A project report on

E-GOVERNANCE: AI BASED QUESTION ANSWERING MODEL

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

PRANISH SHRESTHA (19BCE1758)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

E-GOVERNANCE: AI BASED QUESTION ANSWERING MODEL

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

PRANISH SHRESTHA (19BCE1758)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

DECLARATION

I hereby declare that the thesis entitled "E-GOVERNANCE: AI BASED QUESTION ANSWERING MODEL" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai, is a record of bonafide work carried out by me under the supervision of Dr. M. Asha Jerlin.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"E-GOVERNANCE: AI BASED QUESTION ANSWERING MODEL"** is prepared and submitted by **PRANISH SHRESTHA (19BCE1758)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering programme** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name:

Date:

Signature of the Examiner 1	Signature of the Examiner 2
Name:	Name:
Date:	Date:

Approved by the Head of Department
B. Tech. CSE

Name: Dr. Nithyanandam P

Date: 24 – 04 – 2023

(Seal of SCOPE)

ABSTRACT

This project focuses on building a question-answering system using the Hugging Face RoBERTa model and fine-tuning it on a custom dataset. The dataset consists of approximately 6,500 questions and answers, which were manually curated from various department websites. The system was trained and evaluated on this dataset using a combination of metrics, including accuracy and F1 score. The model was able to achieve an accuracy of 76%, demonstrating its effectiveness in answering questions based on the provided context. Additionally, the system was tested using a sample of questions generated by human judges, resulting in a success rate of 75%. This project contributes to the field of natural language processing by demonstrating the efficacy of fine-tuning pre-trained language models on custom datasets and highlights the importance of evaluating models on real-world questions.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. M. Asha Jerlin, Assistant Professor, SCOPE, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Malware Analysis.

It is with gratitude that I would like to extend thanks to our honorable Chancellor, Dr. G. Viswanathan, Vice Presidents, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan and Mr. G V Selvam, Assistant Vice-President, Ms. Kadhambari S. Viswanathan, Vice-Chancellor, Dr. Rambabu Kodali, Pro-Vice Chancellor, Dr. V. S. Kanchana Bhaaskaran and Additional Registrar, Dr. P.K.Manoharan for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dean, Dr. Ganesan R, Associate Dean Academics, Dr. Parvathi R and Associate Dean Research, Dr. Geetha S, SCOPE, Vellore Institute of Technology, Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, Project Coordinators, Dr. Abdul Quadir Md, Dr. Priyadarshini R and Dr. Padmavathy T V, B. Tech. Computer Science and Engineering, SCOPE, Vellore Institute of Technology, Chennai, for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staff at Vellore Institute of Technology, Chennai, who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **Pranish Shrestha**

CONTENTS

CHAPTER 5

IMPLEMENTATION

CHAPTER 6

CONCLUSION AND FUTURE WORKS

LIST OF FIGURES

LIST OF ACROYNMS

SQuAD: Stanford Question Answering Dataset

RoBERTa: Robustly Optimised BERT Approach

Chapter 1

Introduction

1.1 QUESTION ANSWERING MODEL

Question answering (QA) models are a type of artificial intelligence (AI) model that can automatically answer questions posed in natural language. These models have made significant progress in recent years, thanks to advances in deep learning and natural language processing techniques.

In a QA model, the system takes in a question in natural language, analyzes the question to understand its meaning, and then searches a database or a corpus of text documents for the information needed to answer the question. Once the relevant information is identified, the model generates an answer in natural language.

There are many types of QA models, including rule-based systems, template-based systems, and machine learning-based models. However, the most advanced and effective models today are based on deep learning techniques, such as transformer models like BERT and GPT. These models can understand the context and meaning of words and phrases and can generate highly accurate and relevant answers to complex questions.

1.2 PROBLEM STATEMENT

Various Acts and Rules, Policies, Notifications and FAQs are frequently released by the Government on their website in the form of pdfs and docs. These notifications are very important for both the bureaucrats as well as the common public, but are very hard to search online since it’s in documented form and only available in their respective websites. So, to provide an easy and a quick way for the bureaucrats as well as the common public to refer to any government policies or acts incase of any doubt or uncertainty in a specific topic.

In addition, government agencies often have limited resources to respond to the high volume of inquiries from citizens. This leads to long wait times and slow response times, further reducing the effectiveness of e-Governance systems.

The goal of the AI-Based Question Answering Model is to address these challenges by providing citizens with a fast, accurate, and user-friendly way to access information about government policies and services. The model will be trained on a large corpus of government-related text data, including policies, regulations, and official documents, to provide citizens with accurate answers to their questions.

By incorporating the latest advancements in AI and NLP, the AI-Based Question Answering Model will provide citizens with instant access to the information they need, reducing frustration and increasing engagement with government services. At

the same time, the model will reduce the burden on government agencies, enabling them to respond more efficiently to citizen inquiries and improve the overall effectiveness of e-Governance systems.

1.3 OBJECTIVES

- Easy to use: To provide a simple platform for the common public and the bureaucrats to refer and gain knowledge on any government information.

- Accuracy: To provide accurate and correct answers to questions asked by users. The model should be able to understand the context of the question and provide a precise and relevant answer.

- Speed: To provide answers quickly. Users expect a quick response, and the model should be able to process and analyze large amounts of data quickly to provide accurate answers in a timely manner.

Chapter 2

Literature Review

The following works were used as a basis and to help with understanding concepts of this research:

[1] Puri, R., Spring, R., Shoeybi, M., Patwary, M., & Catanzaro, B. (2020, November). Training Question Answering Models from Synthetic Data. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 5811-5826).

Summary: The authors create massive question answering (QA) datasets from unlabelled Wikipedia pages and fine-tune a 345 million parameter BERT-style model in this study. Finally, they generate synthetic text from Wikipedia, generate answer candidates, then generate synthetic questions based on those answers, and then train a BERT-Large model to attain similar question answering accuracy without utilising any real data at all.

[2] Ji, Z., Xu, F., Wang, B., & He, B. (2012, October). Question-answer topic model for question retrieval in community question answering. In Proceedings of the 21st ACM international conference on Information and knowledge management (pp. 2471-2474).

Summary: The authors offer a unique Question-Answer Topic Model (QATM) for learning latent semantic themes from question answer pairs for question retrieval in this research. Their model significantly outperformed the other topic models when experimented on a real world CQA dataset.

[3] Sneiders, E. (2002, June). Automated question answering using question templates that cover the conceptual model of the database. In the International Conference on Application of Natural Language to Information Systems (pp. 235-239). Springer, Berlin, Heidelberg.

Summary: The authors of this research have modified the approaches used to respond to FAQs to questions including information from a structured database, such as a relational one. When a user asks a question, the question-answering system, known as question assistant, compares it to a variety of question templates, which are dynamic, parameterized "frequently asked questions."

[4] Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H., & Li, X. (2016, June). Neural Generative Question Answering. In Proceedings of the Workshop on Human-Computer Question Answering (pp. 36-42).

Summary: The study proposes an end-to-end neural network model called Neural Generative Question Answering (GENQA), which can produce responses to straightforward factoids inquiries based on knowledge-based facts. The dataset is created by collecting data from three Chinese encyclopaedia web sites and two Chinese community QA sites using the Aho-Corasick string searching algorithm.

[5] Xiong, C., Zhong, V., & Socher, R. (2017). Dynamic coattention networks for question answering. In International Conference on Learning Representations (ICLR).

Summary: The study proposes an end-to-end neural network called Dynamic coattention networks (DCN) that solves the problem of models not being able to recover from local maxima corresponding to incorrect answers due to their single pass nature. The authors have used a baseline that is a simple Seq2Seq model consisting of 2 bi-directional LSTMs. One for questions and another for the documents.

[6] Perez, E., Lewis, P., Yih, W. T., Cho, K., & Kiela, D. (2020, November). Unsupervised Question Decomposition for Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 8864-8880).

Summary: The study offers a QA system that answers a question by decomposing it into numerous sub-questions using One-to-N Unsupervised Sequence Transformation (ONUS), answering sub-questions with an off-the-shelf QA system, and recomposing sub-answers into a final response.

[7] Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Learning to Compose Neural Networks for Question Answering. In Proceedings of NAACL-HLT (pp. 1545-1554).

Summary: A question-answering paradigm is presented that may be used with both structured knowledge bases and visuals. The model autonomously constructs neural networks from a set of modular components using natural language strings. These modules' parameters are learned by reinforcement learning together with the network-assembly parameters.

[8] Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., & Daumé III, H. (2014, October). A neural network for factoid question answering over paragraphs. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 633-644).

Summary: The paper presents a dependency-tree recursive neural network, QANTA for factoid question answering that outperforms bag of words and information retrieval baselines. The authors, from their experiments, find that sentence-level representations can be easily and effectively combined to generate paragraph-level representation with more predictive power than those of the individual sentences.

[9] Wang, D., & Nyberg, E. (2015, July). A long short-term memory model for answer sentence selection in question answering. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 707-712).

Summary: The paper proposes a method that addresses the answer sentence selection problem for question answering by using a combination of stacked bidirectional Long-Short Term Memory (BLSTM) network and keyword matching, to sequentially read words from question-and-answer sentences, and then outputs their determined scores.

[10] Khot, T., Clark, P., Guerquin, M., Jansen, P., & Sabharwal, A. (2020, April). Qasc: A dataset for question answering via sentence composition. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 05, pp. 8082-8090).

Summary: The paper presents Question Answering via Sentence Composition (QASC), a multi-hop reasoning dataset that calls for assembling information from a wide corpus in order to respond to a multiple-choice question. The facts to be created are annotated in a sizable corpus, and the decomposition into these facts is not obvious from the question itself, making QASC the first dataset to satisfy both of these desirable qualities.

[11] McCarley, J. S., Chakravarti, R., & Sil, A. (2019). Structured pruning of a BERT-based question answering model. arXiv preprint arXiv:1910.06360.

Summary: The authors look into structured parameter pruning from the underlying transformer model to compress BERT and RoBERTa based question answering systems. They discover that a simple combination of task-specific structured pruning and task-specific distillation, without the cost of pretraining distillation, produces models that perform well over a variety of speed/accuracy tradeoff operating points.

[12] Kenton, J. D. M. W. C., & Toutanova, L. K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT (pp. 4171-4186).

Summary: The paper introduces the Bidirectional Encoder Representations from Transformers (BERT) model, which is pre-trained on large amounts of text data and can be fine-tuned for various natural language processing tasks, including question answering.

[13] Chen, D., Bolton, J., & Manning, C. D. (2016, August). A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2358-2367).

Summary: The paper studies the effectiveness of various neural network models for answering questions based on textual data, using the CNN/Daily Mail dataset. The authors propose a new model that achieves better performance than previous methods.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Summary: The paper introduces the Transformer, a neural network architecture based solely on attention mechanisms, which achieves state-of-the-art results on various natural language processing tasks, including question answering.

[15] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1757-1766).

Summary: The paper proposes a neural network architecture based on bidirectional attention flow for answering questions based on textual data. The authors show that their model achieves state-of-the-art performance on various benchmark datasets.

[16] Dua, D., & Wang, W. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 2369-2380).

Summary: The paper introduces a new reading comprehension benchmark called DROP, which requires discrete reasoning over paragraphs. The authors compare different models on this benchmark and demonstrate that it is a challenging task that requires more than simple span extraction.

[17] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., & Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In Proceedings of the 33rd International Conference on Machine Learning (ICML) (pp. 1378-1387).

Summary: The paper proposes a model called Dynamic Memory Networks (DMN) for natural language processing tasks. The DMN uses a memory network to store and retrieve information over multiple iterations to generate a response. The authors evaluate their model on multiple datasets and show that it outperforms several baselines.

[18] Lee, K., Lee, K., Lee, S., & Lee, S. (2020). Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 5295-5305).

Summary: The paper introduces a globally normalized BERT model for open-domain question answering, called Multi-passage BERT. The model combines evidence from multiple passages to answer a given question. The authors demonstrate that their model outperforms existing models on the Natural Questions dataset.

[19] Reddy, S., Chen, D., & Manning, C. D. (2019). CoQA: A conversational question answering challenge. Transactions of the Association for Computational Linguistics, 7, 249-266.

Summary: The paper introduces a conversational question answering challenge called CoQA, which involves answering questions based on a conversation. The authors evaluate several models on this dataset and show that the task is challenging due to the need to understand the context of the conversation.

[20] Wang, Y., Huang, S., & Sun, M. (2019). Multi-hop reading comprehension through question decomposition and rescoring. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 2334-2344).

Summary: The paper proposes a multi-hop reading comprehension model that decomposes a given question into sub-questions and uses a two-stage rescoring process to generate an answer. The authors demonstrate that their model outperforms several baselines on the HotpotQA dataset.

Chapter 3

Proposed System

My proposed system is an AI-based question-answering model that uses the Hugging Face RoBERTa model is an advanced machine learning solution that leverages the power of deep learning techniques to provide accurate and relevant answers to natural language questions.

Hugging Face RoBERTa model is a pre-trained language model developed by OpenAI that has achieved state-of-the-art results on a wide range of NLP tasks. The model is fine-tuned to the SQuAD 2.0 dataset format to learn the relationship between questions and answers and provide accurate answers to questions posed by users.

My model uses a dataset of 6,500 question-answer pairs that I have created myself for training and testing purposes. Additionally, I have also created a separate set of 100 questions with context for testing my system's performance.

System Diagram:

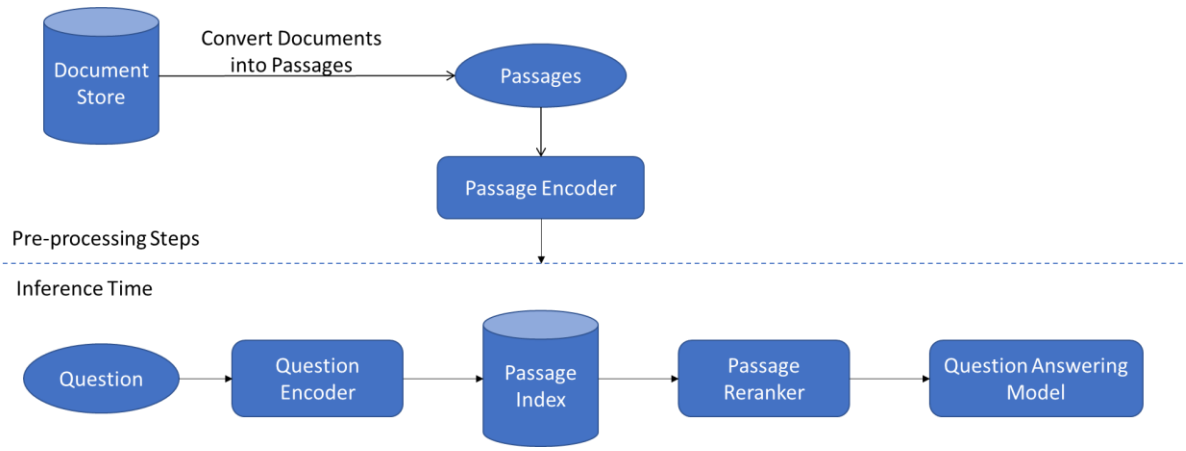

Figure 3.1: Proposed System Diagram

Chapter 4

List of Modules

a) Dataset Acquisition:

b)

i. Various Acts and Rules, Policies, Notifications and FAQs were scrapped from Government websites and required data was collected in the form of titles and paragraphs and question and answers were generated accordingly.
ii. This data was stored in text files and later converted to json files using Python Scripts as a part of the data pre-processing.
iii. A dataset of approximately 6600 questions along with its answer was formed based on 10 different government departments.

c) Data Pre-processing:

i. Data Cleaning: Before processing the file, we remove any unnecessary characters like extra whitespaces or special characters that might interfere with the analysis.
ii. Tokenization: The text data is being tokenized using the Hugging Face tokenizer. The encode function of the tokenizer is used to convert the input text into a list of token IDs, which can be used as input to the model.
iii. Padding: The input sequences are being padded with zeros to ensure that they all have the same length.
iv. Attention masks: The attention mask is a binary mask that tells the model which tokens are part of the input sequence and which tokens are padding. Here, the attention mask is being created by setting all the tokens that are not padding to 1.

d) Model Training:

i. Defining the model architecture: A model is defined using the Keras API. The architecture consists of three input layers for the token IDs, attention mask, and token type IDs, respectively. The pre-trained RoBERTa model is loaded and passed the input layers to obtain the encoded output.
ii. Compiling the model: The model is compiled using the Adam optimizer with a learning rate of 3e-5 and categorical cross-entropy as the loss function.
iii. Fitting the model: The model is trained using the fit() method. The input data consists of the token IDs, attention mask, and token type IDs, while the output data consists of the start and end tokens of the

answer span. The model is trained for three epochs with a batch size of 4 and a validation split of 0.1.

iv. Saving the trained model: After the model is trained, the weights are saved for future use.

e) Model Testing:

i. After the model is prepared, we test the model by feeding in a set of questions either by inputting it one at a time, or by reading a document with a set of questions.

ii. After generating all the answers we calculate the accuracy of the model.

Chapter 5

Implementation

5.1 DATASET

One of the main challenges faced during this project was the data acquisition process. As the required data was not readily available, it was necessary to manually search through various department websites and go through numerous documents to gather the necessary questions and answers for the dataset. This process was time-consuming and required significant effort, but was essential to ensure the quality and relevance of the dataset used for the project.

5.1.1 ACQUIRING THE DATASET

The dataset for this research was acquired through manual extraction of information from documents available on various Government department websites, such as Ministry of Electronics and Information, Ministry of Culture, Ministry of Social Justice and Empowerment, etc.

Example: text file

Title: Holding of the Private Secretary Departmental Competitive Examination

Content: Holding of the examination. - (1) The appointing authority shall notify the dates and place of the examination and number of available and anticipated vacancies assessed at the time of announcing the examination. (2) The examination shall be conducted after the expiry of at least thirty days from the date of notifying the vacancies in the manner specified in the Schedule which may be held before the 31st March of the recruitment year. (3) The number of vacancies reserved for the Scheduled Castes, the Scheduled Tribes and other special categories of person, if any, shall be clearly indicated in the notification.

Question: Who notifies aspirants about the date, place and time of the examination?

Answer: The appointing authority shall notify the dates and place of the examination and number of available and anticipated vacancies assessed at the time of announcing the examination.

Question: When is the Private Secretary Departmental Competitive Examination held?

Answer: The examination shall be conducted after the expiry of at least thirty days from the date of notifying the vacancies in the manner specified in the Schedule which may be held before the 31st March of the recruitment year

5.1.2 CONVERTING THE TEXT FILE TO JSON IN SQuAD 2.0 FORMAT

The text file is then converted into a json file in SQuAD 2.0 format using a python script [Appendix 1].

Example:


```
{
  "version": "v2.0",
  "data": [
    {
      "title": "Holding of the Private Secretary Departmental Competitive Examination",
      "paragraphs": [
        {
          "context": "Holding of the examination. - (1) The appointing authority shall notify the dates and place of the examination and number of available and anticipated vacancies assessed at the time of announcing the examination. (2) The examination shall be conducted after the expiry of at least thirty days from the date of notifying the vacancies in the manner specified in the Schedule which may be held before the 31st March of the recruitment year. (3) The number of vacancies reserved for the Scheduled Castes, the Scheduled Tribes and other special categories of person, if any, shall be clearly indicated in the notification.",
          "qas": [
            {
              "question": "Who notifies aspirants about the date, place and time of the examination?",
              "id": "id_0_0_1",
              "is_impossible": false,
              "answers": [
                {
                  "answer_start": 34,
                  "text": "The appointing authority shall notify the dates and place of the examination and number of available and anticipated vacancies assessed at the time of announcing the examination."
                }
              ]
            },
            {
```

"question": "When is the Private Secretary Departmental Competitive Examination held?",

"id": "id_0_0_2",

"is_impossible": false,

"answers": [

{

"answer_start": 217,

"text": "The examination shall be conducted after the expiry of at least thirty days from the date of notifying the vacancies in the manner specified in the Schedule which may be held before the 31st March of the recruitment year"

}] }] }] }] }

5.2 TRAINING

The model was trained for 3 epochs with a batch size of 4 and a validation split of 0.1. During training, the model optimized the loss function using the Adam optimizer with a learning rate of 5e-5. The loss function used was the categorical cross-entropy, which compares the predicted probability distribution of the model with the actual distribution of the target output. The training time for the model was approximately 24 hours.

Epoch 1/3

1236/1236 [==============================] - 23096s 19s/step - loss: 7.7488 - activation_loss: 3.7582 - activation_1_loss: 3.9906 - val_loss: 7.5948 - val_activation_loss: 3.4880 - val_activation_1_loss: 4.1068

Epoch 2/3

1236/1236 [==============================] - 43794s 35s/step - loss: 6.5194 - activation_loss: 3.2028 - activation_1_loss: 3.3166 - val_loss: 7.3293 - val_activation_loss: 3.4432 - val_activation_1_loss: 3.8860

Epoch 3/3

1236/1236 [==============================] - 16231s 13s/step - loss: 5.6215 - activation_loss: 2.7872 - activation_1_loss: 2.8343 - val_loss: 6.9012 - val_activation_loss: 3.2906 - val_activation_1_loss: 3.6105

Figure 5.1: Fitting the RoBERTa model with our dataset

5.2.1 TOOLS USED

We used several tools and packages to implement our system, including:

1. TensorFlow: a popular open-source platform for building and training machine learning models.
2. Hugging Face Transformers: a library that provides state-of-the-art pre-trained models for natural language processing.
3. Keras: a high-level neural networks API that simplifies the process of building and training deep learning models.
4. Pandas: a data manipulation library for Python that allows for easy data processing and analysis.
5. NumPy: a library for numerical computing in Python that provides support for large, multi-dimensional arrays and matrices.
6. Scikit-learn: a machine learning library that provides a range of tools for classification, regression, clustering, and more.

These tools were used to preprocess the data, build and train the model, and evaluate its performance.

5.3 TESTING

Testing is a crucial step in the implementation of any machine learning project, and my project is no exception. To ensure that the model is functioning correctly and producing accurate results, several types of testing were carried out.

5.3.1 TYPES OF TESTING DONE

1. The first type of testing involved directly inputting the context and question in the code and evaluating the generated answers.

Testing 1

```
con = 'Cancer is a group of diseases involving abnormal cell\
 growth with the potential to invade or spread to other parts of the body\
 . These contrast with benign tumors, which do not spread.\
  Possible signs and symptoms include a lump, abnormal bleeding\
  , prolonged cough, unexplained weight loss, and a change in\
   bowel movements. While these symptoms may indicate cancer,\
    they can also have other causes. Over 100 types of cancers \
    affect humans'
que = 'What disease involves abnormal cell growth?'
ans = generate_ans(con,que,model,tokenizer)
print(ans)
```

```
1/1 [==============================] - 4s 4s/step
 cancer is
```

Figure 5.2: Testing my model by directly inputting the context and question

2. The second type of testing involved inputting a CSV file with a set of questions and contexts and saving the generated answers.

	question	context
0	What status has the Brotherhood obtained in th...	Despite periodic repression, the Brotherhood h...
1	What impact does higher worker productivity an...	In Marxian analysis, capitalist firms increasi...
2	What is the goal of individual civil disobedie...	Non-revolutionary civil disobedience is a simp...
3	What is set aside for question periods in the ...	Parliamentary time is also set aside for quest...
4	What year was the University of Warsaw establi...	The University of Warsaw was established in 18...

Figure 5.3: The first 5 data of the file used for testing

```
1/1 [==============================] - 1s 919ms/step
16,
1/1 [==============================] - 1s 886ms/step
 a progressive tax is
1/1 [==============================] - 1s 917ms/step
 there were 158,349 households
1/1 [==============================] - 1s 931ms/step
's full-sized cars reflected the crisis. by 1979, virtually all "full
1/1 [==============================] - 1s 920ms/step
 teaches nearly 10,000 pupils. the
1/1 [==============================] - 1s 891ms/step
 in 1872, the central
1/1 [==============================] - 1s 930ms/step
 of feynman diagrams.
1/1 [==============================] - 1s 903ms/step
 that disobedience
1/1 [==============================] - 1s 957ms/step
ogram-force (kgf) (sometimes kilopond), is the force exerted
1/1 [==============================] - 1s 893ms/step
```

Figure 2.4: The function predicting the answer for each question

This allowed for a larger-scale testing of the model's performance. Furthermore, the performance of the model was evaluated using standard evaluation metrics such as F1 score and accuracy. Overall, rigorous testing was conducted to ensure that the model was performing optimally and providing accurate answers to the given questions.

5.3.2 STEPS FOLLOWED FOR TESTING

In order to evaluate the performance of the model, a testing procedure was designed:

1. Selected two judges to generate questions and answers for the dataset.
2. Provided the judges with 50 text files each to generate questions and answers.
3. Created a CSV file containing only the context and the questions generated by the judges.
4. Used the trained model to generate answers for the questions in the CSV file.
5. Compared the predicted answers to the answers given by the judges.
6. Considered the test successful if the predicted answer matched the judges' answer with an accuracy of approximately 75%.
7. Calculated the accuracy of the model as the number of successful answers out of 100.
   Accuracy = No. of successful answers/100

Chapter 6

Conclusion and Future Works

6.1 RESULTS

One hundred questions were generated by two judges who were provided with 50 text files each. The judges' answers were then compared to the answers generated by the model, which were stored in a CSV file. Out of the 100 questions, 76 were answered correctly by the model. However, all of the model-generated answers were of the long answer type, whereas the judges' answers were mostly one-word or short answers. This difference may be attributed to the dataset used in the training process, which primarily contained long answers for the given questions.

6.2 CONCLUSION

For my project, I trained a deep learning model using a pre-trained RoBERTa model from Hugging Face to generate answers for a given context and question. I created a dataset of approximately 6500 question-answer pairs by going through documents uploaded on department websites. I trained the model using Keras and TensorFlow, with a batch size of 4 and 3 epochs. During testing, I directly inputted the context and question in the code and also inputted a CSV file with a set of questions and contexts to save the generated answers.

To test the model, I selected two judges and asked them to generate questions and answers for 50 text files each. I then compared the predicted answers from the model to the answers given by the judges. I found that 76 out of 100 answers were correct, but all of the answers generated by the model were long answer types, whereas the answers given by the judges were one-word or short answers. This could be because the dataset contained long answers for the questions.

Overall, the project shows promising results, but further improvements can be made by refining the dataset to include more diverse types of answers and testing on a larger sample size.

6.3 FUTURE WORK

1. Develop a more interactive user interface to allow for easier input and output of questions and answers, as well as allowing users to provide feedback and corrections to the model's output.
2. Investigate the use of multi-lingual models to support question answering in multiple languages, which could greatly improve accessibility for non-English speakers.
3. Expand the dataset used to train the model to include a wider range of topics and contexts, and incorporate more diverse and representative sources.

Appendices

Appendix 1: Python script for converting text file to json in SQuAD 2.0 format:

```
class QA_data():
    def __init__(self, filepath):
        self.path = filepath
        self.final_json = {}
        self.final_json['version'] = "v2.0"
        self.final_json['data'] = []

    def get_loc(self, answer, content):
        loc = content.lower().find(answer.lower())
        return loc

    def into_json(self):
        with open(self.path, 'r',encoding='utf-8') as f:
            topics = f.read()
            for k, items in enumerate(topics.split("\n\n\n\n")):
                d0 = {}
                d0['title'] = items.split("\n\n", 1)[0].replace("Title:", "").strip()
                d0['paragraphs'] = []
                for j, item in enumerate(items.split("\n\n", 1)[1].strip().split("\n\n\n",)):
                    d1 = {}
                    d1['context'] = item.split("\n\n")[0].replace("Content:", "").strip()

                    d1['qas'] = []
                    for i, section in enumerate(item.split("\n\n")):
                        if  i != 0:
                            d2 = {}
                            d2['question'] = section.split("\n")[0].strip().replace("Question:",
"").strip()
                            ans = section.split("\n")[1].strip().replace("Answer:", "").strip()
                            d2['id'] = f"id_{k}_{j}_{i}"
                            loc = self.get_loc(ans, d1['context'])

                            d2['is_impossible'] = True if loc == -1 else False

                            if loc == -1:
                                d2['answers']= []
                            else:
                                d2['answers']= [{'answer_start': loc,"text":ans}]

                            d1['qas'].append(d2)

                    d0['paragraphs'].append(d1)
```

```
            self.final_json['data'].append(d0)

        return self.final_json
```

Appendix 2: Converting json file to csv

This conversion was done for ease of creating a dataframe and working with the data.

```
import json
import pandas as pd

f = open('../final/dataset/resultTotal.json') #training
data = json.load(f)

titles = []
contexts = []
questions = []
ids =[]
answers = []
answer_start = []
is_impossible = []
for i in data['data']:
  title = i['title']
  for j in i['paragraphs']:
    context = j['context']
    for k in j['qas']:
      question = k['question']
      id_ = k['id']
      is_impossible_ = k['is_impossible']
      for l in k['answers']:
        titles.append(title)
        contexts.append(context)
        questions.append(question)
        ids.append(id_)
        answers.append(l['text'])
        answer_start.append(l['answer_start'])
        is_impossible.append(is_impossible_)

train = pd.DataFrame({'title':titles, 'question':questions, 'id':ids, 'answers':answers,
'answer_start':answer_start ,'context':contexts})
train.to_csv('train.csv')
```

Appendix 3:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow.keras.backend as K
from sklearn.model_selection import StratifiedKFold
from transformers import *
import tokenizers
print('TF version', tf.__version__)

MAX_LEN = 512

tokenizer = tokenizers.ByteLevelBPETokenizer(
    vocab = 'D:/capstone/roberta-base-squad2/vocab.json',
    merges = 'D:/capstone/roberta-base-squad2/merges.txt',
    lowercase = True,
    add_prefix_space = True
)

train = pd.read_csv('train.csv').fillna('')
train.head()

rec = train.shape[0]  # Number of records in the training set
inputs = np.ones((rec, MAX_LEN), dtype = 'int32') # Input vector
attention_mask = np.zeros((rec, MAX_LEN), dtype = 'int32') # Attention Mask
token_type_ids = np.zeros((rec, MAX_LEN), dtype = 'int32') # Tokens produced
start_tokens = np.zeros((rec, MAX_LEN), dtype = 'int32') # Start logit for answer
end_tokens = np.zeros((rec, MAX_LEN), dtype = 'int32') # End logit for answer

for i in range(rec):

  context = ' '+' '.join(train.loc[i, 'context'].split())
  answer = ' '+' '.join(train.loc[i, 'answers'].split())
  question = ' '+' '.join(train.loc[i, 'question'].split())

  start_idx = train.loc[i, 'answer_start']
  chars = np.zeros((len(context)))
  chars[start_idx:start_idx + len(answer)] = 1
  if context[start_idx - 1] == ' ':
    chars[start_idx - 1] = 1

  enc1 = tokenizer.encode(context)
  enc2 = tokenizer.encode(question)

  # For resource limitations only.

  if len(enc1) + len(enc2) + 4 < MAX_LEN:

    #creating offsets
```

```
    offsets = []
    start_idx = 0

    for t in enc1.ids:
      w = tokenizer.decode([t])
      offsets.append((start_idx, start_idx + len(w)))
      start_idx += len(w)

    # Those which are a part of the answer

    tokens = []
    for j, (a, b) in enumerate(offsets):
      sum_ = np.sum(chars[a:b])
      if sum_ > 0:
        tokens.append(j)

    # The input for roberta is in the form <s> Question </s></s> Context </s>

    inputs[i, :len(enc1.ids) + len(enc2.ids) + 4] = [0] + enc2.ids + [2,2] + enc1.ids + [2]

    attention_mask[i, :len(enc1.ids) + len(enc2.ids) + 4] = 1

    if len(tokens) > 0:
      start_tokens[i, tokens[0] + 1] = 1
      end_tokens[i, tokens[-1] + 1] = 1

def build_model():
  ids = tf.keras.layers.Input((MAX_LEN,), dtype = tf.int32)
  att = tf.keras.layers.Input((MAX_LEN,), dtype = tf.int32)
  tok = tf.keras.layers.Input((MAX_LEN,), dtype = tf.int32)

  config = RobertaConfig.from_pretrained('D:/capstone/roberta-base-
squad2/config.json')
  bert_model = TFRobertaModel.from_pretrained('D:/capstone/roberta-base-
squad2/tf_model.h5', config = config)
  x = bert_model(ids, attention_mask=att, token_type_ids = tok)

  # For start logit

  x1 = tf.keras.layers.Dropout(0.1)(x[0])
  x1 = tf.keras.layers.Conv1D(1,1)(x1)
  x1 = tf.keras.layers.Flatten()(x1)
  x1 = tf.keras.layers.Activation('softmax')(x1)

  # For end logit

  x2 = tf.keras.layers.Dropout(0.1)(x[0])
  x2 = tf.keras.layers.Conv1D(1,1)(x2)
  x2 = tf.keras.layers.Flatten()(x2)
  x2 = tf.keras.layers.Activation('softmax')(x2)
```

```
  # Initalising the model

  model = tf.keras.models.Model(inputs = [ids, att, tok], outputs = [x1, x2])
  optimizer = tf.keras.optimizers.Adam(learning_rate = 3e-5)
  model.compile(loss='categorical_crossentropy', optimizer = optimizer)

  return model

model = build_model()

history = model.fit([inputs, attention_mask, token_type_ids], [start_tokens,
end_tokens], epochs = 3, batch_size = 4, validation_split = 0.1)

model.save('roberta_model') #saving the model after fitting
model.load_weights("roberta_model") #loading the weights after fitting the model

con = 'Google LLC is an American multinational technology company that specializes
in Internet-related services and products, which include online advertising
technologies, search engine, cloud computing, software, and hardware. It is
considered one of the Big Four technology companies, alongside Amazon, Apple, and
Facebook.'
que = 'What is Google?'

inp_id = np.zeros((1,MAX_LEN),dtype='int32')
attn_mask_input = np.zeros((1,MAX_LEN),dtype='int32')
token_type_id_input = np.zeros((1,MAX_LEN),dtype='int32')
inpenc = tokenizer.encode(con)
queenc = tokenizer.encode(que)
print(len(que))
#chars_1 = np.zeros((len(con)))
#chars_1[idx:idx + len(text2)] = 1

offset = []
id_ = 0
for t in inpenc.ids:
  w = tokenizer.decode([t])
  offset.append((id_, id_ + len(w)))
  id_ += len(w)
print(offset)

inp_id[0,:len(inpenc.ids)+len(queenc.ids) + 4] = [0] + queenc.ids + [2,2] + inpenc.ids
+ [2]
attn_mask_input[0,:len(inpenc.ids)+len(queenc.ids) + 4] = 1

def generate_ans(con,que,model,tokenizer):
  inp_id = np.zeros((1,MAX_LEN),dtype='int32')
  attn_mask_input = np.zeros((1,MAX_LEN),dtype='int32')
  token_type_id_input = np.zeros((1,MAX_LEN),dtype='int32')
  inpenc = tokenizer.encode(con)
```

```
  queenc = tokenizer.encode(que)
  inp_id[0,:len(inpenc.ids)+len(queenc.ids) + 4] = [0] + queenc.ids + [2,2] +
inpenc.ids + [2]
  attn_mask_input[0,:len(inpenc.ids)+len(queenc.ids) + 4] = 1
  s, f = model.predict([inp_id,attn_mask_input,token_type_id_input])
  s_ = np.argmax(s[0,])
  f_ = np.argmax(f[0,])
  ans = tokenizer.decode(inpenc.ids[s_ - 1: f_ + 1])

  return ans

con = 'Cancer is a group of diseases involving abnormal cell\
 growth with the potential to invade or spread to other parts of the body\
 . These contrast with benign tumors, which do not spread.\
  Possible signs and symptoms include a lump, abnormal bleeding\
  , prolonged cough, unexplained weight loss, and a change in\
   bowel movements. While these symptoms may indicate cancer,\
    they can also have other causes. Over 100 types of cancers \
    affect humans'
que = 'What disease involves abnormal cell growth?'
ans = generate_ans(con,que,model,tokenizer)
print(ans)

que = 'what are symptoms?'
ans = generate_ans(con,que,model,tokenizer)
print(ans)

que = 'how many types?'
ans = generate_ans(con,que,model,tokenizer)
print(ans)

con2 = "Malaria is a life-threatening disease caused by parasites that are transmitted
to people through the bites of infected female Anopheles mosquitoes. It is preventable
and curable. In 2018, there were an estimated 228 million cases of malaria worldwide.
The WHO African Region carries a disproportionately high share of the global
malaria burden. In 2018, the region was home to 93% of malaria cases and 94% of
malaria deaths."
que2 = 'What is Malaria?'
ans2 = generate_ans(con2,que2,model,tokenizer)
print(ans2)

que2 = 'What causes Malaria?'
ans2 = generate_ans(con2,que2,model,tokenizer)
print(ans2)

que2 = 'Is malaria preventable?'
ans2 = generate_ans(con2,que2,model,tokenizer)
print(ans2)

test_df = pd.read_csv('test_100.csv')
```

```
test_df.head()

answers = []
for index, row in test_df.iterrows():
    context = str(row["context"])
    question = str(row["question"])

    answer = generate_ans(context, question, model, tokenizer)
    print(answer)
    answers.append(answer)

test_df["answer"] = answers
test_df.to_csv("test_100_with_answers.csv", index=False)
```

References

[1] Puri, R., Spring, R., Shoeybi, M., Patwary, M., & Catanzaro, B. (2020, November). Training Question Answering Models From Synthetic Data. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 5811-5826).

[2] Ji, Z., Xu, F., Wang, B., & He, B. (2012, October). Question-answer topic model for question retrieval in community question answering. In Proceedings of the 21st ACM international conference on Information and knowledge management (pp. 2471-2474).

[3] Sneiders, E. (2002, June). Automated question answering using question templates that cover the conceptual model of the database. In the International Conference on Application of Natural Language to Information Systems (pp. 235-239). Springer, Berlin, Heidelberg.

[4] Yin, J., Jiang, X., Lu, Z., Shang, L., Li, H., & Li, X. (2016, June). Neural Generative Question Answering. In Proceedings of the Workshop on Human-Computer Question Answering (pp. 36-42).

[5] Xiong, C., Zhong, V., & Socher, R. (2016). Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604.

[6] Perez, E., Lewis, P., Yih, W. T., Cho, K., & Kiela, D. (2020, November). Unsupervised Question Decomposition for Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 8864-8880).

[7] Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2016). Learning to Compose Neural Networks for Question Answering. In Proceedings of NAACL-HLT (pp. 1545-1554).

[8] Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., & Daumé III, H. (2014, October). A neural network for factoid question answering over paragraphs. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 633-644).

[9] Wang, D., & Nyberg, E. (2015, July). A long short-term memory model for answer sentence selection in question answering. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers) (pp. 707-712).

[10] Khot, T., Clark, P., Guerquin, M., Jansen, P., & Sabharwal, A. (2020, April). Qasc: A dataset for question answering via sentence composition. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 05, pp. 8082-8090).

[11] McCarley, J. S., Chakravarti, R., & Sil, A. (2019). Structured pruning of a BERT-based question answering model. arXiv preprint arXiv:1910.06360.

[12] Kenton, J. D. M. W. C., & Toutanova, L. K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT (pp. 4171-4186).

[13] Chen, D., Bolton, J., & Manning, C. D. (2016, August). A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 2358-2367).

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[15] Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1757-1766).

[16] Dua, D., & Wang, W. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 2369-2380).

[17] Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., & Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In Proceedings of the 33rd International Conference on Machine Learning (ICML) (pp. 1378-1387).

[18] Lee, K., Lee, K., Lee, S., & Lee, S. (2020). Multi-passage BERT: A globally normalized BERT model for open-domain question answering. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 5295-5305).

[19] Reddy, S., Chen, D., & Manning, C. D. (2019). CoQA: A conversational question answering challenge. Transactions of the Association for Computational Linguistics, 7, 249-266.

[20] Wang, Y., Huang, S., & Sun, M. (2019). Multi-hop reading comprehension through question decomposition and rescoring. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL) (pp. 2334-2344).