methods, which can record detailed information when encountering an unexpected error, and provide the user with a clear error message. If the user does not provide the question

and the context, the system will remind that it needs to be filled out, and all inputs will be validated before they are input to the model. Finally, in order to fully evaluate our Q&A system, we also designed a series of test cases in the code to verify the accuracy of the model, and exported the test results of the model to a csv file for subsequent optimization.

Overall, with the implementation of the above techniques, our demo is more about providing NLP and computer laymen with experiences where they can test and use our model, as well as helping us to better improve and optimization the performance of our model in the future.

Reference

1. The Stanford Question Answering Dataset. (n.d.). Rajpurkar.github.io. https://rajpurkar.github.io/SQuAD-explorer/