Comparative analysis of Distil BERT-based and Roberta-based on Question Answering pre-trained finetuned models

Naing Khant Htet

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of Computing

School of Computing

May 2024

Authorship Declaration

I, Naing Khant Htet, confirm that this dissertation and the work presented in it are my own achievements.

Where I have consulted the published work of others this is always clearly attributed.

Where I have quoted from the work of others the source is always given. With the exception of such quotations, this dissertation is entirely my own work.

I have acknowledged all main sources of help.

If my research follows on from previous work or is part of a larger collaborative research project, I have made clear exactly what was done by others and what I have contributed myself.

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the school's ethical guidelines

Signed:

Date: 1 May 2024

Matriculation no: 40667695

General Data Protection Regulation Declaration

Under the General Data Protection Regulation (GDPR) (EU) 2016/679, the University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Naing Khant Htet

Abstract

This Honours project compares the pre-trained and fine-tuned question-answering (QA) models named: distilbert-base-cased-distilled-squad and deepset/Roberta-base-squad2. Both are easily accessible by importing the Transformers Python library and using the Pipeline to load the models. The comparison evaluates the models using the metric systems: F1 score and Exact Match (EM), and model evaluation time is recorded to determine the efficiency and accuracy of both models. This study contributes to the strengths and weaknesses of each QA model.

Machine Learning (ML) techniques are employed to evaluate these models, using the custom dataset. Initially, data are prepared and cleaned to ensure a fair comparison. Since the models are already pre-trained and finetuned, this project imports those models with specified tasks to evaluate the QA task. Both QA models' performances are captured using the F1 score, EM metrics, and time capturing for the QA process.

For the accuracy result, both models perform impressively, with deepset/Roberta-base-squad2 dominating the viral model in a few scores while the distilbert-base-cased-distilled-squad significantly dominates in efficiency result. This project also highlights the importance of selecting appropriate models for specific application requirements.

Contents

List of Tables

List of Figures

Acknowledgements

Firstly, I would like to thank my Supervisor, Dr. Aye Aye Myint for her guidance and ideas throughout the Project times.

Biggest thanks to my Family for providing me Internet, computer, and tuition fees that provide me to complete my studies.

1 Introduction

1.1 Background

Natural Language Processing (NLP) is the technology that all information seekers need. Searching for the desired information on Google is an example. It matches with the input and the algorithm understands it (SEOMasterz, 2023). NLP advances text-based processes, such as translating, question-answering (QA), text summarisation, customer interaction, and more. It is used in applications that understand text information to bring the relevant information or document from a collection or database, translate the language, and generate the responses in the text (Aggarwal, 2011).

In the earlier days of NLP, there were two closed-domain QA systems named LUNAR and BASEBALL. LUNAR is to answer about the rocks returned by the Apollo Moon missions and BASEBALL is to answer about over a year for Major League Baseball. LUNAR was able to answer 90% of the questions that were demonstrated in 1971. As time goes by, QA systems are improving so that they can determine the purpose of the question and the type of answer as of the year 2001. Open-domain QA systems have been seen and each relies on different kinds of architectures.

People have become lazier about reading a full text on a document just to acknowledge the information they are willing to know, they want to get instant results in a short amount of time without reading many lines of text (Kumar et al., 2022). Aggarwal noted that Intelligent Questing Answering Systems (IQASs) help students to become better readers. There are already QA projects in the real world, and some are implemented as businesses. For example, chatbots for customer support, conversational information extraction like ChatGPT from OpenAI, etc.

Data availability has been increasing as the years go by, so the QA tool is essential for retrieving information efficiently (Patil et al., 2023). I am also writing this report with the help of NLP-based products like Google for the information and ChatGPT for the ideas.

In October 2018, Bidirectional Encoder Representations from Transformers (BERT) was introduced which is simple and powerful enough to take a wide range of tasks with just one additional output layer to the pre-trained BERT model (Devlin et al., 2018). BERT can understand the purpose of a word by looking at other words that come before and after it. With that technology, Google takes a chance to enhance its search experience. Google users can now see the individually extracted answer without going to the relevant website and reading the document. Additionally, search results are becoming more accurate by understanding the subtle nuances of language (Nayak, 2019). Although NLP-based products are becoming popular throughout businesses, the technology still has the potential to improve.

As the programming languages become powerful, libraries already exist that are easy to implement. The Transformer Python library is the place of pre-trained models to perform text, vision, and audio. It relies on the API for developers to use pre-trained models and finetunes with their datasets. Some of the QA projects already exist on the Internet but rely on different algorithms, models, domains, and libraries that retrieve the results.

That comes into the problem of this project. Which models work best for the QA tasks? I have chosen two of the popular QA pre-trained models publicly available from Python's Transformers library. Those are the Distil BERT-base-cased and the Roberta-base models. Distil BERT is the lighter, faster version of the BERT model and Roberta is the enhanced version of the BERT model. Disclaimer, the models this project will be comparing are the finetuned versions. The fine-tuned Distil BERT model is finetuned with the Squad v1 dataset. The original Distil BERT-cased model is trained on the Book Corpus and the English Wikipedia datasets. The finetuned Roberta-base model is finetuned with the Squad v2 dataset. The original Roberta-based model is trained on the Book Corpus, English Wikipedia, CC-News, Open Web Text, and Stories datasets.

Conclusively, this project will be focusing specifically on the comparison of the two closed-domain QA pre-trained finetuned models.

1.2 Problem Statement

The last section discussed the background of QA and NLP technology which stated that the technology is in use by many people and willing to improve more.

By the time of QA systems are invented, the communicating process between human and computers have become more friendly and advanced, enabling computers to understand human language and respond with high accuracy. However, there remains a challenge in determining which models are effective in overall performance and accuracy. This project determines by evaluating and comparing between two models: ‘distilbert-base-cased-distilled-squad’ and ‘deepset/Roberta-base-squad2’.

The core problem is to benchmark those two models to identify their weaknesses and strengths. Specifically, it is vital to understand the performance of models and the exactness from the models’ generated answers. Therefore, this Honours project compares between those models by using F1 and Exact Match metrics system, use pipelines to load the models, and track the models’ processing time to consider their performances.

1.3 Aim and objectives

The aim of this project is to compare the two pre-trained QA models. When comparing, a decent metrics system will be used, F1 score, and Exact match utilizing the help of a custom-made small QA dataset. The comparison will provide the final visual presentation of the result. To achieve the aim, the following objectives will be accomplished.

- To gather existing research papers and projects
- To Analyse the existing question-answering NLP projects
- To Research and study the related Architectures, and models
- To Design and Implement the pre-trained models
- To Develop the metrics system
- To Implement the visual tools to generate the visual presentations of the result
- To Write the documents

1.4 Structure of the Report

The report's structure is organized as follows: In Chapter 2, related background theories are mentioned to clarify and acknowledge the root of the project problem. In

Chapter 3, related works and foundations will be discussed and reviewed by mentioning their approaches to QA model comparison, the outcomes from their approaches, and the challenges they faced. Chapter 4 documents the methodology this project will be using. In Chapter 5, the results from the developed system will be analysed and discussed. Finally, Chapter 6 concludes the project report. After that, appendices and references are listed.

2 Background Theories

2.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) processes the work that can be done by human intelligence but done with computer systems (IBM, n.d.-a). It is capable of problem-solving on specific subjects, combined with available technologies. For example, integrating AI into Google Maps for advanced map assistance. Normally, this type of task requires human knowledge and intelligence.

When mentioning AI, Deep Learning and Machine Learning always come up together since they provide the fuel for the development of AI. They are modelled after the decision-making process by human brains, learning the available data and classifying or predicting with the knowledge.

There are two types of AI:

1. Weak AI: also known as ‘Narrow AI’ or ‘Artificial Narrow Intelligence’. As mentioned earlier, integrating AI into specific subjects for advanced computation. Or it supports humans with the best knowledge, but the system is limited since it is not human. For example, self-driving vehicles have an advantage in driving but would not have an advantage in fields other than driving.
2. Strong AI: would have the same intelligence as humans. A machine could have consciousness, the ability to learn, solve, predict, and plan. It is like a human but on a machine. This AI is still theoretical, but it is being researched. The best example would be human AI from the film Blade Runner 2049.

Various applications integrate AI into the system. For example, Healthcare: diagnostics, robotic surgeries, etc. Finance: Stock market detection, Fraud detection, Data analytics, Stock Predictions, etc. Customer Service: Chatbots, Virtual assistants, etc. Entertainment: Movie and Music recommendations, Script and Content creation, etc. ChatGPT is also an AI feature for the question-answering system.

2.2 Machine Learning (ML)

Machine Learning (ML) fuels the development of Algorithms that allow computers to learn from and make decisions based on the datasets (IBM, n.d.-c). ML is to automatically improve the system without human intelligence being involved or manually programmed for specific tasks.

ML works by starting with the Decision Process. ML algorithms make classifications or decisions based on input that can be labelled or unlabelled. For the accuracy of a model, the Error Function compares the model's prediction with known examples. For the model optimization process, weights are autonomously reduced between the known example and the model's prediction. It will reduce until the accuracy reaches a satisfactory level.

ML is classified into 3 categories.

2.2.1 Supervised Learning

In Supervised Machine Learning, algorithms use labelled datasets to train, classify, and predict the results. The model adjusts the input's weight to fit appropriately which is also the process of Cross-validation that makes sure to avoid overfitting and underfitting. A real-world example would be, classifying spam emails. Supervised Learning is also used in Neural Networks, Linear Regression, Logistic Regression, Support Vector Machine, etc.

2.2.2 Unsupervised Learning

Unsupervised Machine Learning uses algorithms to cluster (subset) the unlabelled datasets. The algorithms handle the process without the need for human intelligence. It can search the similarities and differences that are suitable for data analysis tasks. Neural networks and K-Means Clustering are examples of Unsupervised learning.

2.2.3 Semi-supervised Learning

Semi-supervised learning lies in between Supervised and Unsupervised learning. It uses labelled datasets but smaller ones for classification and larger unlabelled datasets for extraction.

2.3 Deep Learning

Deep Learning is a part of Machine Learning and uses Multi-layered Neural Networks to process complex decision-making tasks (IBM, n.d.-b). Different from traditional Machine Learning Deep Learning uses more computational layers, which could be hundreds or thousands of layers to train the models.

It is capable of Unsupervised learning that models can extract relationships, and features from raw and unstructured datasets. The models can also refine the outputs for better accuracy. Deep Learning is the reason for automation in many analytical

tasks without human intervention. Digital assistants, Fraud detection, and generative AI are supported by Deep Learning.

It is inspired by the function of the brain called Artificial Neural Networks. Input data, weights, and biases all work together to classify and describe the objects. Deep neural networks consist of multiple layers interconnected with nodes, each layer builds upon the previous layer for better precision. The process of the computation is called Forward Propagation, input and outputs are called Visible Layers. There is a process called Backpropagation that is the opposite name of the Deep Learning process, it calculates errors in Prediction and moves backwards through the layers to adjust the weights and bias of the function. Therefore, the models can be more accurate by combining Forward propagation and Backpropagation.

Here are some of the Deep Learning Models.

2.3.1 CNNs

Convolutional Neural Networks (CNNs) are the specific types of Neural Networks that are used for vision classification problems. It can detect patterns within the image or the videos, offering tasks like Object detection, Face recognition, etc.

CNNs are composed of node layers, each containing an input layer, a hidden layer(s), and an output layer. Each node connects to another node having weight and threshold. The data will be sent to the next layer if the output from the node is top over the specified threshold. CNN is made up of at least three main types of layers: Convolutional Layer, Pooling Layer, and Fully Connected Layer. The process will be more complex by each layer. It enables data exchange between layers to support efficient processing.

2.3.2 RNN

Recurrent Neural Networks (RNNs) are used in NLP and speech recognition tasks. It uses sequential data to predict future outcomes. Stock market prediction, Language translation, Speech Recognition, and Caption generation are examples of RNNs.

RNNs use Memory and Binary data Processing to produce not only one output but also one-to-many, many-to-one, or many-to-many outputs. It uses Memory to predict the future events, determining the output of a given sequence. Parameters are shared across the layers. Through the Backpropagation and Gradient descent processes, weights are adjusted, and the same weight parameters are shared in each layer. To

determine the Gradients, the Backpropagation Through Time (BTTT) algorithm is used.

2.3.3 Diffusion models

Diffusion Models are used for Generating mostly images by overwriting the training data. It uses the Forward and Reverse diffusion process of Denoising and Noise-addition, meaning the Diffusion Models will add Noise to the data and then learn the denoising process. The models produce the replicated version of authentic data but not the same. The training process is light since it does not require adversarial training.

2.3.4 Transformer

Since this project makes use of Transformer models, details will be reviewed in the upcoming sections. Overall, Transformers are for NLP tasks, containing Encoder and Decoder architecture. The models process words parallelly which is efficient for training. Also, it is the successor of RNNs but requires heavy computational resources to train. Classification, Name Entity Recognition, Language Translation, Text summarization, and QA are examples of Transformers.

2.4 Natural Language Processing (NLP)

NLP helps understand the language of Humans to communicate by using AI. It is already a part of everyday lives, powering search engines, customer service chatbots, voice-operated GPS systems, and QA assistants (IBM, n.d.-d). It enables the computer to recognize the human's language by text and voice.

These tasks are examples of NLP: Spam detection, Text summarization, Language translation, QA, Sentiment analysis, Grammar correction, and many more. NLP includes two main types of analysis: Syntactical and Semantic. Syntactical determines the meaning of the words while Semantic explains the meaning of words within the sentence structure.

There are three different approaches to NLP:

Rules-based NLP: Traditional programming if-then decision trees. It can only provide answers based on scope size and cannot bring any further than that.

Statistical NLP: This enables mapping words to a vector representation and can be modelled using mathematical methods. It classifies and labels elements from text and voice data to assign the meaning of the elements.

Deep Learning NLP: It can process even from raw and unstructured data including text and voice, using Neural Network models. There are Sequence-to-Sequence (seq2seq) models, Transformer models, Autoregressive models, and Foundation models. Seq2seq are based on RNNs and are used for Translation. There are famous transformer models such as BERT, use Tokenization and Self-attention to learn the relation of texts. Autoregressive models try to predict the next words in a sequence, GPT is an example. Foundation models and Prebuilt models can enhance the speed of NLP tasks, such as IBM Granite foundation models support Content generation and Insight extraction tasks.

NLP can help organizations discover insights in a faster way that finds relationships between trending pieces of content for deeper analysis without the extra need for human intervention. With those advantages, organizations can save budget by automating heavy processes.

Although NLP processes can reduce the budget, they can also skew the answers based on the type of data used. If the processes become diverse, the results will also be diverse. Incorrect grammar, advanced idioms or such sophisticated use of words can confuse the process.

2.5 Transformers Architecture

The Transformers architecture was introduced by (Vaswani et al., 2017). The architecture that changes NLP relies on the Self-Attention mechanisms instead of Recurrent Neural Network (RNN). RNN is the process of one element at a time while carrying through the processes with the Hidden State h_t. It is updated based on the position t and the previously hidden state h_{t-1}. RNN has the limitation of parallelization process that could save time training the data.

However, the Transformers can train the data parallelly while capturing the long-range dependencies in the input sequence. It can also take large datasets suitable for essential NLP tasks such as Text summarization, QA, etc. As the author stated, the model is trained on 8 Nvidia P100 GPUs, 10,000 steps for the base model with the English-German dataset.

The following sub-sections will provide the architecture of the Transformers.

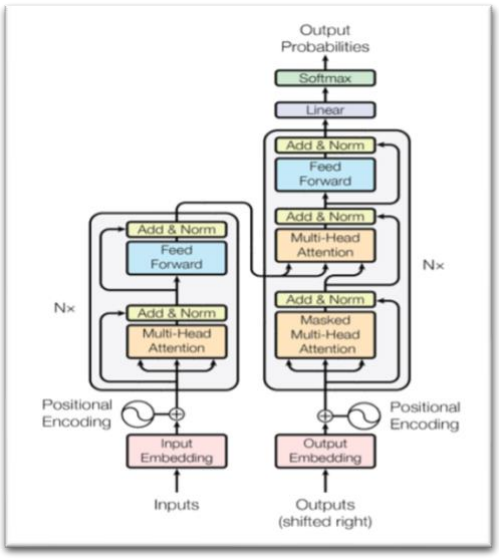

Figure 2.5-1 The Transformer - Model Architecture (Vaswani et al., 2017)

2.5.1 Encoder-Decoder Structure

The architecture consists of two structures, an Encoder and a Decoder. The encoder consists of a stack of $N = 6$ identical layers and the Decoder consists of a stack of $N = 6$ identical layers with the additional layer that performs the multi-head attention for the output of the Encoder layer.

2.5.1.1 Encoder Layer

As mentioned, this layer has $N = 6$ identical layers, but contains two sub-layers for each Encoder layer. They are the Multi-Head Self Attention Mechanism and the Position-wise Fully Connected Feed-Forward Network. Each layer has the layer normalization which is so-called the Residual Connection. Each layer's output can be represented as, $LayerNorm(x + Sublayer(x))$.

2.5.1.2 Decoder Layer

The Decoder layer contains three sub-layers, namely the Masked Multi-Head Self-Attention Mechanism, the Multi-Head Attention Mechanism (processing to the Encoder's output), and the Position-wise Fully Connected Feed-Forward Network. These layers also have their Normalization layer.

2.5.2 Attention mechanisms

The attention can be considered as mapping the query and a set of Key-Value pairs to the output that vectors are all the Outputs, Keys, Values, and Queries.

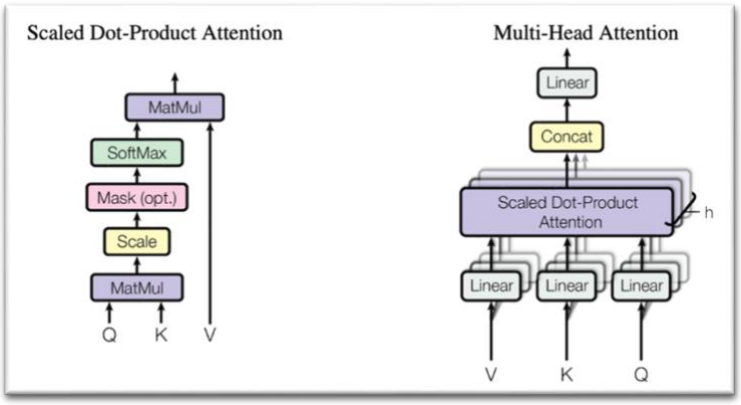

Figure 2.5-2 Scaled Dot-Product Attention and Multi-Head Attention (Vaswani et al., 2017)

2.5.2.1 Scaled Dot-Product Attention

The Dot-Product attention is identical to the algorithm. It has the inputs of queries Q, keys K, values V, keys of dimension d_k, and values of dimension d_v. It is computed as:

Equation 2.5-1 Dot-product Attention

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{\mathrm{d_k}}})V$$

2.5.2.2 Multi-Head Attention

This attention performs multiple h attention heads instead of a single attention function with d_{model} dimension keys. From the different representation subspaces at different positions, the model can attend to the data.
It is computed as:

Equation 2.5-2 Multi-Head Attention

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^0$$

Where each head is computed as:

Equation 2.5-3 Each head in Multi-Head

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

2.5.3 Position-wise Feed-Forward Networks

The feed-forward Network is the additional connection of each layer in the Attention sub-layers. It contains a RELU activation between two linear transformations. They are the same across different positions but use different parameters. The network is computed as:

Equation 2.5-4 Feed-Forward Networks

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$$

2.5.4 Embedding and SoftMax

2.5.4.1 Embedding

Embedding converts the words (tokens) into fixed-size vectors d_{model}. Transformers use Positional Encoding (PE) to the Embedding vectors for each token's position information. PE is computed as:

Equation 2.5-5 Positional Encodings

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Where pos is the position and i is the dimension.

2.5.4.2 SoftMax

SoftMax converts the output of the Decoder to probabilities of the next tokens. The same weight matrix is shared between two Embedding layers and the pre-state of SoftMax linear transformation. In the Embedding layers, weights are multiplied by the $\sqrt{d_{model}}$.

2.6 BERT

BERT is introduced by (Devlin et al., 2018). It is the pre-trained model to improve NLP tasks by relying on the Transformers architecture. It introduces the bi-directional processing where the model will consider the previous and next words at the same time. That is possible with the help of the Masked Language Model (MLM) pre-training objective that the model will randomly mask 15% of the input tokens. It predicts the original vocabulary ID of the masked token with only its context.

BERT uses Next Sentence Prediction (NSP) to understand the relationship of sentences. NSP task predicts if sentence B follows sentence A which enhances the prediction by understanding the sentence relationship.

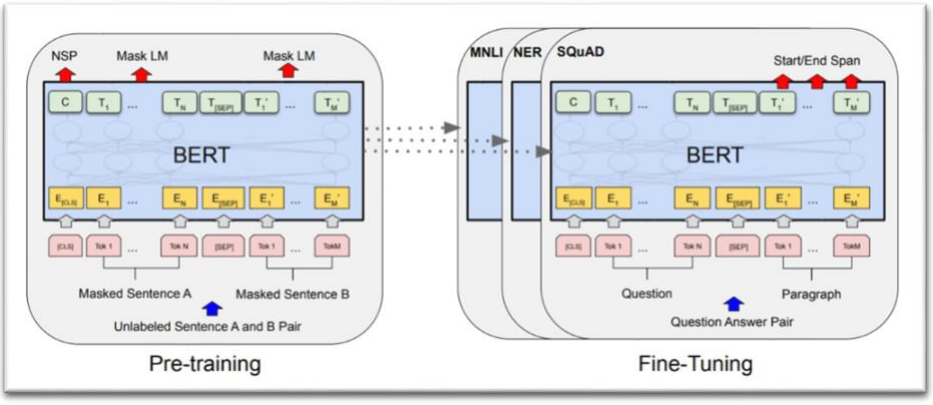

Figure 2.6-1 Scaled Dot-Product Attention and Multi-Head Attention (Vaswani et al., 2017)

The model goes through two stages: Pre-Training and Fine-Tuning.

2.6.1.1 Pre-Training

In the pre-training stage, the model uses MLM and NSP. MLM randomly masks 15% of the token in the input sequence, replaces 80% of the masked tokens with [MASK], 10% with random tokens, and keeps the final 10% unchanged. NSP creates sentence pairs. (IsNext) is labelled if sentence B is the actual next sentence and (NotNext) is labelled if sentence B is random. [CLS] Classification token is added at the start of each sentence pair and [SEP] Separator token is added to the end. The model is trained on Books Corpus (800M words) and English Wikipedia (2,500M words except for tables, lists, and headers).

2.6.1.2 Fine-Tuning

Fine-tuning is the process of pre-trained BERT on specific downstream tasks by adding a task-specific layer depending on the task. If the task is Text Classification, [CLS] token representation will be fed into an output layer for classification. If the task is QA, two classification layers will be then to predict the start and end position of the answer.

2.7 Distil BERT model

This section will break down the Distil BERT model as this project makes use of the Distil BERT model. As mentioned earlier in this report, Distil BERT is the lighter, smaller, faster version of the original BERT. This model is introduced by (Sanh et al., 2019). It helps in maintaining the performance of the original BERT while reducing a lot of parameters. But how does it make that possible?

This model makes use of the ‘Knowledge Distillation’ which is a compression technique. For example, a student learns from a teacher (larger model) and then

reproduces the acquired knowledge. How does a student learn something from a teacher? It learns by using a Triple Loss.

2.7.1 Triple Loss

1. Distillation Loss (L_{ce})
    a. It measures the difference between the student's probabilities and the teacher's soft target probabilities (Bergmann D, 2024).
    b. It is defined as $L_{ce} = -\Sigma_i q_i \times \log(p_i)$, where q_i is the teacher's soft target probabilities, and p_i is the student's probabilities.
2. Masked Language Modelling Loss (L_{mlm})
3. Cosine Embedding Loss (L_{cos})
    a. It aligns the direction of the hidden states from the student and the teacher.

With the linear combination of these three, it trains the Distil BERT.

2.7.2 Training procedure of Distil BERT

Distil BERT is trained with the English Wikipedia and Toronto Book Corpus like the Original BERT trained with. It was trained on 8 Nvidia V100 GPUs for approximately 90 hours. (Sanh et al., 2019) said the model was trained efficiently by using gradient accumulation and dynamic masking without predicting the next sequence objective.

2.7.3 Results of Distil BERT

The authors evaluate this model with different benchmarks. On the General Language Understanding Evaluation benchmark (GLUE), the model maintains 97% of the original BERT's performance with significantly fewer parameters.

Model	Score	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Figure 2.7-1 Distil BERT retains 97% of BERT performance (Sanh et al., 2019)

The model also performed efficiently on other benchmarks like IMDb sentiment classification and Squad 1.1 QA.

2.8 Roberta model

The Roberta model's backronym is named the Robustly Optimized BERT Approach. Unlike the Distil BERT model, this model is neither smaller nor lighter than the original

BERT model. Instead, it improves from the BERT by building upon it. The model is made because (Liu et al., 2019) proposed the original BERT model was undertrained and wanted to give a better recipe for training BERT models.

2.8.1 Key features of Roberta

Roberta modified the original BERT with the following key features.

- It uses larger batch sizes for bigger training.
- The bigger the batch size, the longer the time to train the model.
- Original BERT used static masking where the model uses the same mask for every training. But Roberta uses Dynamic masking which means the masking pattern changes every time the model is fed with the sequence.
- BERT used the Next Sentence Prediction (NSP) which is the binary classification loss for predicting whether two segments are following each other or not in the original text. But Roberta removes NSP.
- It uses Byte-Pair Encoding (BPE) which is the sub-word tokenization algorithm, it breaks down the text into sub-word units instead of breaking down into words.
- The Roberta base model uses 12 layers, 12 Attention heads, and 768 hidden units.
- It was trained with these datasets: English Wikipedia, Book Corpus, CC-News, Open Web Text, and Stories from the Common Crawl.

2.8.2 Results of Roberta base model

The model is evaluated on multiple benchmarks: GLUE, Reading Comprehension from Examinations (RACE), and Squad. Here is the result of the model on GLUE.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS
$\text{RoBERTa}_{\text{BASE}}$ + all data + 500k steps	87.6	92.8	91.9	78.7	94.8	90.2	63.6	91.2

Figure 2.8-1 Roberta base model results on GLUE (Liu et al., 2019)

3 Literature review of QA

This Chapter provides an overview of the current QA projects using Distil BERT and Roberta models. By reviewing the works of literature, it can find out gaps in the existing projects that will contribute to this project.

There are many NLP-based systems such as customer support, text summarization, QA, etc. By the time Transformers architecture was released, the NLP systems had enhanced their performance and accuracy. BERT has also become significant among these NLP systems due to its efficiency and performance. Moreover, there are also variants of the original BERT model named Distil BERT and Roberta which have brought impressive NLP results. They can do various NLP tasks including the QA process.

The QA system will try to understand the question first by breaking it down and then the answer extraction process begins. For the text resource, it can be open-domain or closed-domain where the open-domain indicates the text resource is from the World Wide Web (WWW) and the closed-domain is from a specific subject field such as Music, Health, Weather Forecasting, etc. The Closed-Domain Question Answering (CDQA) system uses more NLP processes (Singh et al., 2016). When making the QA system, it should understand the question and the relationship between different resources from the knowledge base (Acharya et al., 2022). Different IQASs use different NLP techniques that can be Structural which focuses on Grammar, Logic, or Non-Structural which focuses on statistics and words (Aggarwal, 2011).

There are existing Python libraries that include the NLP functions that are ready to implement, and some projects make use of them. Different algorithms, Different formulas, and different techniques are used for individual advantages based on the field of their purposes. However, their performances can be crucial for consumers' experience. If the model is taking too long to load, a user would consider using it again or not, and if the model is not accurate, a user would not trust the system.

3.1 Performance comparisons on QA

3.1.1 Summary of the comparative study of BERT and Roberta on QA

(Akhila, 2023) provided the study on the performance comparison of BERT and Roberta models for the QA system. BERT introduced by (Devlin et al., 2018) used Bidirectional training which can understand the text more efficiently and Roberta

introduced by (Liu et al., 2019) enhanced the BERT model for performance using large batch sizes, etc.

This study also used the F1 score and the Exact Match for the Metric system. It indicates these two systems are vital to compare the accuracy of QA models. F1 is popular when dealing with classification in QA (Akhila, 2023).

BERT can train the data in a bidirectional way which can understand context from left to right or right to left. In that way, BERT has the advantage of understanding text and answering questions better.

Since Roberta takes a longer time to study, larger batch size to train and uses the Dynamic masking method, the Roberta model can outperform the BERT model in QA tasks.

The study used the Squad dataset in JSON format including both answerable and unanswerable QA pairs. The study used a technique called Inter Sent for Sentence Embedding. The models were trained for 3 epochs including the models with parameters such as IDs, attention masks, start positions, and end positions.

3.1.1.1 Analysis of the Performance

As the study shows, both models perform well but Roberta has better results compared to Distil BERT due to its enhancement techniques such as handling more data. After 3 epochs, the Roberta results are better because the model's training and validation loss is lower than the BERT and Distil BERT models. Since the Distil BERT is a lighter version of BERT, the model has a higher Validation loss than the competitor.

Table 3.1-1 Distil BERT base cased vs Roberta base

Model	**Training loss**	**Validation loss**
Distil Bert base cased	0.6412	1.3518
Roberta base	0.6517	1.0743

3.1.2 Analysis of the Squad v2 dataset

A study by (ÖZKURT, 2024) compares the performance of QA models, BERT, Roberta, Distil BERT, and ALBERT (A Lite BERT) on the Squad v2 dataset. ALBERT was introduced by (Lan et al., 2019), the model tries to maintain the original BERT performance while using parameter-sharing and factorized techniques.

F1 score and Exact match metric systems are involved in this study. Squad v2 dataset involves challenging data for the QA models. All models were equally trained for 3 epochs.

3.1.2.1 Analysis of the Performance

Table 3.1-2 QA models comparison (ÖZKURT, 2024)

Models	**F1 Score**	**Exact Match**
BERT Medium	70,11%	65,95%
Distil BERT	68,17%	64,88%
Roberta	82,91%	79,87%
ALBERT	89,91%	86,85%

According to the paper, Roberta achieved second place in the F1 score, it can understand the context and handle the unanswerable questions. Distil BERT has a smaller size and has the smallest F1 scores but delivers impressive results that can compete with its bigger models. However, the ALBERT competes over all the other models in this study with consistent results for both answerable and unanswerable questions.

3.2 Various NLP tasks

3.2.1 Sentiment Analysis

This study evaluates the NLP models BERT, Distil BERT, and Roberta for the analysis of Sentiment. Although this Honors project focuses on the QA task, this study evaluated by (Joshy & Sundar, 2022) also took QA tasks into account.

The study uses popular datasets for Sentiment analysis such Sentiment 140 dataset including 16 million tweets, and the Tweet dataset about Coronavirus including 44,955 tweets. The evaluation process includes tokenizing the words, removing the stop words, and lowercasing which are the tasks of cleaning the data. Then, the models are finetuned on those datasets and evaluated the performance with Accuracy, F1 score, Precision, and Recall metrics.

3.2.1.1 Analysis of the Performance for Sentiment

In the paper, the original BERT significantly outperformed both Roberta and Distil BERT models on Sentiment analysis tasks. It got better accuracy on both datasets.

But for the other two models, Roberta got slightly better results in terms of overall accuracy except for the Training accuracy on the Sentiment 140 dataset. Although Roberta is an enhanced version of BERT, Distil BERT can follow up with Roberta's Accuracy. This table shows the difference in performance of those two models on the Sentiment 140 dataset.

Table 3.2-1 Distil BERT vs Roberta in Sentiment 140 dataset (Joshy & Sundar, 2022)

Models	**Training accuracy**	**Validation accuracy**	**Testing accuracy**
Distil BERT	97%	75.7%	77.10%
Roberta	80.3%	78.13%	78.20%

3.2.2 Empirical comparisons

This study also compares the pre-trained models, Distil BERT, Roberta, BERT, and others for other than QA tasks. As the pre-trained models become popular among the Machine Learning community, the authors (Casola et al., 2022) chose to study the performance of pre-trained models on various NLP tasks.

Models are evaluated on multiple datasets for various NLP tasks including QA, Sentiment analysis, Named Entity Recognition, and text classification. Datasets are cleaned to achieve accurate performance from those pre-trained models on NLP tasks. Models are also finetuned for targeting tasks using task-specific training data. Tokenization, data normalization, and such data cleaning techniques are applied to process the Evaluation. For the metric systems, F1 score, and Exact match are used. This study approaches QA tasks in two tasks, Answer Sentence selection (AS2) and Machine Reading (MR). AS2 helps rank the question/sentence pair which can be considered as sentence classification and MR attracts attention in the Science Literature. For the QA task, the squad v1.1 dataset is used. The author used a constant learning rate for the first steps.

3.2.2.1 Analysis of the performance of Empirical

Roberta significantly outperforms the BERT, Distil BERT, and ALBERT. It achieves better results in both metrics. Compared to the study in Sentiment analysis stated in the previous section, Roberta outperforms the original BERT in this study since the task is QA and the chosen dataset is QA based. Distil BERT has a lower score than the BERT model and the lowest rank among the competition.

Table 3.2-2 Pre-trained models on Squad v1.1 dataset (Casola et al., 2022)

Metric	BERT	Roberta	Distil BERT	ALBERT
F1	87.51	91.82	84.13	90.25
EM	79.71	85.42	75.34	82.69

The study makes use of different learning approaches, Constant learning without a warmup, Constant learning with a warmup, and Linear learning with a warmup. The study shows there is very little difference between these learning approaches.

3.2.3 Comparison for Conversational QA (CoQA)

The study by (Staliunaite & Iacobacci, 2020) focuses on various Linguistic tasks such as Numerical questions, Negation, Sentiment, and Semantic role Labelling. As usual, BERT, Distil BERT, and Roberta models are tested on the conversation QA dataset. The dataset contains stories, QA pairs, and dialogues for Histories which are specifically designed to test the models' ability to generate the answers.

The authors applied tokenization to process the evaluation. Models' performances are measured with the F1 score metrics and trained for 4 epochs with the AdamW optimizer.

3.2.3.1 Analysis of the Performance of CoQA

Roberta model outperforms the other models followed by BERT and Distil BERT. Roberta dominates with the overall F1 score but with some disadvantages on Numerical tasks. Distil BERT ranked at the bottom due to its smaller size for the model. This is the overall F1 score for the models.

Table 3.2-3 Models on CoQA dataset

Metric	Roberta	BERT	Distil BERT
Overall F1	81.2	76.9	66.6

3.3 Summary of Literature Review

These models are not only tested on the QA dataset but also on the various datasets for various NLP tasks. From the results, Roberta takes the highest place among the other models on overall NLP tasks with slight performance loss on some tasks. But each model has its advantages.

Roberta is the enhanced version of BERT, achieving good performance through training longer and bigger, and employing Dynamic masking but takes computation resources compared to Distil BERT which is the efficient version of BERT.

From those findings, transformer-based models advance the NLP tasks with high accuracy. But they depend on the fine-tuning processes, resources, and duration. Roberta takes over others because it uses more resources than others. On the other hand, the Distil BERT model is ideal for a faster experience sacrificing the best accuracy.

Those findings could have focused on fewer computation resources to test the models' abilities, making the NLP models easily accessible from even smaller devices. Those papers provided a fair comparison between the models by using several techniques that need a quiet process to evaluate the results or manually loading the models and writing extensive codes to get the job done.

Some libraries support simpler implementation processes such as Pipeline for implementing the models without spending extra time on code while supporting efficiency. Because of that purpose, this project makes use of efficient tools to compare between the models.

4 Methodology

This project is the comparison of two pre-trained QA models that are finetuned on the parent model that is not specifically made for QA tasks. As mentioned earlier in the Background section, the models are named ‘Distil BERT base cased distilled Squad’ and ‘Roberta base model’. Both are finetuned with the QA datasets and made for QA tasks where a model will extract answers from the given context, which can also be considered closed-domain QA tasks (CDQA).

To compare these two models, a good metric system and a good testing dataset is essential. The word ‘good’ means, it must be effective and reasonable that bring a good contribution to the project. Therefore, the project uses not only one metric but two metric systems to compare and they are named the F1 score and the Exact Match. The dataset should be effective enough to compare between these models so that the project can evaluate their performance and accuracy. Since the model evaluation tasks are heavy on the machine, an effective and small dataset is chosen for the project. These pre-trained models are also already trained on well-known QA datasets, squad versions 1.1 and 2.0.

Here is the list of resources used for this project just in case to clarify. Also, some of these are not directly engaged with the project but it is also vital to provide the foundation of the models.

- Transformers Architecture
- Bidirectional Encoder Representations from Transformers (BERT)
- Distil BERT model
- Base Distil BERT QA model
- Roberta model
- Base Roberta QA model
- Customized QA dataset
- Exact Match metrics

- F1 Score metrics

4.1 Custom QA Dataset

The dataset contains the context, the question, and the answer. The context is a paragraph or sentence where the answer is involved. The question is the input sentence to be fed into the model, it can contain question types such as ‘Which’, ‘How’, ‘Why’, ‘What’, ‘When’, etc. The answer is the result of the combination of the context and the question.

For this project, JSON is used for the dataset format. It contains 30 QA data, each with different contexts involving several question types. Contexts are straightforward, but some questions are not used to be direct to trick the QA model. For example, the context is “Francis Ford Coppola had a problem with Soldiers while filming the Apocalypse Now film” and the question goes “What happened to Francis while filming?”. The question contains fewer exact words from the given context and lets the model predict. In this case, the answer is “a problem with Soldiers”.

These are the few data from the custom dataset used in this project.

Table 4.1-1 Custom dataset example

Context	Question	Answer
That Donut is Jason’s.	Whose Donut is that?	Jason
Post Malone is into the Country Music	Which music category does Post Malone into?	The Country Music
Scott went to the Supermarket at 10 AM.	When did Scott go?	10 AM

```
import json

# load dataset
with open('dataset.json', 'r') as file:
    test_data = json.load(file)
```

Figure 4.1-1 Reading JSON and storing in List object

For implementation, the dataset must be turned into the readable object that the QA models will be using at a time. Since the dataset is stored in JSON format, Python will read the file and store each of the data in the List object with ‘json.load’ function. In this way, two QA models can start the QA process.

4.2 Pipeline

The pipeline function is provided by the Hugging Face Transformers library. The function helps streamline the pre-trained models for various NLP tasks. Not only the QA tasks but also the other NLP tasks such as Text classification, Named Entity Recognition, Text summarization, etc. The pipeline is useful because it can reduce the complexity of installation for the project.

Without the pipeline, models must be manually set up like in the papers mentioned in the Literature Review. The tasks include Tokenization, Model loading, and others.

Pipeline is easy to use and allows the pre-trained models to be loaded in just a few lines of code. It can also support less computation resources, running the models efficiently. It can be loaded by importing the Transformers Library, defining the task, and setting the model's name. This is how the pre-trained models can be loaded with several types of QA tasks.

```
from transformers import pipeline

# Text summarization with Google T5 model
summarization = pipeline("summarization", model="google-t5/t5-base", tokenizer="google-t5/t5-base")
# English to France Translation
en_fr_translator = pipeline("translation_en_to_fr")
```

Figure 4.2-1 Using Pipeline for several NLP tasks

4.3 Implementation of Distil BERT QA model

This project makes use of the pre-trained and finetuned Distil BERT model that is developed by the Hugging Face. The model is finetuned on the Squad v1.1 dataset for specific QA tasks. The project will use the model `distilbert-base-cased-distilled-squad`.

To implement the model, import the Hugging Face Transformers library. Then, load the pre-trained model and task type. For this specific project 'question-answering' task is used.

```
from transformers import pipeline

# Load the pre-trained distil bert model
distilbert = pipeline("question-answering", model="distilbert-base-cased-distilled-squad")
```

Figure 4.3-1 Loading the Distil BERT Squad QA model

The model is loaded and then it is ready to start the QA task. The project used the custom dataset in JSON format. A dataset is loaded as a variable so, while the data is

looping by each, the model will perform a task for each context and question. All results from the model's task will then be appended to a specific result list at each process.

```
distilbert_predictions = []

for item in test_data:
    context = item['context']
    question = item['question']

    distilbert_answer = distilbert(question=question, context=context)

    distilbert_predictions.append(distilbert_answer['answer'])
```

Figure 4.3-2 Looping a dataset and performing Distil BERT QA task

That was the implementation of the pre-trained and finetuned Distil BERT model. Since it is made ready to use, the process was quick and easy. The next section will be implementing the Roberta model, and the process is just like this one.

4.4 Implementation of Roberta QA model

The project will use the `deepset/Roberta-base-squad2` model. It is based on the Roberta base model and finetuned with the Squad 2.0 dataset, which is made especially for extractive QA tasks. The dataset contains both answerable and unanswerable questions. As the Hugging Face stated in the model description, it uses 96 batch sizes, 2 epochs, 386 maximum sequence length, 3e-5 learning rate, 128 Document strides, and 64 maximum query lengths.

Just like the Distil BERT QA model, an implementation of this model is similar. The Transformers library is already loaded, load the Roberta model along with the tokenizer and the type of task.

```
# Load the pre-trained Roberta model
roberta = pipeline("question-answering", model="deepset/roberta-base-squad2", tokenizer="deepset/roberta-base-squad2")
```

Figure 4.4-1 Loading the Roberta base Squad2 QA model

The process of the QA task is also the same as Distil BERT, loop the dataset variable and assign the context and question to the model. Then, the model will process the Extractive QA task, and insert the results into a specific result list.

```
roberta_predictions = []

for item in test_data:
    context = item['context']
    question = item['question']

    roberta_answer = roberta(question=question, context=context)

    roberta_predictions.append(roberta_answer['answer'])
```

Figure 4.4-2 Looping a dataset and performing Roberta QA task

That was the implementation of two models. Simply, import, load, and then assign the task. The following sections will break down the set-up of Metric systems to compare these two models.

4.5 Implementation of Time

It is essential to know the difference in time to produce the results by each model. To capture the time of making the QA results, the ‘Time’ library by Python is imported.

```
import time
```

Figure 4.5-1 Import Time

To capture the process time for each process, the empty list for time is created initially. As usual, loop the dataset for the model to start the QA process. But before feeding context and questions to the Distil BERT and Roberta QA models, the current time (starting time) is recorded. After that, the model processes the QA task by taking a context and a question. Then, the current time (ending time) is recorded again. To get the actual time taken to process for one question, the starting time is subtracted from the ending time and the subtracted value is then stored in the list.

```
distilbert_predictions = []
distilbert_times = []

for item in test_data:
    context = item['context']
    question = item['question']

    # Capture Distil BERT process time
    start_time = time.time()
    distilbert_answer = distilbert(question=question, context=context)
    end_time = time.time()
    distilbert_predictions.append(distilbert_answer['answer'])
    distilbert_times.append(end_time - start_time)
```

Figure 4.5-2 Count time for each process (Distil BERT)

When the whole process for every context is finished, the time list has all the values of seconds the model takes to process. All elements from the time list are summed to get the total time for the model to process 30 QA.

```
# Get the total process time
total_distilbert_time = sum(distilbert_times)
```

Figure 4.5-3 Sum each process time for total time (Distil BERT)

That was the implementation of time for the Distil BERT model, and the same process applies to the Roberta model with a few changes of name. The total process times are stored as the values for Visualization.

4.6 Implementation of Metrics system

Before calculating both the F1 score and the Exact match scores, the model results and the dataset answers should be cleaned. By following that procedure, the comparison process will be effective with fairness.

4.6.1 Clean the answers

Initially, the answers are turned into lowercase so that there is no difference between the model answer and the dataset answer. The comparison result would be different than it should be without lower casing them all.

Secondly, there might be punctuation in the answers such as exclamation marks or question marks. They need to be cleaned too because their presence will affect the comparison result. Therefore, removing it will make the answers identical.

Thirdly, articles in the answers should be removed. Those are ‘a’, ‘an’, and ‘the’, such words that do not matter for the answer accuracy. The comparison data will be different without removing the articles from the answers.

Finally, unnecessary white spaces should be got rid of from the answers. The presence of extra white spaces will disturb the comparison results. Even the spacing of one can cause the difference.

Therefore, both answers should be cleaned by following those cleaning procedures. Here is the cleaning process with the coding implementation.

```
import re
import string
from collections import Counter


def remove_whitespace(text):
    """return removed white spaces text
    by splitting into words and join them again"""
    return ' '.join(text.split())


def remove_articles(text):
    """return removed articles text
    by using Regular Expression"""
    return re.sub(r'\b(a|an|the)\b', ' ', text)


def remove_punctuation(text):
    """return removed punctuation text
    by using string.punctuation"""
    return ''.join(ch for ch in text if ch not in set(string.punctuation))


def clean_answer(ans):
    """return lowercased text, removed punctuation,
    articles, and unnessary whitespaces."""
    return remove_whitespace(remove_articles(remove_punctuation(ans.lower())))
```

Figure 4.6-1 The methods of answer cleaning

White spaces are removed by splitting the words. Those words are then rejoined after. To remove the articles from the text, the project used the Regular expression. To remove the punctuation, the text is looped through to each character, ignoring the punctuation words, and then remade as a whole text. By implementing all those methods, the cleaned model answer is ready to be evaluated.

4.6.2 F1 scores

The F1 score is the evaluation metric to test the accuracy of the model. It is a reliable metric system that computes the prediction accuracy along the QA process with the specific dataset. It calculates by considering precision and recall.

The Precision is the number of true positives divided by false positives plus true positives which is how many positive results are predicted correctly by the model.

Equation 4.6-1 Precision

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

The recall is the number of true positives divided by false negatives plus true positives which is also how the model correctly identifies positive observation from the actual dataset.

Equation 4.6-2 Recall

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

This is the equation of the F1 score. The equation is in the harmonic means of Precision and Recall.

Equation 4.6-3 F1

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The implementation of the F1 score with Python is quite a complex process. It is wrapped into a function called `f1_scores`. The function is shown below.

```
def f1_scores(prediction, true_answer):
    """return the f1 scores"""
    # split the model answers into a list
    pred_tokens = clean_answer(prediction).split()
    # split the true answers into a list
    truth_tokens = clean_answer(true_answer).split()
    # common token between model tokens and true tokens
    common = Counter(pred_tokens) & Counter(truth_tokens)
    sum_common = sum(common.values())
    if sum_common == 0:
        return 0
    precision = sum_common / len(pred_tokens)
    recall = sum_common / len(truth_tokens)
    f1 = 2 * (precision * recall) / (precision + recall)
    return f1
```

Figure 4.6-2 The Implementation of F1 score metric

The function starts by splitting the model's prediction output texts into words (tokens) which means each token is stored as an element in a list of the text (e.g., ['from', '2003]). Then start splitting the actual answers from the dataset like splitting the model answers.

After splitting the text into tokens from both model and dataset answers, the function counts the frequency of each token. 'Counter' is a built-in Python class that counts the hash table items with the dictionary parameter. In this case, 'Counter' will store the token frequency as the dictionary variable. That will count the frequency of each token from both answers and then store it as a new variable where only common tokens (mutual tokens between model and dataset tokens) are taken with the Intersection

operator. All the common token values are then summed up to get the total number of both tokens. If there are no common token between both, the function returns 0.
To get the Precision value, the sum of the common token is divided by the length of the model's predicted tokens.

Equation 4.6-4 Precision in Project

$$Precision = \frac{sum\ of\ common\ token}{Length\ of\ the\ Model's\ predicted\ tokens}$$

To get the Recall value, the sum of the common token is divided by the length of the dataset's tokens.

Equation 4.6-5 Recall in Project

$$Recall = \frac{sum\ of\ common\ token}{Length\ of\ the\ Dataset\ tokens}$$

Those two values are then used in the F1 equation to calculate the F1 score.

4.6.3 Exact Match

The exact match metric system is straightforward to implement. It is also wrapped into a function called `exact_match_scores`. The function takes two parameters, model answers and dataset answers. The comparing answers are cleaned by calling the function `clean_answer`. It compares both answers as if they are matched or not. It returns the Boolean value based on the condition, True if both values are matched but False if both values are not the same. The implementation of the exact match function is shown below.

```
def exact_match_scores(prediction, true_answer):
    """Calculate Exact Match scores"""
    return clean_answer(prediction) == clean_answer(true_answer)
```

Figure 4.6-3 The Implementation of the Exact Match Score Metric

4.6.4 The evaluation process of both Metric systems

This section will be the purpose of this project, the evaluation of two models' answers with the F1 scores and the Exact Match. As those two metric system functions are stated previously, this section will use those functions to evaluate the model results. Another function called 'evaluate' is built. Here is the evaluation function.

```
# Function to evaluate
def evaluate(model_answers, dataset):
    """Calculate f1 score and exact match"""
    total_f1 = 0
    total_em = 0
    for model_ans, item in zip(model_answers, dataset):
        # extract only answer section from dataset
        dataset_answers = item['answer']
        # get a highest value from the f1 scores
        f1 = max(f1_scores(model_ans, da) for da in dataset_answers)
        # get a highest value from the em scores
        em = max(exact_match_scores(model_ans, da) for da in dataset_answers)
        total_f1 += f1
        total_em += em
    f1 = total_f1 / len(dataset)
    exact_match = total_em / len(dataset)
    return {'exact_match': exact_match, 'f1': f1}
```

Figure 4.6-4 The Implementation of the Evaluate function

The function starts with two values set as 0. They are the total values of the F1 score and the Exact match, ‘total_f1’ and ‘total_em’. These two values will be summed at each loop of calculating metric scores.

Model answers and Dataset answers are the parameters of the function. Model answers come with the List value. Dataset answers come with the Dictionary value in the List. The function will loop through both with the help of the Zip class which takes multiple iterable into a single iterable.

```
for model_ans, item in zip(model_answers, dataset):
    # extract only answer section from dataset
    dataset_answers = item['answer']
```

Figure 4.6-5 Getting only answers from the dataset

While looping two parameters at the same time as a pair, only the ‘answer’ key value is extracted from the dataset values for the sake of comparison. The extracted answer will outcome in the List value.

```
# get a highest value from the f1 scores
f1 = max(f1_scores(model_ans, da) for da in dataset_answers)
# get a highest value from the em scores
em = max(exact_match_scores(model_ans, da) for da in dataset_answers)
```

Figure 4.6-6 F1 and Exact Match

Another looping is made which iterates through each of the dataset List values. At each iteration, the F1 score function is called with two parameters, the model answer, and the dataset answer. The highest f1 score is taken as a new value called ‘f1’ and they

are summed to the previously assigned value called ‘total_f1’ at each iteration of List data. Calculating the Exact match is also the same as calculating the F1 score.

```
    total_f1 += f1
    total_em += em
f1 = total_f1 / len(dataset)
exact_match = total_em / len(dataset)
return {'exact_match': exact_match, 'f1': f1}
```

Figure 4.6-7 Storing Evaluation results

When the loop ends, two important values are made, those are the ‘total_f1’ and the ‘total_em’ scores. Each total value is divided by the length of the dataset (30 in this case). The reason is to provide the average scores across all the QA results.

The function returns the Dictionary value assigning the F1 result to the F1 key and assigning the Exact match result to the Exact Match key.

Conclusively, the evaluate function implements the f1 and the exact match functions by iterating both List values of the answers. The average score of the F1 and the Exact Match are the results of this Evaluation function. Therefore, the evaluate function is ready to be called with the real values placed in two parameters.

```
# Evaluate two models
distilbert_results = evaluate(distilbert_predictions, test_data)
roberta_results = evaluate(roberta_predictions, test_data)
```

Figure 4.6-8 Calling evaluate function

The model answers and the dataset values are already set up previously. Those are then called in the evaluate function. Distil BERT and Roberta evaluation results are stored in each Dictionary variable for visualization purposes.

4.7 Implementation of Visualization

4.7.1 Visualization for F1 score and Exact Match

After the results are stored as the Dictionary, Visualization is ready to be implemented with the result. Bar plots graph is chosen since it is a good option for showing the result effectively. This is how the visualization step is implemented.

```
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
```

Figure 4.7-1 Importing Matplotlib, Seaborn and Pandas

Matplotlib library is a popular library for creating plots, and interactive visualization. Seaborn is imported to enhance the visualization and Panda is for dealing with the result data.

```
# Storing results as Dictionary
results = {
    "Model": ["Distil BERT", "RoBERTa"],
    "Exact Match": [distilbert_results['exact_match'], roberta_results['exact_match']],
    "F1 Score": [distilbert_results['f1'], roberta_results['f1']]
}
```

Figure 4.7-2 Storing results as Dictionary

Previously, each model's results are stored as a Dictionary with its value. They are rearranged accordingly for visualization purposes. The dictionary contains 3 keys, and they are 'Model', 'Exact Match', and 'F1 score'. A list of values is set as the dictionary value with their key. In that way, the implementation process becomes clearer.

```
df = pd.DataFrame(results)
```

Figure 4.7-3 Dictionary to Pandas Data Frame

The stored dictionary value is converted into the Data Frame by using the Pandas library for easier implementation for making the plots.

```
# Figure size
plt.figure(figsize=(10, 5))
```

Figure 4.7-4 Plotting Exact Match scores

This code configures the size for the plot output. In this case, it is 10 inches in width and 5 inches in height.

```
# Exact Match Scores
plt.subplot(1, 2, 1)
sb.barplot(x="Model", y="Exact Match", data=df)
plt.title("Exact Match Scores")
plt.ylim(0, 1)
plt.ylabel("Exact Match Score")
plt.xlabel("Models")
```

Figure 4.7-5 Plotting Exact Match scores

The first line of code selects the first plot, configuring 1 row and 2 columns. Data is added into the plot with the seaborn library, the X axis is set as 'Model', and the Y axis is set as 'Exact Match' by using the data from the previously built Data Frame. For the Y axis, the number is limited from 0 to 1. X label and Y label are named accordingly.

```
# F1 Scores
plt.subplot(1, 2, 2)
sb.barplot(x="Model", y="F1 Score", data=df)
plt.title("F1 Scores")
plt.ylim(0, 1)
plt.ylabel("F1 Score")
plt.xlabel("Models")
```

Figure 4.7-6 Plotting F1 scores

This step is same as the creating a plot for the Exact Match plot but changes a few data for the F1 score. Replacing Exact Match data with the F1 score data for creating the Bar plots, and for Labels. In this way, the visualization is set up to evaluate the result of the Distil BERT base and Roberta base models.

4.7.2 Visualization for Total process time

Creating the plot for the total time of two QA models' process is the same as creating the plot for the F1 score and exact match. Previously, the total time process of each model is stored as the value. That value will be used in this step to create a visual presentation.

```
time_results = {
    "Model": ["Distil BERT base", "RoBERTa base"],
    "Time": [total_distilbert_time, total_roberta_time]
}
```

Figure 4.7-7 Storing the total time results and model names as Dictionary

Initially, the total time results are stored as a value for the key ‘Time’, and two models are stored as a value for the key ‘Model’. The dictionary data is ready to be used for creating a plot.

```
dft = pd.DataFrame(time_results)

plt.figure(figsize=(5, 5))
sb.barplot(x="Model", y="Time", data=dft)
plt.title("Total process time for Distil BERT base and Roberta base")
plt.ylabel("Total time (seconds)")
plt.xlabel("Models")
```

Figure 4.7-8 Plotting total process time of two models

The plot-creating process is the same but changes of name and data. This time, the plot size is set to 5 inches for both width and height. The Y label is set for Total time in seconds and the X label is set for Models. In this way, visualization for the total process time of two QA models is implemented.

5 Results and discussion

5.1 Results

The performance comparison of the two pre-trained QA models, Distil BERT (distilbert-base-cased-distilled-squad) and Roberta (deepset/Roberta-base-squad2), was evaluated using the F1 score and Exact Match (EM) metric systems.

F1 measures the models' accuracy by considering Precision and Recall. It is a good metric system because it offers a single result, including both False positives and False negatives. EM measures the models' accuracy by comparing the output to the ground truth answers.

Also, the processing time for each model is measured using Python's time library to understand the efficiency of the process. The QA results made by models will never change but process time makes new results whenever the process runs again. Therefore, this test runs the models 3 times to clarify. The models run on M1 MacBook Air with 8 GB of Memory.

These are the results of the models' evaluation:

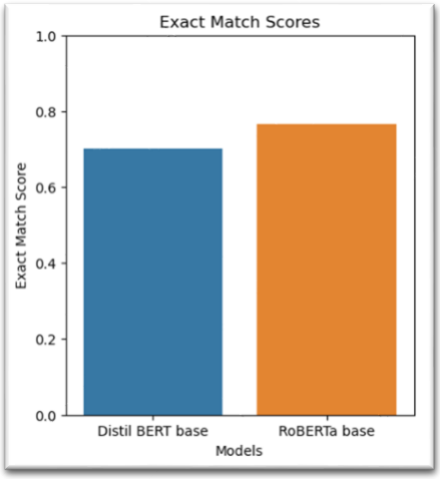

Figure 5.1-1 Exact Match results

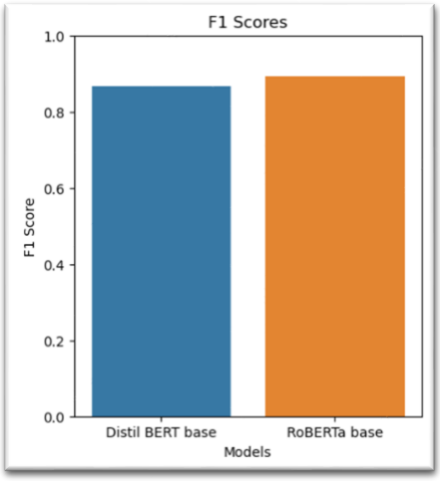

Figure 5.1-2 F1 Score result

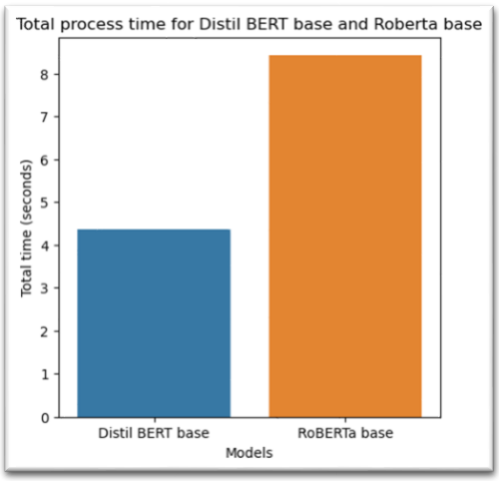

Figure 5.1-3 Processing time after running 3 times

Table 5.1-1 Pre-trained Distil BERT base and Roberta base results after running 3 times

Models	**EM**	**F1**	**Processing time**
distilbert-base-cased-distilled-squad	0.7	0.866	4.352 sec
Roberta-base-squad2	0.766	0.893	8.427 sec

5.2 Discussion

5.2.1 Models' Performance metrics

The Exact Match (EM) evaluates the model's performance by matching the model predictions and ground truth answers exactly. Distil BERT achieved an EM score of 0.7, while Roberta achieved higher EM of 0.76666666667. These results show that the

Roberta model is more accurate and can provide exact answers compared to the lighter version of BERT, Distil BERT.

The F1 score measures the models' performance by considering Precision and Recall. Distil BERT achieved an F1 score of 0.866464, while Roberta achieved a higher F1 score of 0.893015. This indicates that Roberta also outperforms the Distil BERT as the pre-trained model.

5.2.2 Processing Time

Processing time for the models is also important when making real-world applications. But for this experimental comparison between the pre-trained Distil BERT and Roberta models, the Distil BERT has a significant advantage in process time. It took 4.3525 seconds to process 30 questions while the Roberta model took 8.4274 seconds. Also, these are the results after running for 3 times.

Roberta took nearly 2 times increase in processing time. It shows that the Roberta model provides more accurate results but reduces the efficiency of computation compared to the Distil BERT.

These results show the trade-off between accuracy and processing time. Roberta's accuracy and higher score of metrics are good for the system that wants to be accurate. For example, critical decision-making, Medical QA, and political QA systems. On the other hand, Distil BERT's faster response is also suitable for applications that want the process to be efficient and do not require a high level of accuracy. For example, customer service bots. The reduced processing time can enhance the user experience but a few sacrifices on accuracy. Due to the results from this project, Distil BERT accuracy may be slightly less than Roberta's, but processing time is significantly efficient. Therefore, the Distil BERT model offers a balanced choice.

6 Conclusion

6.1 What this project achieved

This project evaluated the performance of finetuned pre-trained QA models, Distil BERT (distilbert-base-cased-distilled-squad) and Roberta (Roberta-base-squad2), analysing with the Exact Match (EM) and the F1 score metrics, measured for the accuracy of the models' output. The efficiency of the models' processing was also measured. The results highlight the trade-offs between each model.

The Roberta model achieved better performance in terms of accuracy got the EM score of 0.766666 and the F1 score of 0.8930. On the other hand, the Distil BERT model got an EM score of 0.70 and an F1 score of 0.86664. This shows that Roberta is desirable for making accurate answers. However, better accuracy comes with lower efficiency in processing time. Roberta took 8.42 seconds after running 3 times while the Distil BERT is 2 times more efficient and took only 4.3525 seconds after running 3 times. Integrating the models into the projects depends on the purpose. Distil BERT is a good choice for a faster experience since the model outputs the answer efficiently. Roberta is a good choice when a user wants the answer to be at best accuracy.

Conclusively, this Honors project provides insights into the performance and efficiency of two pre-trained models that are made easy to implement. These two models can be researched further as mentioned in future works.

6.2 Future works

Future works can focus on better usability and better performance of QA models. And can also integrate the pre-trained models into the projects by simply using pipelines. For the sake of models' optimization, models' parameter size can be reduced, and the speed of the process without impacting the accuracy heavily. Or applying knowledge distillation that trains small but maintains the accuracy with better efficiency. These two models can also be combined in a system where a user has the option to choose between these models. These Models can be finetuned to focus on specialized fields such as Medical, Political, Financial, etc. Projects can be incorporated with users to generate more accurate answers. For example, getting user feedback on generated answers' accuracy. New trustworthy metric systems can be also applied for deeper

insights into the models' performance and differences instead of relying only on F1 and Exact Match metrics.

6.3 Personal statement

I am excited to make this Research project since it is my very first professional project to make. By the fact of this is the first time, it comes with challenges. The experience was walking in the dark dessert, following the beams of light. Although, this field of study is trending and resource rich, it was hard to do, not having the experience of these.

This project is supported by academic papers, website articles, and books. Some of the papers are thoughtfully read and documented but are removed from the report because they became unfamiliar with the project scope. If I have a chance to engage more time on the Development, I would have used more prebuilt libraries to calculate metric scores.

This project structure is thorough and require details in each section. That motivates to get into details and give me more challenges.

References

Acharya, S., Sornalakshmi, K., Paul, B., & Singh, A. (2022). Question Answering System using NLP and BERT. *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, 925–929. https://doi.org/10.1109/ICOSEC54921.2022.9952050

Aggarwal, M. (2011). *Information Retrieval and Question Answering NLP Approach: An Artificial Intelligence Application*. https://api.semanticscholar.org/CorpusID:16143204

Akhila, N. (2023). Comparative study of bert models and roberta in transformer based question answering. *2023 3rd International Conference on Intelligent Technologies (CONIT)*, 1–5.

Bergmann D. (2024, April 16). *What is knowledge distillation?*

Casola, S., Lauriola, I., & Lavelli, A. (2022). Pre-trained transformers: an empirical comparison. *Machine Learning with Applications*, *9*, 100334.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv Preprint ArXiv:1810.04805*.

IBM. (n.d.-a). *What is Artificial Intelligence*. Retrieved July 15, 2024, from https://www.ibm.com/topics/artificial-intelligence

IBM. (n.d.-b). *What is Deep Learning?* Retrieved July 18, 2024, from https://www.ibm.com/topics/deep-learning

IBM. (n.d.-c). *What is Machine Learning*. Retrieved July 18, 2024, from https://www.ibm.com/topics/machine-learning

IBM. (n.d.-d). *What is NLP*. Retrieved July 18, 2024, from https://www.ibm.com/topics/natural-language-processing

Joshy, A., & Sundar, S. (2022). Analyzing the performance of sentiment analysis using Bert, Distilbert, and Roberta. *2022 IEEE International Power and Renewable Energy Conference (IPRECON)*, 1–6.

Kumar, A., Panwar, A., & Rawat, A. (2022). *Research Paper on Question Answering System using BERT*. https://doi.org/10.13140/RG.2.2.32542.20800

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv*, *abs/1909.11942*. https://api.semanticscholar.org/CorpusID:202888986

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv Preprint ArXiv:1907.11692*.

Nayak, P. (2019). *Understanding searches better than ever before*.

ÖZKURT, C. (2024). *Comparative Analysis of State-of-the-Art Q\&A Models: BERT, RoBERTa, DistilBERT, and ALBERT on SQuAD v2 Dataset*.

Patil, R., Patil, P. D., Joshi, Y., Khandelwal, S., Nalawade, S., & Palve, B. (2023). NLP Based Question Answering System. *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 1–5. https://doi.org/10.1109/ICCUBEA58933.2023.10392202

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv Preprint ArXiv:1910.01108*.

SEOMasterz. (2023, September 5). *Understanding Google's Semantic Search and Natural Language Processing (NLP)*.

Singh, S., Das, N., Michael, R., & Tanwar, P. (2016). The question answering system using NLP and AI. *International Journal of Scient Ific & Engineering Research*, *7*(12), 2229–5518.

Staliunaite, I., & Iacobacci, I. (2020). Compositional and Lexical Semantics in RoBERTa, BERT and DistilBERT: A Case Study on CoQA. *ArXiv*, *abs/2009.08257*. https://api.semanticscholar.org/CorpusID:221761454

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Appendix 1 Initial Project Overview

Title

Ask Me website: Q&A feature for your input content

Background

Everyone associated with the Internet and Technology may have used the Google Search Engine. Whenever the user asks the question, Google responds with the highlighted text as the top answer that is extracted from the web page source. That saves time visiting the web page and searching for the desired result on that page by the user himself. “Reading complex documents returned by a query to discriminate them or gathering them might be too time expensive.” (Kierszbaum & Lapasset, 2020, p. 1). Generally, it is the ability to understand the source context and the ability to offer a short answer with the help of BERT (Bidirectional Encoder Representations from Transformers). Google introduced “a neural network-based technique for natural language processing (NLP) pre-training called BERT” (Pandu Nayak, 2019). “BERT has inspired many recent NLP architectures, training approaches, and language models, such as Google’s TransformerXL, OpenAI’s GPT-2, etc.” (Basu et al., 2021, p. 2). With the rise of those technologies, this project tends to understand the input the user provides and give conversational features that will save a lot of time reading by using TensorFlow’s packages. Technically, it is a pre-trained BERT model npm package since this is for the website.

Aim and objectives

To provide a short answer for appropriate questions from the text resource the user provided.

1. Proper time management.
2. Research the machine learning architecture.
3. Search the relevant papers and the Literature.
4. Review the self-Literature.
5. Learn how to implement Packages and Library.
6. Develop the system.
7. Report the result to the Supervisor.
8. Conclude the project.

Plan

Stage	Name	Activities	Target date
1	Software Library	Search the package or library for the later use in software development (Early search may help in mid project phase)	6-Dec-23
2	IPO	Finish the Initial Project Overview, may edit later	15-Dec-23
3	Manage task	Manage time for upcoming tasks	17-Dec-23
4	Research Papers	Search papers that relevant to the project title and read	8-Jan-24
5	Literature Review	Research how to write the good literature review, define the methodology	29-Apr-24
6	Research algorithm	Try to understand the architecture that the author will be using, how they work out	17-May-24
6	Pratical work	Start the development by testing the searched packages, and then start building	10-Jun-24
7	Report and fix	Report the outcome to supervisor, make discussion and fix the bugs that may occur and change things that supervisor recommends	1-Jul-24
8	Conclude	Finish the discussion and evaluate the Project	10-Jul-24

Main deliverables

1. Literature Reviews
2. A finished Project Report
3. A Working system

Personal development

- To know Machine Learning, and to learn how it can be implemented on the website.
- To have a decent skill in using Google Search or Edinburgh Napier Library for better results.
- BERT architecture is unnoticeable for a typical user, but it has been attached to daily life. Knowing how this work is challenging.
- Need to know how to implement the chosen Library and Package to work and the background working techniques.
- Have learned the basic React front-end library but will have to learn the implementation with Machine Learning which will lead to an Advanced React usage level.
- Need to learn how to make good Reference using a tool (e.g., Mendeley).

Key challenges

- Not knowing the actuality of Machine Learning is the main challenge for the project since it is uncertain and new. To recover this issue, starting by getting recommendations from Chat-GPT, and reading a lot of relevant articles will help in understanding.
- Knowing the pre-trained Machine Learning Library or packages which is the TensorFlow in case, is quite an easy step after knowing Machine Learning but it will be frustrating to search for the right one so the real world and example projects that are implementing the desired Library should be reviewed. Choosing the wrong package can be a high risk because it may ruin the project flow.
- Implementing it in the traditional website because Machine Learning is new and using that package is also new, knowing how to implement is a hard task but it is not a big deal compared to choosing the right package.
- Having only basic knowledge of React Javascript Library. The author has built traditional websites and Express JS but has not tried the React library in production. So, that would be a challenge and would not be considered as risky since there are tons of React information on the Internet.

References

Basu, S., Gaddala, A., Chetan, P., Tiwari, G., Darapaneni, N., Parvathaneni, S., & Paduri, A. (2021). *Building a Question and Answer System for News Domain*.

Kierszbaum, S., & Lapasset, L. (2020). Applying Distilled BERT for Question Answering on ASRS Reports. *2020 New Trends in Civil Aviation (NTCA)*, 33–38. https://doi.org/10.23919/NTCA50409.2020.9291241

Pandu Nayak, G. F. and V. P. (2019, October 25). *Understanding searches better than ever before*.

Appendix 2 Project Diary with Supervisor

Student: Naing Khant Htet (40667695)

Supervisor: Dr. Aye Aye Myint

Date: 14 December 2023

Last diary date: 18 July 2024

Objectives:

To show the IPO.

Progress:

Have finished IPO with unreviewed Objectives and Project Background.

Supervisor's Comments:

explored background theory and set project objectives. Need to explore literature and theory background in details.

Date: 29 April 2024

Last diary date: 18 July 2024

Objectives:

To show my 1st Seminar Presentation.

Progress:

Done working on Keynote for 1st Seminar Presentation, including the Background, Objectives, and sketch of Methodologies.

Supervisor's Comments:

Studied project related methodologies to some extent. Need to do review on some related papers.

Date: 9 May 2024

Last diary date: 18 July 2024

Objectives:

To discuss my Project's Introduction and on-going Literature Review.

Progress:

Started working on Project Report Introduction including Background, Objectives, Aims and Structure of the Report.

Supervisor's Comments:

You studied some related papers. You need to explore the dataset that is to be used, library, tool.

Date: 15 May 2024

Last diary date: 18 July 2024

Objectives:

To show my 2nd Seminar presentation.

Progress:

Done the Keynote for 2nd Seminar Presentation including Gantt, Background Theories, Aims, Datasets, and Results.

Supervisor's Comments:

You have specified to use custom dataset. You need to consider comparing two algorithms.

Date: 10 June 2024

Last diary date: 18 July 2024

Objectives:

To discuss my fixed document from 2nd Seminar reviews.

Progress:

Fixed my Project document and Implementation as from the results of 2nd Seminar feedback.

Supervisor's Comments:

You designed tools, algorithms, library and comparison on two algorithms. You need to emphasis on development and evaluation.

Date: 19 June 2024

Last diary date: 18 July 2024

Objectives:

To show my on-going Development.

Progress:

In the middle of the Development progress.

Supervisor's Comments:

You finished partial implementation though.

You need to add result and discussion about evaluation metric.

Date: 1 July 2024

Last diary date: 18 July 2024

Objectives:

To show my on-going Project Document.

Progress:

Have done pretty much about Project Document: Introduction, Literature Reviews, Methodologies, Results, Discussion sections.

Supervisor's Comments:

You have done implementation, results and discussion work. You need to review thesis book in holistic view.

Date: 15 July 2024

Last diary date: 18 July 2024

Objectives:

To show my finalized Project.

Progress:

All sections of Project documents and Development are completed.

Supervisor's Comments:

Necessary tables, figures and lists are completed. Please check on problem statement and objectives.

Appendix 3 Gantt Chart

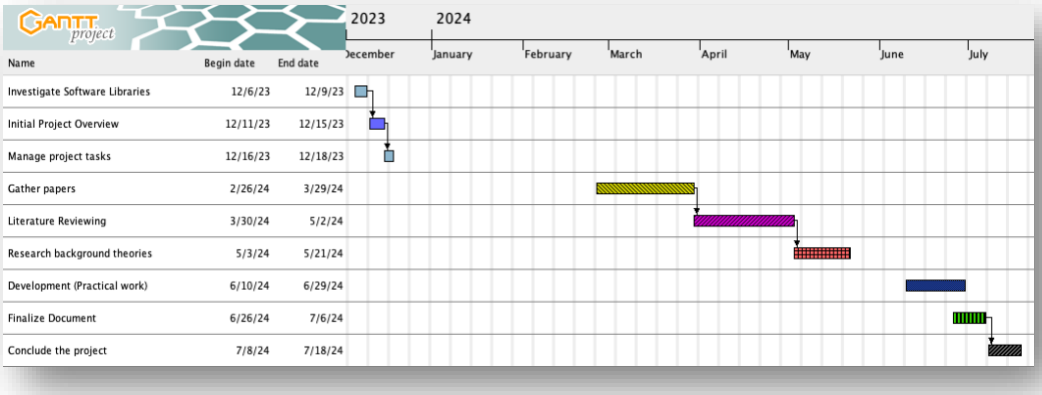

Figure 3.2.1: Project plan (Gantt Chart)