Heriot-Watt University

Masters Thesis

Open Domain Question Answering Agent (ODQA)

Author:
Shreyas Arunesh (*H00357095*)

Supervisor:
Dr. Arash Eshghi

A thesis submitted in fulfilment of the requirements for the degree of MSc. Artificial Intelligence

in the

School of Mathematical and Computer Sciences

September 2022

Declaration of Authorship

I, Shreyas Arunesh (*H00357095*), declare that this thesis titled, 'Open Domain Question Answering Agent (ODQA)' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: *Shreyas Arunesh*

Date: *Monday 15th of August 2022*

Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F21MP – Masters Project and Dissertation
Type of assessment:	**Individual**
Coursework Title:	Open Domain Question Answering Agent Masters Project
Student Name:	Shreyas Arunesh
Student ID Number:	H00357095

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Shreyas Arunesh*

Date: *15/08/2022*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

FIGURE 1: Student declaration of Authorship.

"I want an AI-powered society because I see so many ways that AI can make human life better. We can make so many decisions more systematically or automate away repetitive tasks and save so much human time."

Andrew Ng

Abstract

The fundamental challenge of Open Domain Question Answering in Natural Language Processing (NLP) involves retrieving documents relevant to a specific users query and generating answers that are not domain restricted. Question Answering (QA) is advancing from sophisticated pipelined systems to end-to-end deep neural networks since most of the models answer queries using a single information source, often text from an encyclopedia or a single knowledge Base (KB).

In this masters thesis, Open Domain Question Answering Agent is developed. In the most basic form, the agent takes users queries unrestricted to any domain and provides a short span of answer. This process has two phases, firstly the retriever model extracts a set of relevant documents based on the users input query. Secondly, the Machine Reading Comprehension (MRC) model takes the short listed documents and that extract the span of answer to the query.

The model performance is evaluated using commonly available intrinsic evaluation metrics to examine how well the agent can answer open domain questions. Lastly, the model is compared with other Question Answering models to perform necessary error analysis.

Acknowledgements

I want to thank my supervisor Dr. Arash Eshghi for his consistent guidance throughout the semester in the NLP sub-field of conversational agents and without help help and guidance this project would not be possible.

I also want to thank my parents, siblings and friends for their support throughout this project.

Contents

Declaration of Authorship i

Abstract iv

Acknowledgements v

Contents vi

List of Figures x

List of Tables xi

Abbreviations xii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 2
- 1.3 Research question . . . 3
- 1.4 Aims and Objectives . . . 4
  - 1.4.1 Project Outline . . . 5

2 Literature Review 6
- 2.1 Introduction to Natural Language Processing (NLP) . . . 6
- 2.2 Question Answering (QA) . . . 6
  - 2.2.1 Closed Domain Question Answering . . . 7
  - 2.2.2 Open Domain Question Answering . . . 7
- 2.3 Evolution of QA models . . . 8
  - 2.3.1 Traditional Methods . . . 8
    - 2.3.1.1 Knowledge Based QA Systems . . . 8
    - 2.3.1.2 Early Deep learning based Systems . . . 8
  - 2.3.2 Modern Methods . . . 9
    - 2.3.2.1 Word Embeddings and Contextual Word Embeddings . . . 9
    - 2.3.2.2 Attention Mechanism . . . 10
    - 2.3.2.3 Bidirectional Attention Flow (BiDAF) . . . 10
    - 2.3.2.4 Transformers . . . 11

2.3.3 Large Scale language Models . . . 14
2.3.3.1 Long Short-Term Memory (LSTMS) . . . 14
Limitations of LSTM models: . . . 15
2.3.3.2 Bidirectional Encoder Representations from Transformers (BERT) . . . 16
Advantages of BERT . . . 18
2.3.3.3 Generative Pre-trained Transformer (GPT-3) . . . 18
Limitations of GPT models . . . 18
2.4 Information Retrieval (IR) . . . 18
2.4.1 Types of Information Retrieval Models . . . 19
2.4.1.1 Sparse Retriever Model (TF-IDF/ BM25) . . . 20
2.4.1.2 Dense Passage Retriever (DPR) . . . 20
2.5 Machine Reading Comprehension (MRC) . . . 21
2.5.1 Extractive Question Answering . . . 21
2.5.2 Generative Question Answering . . . 22
2.6 Machine learning concepts . . . 22
2.6.1 Model training and train/ test dataset . . . 22
2.6.2 Cross validation and validation dataset. . . . 22
2.6.3 Transfer Learning . . . 23
2.7 Evaluation . . . 23
2.7.1 Evaluation of Retrieval Model . . . 23
Metrics for Retriever model Evaluation . . . 24
2.7.2 Evaluation of reader models . . . 25
2.7.2.1 Metrics for reader model Evaluation . . . 25
Exact Match (EM) . . . 25
F1-Score . . . 25
2.7.3 Machine Reading Comprehension datasets . . . 27
2.7.3.1 Stanford Question Answering Dataset (SQAD) . . . 27
2.7.3.2 Question Rewriting in Conversational Context (QReCC) 27

3 Requirement Analysis 28
3.1 MosCoW Analysis . . . 29
3.2 Language and Frameworks . . . 30
3.3 Datasets . . . 30
3.3.1 English Wikipedia Dump . . . 30
3.3.2 SQuAD . . . 31
3.4 Computing resources . . . 31

4 Models and implementation 32
4.1 Information Retriever . . . 32
4.1.1 Wikipedia Indexing . . . 32
4.1.1.1 Parsing XML dump and data pre-processing . . . 32
4.1.1.2 Single-pass in-memory indexing (SPIMI) . . . 33
4.1.2 Wikipedia Search Engine . . . 34
4.1.2.1 Single stage Retriever (BM25) . . . 35
4.1.2.2 Multistage Retriever (Wikipedia API +BM25) . . . 36
4.2 Reader models . . . 37

4.2.1 Data pre-processing . . . 37
4.2.1.1 Data Splitting and structuring . . . 37
4.2.1.2 Find the Start/ end position character . . . 38
4.2.1.3 Tokenisation using Glove word embedding for DrQA . . . 38
4.2.1.4 Tokenisation and additional stages for BERT-base . . . 39
4.2.1.5 Create a Dataset class . . . 40
4.3 DrQA . . . 40
4.3.1 Training , Validation and hyper-parameters details of DrQA . . . 42
4.3.2 Bidirectional Encoder Representations from Transformers (BERT-Base) . . . 43
4.3.3 Training, validation and hyper-parameter details of BERT-base . . . 43
4.4 Fully connected pipeline . . . 44

5 Evaluation and Results **45**
5.1 Retriever Evaluation . . . 45
5.1.1 Reason for using SQuAD dataset . . . 45
5.1.1.1 Challenges in SQuAD V2.0: . . . 46
5.1.2 Experimental setup . . . 46
5.1.2.1 Evaluation metrics . . . 46
5.1.2.2 Evaluation Methodology . . . 47
5.1.3 Retriever Results . . . 48
5.1.3.1 Results interpretation . . . 49
5.1.3.2 Drawbacks of Retriever . . . 49
5.2 Reader Evaluation . . . 50
5.2.1 Experimental setup . . . 50
5.2.2 Reader Results . . . 51
5.2.2.1 Results Interpretation . . . 53
5.2.2.2 Comparison of models. . . . 53
5.2.2.3 Drawbacks of Reader models . . . 54
DrQA: . . . 54
BERT-base: . . . 55

6 Discussion and conclusion **56**
6.1 Error Analysis . . . 56
6.2 Incomplete Experiments . . . 58
6.2.1 Model ablations . . . 58
6.2.2 Data ablations . . . 59
6.3 Problems and issues . . . 59
6.3.1 Memory issues . . . 59
6.3.2 Library issues . . . 60
6.3.3 Run time limitations . . . 60
6.3.4 Debugging and understanding DrQA and Pytorch . . . 61
6.4 Future work . . . 61
6.5 Conclusion . . . 62

A Reader Hyper-parameters **63**

A.1 DrQA hyper-parameters 63
A.2 BERT-base hyper-parameters: 63
A.2.1 BERT-base hyper-parameter values 64

B Interactive Session **65**
B.1 Fully connected pipeline interactive session 65

C GitHub/ MACS-GitLab Project repository **67**
C.1 Project code and repository. 67

Bibliography **68**

List of Figures

1 Student declaration of Authorship. ii

1.1 General architecture of this project. 3

2.1 Bidirectional Attention Flow (BiDAF) Model [Seo et al., 2016]. 11
2.2 Transformers architecture [Vaswani et al., 2017]. 12
2.3 Attention Mechanism of Scalled Dot Product Attention (Left) and Multi-Headed Attention (Right) [Vaswani et al., 2017]. 13
2.4 Arcitecture of BiLSTM [Soutner and Müller, 2013]. 15
2.5 BERT architecture [Devlin et al., 2018]. 16
2.6 Information Retrieval Process [Lewis and Jones, 1996]. 19

4.1 BERT-tokeniser downloaded from Huggingface library. 40
4.2 Flowchart demonstrating the flow of DrQA model. 41

5.1 Retriever output for No of articles = 15 & No of paragraphs = 20 (left) and No of articles = 25 & No of paragraphs = 15 (Right) 48
5.2 Retriever output for No of articles = 35 & No of paragraphs = 10. 49
5.3 DrQA model results on SQuAD V1.0 dataset. 51
5.4 DrQA model results on SQuAD V2.0 dataset. 52
5.5 BERT-base model results on SQuAD V1.0 dataset. 52
5.6 BERT-base model results on SQuAD V2.0 dataset. 52

B.1 First interactive sample of ODQA for N=10 K=10. 65
B.2 Second interactive sample of ODQA for N=10 K=10. 66

List of Tables

2.1 Confusion Matrix [Visa et al., 2011]. 26

3.1 Project requirements. 29

4.1 Dataset statistics of SQuAD V1.0 V2.0 38
4.2 BERT tokens and purposes. 39

5.1 Evaluation score of reader models on both SQuAD V1.0 and V2.0. . . . 54

6.1 Time consumed for each process. 61

A.1 DrQA training hyper-parameters. 63
A.2 BERT-base training hyper-parameter values. 64

Abbreviations

QA	**Q**uestion **A**nswering
SPMI	**S**ingle **P**ass **I**n **M**emory **I**ndexing
KB	**K**nowledge **B**ase
MC	**M**achine **C**omprehension
SOTA	**S**tate **O**f **T**he **A**rt
QAS	**Q**uestion **A**nswering **S**ystems
NLP	**N**atural **L**anguage **P**rocessing
SQuAD	**S**tanford **Qu**estion **A**nswering **D**ataset
RNN	**R**ecurrent **R**eural **N**etwork
ANN	**A**rtificial **N**eural **N**etwork Architecture
CNN	**C**onvolutional **N**eural **N**etwork
NLIDB	**N**atural **L**anguage **I**nterfaces to **D**ata**b**ases
BiDAF	**Bii**irectional **A**ttention **F**low
CoQA	**C**onversational **Q**uestion **A**nswering **S**ystem
QuAC	**Qu**estion **A**nswering **I**n **C**ontext
LSTM	**L**ong **S**hort **T**erm **M**emory
BERT	**B**idirectional **E**ncoder **R**epresentation **T**ransformers
MLM	**M**asked **L**abel **M**ask
NSP	**N**ext **S**entance **P**rediction

Dedicated to everyone who supported me in this project.....

Chapter 1

Introduction

1.1 Background

Language is our primary form of communication among humans, therefore for decades, many researchers in the AI community have been working towards getting machines to understand the language. Natural language is the finest information access tool for people in human-computer interaction. As a result, Question Answering function was integrated into several web search engines, such as Google, Yahoo and etc... to advance their level of intelligence. With the use of these technologies, search engines can now provide exact answers to certain types of queries. The primary goal of Question Answering Systems (QAS) is to respond to factual questions in natural language on a specific topic.

Given the tremendous success of deep learning in natural language processing, answer selection approaches based on neural networks has been well investigated. However, the traditional neural networks i.e RNN and CNN typically suffer from acquiring global text information due to their working principles. Recent breakthroughs in neural architectures, such as the Transformer, combined with the introduction of large-scale pre-trained models such as BERT and GPT has illuded NLP community for a long time and transformed Natural Language Processing (NLP) applications by pushing the state of the art (SOTA) to new heights. Transfer learning is a popular approach in deep learning that enables improvement in learning new activities such as Question Answering, Machine Translation, Summarisation by transferring previously gained knowledge to new downstreaming tasks. Question-and-answer systems are used on a regular basis in commercial systems such as Alexa, Alana, and Woebot.

1.2 Motivation

Earlier QA systems such as Baseball [Green Jr et al., 1961], intended to provide information about American baseball games, and LUNAR [Woods, 1973], that was created as a powerful tool to support research work of lunar geologists were all centred on restricted domain or closed-domain settings in which all the relevant information was held within the system in a well-defined dictionary, and user queries are translated into query statements which mainly employs linguistic analysis and syntactic pattern matching techniques to obtain final answers from the dictionary.

OpenQA aims to provide an answer to a question without providing any background as illustrated in the example bellow. Typically, the system must first look for relevant articles that provide context for a given question either in a local repository or on the World Wide Web (WWW), and then generate the appropriate response. [Harabagiu et al., 2003]

Question: Who was the prime minister of United Kingdom in 2021?
Answer: Boris Johnson.

Although Open domain QA are very powerful technique, they are "one-shot" QA systems, meaning they don't keep track of prior questions or subjects in the dialogue. This is when conversational elements come into play to perform Conversational Open Domain Question Answering.

For a system to respond in the way described bellow, it would need to be based in two contexts. To begin, it would need to understand the world in which it lives to respond factual queries about it. Secondly, the system should remember the context of the conversation to respond to follow-up inquiries by resolving the co reference resolution to understand the bellow example i.e "the country" refers to United States of America and "that time" refers to 11 September 200.

Question: When did World Trade Centre collapse?
Answer: 11 September 2001.
Question: Where is it located?
Answer: United States of America.
Question: Who was the president of the country during that time?
Answer: George W. Bush.

Wikipedia is a multilingual online encyclopedia, making it one of the largest and most popular information sources on the internet. It is constantly evolving with the most recent articles that people are interested in reading, and is also a popular resource for open-domain question answering.

The goal of this project is to implement an Open Domain Question Answering Agent to answer any factoid question using Wikipedia as the unique knowledge source. This task of large scale machine reading is combined with the challenges of document retrieval (finding the relevant articles from Wikipedia) and machine comprehension of text (locating the answer spans from those articles). Therefore, this project consists of two components: (1) Information Retrieval system to extract relevant documents among the collection of more than 6 million Wikipedia articles and (2) Machine Reading Comprehension to scan the retrieved documents to find the answer. Lastly, the performance of both components are assessed individually using the intrinsic evaluation metrics. The general architecture of this project is illustrated in the Figure 1.1 bellow.

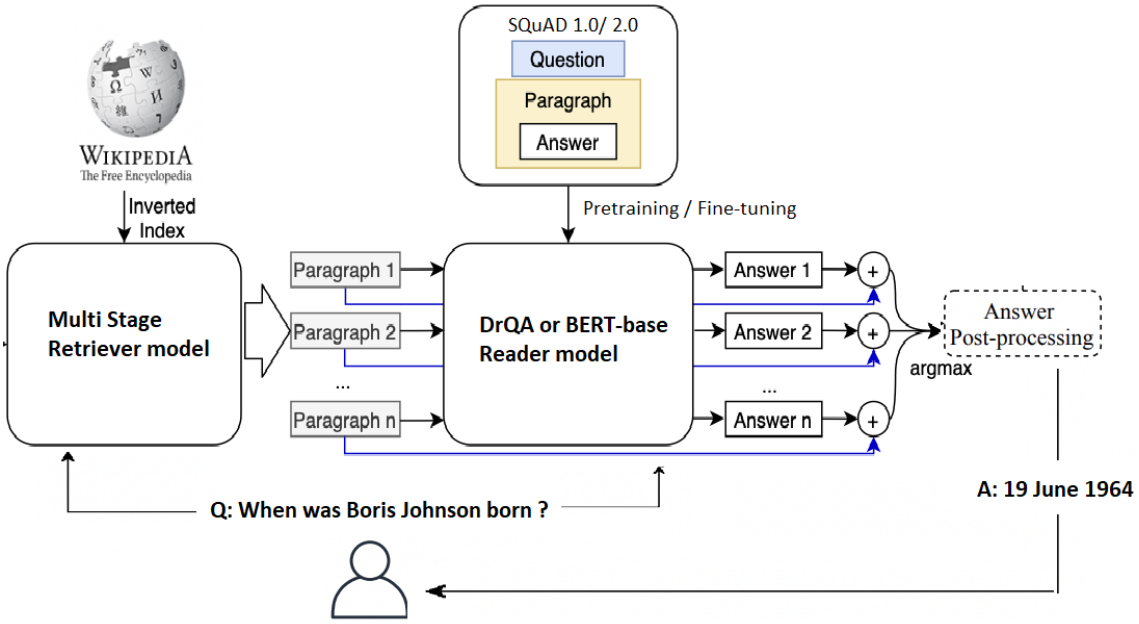

FIGURE 1.1: General architecture of this project.

1.3 Research question

Which language model, amongst "transformer-based models" and "recurrent neural network-based models," is most suitable for the task of Open Domain Question Answering ?

1.4 Aims and Objectives

The aim of this project is to develop an Open domain Question Answering system using Wikipedia as the primary source of information where the answer to each factual query is a text span from a Wikipedia article. The system is three folded, For a given search query, Firstly Sparse Information Retrieval model is reproduced to index and extract relevant documents from Wikipedia. Secondly, two reader models from successive generations are re-implemented to extract the span of answer from retrieved documents. Finally all models are evaluated to assess and compare their performances.

The Project objective are as follows:

1. Indexing of the latest English Wikipedia dump using Single-Pass In-Memory Indexing (SPMI); enwiki-20220101 [EnWiki, 2022]

2. Re-implementing Wikipedia Information retrieval system using BM25 retrieval model to extract query relevant passages from Wikipedia. [Svore and Burges, 2009]

3. Re-implementing a traditional Recurrent Neural network (BiLSTM) based Machine Reading Comprehension to extract span of answers on the information retrieved using DrQA language model. [Chen et al., 2017]

4. Training and validating DrQA language model using the public datasets; SQuAD V1.0 & V2.0. Rajpurkar et al. [2016]

5. Fine tuning and validating the modern-transformer based BERT-base language model using the public datasets; SQuAD V1.0 & V2.0. Rajpurkar et al. [2016]

6. Evaluating the Information Retriever model using intrinsic evaluation metrics (such as Precision@N) on the questions and contexts of public dataset; SQuAD. Rajpurkar et al. [2016]

7. Evaluating both reader models using intrinsic evaluation metrics (such as Exact Match and F1 Score) for performance comparison.

8. Optionally perform necessary error analysis using *model ablations* and *data ablations* and implement Conversational aspect in the base systems to make the system interactive using Question Rewriting in public dataset; QReCC [Anantha et al., 2020]

1.4.1 Project Outline

The following topics will be covered in the subsequent chapters:

- Chapter 2: Literature Review chapter will discuss about evolution of QA models, Large scale language models, Machine Comprehension and Evaluation Metrics and Machine Reading comprehension datasets.
- Chapter 3: Requirement analysis chapter will discuss about the tools and resources required to execute this project.
- Chapter 4: Models and implementation chapter will illustrate various models and implementation methodology followed in this project.
- Chapter 5: This chapter will discuss about the evaluation methods and results obtained in this project.
- Chapter 6: This chapters will discuss about the error analysis, various problems and potential future works.

Chapter 2

Literature Review

The literature review chapter discusses about the Question Answering, Open domain and closed domain Question Answering systems followed by the evolution of Question Answering models to large scale language models. An emphasis will be placed in literature about Information Retriever and Machine Reading Comprehension models. Finally, the evaluation of the systems using various evaluation metrics is illustrated using commonly available machine reading comprehension datasets.

2.1 Introduction to Natural Language Processing (NLP)

Natural Language processing (NLP) focuses on the effective communication between computers and Human being for theoretical and practical aspects of information sharing and information extraction. In the NLP domain, activities like information retrieval, natural language understanding, Natural Language Generation etc... has been a growing theme for Knowledge sharing [Gupta and Gupta, 2012].

2.2 Question Answering (QA)

Question Answering is a technique of building systems that answers human questions automatically within the field of Natural Language Processing (NLP). Search engines like Google, Yahoo are used to find ranked list of relevant documents on formulated queries using the techniques of keyword matching and frequencies of accessing documents. Later users go through the short listed documents to find the answers [Saini and Yadav, 2017]. However, Question Answering System are used to retrieve most relevant and accurate spans based on response-oriented approach to answer human queries. Some applications

require information retrieval on general topic and others from a particular domain. Based on this answering domain, Question Answering system is broadly categorised into two types:

2.2.1 Closed Domain Question Answering

It's a type of QA systems which don't rely on pre-existing knowledge base and questions are based on specified context while addressing human queries. Closed domain QA systems are designed to retrieve answers from structured, unstructured or semi-structured data using domain specific terminology and ontology [Vargas-Vera and Lytras, 2010]. One famous application of this system is Green's Baseball introduced by Bdour and Gharaibeh [2013] in 1961 to provide information on American baseball games, including the start time, venue, and club name. Another well-known QA systems, LUNAR [Woods, 1973], developed in 1973, and MURAX [Kupiec, 1993] established in 1993 as a powerful tools to support the work of lunar geologists. All these systems stores relevant information within the systems in a well-defined dictionary. User questions are converted into query statements applying linguistic techniques to obtain final answer.

2.2.2 Open Domain Question Answering

These QA systems has access to the knowledge- repository and is not restricted to specific domain while addressing incoming human queries. These models perform a large-scale search on World Wide Web information through combination of information retrieval that picks a limited number of documents relevant to the topic and reading comprehension generates an answer by applying general Vocabulary from the selected documents on any domain [Zhu et al., 2021]. The first OpenQA application was designed in 1999 to extract the top 5 snippets containing the right answer from a collection of news stories in Text REtrieval Conference's QA track (TREC) [Voorhees et al., 1999]. Other famous applications of OpenQA is IBM-Watson, a framework of Information retriever with deep learning-based Ranker and Reader [Wang et al., 2018].

2.3 Evolution of QA models

Introduction of transformers in 2017 by Vaswani et al. [2017] became a new cutting edge in the field of NLP. Most recent advances in NLP have used Transformer models (see Section 2.3.2.4) and its subsets like BERT [Devlin et al., 2018] (see Section 2.3.3.2) and GPT [Floridi and Chiriatti, 2020] (see Section 2.3.3.3) which is a encoder-decoder architecture achieving a uniqueness in the state of art for tasks such as Question Answering, Machine Translation and also text summarising. [Reddy et al., 2019]

2.3.1 Traditional Methods

In contrast to modern neural approaches, traditional Open Domain QA models uses structured data for accomplishing language into database queries. This section will go through some of the traditional techniques for Question Answering tasks that use peripheral approaches.

2.3.1.1 Knowledge Based QA Systems

knowledge Based QA systems uses Natural language Interfaces to Databases (NLIDB) technique to query huge, structured sets of knowledge triples [Zhang et al., 2022]. A knowledge triple, or semantic triple, is a common way to store semantic data in the form of subject-predicate-object.

Fader et al. [2013] proposed a large-scale Knowledge Based QA methodology that could answer open domain queries employing semantic parsing to generate entities, relations, and query patterns from a corpus of question answer pairs which may subsequently be used to match responses in a Knowledge Base [Kim, 1996]. The major drawback of this system is when there are no answers to the question in the Knowledge base, the system fails to map entities and relations of knowledge base to unanswerable questions. [Kim, 1996]

2.3.1.2 Early Deep learning based Systems

In contrast to knowledge-based QA systems, Information Retrieval QA systems extracts answers from a corpus of documents using a pipeline of text-based approaches. The necessary texts are located using linguistic intuitions and deep neural networks, and answers are extracted from segments of text within these documents. Some early Deep

Learning approach to answer selection was; Word2vec introduced by Mikolov et al. [2013], word vectors was used to find the insightful relationship and similarities between the words using encoding mechanism.

Later Yu et al. [2014] introduced a novel text-based supervised learning model which often relied on lexical semantic resources which demonstrated that it's possible to use a neural network approach to learn a semantic similarity between question answer pairs. Recurrent Neural Network (RNN) was established as a result of this, which allowed the deep neural nets to consider the order of words and their effect on preceding and subsequent words by updating the weights while training the neural network. This improved QA abilities but suffered from vanishing gradients problems in the long run [Chen et al., 2017].

2.3.2 Modern Methods

To address this issue of vanishing gradient, Long Short Term Memory (LSTM) (see Section 2.3.3.1) was introduced by Bai [2018] to enable the learning of long-term dependencies by allowing knowledge from earlier in a sequence to be preserved and applied to states later in the process.

A unidirectional LSTM preservers the information of past inputs only. Since it was essential to preserve information of both past and present in real world application, Seo et al. [2016] introduced BiDAF (see Section 2.3.2.3).

2.3.2.1 Word Embeddings and Contextual Word Embeddings

Embeddings aims to learn a continuous vector representation for each word in the document obtained from models using unsupervised learning for numerous machine learning tasks. Traditional word embedding techniques provided by word2vec, Glove (See section 2.3.1.2) learns a global word embedding. They start by developing a global vocabulary by disregarding the meaning of words in diverse contexts while employing unique terms in the documents. Then, comparable representations are learned for the terms that occurred in the texts that are close to each other. As a result, no matter where a word appears in a phrase, its embeddings are always the same therefore, the pre-trained word embeddings are static.

The biggest downfall of word embeddings is that the contextual meaning i.e the meaning gained from the words surroundings is overlooked. Contextual word embeddings are dynamically generated from pre-trained/ fine-tuned models provided by Transformers (See Section 2.3.2.4) to helps overcome this issue as they learn sequence-level semantics

by considering the sequence of all words in the documents. The embeddings generated for each word in a sentence are determined by other words in the context. As a result, Contextual word embeddings learn several representations for polysemous words.

For example, in the sentence "I left my wallet on the left side of the car", both the words left has same representation of embeddings in word2vec. Whereas embeddings in transformer-based models has different representation of embeddings for the polysemous word "left".

2.3.2.2 Attention Mechanism

An Attention function can be illustrated as a mapping of input query and set of keys-value pair of the output, where the query, value, output, and keys are all inferred as vectors. The output is computed as a weighted sum of the values, where the weight allotted to each value is calculated by a compatibility function relevant to the query key. [Vaswani et al., 2017].

Self-attention, which is also referred as intra-attention which is a mechanism that connects distinct points of a single sequence to compute a representation of the entire sequence. They are labelled as self-attention since the inputs are encoder or decoder of a particular keys, values, or queries [Vaswani et al., 2017]. Transformers (see Section 2.3.2.4) uses Self- attention mechanism to examine the relationship between a word and its neighbours in contextual word embeddings. (see Section 2.3.2.1). Because of this self-attention based contextual embeddings, large language models (such as BERT, GPT) (see Section 2.3.3.3) derived from Transformers (see Section 2.3.2.4) are important for today's modern Machine Reading Comprehension (MRC) (see Section 2.5) [Vaswani et al., 2017].

2.3.2.3 Bidirectional Attention Flow (BiDAF)

Bidirectional Attention Flow (BiDAF) is a predecessor to transformers (see Section 2.3.2.4) [Seo et al., 2016]. It's a closed-domain, extractive QA model that can only answer factoid questions. Arguably it was the best QA model for Machine Comprehension (MC) at that time because of its neural attention mechanism, that allows the system to focus on a specific areas inside a context. It follows a multistage hierarchical process that describes the context at multiple levels of granularity and employs a bi-directional attention flow technique. [Seo et al., 2016]

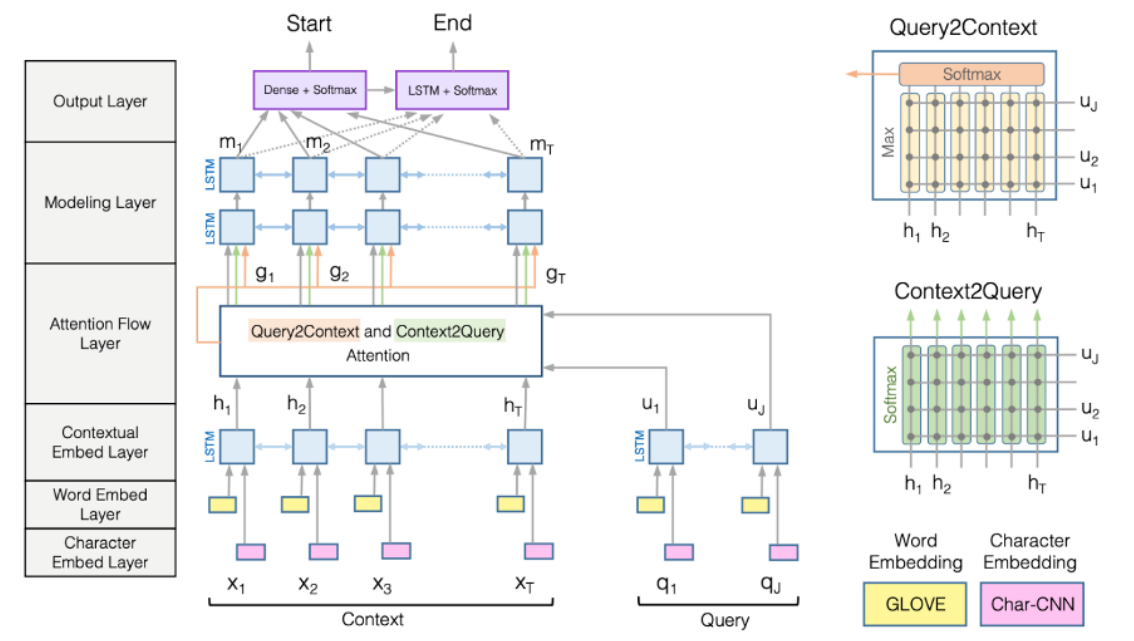

FIGURE 2.1: Bidirectional Attention Flow (BiDAF) Model [Seo et al., 2016].

As illustrated in the Figure B.2 above, BiDAF has a modular memory-less architecture, which has a composite structure made up of Lego blocks of standard elements such as GloVe, RNN, LSTM and Attention [Seo et al., 2016]. Instead of relying on the previous time step's attention vector, a new attention vector is calculated at every time step. BiDAF contains 3 parts of 6 layers; Embedding layer, to change the representation of words from strings to vectors in the Query and Context. These query and context are then passed to attention and modelling layer where several matrix operates to fuse the information. The output of this layer is another context representation that includes information from the query which is referred as "Query-aware Context representation". This is then passed to the Output layer which will transform it to a bunch of probability values to determine the start and end indices of the answer in the context. [Seo et al., 2016]

2.3.2.4 Transformers

Transformers are simply Artificial Neural Network Architecture (ANN) which relying entirely on a stacked self-attention mechanism, fully connected layers to draw global dependencies between input and output [Vaswani et al., 2017]. This implies the architecture has 2 main components; an encoder to map the input sequence of continuous representation (x1,...,xn) and the decoder then produces an output sequence (y1,...,ym) of symbols one element at a time [Graves, 2013]. These models are trained to learn contextual associations between words or sub-words in a text corpus and then the models

can be fine-tuned for the specific downstream tasks since transformer design is amenable to pre-training on huge text corpora, resulting in significant gains in accuracy [Vaswani et al., 2017].

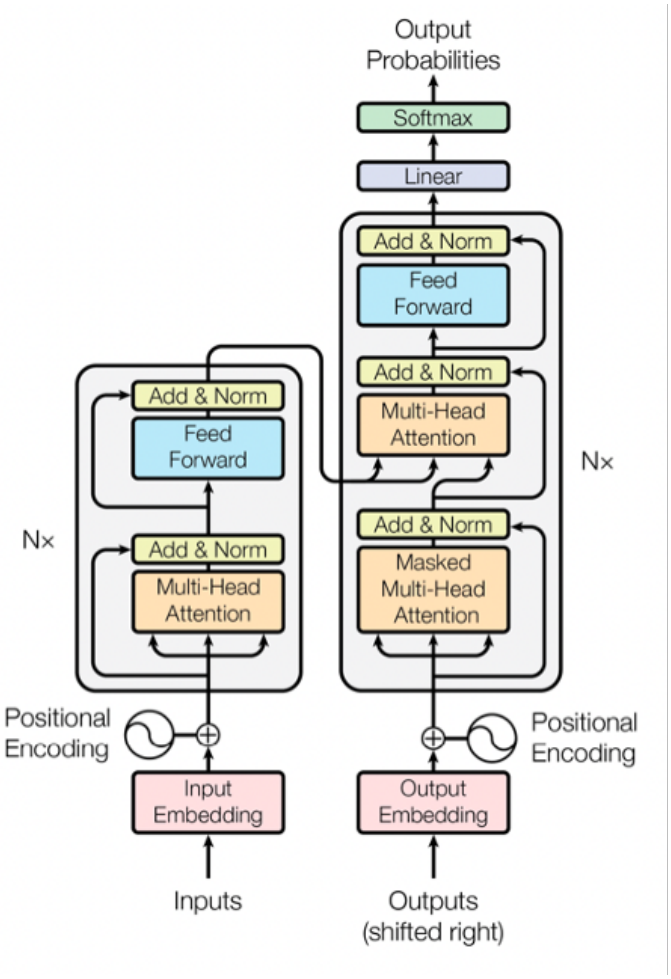

FIGURE 2.2: Transformers architecture [Vaswani et al., 2017].

The Figure 2.2 above illustrates the Transformer architecture which consists of two major components encoder-decoder stack and attention. Encoder and decoders both consists of equal number of multiple neural layers embedded in the attention. Some of the extensive descriptions of these components are provided below to help comprehend these definitions [Vaswani et al., 2017].

1. **Encoders are decoders** are multiple stacks of neural network composed of multi-head self-attention and a feed-forward neural network.

2. **Tokenizers** are the implementation of production grade for transformers. These spilt all the sentences into tokens / words and assign them id's and store them as vectors [Adel and Shi, 2021].

3. **Masks** are flags that are generally implemented in decoders using numerical values that allow the neural network to ignore future values in the decoder input.

4. **Pads** are flags that are either implemented in encoder or decoder. Pads help to convert numbers to huge negative values that are discarded by activation functions. Padding is essential due to the presence of varying input sequences (strings of words).

5. **Positional Representation** is the process of encoding positional information inside the transformers.

6. **Embedding** is the representation of every word / token in each sequence as vectors and numerical values in the predetermined vector space.

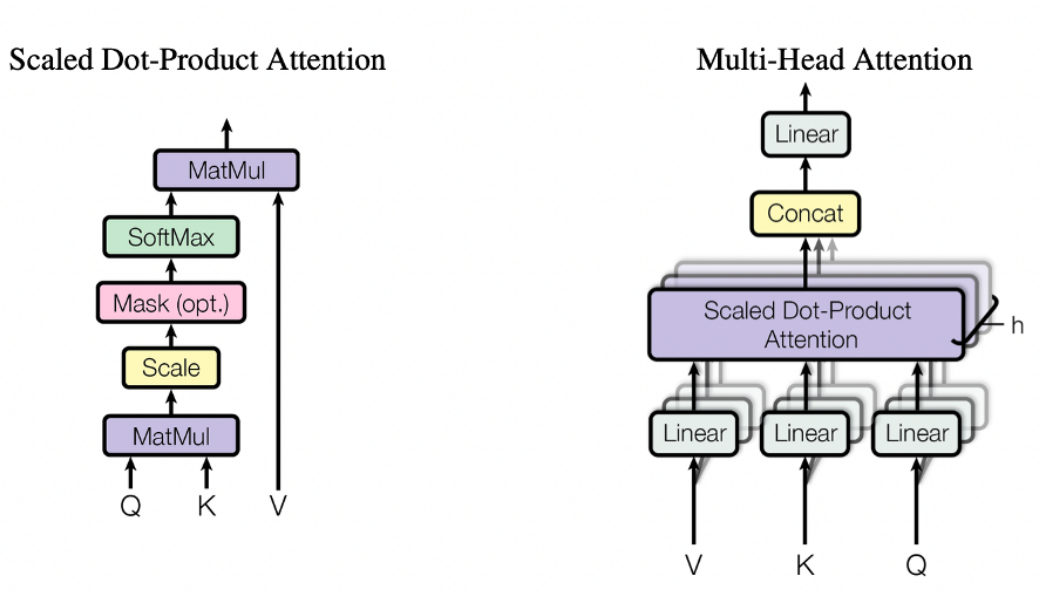

FIGURE 2.3: Attention Mechanism of Scalled Dot Product Attention (Left) and Multi-Headed Attention (Right) [Vaswani et al., 2017].

As illustrated in the Figure 2.3, authors described that Encoder (Left) receives vector inputs, such as embedding and positional encoding, and sends to Multi Head Self Attention block to extract the contextual features [Vaswani et al., 2017]. Single attention, as shown in Figure 2.3, represents the input sequence processed by attention to extract the context between the words, where Q denotes the single word vector representation and K denotes the whole word vector representation in the sequence. The encoder is coupled to the decoder's Multi-head attention (encoder-decoder). They are used to construct the next prediction by combining the encoder's output with that of the previous decoder neural layer. The Feed-Forward layers are modelled after key-value memory, in which keys correspond to human-interpretable input patterns and values correspond to

the next-token distribution of patterns in the associated key. The model's output is the combination of these distributions that was initially composed to form individual output layer and eventually redefining throughout the model [Geva et al., 2020]. BERT (see Section 2.3.3.2) is an example of a transformers architecture that can be implemented in a variety of NLP applications and will also be employed in this project.

2.3.3 Large Scale language Models

This section will discuss about the large scale language models.

2.3.3.1 Long Short-Term Memory (LSTMS)

Long Short-Term Memory (LSTM) is a different type of Recurrent Neural network (RNN) which by training can discovers both long and short patterns present in data and eliminates the problem of vanishing gradient. In traditional RNN, since neurons in the beginning of the layers tends to learn slower compared to neurons at the end of the layer the gradient tends to be vanishing (Exploding). To overcome this gradient vanishing problem, LSTM Cell was introduced [Graves, 2013].

LSTM was first introduced as a "Gating based architecture". These gates determine when values are allowed to flow in or out of the block's memory. The model basically includes 3 gates; input gate which includes part of new information from input value to new state, output gate to control which part of the output value from the hidden layer can be the final output and finally forget gate determines which part of the information stored in the current state needs to be forgotten [Soutner and Müller, 2013].

In Bidirectional LSTM, the input travels in both directions, and it may use information from both sides as depicted in Figure 2.4 bellow. It's also an effective tool for simulating the sequential relationships between words and sentences in both directions. BiLSTM contains an additional LSTM layer that reverses the direction of information flow. In a nutshell, it implies that the input sequence is reversed in the additional LSTM layer. The outputs of both LSTM layers are then combined in a variety of methods, including average, sum, multiplication, and concatenation.

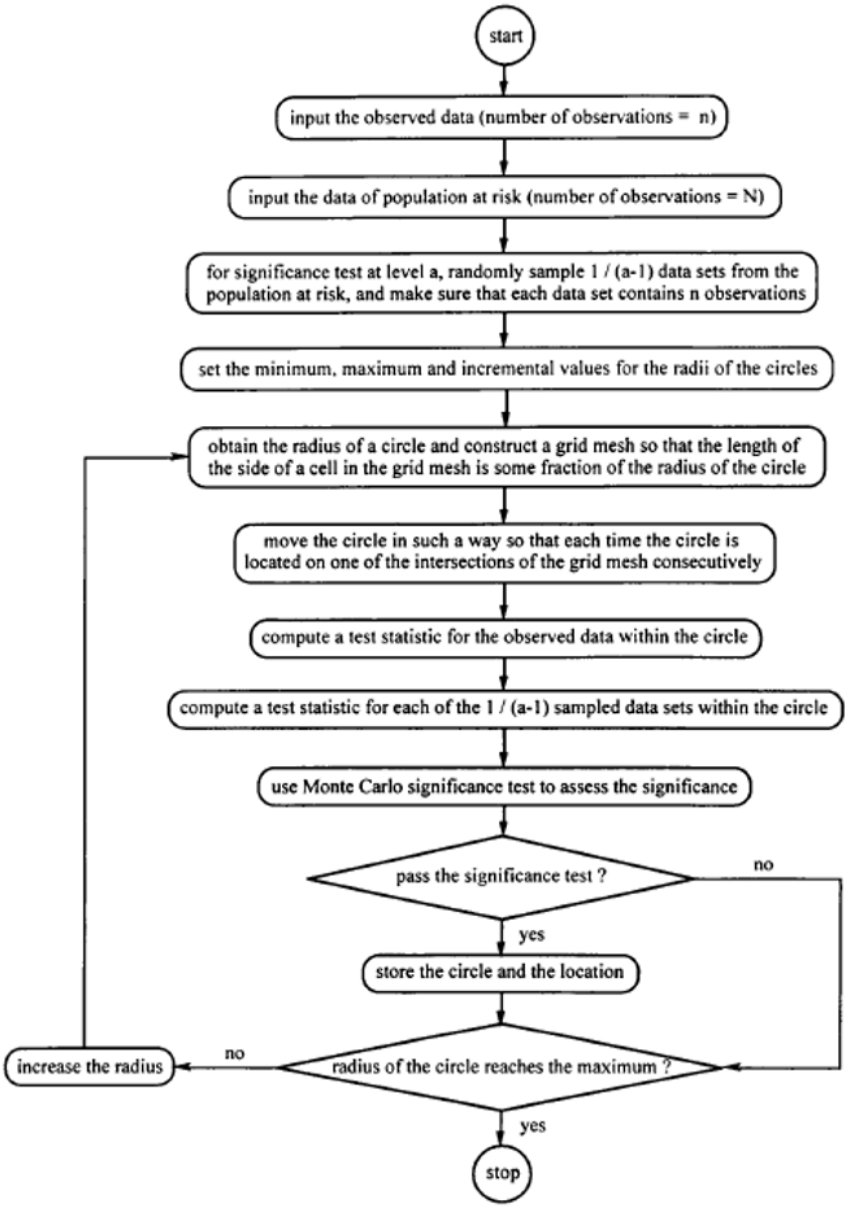

FIGURE 2.4: Arcitecture of BiLSTM [Soutner and Müller, 2013].

Limitations of LSTM models:

Although LSTM can represent any Context Free Language (CFL) and has achieved a great success in numerous NLP tasks [Sutskever et al., 2014]. Existing analysis proves that Transformers can perform better and can achieve state of the art performance earlier than LSTM this is because LSTM is incapable of factorising its latent space to encode the state and many elements of the stack without explicit supervision [Soutner and Müller, 2013]. Tasks like Question and Answering, Language recognition is not a suitable task with LSTM since the results was found sensitive to PDAs (the hyperparameters of the

models, and the training methods). LSTM is not enabled with self-attention mechanism and they are slow to train as they are recurrent [Shi et al., 2021].

2.3.3.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT introduced by Devlin et al. [2018], a novel approach of deep learning algorithm designed to include bidirectionality in a single transformer model. It is aimed to constrain both left and right context in all layers to pre-train deep bidirectional representations from unlabelled text. The architecture of BERT is identical to the transformers (see Section 2.3.2.4) with variable number of self-attention head and encoder layers.

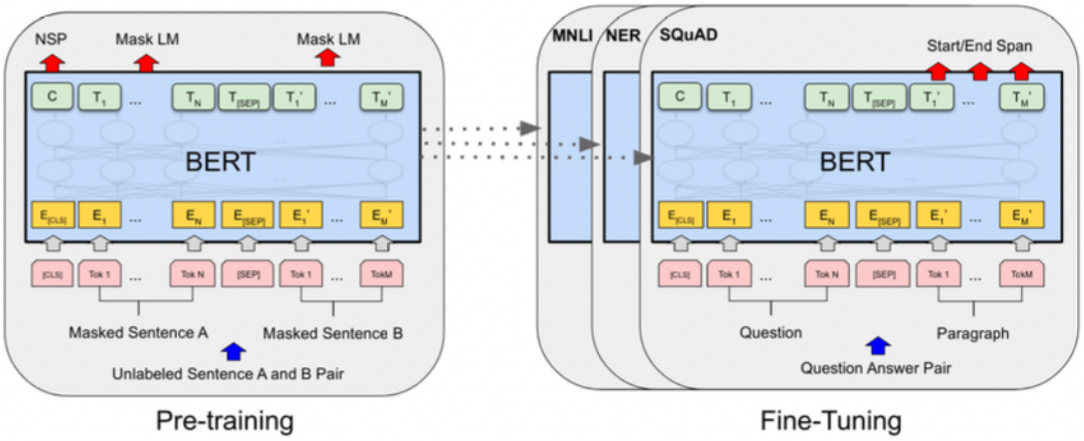

FIGURE 2.5: BERT architecture [Devlin et al., 2018].

The Figure 2.5 above represents the BERT architecture used for overall pretraining and finetuning with Input and output representations used to initialise the model for various down streaming tasks like Question Answering using WordPiece embeddings [Wu et al., 2016]. There are 2 variations in BERT architecture, BERT-Base (with 12 layers) and BERT- Large (with 24 layers) both are restricted to 512 tokens of input sequence. Every sequence always starts with a special classification token where the final hidden state corresponding to this token is utilised as the aggregate sequence representation. Sentence pairs are packed into a single sequence separated by special token and add the learned embedding to differentiate every sentences. The input representation for a particular token is the sum of the associated token, segment, and position embeddings [Devlin et al., 2018].

Pretraining is a machine learning procedure which aims to create a baseline for the model developed by training on unlabelled data. Developers used Books Corpus and the English Wikipedia as the pertaining data for BERT. The two steps followed to pretrain these models are Masked Label Mask (MLM) referred as cloze task which masks some

percentage of the input tokens at random, and then predict those masked tokens and Next sentence prediction (NSP) which is used train the model for understanding the sentence relationships which is more beneficial for Question and Answering (QA) tasks [Devlin et al., 2018].

Advantages of BERT

BERT is a simple bidirectional architecture that is empirically very powerful as it could achieve state-of-the-art results for various tasks including question answering. This model is pretrained on an incredible amount of data named BooksCorpus: 800 million words, and by adding a simple classification layer and fine tuning the model without major task-specific architectural modification can make a big difference in performance compared to traditional language models. They outperform conventional approaches in terms of understanding contexts of massive amounts of text and language. Moreover, This model is open source, and it can be found in the Hugging face library. With a few lines of code, developers may customise this model to promote and utilise the existing concepts.

2.3.3.3 Generative Pre-trained Transformer (GPT-3)

Generative Pre-trained Transformer (GPT) is a transformer based architecture introduced by OpenAI is one of the most powerful language models [Mager et al., 2020]. GPT-3 was developed using a combination of five different corpora such as Common Crawl, WebText2, Books1, Books2 and Wikipedia. The performance of the model was impressive on tasks like closed book Question Answering, machine translation and better than state of art for language models with few settings. The model was also evaluated on synthetic tasks like arithmetic addition, unscrambling of words, news article generation, learning and using novel words etc. The performance drastically increased with the increasing parameters [Floridi and Chiriatti, 2020]

Limitations of GPT models

Due to the unidirectionality of the model, it occasionally loses coherency when composing large phrases and repeats text sequences. Other major drawback is GPT-3 include complicated and expensive inference from the model due to its complex architecture, lower interpret-ability of the language and confusion about what makes the model accomplish its few shots learning behaviour [Floridi and Chiriatti, 2020].

2.4 Information Retrieval (IR)

Information retrieval is defined as the process of accessing and retrieving the most appropriate information based on user's query. Information retrieval system is the set

of algorithms that facilitate the process of searching relevant documents, sorting, and ranking these documents based on user's queries.

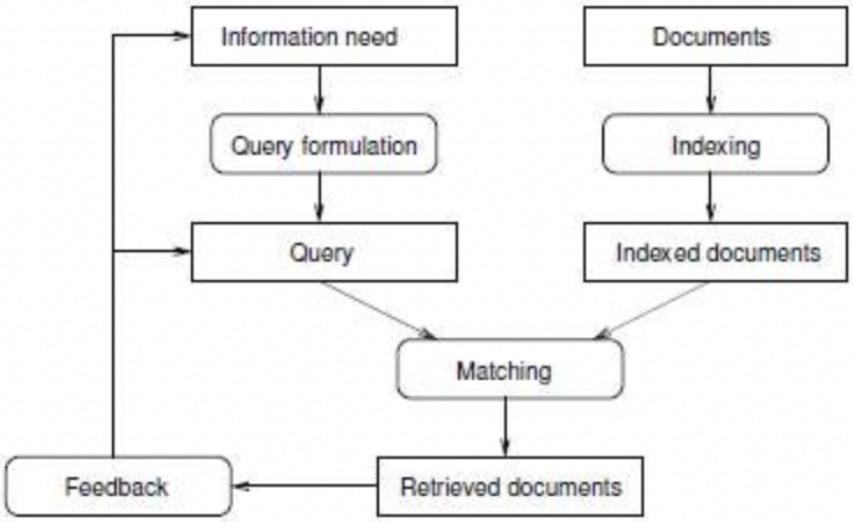

FIGURE 2.6: Information Retrieval Process [Lewis and Jones, 1996].

The above Figure 2.6 illustrates the complete Information Retrieval flow starting from formulating users request to retrieving relevant documents in the form of documents. The operating procedure involves indexing the documents collected, transforming the query in the same way as the document content is represented, comparing the description of each document with that of the query and listing the results in order of relevancy [Lewis and Jones, 1996].
Some of the key components of Information Retrieval system are:

1. D: Document Representation.
2. Q: Query Representation.
3. F: A framework to match and establish a relationship between D and Q.
4. R(q, di): A ranking function that determines the similarity between the query and the document to display relevant information.

2.4.1 Types of Information Retrieval Models

Broadly IR models are classified as:

1. Classical IR Model: These are widely used IR models based on mathematical concepts where the retrieval of information depends on documents containing the

defined set of queries. These are further classified as Vector-space, Boolean and Probabilistic IR Models.

2. Non-Classical IR Models: These IR models are built upon propositional logic. Examples of non-classical IR models include Information Logic, Situation Theory, and

2.4.1.1 Sparse Retriever Model (TF-IDF/ BM25)

This is a traditional type of Information Retrieval system relays on sparse vectors for the purpose of extracting relevant information from our document repositories. TF-IDF is a simple algorithm used by these systems to calculate the similarities between the texts. TF refers to how many words in the query are found in the context where as IDF is the inverse of the fraction of documents containing this word. The product of TF and IDF results in TF-IDF score. BM25 is a variation of TF-IDF where TF score is dampened after returning large numbers of matches between the query and contexts. Additionally, it also considers the document length. The TF-IDF score is normalized so that short documents will score better than long documents given they both have the same number of word matches [Svore and Burges, 2009].

The 3 factors that affect the weight of each query are:

1. Term Frequency (TF): The frequency of which the query term appears in the document.
2. Inverse Document Frequency (IDF): Some words (such as "in") appear in almost all the documents but it's not as important as the actual query word.
3. Length of Document: A short document that has repetitive query is more important than long document.

2.4.1.2 Dense Passage Retriever (DPR)

DPR is an alternative to the traditional TF-IDF / BM25 model which aims to index all the passages in a low-dimensional and continuous space to retrieve top k passages relevant to the input question. DPR utilises a dense encoder E_p () which converts every text passage into a d-dimensional real-valued vector and generates an index for all M passages to be retrieved and different encoder E_q () to map the input queries to d dimensional vectors and retrieves k passages containing the vectors that are the most

similar to the query vector [Karpukhin et al., 2020] .

The similarities between queries and passages is defined in the Equation 2.1:

$$\text{sim}_{(\mathbf{p},\mathbf{q})} = \mathbf{E_Q}(\mathbf{q})^{\mathbf{T}} \cdot \mathbf{E_P}(\mathbf{p}) \tag{2.1}$$

2.5 Machine Reading Comprehension (MRC)

Machine Reading Comprehension (MRC) is defined as the process of finding an Answer (A) to the Question (Q) by learning the function (F) such that A = F (C, Q). MRC could be widely applied in many NLP systems such as search engines and dialogue systems. This can be adopted to tasks like Question Answering (QA) to span relevant answers from the Wikipedia retrieved document [Chen et al., 2017]. MRC can be broadly divided into two types based on the answer predicting approach.

2.5.1 Extractive Question Answering

Extractive question answering is a specific type of system that effectively used to answer the question in the given context. The goal is to classify the start and end position of the span in the context [Rajpurkar et al., 2016]. In this system, the encoder first receives a set of questions and a set of relevant contexts referred as tokens in question and context. It then produces contextual representation denoted by h. The final two linear layer predicts the probability of each token h to classify start and end position independently [Yang et al., 2020]. Here the training objective is to minimize the loss function.

The loss function is defined in the Equation 2.2:

$$L_{Ext} = -\operatorname{lop}\left(P_{(start,s)}\right) - \log\left(P_{(end,e)}\right) \tag{2.2}$$

where,

$$P_{start} = \text{softmax}\left(w_{\text{start}}\, h\right) \tag{2.3}$$

$$P_{end} = \text{softmax}\left(w_{end} h\right) \tag{2.4}$$

P(start,s) in Equation 2.3 and P(end,e) in Equation 2.4 denote the probability ground truth of start and end tokens of answer.

2.5.2 Generative Question Answering

Generative Question Answering system aims to generate an abstractive answer to a given question instead of extracting an answer span from a provided passage by autoregressively predicting tokens [Yang et al., 2020]. In this system, an encoder takes a question Q and a context C as input and outputs contextual representation h. A decoder takes the previously generated answer tokens as input and performs attention over h to generates a tokens.

2.6 Machine learning concepts

Machine Learning is the process of teaching computers to program themselves. If programming is automation, then machine learning is the technique of automating automation. This section will discuss some of the general machine learning methods and concepts.

2.6.1 Model training and train/ test dataset

Training an ML model involves providing training data to an ML algorithm i.e learning algorithms that updates weights and bias. The term ML model refers to the model artefact produced after the training process. A training set is the largest portion of the entire dataset that is used to train the model (update the model's weights). During every epochs, the same dataset is fed to the neural network so that the model learns the features of the data repeatedly. The training set should include a diverse collection of inputs so that the model can be trained in all settings and predict any previously unknown data sample that may arise in the future. Generally 70 % of the entire dataset is used as a training set.

A test set in machine learning is a secondary data set that is used to test a machine learning model performance after it has been trained on an initial training data set. Generally 10 % of the entire dataset is used as a test dataset.

2.6.2 Cross validation and validation dataset.

Cross validation is typically used to measure the model's performance accuracy on unseen data at the end of each epoch. That is, to use a subset of data to estimate how the model will perform in general when used to generate predictions on data that was not utilised

during the model's training. In the famous k-Fold Cross Validation, the data is separated into k subsets (usually K=5, K=10). The approach is now repeated k times, with one of the k subsets serving as the validation set and the other k-1 subsets as training set each time. This enables efficient utilisation of features in data and considerably decreases bias and variance thus avoiding over-fitting.

The validation set is a different portion of the dataset that will be used during the training process to see how well your model performs on the data that were not used in training. During training, its usual to report the validation metrics such as validation loss after each training epoch as it helps to determine whether the model has achieved the highest performance possible on validation set. The training should be cease at this stage which is referred as "early stopping". Generally 20 % of the entire dataset is reserved as validation set.

2.6.3 Transfer Learning

Transfer Learning is a machine learning approach in which knowledge gained from a model used in one task is reused as the foundation for another task. Transfer learning may be utilised to achieve excellent performance with minimal human supervision on many down streaming tasks including Question Answering. In BERT, Fine tuning is relatively inexpensive procedure of modifying a pretrained model using the task specific dataset by adding just one additional output layer for classification and retraining the model (updating the weights) on new dataset for few Epochs [Wolf et al., 2020].

2.7 Evaluation

It is critical to assess the model's performance. In general, the loss function is used in machine learning methods to determine how close or far away the model was to making the correct prediction when compared to the actual answer. In Open Domain QA. , the Retriever and Reader models are often assessed independently. This section discusses several popular datasets and metrics for both the retriever and the reader.

2.7.1 Evaluation of Retrieval Model

In an information retrieval system, evaluation is a critical and tedious task that relies around the concept of relevant and irrelevant documents. The efficiency of information retrieval systems is evaluated by comparing performance, functionality, and systematic

approach on a common set of queries and documents (like SQuAD). This section will discuss some of the evaluation metrics used for evaluating retriever model.

Metrics for Retriever model Evaluation

1. **Recall@k**: Recall is to access the systems ability to extract relevant documents. Recall@k is defined as the percentage of relevant documents retrieved in the top-k recommendations. Suppose recall at 10 is calculated as 40% in the top-10 recommended documents. This signifies that 40% of the total number of relevant documents appear in the top-k results. Mathematically recall@k is defined in Equation 2.5

$$\text{Recall@k}=\frac{\text{Total no of recommended items @k that are relevant}}{\text{Total number of relevant items}} * 100 \quad (2.5)$$

2. **Precision@k**: Precision is to access the systems ability to reject the non-relevant documents. Precision at k is defined as the percentage of relevant recommended items in the top-k set. It is interpreted as follows. Assume my precision at 10 is 80% in a top-10 recommendations. This signifies that 80% of recommendations are relevant. Mathematically precision@k is defined in Equation 2.6

$$\text{Precision@k}= \frac{\text{Total no of recommended items @ } k \text{ that are relevant}}{\text{Total no of recommended items @k}} * 100 \quad (2.6)$$

2.7.2 Evaluation of reader models

While evaluating language models, usually the retrieval procedure is skipped and the documents containing the answer span is directly passed to the reader. The main objective is to access the systems ability to predict answers that resembles the correct answer. This section will discuss some of the commonly available intrinsic evaluation matrices used to assess the reader models.

2.7.2.1 Metrics for reader model Evaluation

Exact Match (EM)

Exact Match (EM) is strictly all-or-nothing metric. For every question and answer set, if the characters of the predicted answers perfectly match the characters of the true answers, EM = 1, else EM = 0. Exact match ratio is calculated over all questions in the dataset and is defined as the percentage of predicted answers that perfectly match characters of the true answers. The exact match ratio is a fairly precise indicator of model performance. It only rises when the model properly predicts every characters of true answers, with no false positives. Mathematically, exact match ratio is defined in Equation 4.5.

$$\text{Exact match ratio } = \frac{\text{Number of examples with exact label match}}{\text{Total number of examples}} * 100 \tag{2.7}$$

F1-Score The F1 score is a popular metric used in QA and classification problems. It is calculated by comparing the individual words of predicted answer against those of true answer. Therefore F1 score is based on the amount of shared words between predicted and the true answers. Precision is the ratio of common words to total number of words in the predictions. Some of the common terminologies while calculating F1-Score are defined bellow.

1. **True Positive (Tp)**: The total amount of tokens shared by the correct answer and the prediction.
2. **False Positive (Fp)**: The amount of tokens in the prediction that aren't in the correct answer.
3. **False Negative (Fn)**: The amount of tokens in the correct answer that aren't in the prediction.

This is depicted in the Table A.2, which is referred as confusion matrix where each row belongs to the predicted class, whereas each column belongs to the actual class. 'True' and 'False' show whether the items are correct otherwise in-correct. The words 'Positive' and 'Negative' show whether the classification results are in the corrected class.

	Actual: Positive	**Actual: Negative**
Predicted: Positive	True Positive (TP)	False Positive (FP)
Predicted: Negative	False Negative (FN)	True Negative (TN)

TABLE 2.1: Confusion Matrix [Visa et al., 2011].

The following equations depicts the formula to calculate Precision in Equation 2.8, Recall in Equation 2.9 and F1 Score in Equation 2.10.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positive}} \tag{2.8}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.9}$$

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.10}$$

Other evaluation metrics for machine reading comprehension models are:

1. **Accuracy** measure is calculated in closed domain evaluation, and readers receives value 1, if the predicted answer has any word overlap with the true answer. This matrix is more efficient compared to Recall and F1 since, the context around the predicted answer will be presented to the user.

2. **BELU** introduced by Papineni et al. [2002] is a "word-overlap" metrics which uses the number of words that overlap between a generated translation and some human translations. It considers the accuracy of a translation by taking a weighted average of the precision obtained for variable length in reference translations.

3. **ROUGE** introduced by Lin [2004] is an evaluation matrices that was created to evaluate automatic summarization models. These models do not have a good correlation with human judgments regarding QA quality.

2.7.3 Machine Reading Comprehension datasets

There are wide range of publicly available datasets built on MRC to evaluate the performance of the model. Some of the famous datasets are discussed in this section.

2.7.3.1 Stanford Question Answering Dataset (SQAD)

Stanford Question Answering Dataset (SQAD) [Rajpurkar et al., 2016] is one of the famous reading comprehension dataset with over 100K question-answer pairs in which one user asks a question about a related Wikipedia article and another responds with a span of the answer in the article. This dataset contains orders of magnitude more data than any previous hand-annotated dataset, making it ideal for neural techniques where performance scales with dataset size. An improved version of this is SQAD 2.0 which also contains unanswerable questions [Rajpurkar et al., 2016] .
Some of the drawbacks of SQAD datasets is that language models are sensitive to adversarial cases that have been trained to confuse them. To overcome this issue, authors developed Adversarial SQuAD, which has distracting lines from Wikipedia articles that are meant to lead QA models to make inaccurate predictions [Jia and Liang, 2017].

2.7.3.2 Question Rewriting in Conversational Context (QReCC)

Question Rewriting in Conversational Context (QReCC) introduced by Anantha et al. [2020] is a dataset that contains 14K conversations with 80K question-answer pairs within a collection of 10M web pages that are split into 54M passages. The key feature of this dataset is to provide annotations to train and evaluate task of question rewriting. It consists of a sequence of Question Answering pairs where every QA is annotated with a question rewrite. Previously mentioned Machine comprehension datasets are limited to conversation in a single document. QReCC was the first dataset to provide resources for the community to develop, evaluate and incorporate open Domain Question Answering with scripts for building a collection of passages from the Common Crawl and the Wayback Machine for passage retrieval. [Anantha et al., 2020]

Chapter 3

Requirement Analysis

This Requirement Analysis chapter will discuss about the MosCoW analysis, language and frameworks, dataset and computing resource used for training, testing and finetuning the language models.

3.1 MosCoW Analysis

The MoSCoW technique described in [Walker et al., 1997], is used to prioritize requirements of the projects where each requirement is categorised by 'Must have', 'Should have', 'Could have' and 'Won't have this time' to indicate their respective priority. The Table 3.1 illustrates the MosCoW analysis of this project.

N°	Requirement Description	MoSCow	Priority	Requirement Type
1	Indexing latest Wikipedia dump	Must	High	Functional
2	Implementing BM25 based search engine for information retrieval	Must	High	Functional
3	Evaluate Retriever model using intrinsic evaluation metrics.	Should	Medium	Functional
4	Training and validating DrQA reader model.	Must	High	Functional
5	Fine tuning and validating BERT-base reader model on SQuAD dataset.	Should	Medium	Functional
6	Evaluate both reader models using intrinsic evaluation metrics	Must	High	Functional
7	Compare the base model performances with other SOTA models.	Could	High	Non-Functional
8	Error analysis using model and data ablations.	Wont have(this time)	Low	Functional
9	Implementing conversational aspect with QuReCC dataset.	Wont have(this time)	Low	Functional

Table 3.1: Project requirements.

3.2 Language and Frameworks

Python programming language will be used for the programming aspects of this project. Python has become the industry standard for AI practitioners because of the vast community supports and the availability of popular machine learning libraries such as Tensor flow and PyTorch [Abadi et al., 2016]

This project can be implemented in both TensorFlow and PyTorch. But PyTorch will be chosen for this project as it adheres to conventional pythonic programming practises opposed to TensorFlow. PyTorch make it simple to define the layers in neural networks and also to train the models effectively utilising low-level optimizations that are hidden from the end user. Other reason for choosing PyTorch is because of the fact that the author is better experienced with PyTorch libraries and tools.

Keras is another library that was considered. On top of existing Deep Learning frameworks. This high-level AI framework is ideal for training neural networks using simple syntax. This will not be utilised because it does not run on top of PyTorch and all of the layers in BERT and BiLSTM can be better implemented in PyTorch. [Chollet, 2016]

3.3 Datasets

This project uses public datasets for the task of information retrieval and machine reading comprehension. Other popular Machine Comprehension datasets are discussed in the literature (see Section 2.5). This section will discuss about the datasets used in this project:

3.3.1 English Wikipedia Dump

In this project English Wikipedia serves as a knowledge source for finding answers. The details of latest Wikipedia dump are as follows:

1. Size: 89 GB
2. Total number of Articles: 6 Million
3. Total number of words: 3.9 billion
4. Dump date: 01-January-2022

3.3.2 SQuAD

SQuAD is the public dataset used in this project to train, test and evaluate the DrQA and BERT-base language models. Some key factors taken into consideration before choosing SQuAD dataset over others datasets are

1. **SQuAD is non-conversational:** The questions in SQuAD dataset are non-conversational in nature, making this dataset unique for training and evaluating language models in this project, as the primary objective of this project is to implement a QA system that can answer open domain questions in a single turn while disregarding the contexts of previous turns.

2. **SQuAD is huge and is Wikipedia based:** This dataset contains 100K question-answer pairs to aid in the development and evaluation of intensive and complex language models. (See Section 2.7.3.1). Furthermore, the contexts in the dataset are taken from Wikipedia, making this feature ideal for retriever evaluation since Wikipedia is the primary knowledge source in this project.

3. **SQuAD is challenging:** SQuAD V2.0 focus on unanswerable questions which makes this dataset much challenging compared to other datasets. In order to do effectively on SQuAD 2.0, systems must not only answer questions when they are possible, but also recognise when no answer is supported by the paragraph and avoid from answering. See Section 5.1.1.1 for challenges in the upgraded version.

3.4 Computing resources

The entire code in this project will be developed and tested on local Pycharm environment powered by Apple M1 Max processor with 10-core CPU, 32-core GPU & 16 GB RAM but not equipped with CUDA.

Training large scale language models requires Pytorch-CUDA enabled GUP. Therefore, this project majorly uses NVIDIA TESLA P100 with 32GB RAM for GPU and memory acceleration while train DrQA and BERT-base language models on SQuAD V1.0 & V2.0 datasets. This resource will be accessed through Google Colab PRO+ which is totally cloud-based. The model developed in local environment is uploaded to Google Colab using Jupyter Notebook environment solely for training, validation and evaluation.

Chapter 4

Models and implementation

4.1 Information Retriever

Wikipedia being one of the Internet's largest and most popular information websites. This project employs an efficient non-machine learning based document retrieval method to first narrow down the search space and focus on reading specific paragraphs of articles that are likely to be relevant for the question. Information Retriever is mainly composed of two components. Firstly, Wikipedia Indexing for web scraping of Wikipedia pages manually to index nearly 6 million Wikipedia article titles and secondly, Wikipedia search engine to extract the relevant top_K_N paragraphs where the span of answer is most likely to appear. This section will discuss about retrieving relevant paragraphs from the latest English Wikipedia dump (see Section 3.3.1).

4.1.1 Wikipedia Indexing

4.1.1.1 Parsing XML dump and data pre-processing

In this project SAX parser is used to parse the XML file and extract necessary data from the Wikipedia dump. SAX Parser provides a technique for processing XML documents using event handlers [Pan et al., 2008] . Only texts and titles from the XML file will be utilised for indexing by stripping away images, figures, tables. Each title will be allocated with a self-provided id using the variable number of paragraphs and tokens in each page.

Text processing and page processing are two aspects of data pre-processing. This is necessary to avoid adding irrelevant tokens to the Index. The following are the major text processing steps:

1. Removing punctuations like. ,!$( )*%@
2. Eliminating stop words using nltk library.
3. Text tokenisation using nltk library
4. Removing HTML tags and non-ASCII characters.
5. Stemming- Python library called PyStemmer is used for this task.
6. Lemmatization- Using nltk library.

Every Wikipedia page is processed by extracting various fields for simple and multi-field querying. In this process, the six fields indexed are: title, body, category, infobox, links, and references. This enables users to search information through generic or field-specific queries on these fields indexed.

4.1.1.2 Single-pass in-memory indexing (SPIMI)

Creating the final Index directly from a huge collection of data, such as a Wikipedia dump, will be a time-consuming process that can also blow up memory. Therefore, initial, and intermediate indexes are created before merging it into the final index. Indexing of the Wikipedia page titles is performed using SPIMI approach which uses terms and termIDs for indexing. SPIMI algorithm writes each block of dictionary to the disk, and then starts a new dictionary for the next block while parsing and converting documents into a stream of term-docID pairs (referred to as tokens). SPIMI-INVERT is called on the token stream repeatedly until the entire collection of documents is processed. [Mo'minov, 2017]

During each subsequent SPIMI-INVERT call, tokens are handled one at a time. If a term appears for the first time, it will be added to the dictionary (ideally referred as a hash) and a new list will be created. SPIMI has a critical component known as compression, which allows both the postings and the dictionary terms to be saved on disc using less disc space. This increases the algorithm's performance by eliminating expensive document sorting methods.

After creating an initial indexing, Intermediate index is created to handle instances when the token information is scattered over many files. This is done by merging the initial index. Unlike initial indexes, the intermediate files are divided into fields. The format of initial and intermediate index files looks like:

Initial index: banana-2141:5;1232:1;5432:78;
Intermediate index : banana-5985:t3b6i2r1;4958:t5c8b3i1;

In the above example, "banana" is a token. After the hyphen, each pair is separated by ";". The first value before ":" represents the docID of the token, followed by the associated field and value representing frequencies. The index t3b6i2r1 represents the token "banana" appears thrice in the title, six times in the body, twice in the infobox, and once in the reference.

The final index is generated from intermediate index to allow faster information retrieval process to save the disk space. The original and intermediate indexing files are removed at the end of this step. The final index format is illustrated below. The token "banana" has a value of 345, which corresponds to the token's frequency across the Wikipedia dump. All succeeding values are field_files where the token appears.

Final index: banana-345-1-3-4–2-4-6-'.split('-'): [banana, 345 '1', '3', '4', '', '2', '4', '6', '']

The fourth element ('3') in the above example indicates that the token 'banana' appears on line 3 of file number 1 of title file. Similarly, '4' indicates that the token appears on line 4 of the body file. The token does not appear in the category file since it is an empty string ('). Further, token appears in Line 2 of the infobox file, line 4 of the link file, and line 6 of the reference file. If the token appears in any field, it will only be present in that exact file number and nowhere else.

4.1.2 Wikipedia Search Engine

The crucial part of Information retriever is searching for relevant articles for a given query and implementing ranking functionality to arrange the documents according to its relevance where the most relevant document appear first.

This project incorporates two levels of filtering to build an effective information retrieval system that narrows the search space. Firstly, Single stage Retriever extracts and ranks the top K Wikipedia Article titles where the span of answer for a given question might appear in the content of those Articles. Secondly, Multi Stage Retriever extracts and ranks the top_N individual paragraphs from the extracted Wikipedia titles where the span of answer might contain in those Top_N retrieved paragraphs.

Query parsing and pre-processing is same as text pre-processing while indexing (see Section 4.1.1.1). One of the main advantages of this search engine is the ability to search for information using both simple and multi-field queries i.e Users are allowed to search information either using a simple query or using Simple and field specific query or simple and field specific query as illustrated bellow.

Type 1 (Simple Query):- What is the capital of Scotland ?
Type 2 (Field Query):- b:Who is the prime minister of India? i:2022
Type 3 (Simple+ Field):- Who is Boris Johnson? t:United Kingdom i:2022.

4.1.2.1 Single stage Retriever (BM25)

This stage retrieves and sorts the top K relevant Wikipedia article titles. For a given question, the File traverser finds each token of the question in the indexed vocabulary and records the file number, document ID, and page frequency. The Query results gets all the title indexed. This data is then sent into the BM25 ranking algorithm, which returns the top k titles.

The ranking function is the re-implementation of Okapi BM25 model which consists of 4 major components illustrated by following definitions

1. **Parser module:** The parser module creates a list and a dictionary from the query and corpus files.
2. **Query processor**: The query processor scores the documents depending on the terms in each query of the query list.
3. **Ranking function:** The ranking function uses the natural algorithm to rank the documents retrieved.
4. **Data structures:** An inverted index and a document length table are included in the data structures module. Each word is mapped to a dictionary in the inverted index, and this secondary dictionary maps each document id to the word frequency in the outer dictionary. The document length table includes the length of each document as well as a method to compute the collection's average document length.

The BM25 scoring function is defined in the Equation 4.1:

$$\text{score}(q,d) = \sum_{i=1}^{|q|} \text{idf}\left(q_i\right) \cdot \frac{tf\left(q_i,d\right) \cdot \left(k_1+1\right)}{tf\left(q_i,d\right) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)} \tag{4.1}$$

Where,

1. f(q_i,d) correlates to the term's frequency, defined as the number of times query term qi appears in the document d.

2. d is the length of the document in words (terms), defined in the Equation 4.2 :

$$|d| = \frac{1}{(\text{ norm } * \text{ norm })} \tag{4.2}$$

   where norm is the score factor used by Lucene's default similarity function in Fang and Zhai [2009]

3. avgdl is the average document length over all the documents of the collection.

4. k1 and b are free parameters, usually chosen as k1 = 2.0 and b = 0.75.

5. idf(q_i)is the inverse document frequency weight of the query term qi.

4.1.2.2 Multistage Retriever (Wikipedia API +BM25)

At this stage of the information retrieval process, the algorithm extracts and ranks the top N Wikipedia paragraphs that are relevant to the query where the span of answer might exist in the extracted paragraphs.

Unlike SSR, the MSR will not manually web scraping Wikipedia passages as this is time consuming and expensive process that might blow up the memory. To simplify the process, the Wikipedia API will be utilised to extract the full plain text of a Wikipedia page, removing HTML elements, URLs, picture references, equations, tables, and navigational components. This is done by simply sending the Wikipedia article title obtained from the Single stage retriever as an argument to the function get_text("page title"). After that, the plain text is processed to extract individual paragraphs by removing any leftover special characters, references, and links. Any paragraph fewer than 400 characters is filtered.

These extracted paragraphs are then passed to the BM25 ranking algorithm, which sorts the top N paragraphs. These top N paragraphs are treated as individual documents for the Machine comprehension model. Some of the Hyper-parameters for Information retrieval system are **K1**: 1.2, **K2**: 100, **b**: 0.75, **K**: Number of Articles, **T**: Threshold, **N**: Number of Paragraphs, **Field Weightage**: 'title':1.0, 'body':0.6, 'category':0.4, 'infobox':0.75, 'link':0.20, 'reference':0.25

4.2 Reader models

The reader models learns the ability to solve the reading comprehension challenge. In this project the reading comprehension task is accomplished by two language models, the first model is a traditional neural network based model inspired from DrQA proposed by Facebook [Chen et al., 2017] and the second model is inspired from the recent success of the self attention Transformer based model BERT-base as discussed in the literature 2.3.3.2. Both models are extractive question answering system which involves extracting the answer to a specific question from a document. (see Section 2.5.1)

This section will discuss about the data pre-processing, re-implementation techniques of DrQA model including the training procedure on SQuAD datasets and implementation of fine-tuning technique for pertained BERT-base language model on SQuAD datasets for the down streaming task of question Answering.

4.2.1 Data pre-processing

This section will discuss about the pre-processing of SQuAD datasets (V1.0 & V2.0) before feeding it to the reader models. As discussed in the literature (see Section 2.7.3.1), SQuAD comprises of 100,000 questions on a collection of Wikipedia articles, with the answer to each question is a text segment, or span, from the relevant reading passage. In second version (SQuAD V2.0), 50,000 unanswerable questions were added to make them appear like answerable questions. As a result, the second version is significantly more complicated than SQuAD V1.0. The objectives of using SQuAD V2.0 dataset is to check if the system could recognise question when no answers is supported by the paragraph and thus avoid from answering.

In this project, the data pre-processing for both DrQA and BERT-base follows the same methodology with different tokenization and additional data pre-processing stages for BERT model. This simplifies the code by avoiding redundant functions. The data pre-processing steps are:

4.2.1.1 Data Splitting and structuring

The SQuAD datasets are offered through train and development json files. The train dataset file was further splitted into 90 % of training dataset and 10 % of validation dataset. The dev dataset file was used for the evaluation of best performing model (see Section 4). The Table bellow summarizes overall statistics of both the versions of the

dataset. All these information was stored to a data frame of columns questions, contexts, and answers.

	SQuAD 1.1	**SQuAD 2.0**
Train		
Total examples	87,599	130,319
Negative examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10,570	11,873
Negative examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9,533	8,862
Negative examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

TABLE 4.1: Dataset statistics of SQuAD V1.0 V2.0

4.2.1.2 Find the Start/ end position character

The answer's start and ending position characters are necessary for reader models. Predicted answers sometimes "eat" one or two characters. For instance, model returns "Six" instead of the word "sixth". Under these circumstances, the strategy of "cutting" the paragraph answer by 1 or 2 characters to match the true answer is adopted to keep up with the true answer the reader model is expecting during training.

4.2.1.3 Tokenisation using Glove word embedding for DrQA

In this project, 100-dimensional Glove word embeddings [Pennington et al., 2014] trained on 840B Web crawl data are applied to extract each token's corresponding word embedding by fine-tuning the 1000 most frequently occurring question words. The Equation bellow 4.3 illustrates the Glove word embedding.

$$f_{emb}\left(p_i\right) = \mathbf{E}\left(p_i\right) \tag{4.3}$$

Finally, truncation and padding are applied to contexts and questions, with contexts truncated/padded to 512 characters and questions to 40. The padding tokens are all zero tensors. It should be noted that a '[UNK]' token is placed at the beginning of each context, and in SQUAD 2.0, the start/end labels of unanswerable questions are indicated as 0, this returns the initial '[UNK]' token for answers to unanswerable questions.

4.2.1.4 Tokenisation and additional stages for BERT-base

BERT employs wordpiece tokenization to reduce the quantity of vocabulary. This enhances training performance by utilising to delimit split tokens. For instance consider the following terms: (run, running, runner). Without wordpiece tokenization , the model will store and learn the meaning of all three words individually. However, with wordpiece tokenization, the model learns the context of the word "ran," and the remaining meaning is stored in the suffix. To represent the embedding of each word in this project, the Huggingface library's BERT-base-uncased tokenizer is utilized. Truncation and padding is performed with a maximum token length of 15 for questions and 512 for contexts. BERT-base comprises several training tokens that should be stored in a word dictionary with an index number for each token and word in the vocabulary. The Table 4.2 depicts the function of several tokens

Tokens	**Purpose**
[CLS]	Classification is usually the first token.
[SEP]	Two sentences are separated.
[END]	End of token.
[PAD]	To truncate the sentence same length is used.
[MASK]	By substituting the original word, Mask can be created.

Table 4.2: BERT tokens and purposes.

The 3 Embedding layers are:

1. Token Embeddings: In the sentence "The man is walking. The dog is barking". Token embedding creates the sentence as "[CLS] the man is walking [SEP] the dog is barking". This is later converted to index from word of dictionary.

2. Segment Embeddings: Two phrases are separated by a segment embedding, which is commonly specified as 0 or 1.

3. Position Embeddings: Each embedding in a sequence is given a position via a position embedding.

The Figure 4.1 bellow depicts the use of BERT tokenizer to get token_ids, attention_mask, sequence_mask both in context or question sequence before feeding into the model.

```
from transformers import BertTokenizerFast
tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')

encodings = tokenizer(train_contexts, train_questions, truncation=True, max_length= 512)
```

FIGURE 4.1: BERT-tokeniser downloaded from Huggingface library.

4.2.1.5 Create a Dataset class

Finally, the Squaddataset class is commonly developed for the DrQA and BERT models to easily transform the train, validation, and evaluation encodings before feeding into the models. Torch.utils.data.Dataset is inherited in this class. This class also helps in tokenising datasets during training and validation rather than keeping all tokens beforehand, which saves RAM.

4.3 DrQA

The main objective of DrQA model is to predict the probability of start and end position of the answer span in the given context. The flow chart of DrQA models is illustrated in the Figure 4.2 bellow. It consists of 2 modules for encoding question and passages tokens. After data pre-processing discussed earlier (see Section 4.2.1), for a query q with l tokens q1, q2...ql and single paragraph with m tokens Z1, Z2...Zm, these tokens are represented as a sequence of feature vectors before passing them as inputs to recurrent neural network.

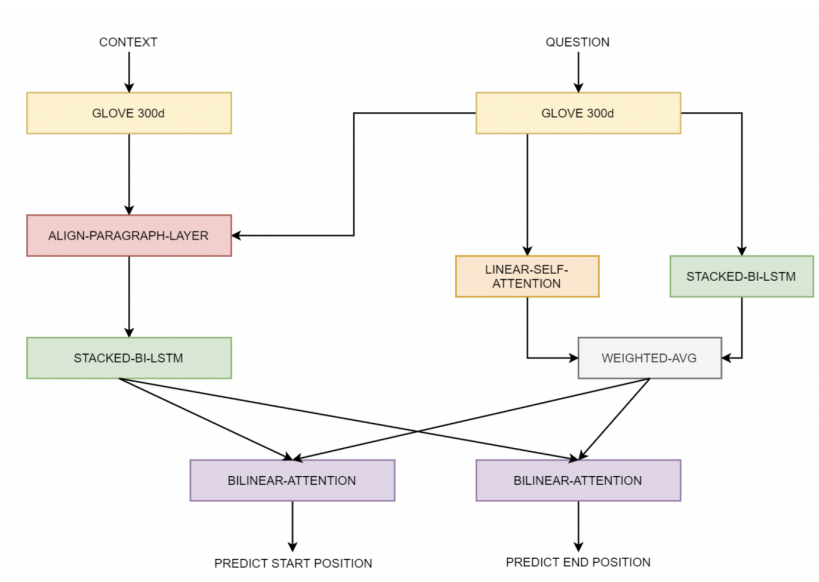

FIGURE 4.2: Flowchart demonstrating the flow of DrQA model.

Passage encoding module consists of sequence to feature embeddings with multi-layer bi-directional LSTM (BiLSTM). Concatenation of the hidden units at each layer yields the sequence of feature vectors. The following Equation 4.4 represents the feature vector of passage encoding where relevant contextual information is stored in token Zi.

$$\{\mathbf{Z}_1, \ldots, \mathbf{Z}_m\} = \mathrm{RNN}\left(\left\{\tilde{\mathbf{Z}}_1, \ldots, \tilde{\mathbf{Z}}_m\right\}\right) \tag{4.4}$$

The feature Zi is divided into numerous categories:

1. **Word Embeddings**: As discussed in data pre-processing (see Section 4.2.1), an 800B Web crawl-trained 300d Glove word embedding is used to get the word embedding that corresponds to each token.

2. **Exact match**: This binary feature indicates weather the token Zi appears in question x. The exact match is illustrated in the bellow Equation 4.5:

$$f_{\text{match}} = \mathbb{I}\left(Z_i \in x\right) \tag{4.5}$$

3. **Token features**: Few manual features that represent certain attributes of the token pi are included in its context. It includes POS (part-of-speech) tagging, Named Entity Recognition (NER), and TF (Term-Frequency). The equation of additional feature representation is illustrated in the bellow Equation 4.6.

$$f_{\text{token}}\left(Z_i\right) = \left(\mathrm{POS}\left(Z_i\right), \mathrm{NER}\left(Z_i\right), \mathrm{TF}\left(Z_i\right)\right) \tag{4.6}$$

In question encoding module, the sequence to word embeddings of question tokens are fed into another multi-layer bi-directional LSTM with self-attention on top to extract the question vector which consists of the semantic information of the given question. The attention score is intended to account for inter-sentence matching and similarity between the paragraph token and the query tokens. This feature provides soft alignments between words that are related but not identical.

The Equations 4.7 & 4.8 bellow illustrates the question embeddings with attention layer on top. Where Eg($\dot{}$) is the word embedding from glove and 'alpha' is Relu with single dense layer.

$$f_{\text{align}}\ (z_i) = \sum_j y_{i,j} E_g\left(x_j\right) \tag{4.7}$$

$$y_{i,j} = \frac{\exp\left(\alpha\left(E_g\left(z_i\right)\right)^\top \alpha\left(E_g\left(x_j\right)\right)\right)}{\sum_{j'} \exp\left(\alpha\left(E_g\left(z_i\right)\right)^\top \alpha\left(E_g\left(x_{j'}\right)\right)\right)} \tag{4.8}$$

Once the feature vectors for the question and associated paragraphs are created, the model tries predict the probability of the start and end position of an answer span appearing in the context by performing a linear transformation on these vectors. The Equation **??** represents the maximum probability of start score (Pstart)and end score (Pend) of the span where Ws & We are learned parameters.

$$p_{\text{start}}\ (i_s) \propto \exp\left(\mathbf{z}_{i_s} \boldsymbol{W}_s \mathbf{x}\right) \tag{4.9}$$

$$p_{\text{end}}\ (i_e) \propto \exp\left(\boldsymbol{z}_{i_e} \mathbf{W}_e \mathbf{x}\right) \tag{4.10}$$

$$\text{s.t. } i_s \leq i_e \leq i_s + 15 \tag{4.11}$$

4.3.1 Training , Validation and hyper-parameters details of DrQA

In this project, DrQA model is trained and validated on SQUAD V1.0 & V2.0 training dataset. After obtaining the start and end probability vectors. The similarity of classification logits for start or end vectors is calculated by taking the product of start or end vector and BiLSTM context tokens. At the end of forward propagation, cross-entropy loss is estimated using the Sotmax function, followed by log computations on the start/end logits and ground truth labels. During back propagation, the weights are updated using the Adam optimizer.

Every 1000 iterations, the model is validated on the validation dataset by calculating the Em and F1 scores (see Section 2.7.2.1). After each checkpoint, the best performing model is saved. After every epoch average training and validation loss is calculated. Early stopping trigger is used to determine the maximum number of epochs to avoids over-fitting of the DrQA model i.e training is terminated as soon as the performance on the validation dataset falls below the performance prior to training of that epoch. This is reflected in an increase in validation loss. Here Local minima are also considered to avoid premature termination at the wrong epochs. Hyper-parameters play a vital role in controlling the behaviour of a DrQA model. Therefore, most of the hyper-parameters used for training DrQA model are default parameters reported in the paper [Chen et al., 2017]. The hyper-parameter values used for training and validation of DrQA model is illustrated in Appendix A.1.

4.3.2 Bidirectional Encoder Representations from Transformers (BERT-Base)

The BERT model used in this project is a BERT-base-uncased model downloaded from the Huggingaface transformers library, which acts as a template class for developers to utilise for constructing new classes based on pre-trained models. This is a self-attention-based model that's been pretrained on a massive corpus of English data (BookCorpus) in self-supervised fashion, specifically using two objectives called Masked language modelling (MLM) and Next sentence prediction (NSP), as discussed in the literature (see Section 2.3.3.2).

4.3.3 Training, validation and hyper-parameter details of BERT-base

In this project, the pretrained BERT-base language model is fine-tuned separately on SQUAD V1.0 &V2.0 training dataset for the down streaming task of Question Answering. This is referred as transfer learning as discussed in literature (see Section 2.6.3).

The main objective of this model is to predict the start and end position of answer in a given context. A simple classification layer (refereed as QA-head) is added on top of 12 hidden layers to categorise answer start and end tokens of the context. The pretrained weight and bias values are freezed while loading the model. The pre-processed and tokenised data (see Section 4.2.1) is fed into the Data loader class, which loads the dataset into the model and iterates across batches. The data loader class provides the input ids, attention mask, start positions, end positions, and token type ids to the model for fine-tuning and validation.

During the forward-propagation, the model predicts the start and end position of the answer span by choosing a span with the highest total score (start score + end score) where end index $\geq$ start index. There are two sets of weights within the QA-head, one for the start token and another for the end token, both of which contain same dimensions as the output embeddings. The output embeddings are fed into QA-head, and the dot product between the start/end token weight and the output embeddings is calculated individually. Then, for each start and end token sets, a softmax activation is used to generate a probability distribution over all token sets. The tokens with the highest probability are selected as the start and end tokens.

During back-propagation, AdamW optimiser is used to update the values of weights. The model is validated over SQuAD validation dataset for every 1000 iterations by calculating Em and f1 score and the best model is saved after every checkpoint for further evaluation (see Section 5.2). BERT-base has 110M parameters, therefore in this project most of the hyper-parameters used for fine-tuning are default parameters from [Yang et al., 2019]. Some of major hyper-parameter considered for training and validation are illustrated in appendix (see Appendix A.2.1)

4.4 Fully connected pipeline

The fully connected pipeline follows implementation of BERT-Serini [Yang et al., 2019]. Its composed of two components. For a particular query, First, the Multi Stage Retriever model retrieves the top_K_N Wikipedia documents (for example, 5 paragraphs) and assigns a ranking score to each document (Retriever score). Secondly, one of the reader models, either DrQA or BERT (as selected by the user), determines the span of answer from each retrieved document and assigns a score to each answer (Reader score). Finally, the best answer is selected by linear interpolation of the retriever and reader score as shown in the Equation 4.12 below where µ is a hyper-parameter of range 0-1 The sample of interaction with fully connected pipeline is illustrated in the Appendix B.1.

$$\text{Reader_score} = \text{start_score} + \text{End_score}$$
$$\text{Retriever_score} = \text{BM25 document score}$$

$$S = (1 - \mu) \cdot S_{\text{Retriever_score}} + \mu \cdot S_{\text{Reader_score}} \tag{4.12}$$

Chapter 5

Evaluation and Results

This section will discuss the individual evaluation of Information Retriever and Machine Reading comprehension modules.

5.1 Retriever Evaluation

Firstly,the efficiency of document retriever module is evaluated using SQuAD dataset(both versions have same context). The main objective of retriever module is to find the relevant top N paragraphs where the span of answer for a given question is most likely to appear. This is evaluated by finding "The percentage of SQuAD questions for which the similar gold-context of SQuAD appears in the retrieved top_K_N paragraphs".

5.1.1 Reason for using SQuAD dataset

The Stanford Question Answering Dataset (SQuAD) is a novel reading comprehension dataset that comprises of 100,000+ questions and span of answers on a collection of high quality Wikipedia articles. The context of this dataset includes 23,215 gold paragraphs extracted from English Wikipedia where all the individual paragraphs are above 500 characters stripping away images, figures, and tables [Rajpurkar et al., 2016]. This feature makes the dataset unique for retriever module evaluation since Wikipedia is the source of knowledge for question answering in this project. Other factors are discussed previously (see Section 3.3.2). For comparison with the retrieved passages, SQuAD gold passages are considered as true context.

5.1.1.1 Challenges in SQuAD V2.0:

As discussed earlier SQuAD V2.0 mainly focus on addition of unanswerable questions which makes this dataset complicated compared to SQuAD V1.0. This section will discuss about the five different types of unanswerable questions found in SQuAD V2.0

1. **Entry salad:** A question with nonsensical references to context entities are unanswerable. (ex: What infinite hierarchy implies that the graph isomorphism problem is NQ-complete?))
2. **Topic Error:** A query that refers to an entity in the context but does not concentrate on that subject. Such questions may have answers, but the answer is unlikely to be discovered in the context. (ex: "How many earthquakes occur in California?" while the article's primary focus is on "Southern California,").
3. **Missing Information:** A hypothetical query in which the answer appears to be in the context but is not. (e.g. "What is the highest temperature in January?" while the article is about stock market.)
4. **Content Negation:** A query that seeks information that contradicts the information present in the context. (Ex, "Who didn't trigger the fall of British Empire?")

5.1.2 Experimental setup

As discussed earlier (see Section 4.1.2), Single Stage Retriever (SSR) extracts the top K article titles and Multi Stage Retriever (MSR) retrieves the top N paragraphs from SSR extracted context. In this project, both SSR and MSR are evaluated individually by varying the number of articles (K) and number of paragraphs (N) retrieved. The evaluation metrics and methodology are illustrated in this section.

5.1.2.1 Evaluation metrics

Each paragraphs retrieved by both SSR and MSR are compared with SQuAD gold passages using an automatic evaluation metrics called BERT-Score [Zhang et al., 2019]. This algorithm generates a similarity score for each token in retrieved paragraphs and each token in the gold paragraph of SQuAD. The token similarity is calculated on contextual word embeddings with adversarial paraphrase detection. This enhances the decision-making correlation in better judgement for challenging passages before drawing the final BERT similarity score. The range of BERT-Score is between 0 (Least relevant) to 1 (most relevant).

The final evaluation metric, Precision@K_N is calculated for the entire dataset which is defined as the percentage of questions in the dataset where top_K_N paragraphs retrieved are relevant to the gold paragraph. For example, suppose Precision@K_N is 70%, this is interpreted as 70% of the questions retrieved paragraphs relevant to the gold paragraphs appearing in the top_k_N results.

Mathematically, Precision@K_N is defined by Equation 5.1

$$\text{Precision@K N} = \frac{\text{Total Questions where relevant para is retrieved in top_KN}}{\text{Total no of Questions in dataset}} * 100 \tag{5.1}$$

5.1.2.2 Evaluation Methodology

The following steps describes the evaluation methodology of retriever modules. The methodology remains same for both Single Stage (top_K) and Multi Stage Retriever (top_K_N) models.
Note: In SSR, all the paragraphs containing in top_K articles are extracted and evaluated whereas in MMS, only top_k_N paragraphs are extracted and evaluated.

Step 1: For a given question, BERT similarity score is calculated between the gold paragraphs of SQuAD and each of the predicted paragraphs (top_K for Single Stage Retriever and top_K_N for Multistage Retriever) and is stored in a list.

Step 2: Max score is considered from the list of top_K_N or top_k BERT-Similarity score. At this stage, the only interest is to consider the most relevant passage retrieved (i.e BERT-Score nearly equal to 1).

Step 3: A threshold value is considered to classify weather the retrieved passage is relevant or non-relevant i.e for a given question, BERT similarity score above a given threshold value is considered as relevant to gold context, any value below the threshold is considered as non- relevant.

Step 4: After completing Steps 1-3, repeat the process for all of the questions in the SQuAD dataset. Finally, Threshold@K_N metrics is determined for multistage retriever, whereas Threshold@K is calculated for single stage retriever.

5.1.3 Retriever Results

The Evaluation results are demonstrated in the bellow Figures 5.1, 5.1, 5.2 . As discussed in the previous section, evaluation is performed by varying the combinations of extracted articles and paragraphs. Specifically, by increasing the number of Articles extracted in SSR and decreasing the number of paragraphs extracted by MSR. In the Figure 5.1 bellow, y-axis represents the Precision@K_N for all the questions in SQuAD dataset and x-axis reflects the hyper-parameter threshold (0-1). Higher the threshold value (nearly equal to 1), indicates the percentage of questions for which exact identical SQuAD paragraphs was retrieved.

For consistency the evaluation results are documented in the following order where K- Number of Articles, N- Number of Paragraphs, SSR- Single Stage Retriever and MSR- Multi Stage Retriever.

Number of articles **(K)**: 15 and Number of paragraphs **(N)**: 20

Number of articles **(K)**: 25 and Number of paragraphs **(N)**: 15

Number of articles **(K)**: 35 and Number of paragraphs **(N)**: 10

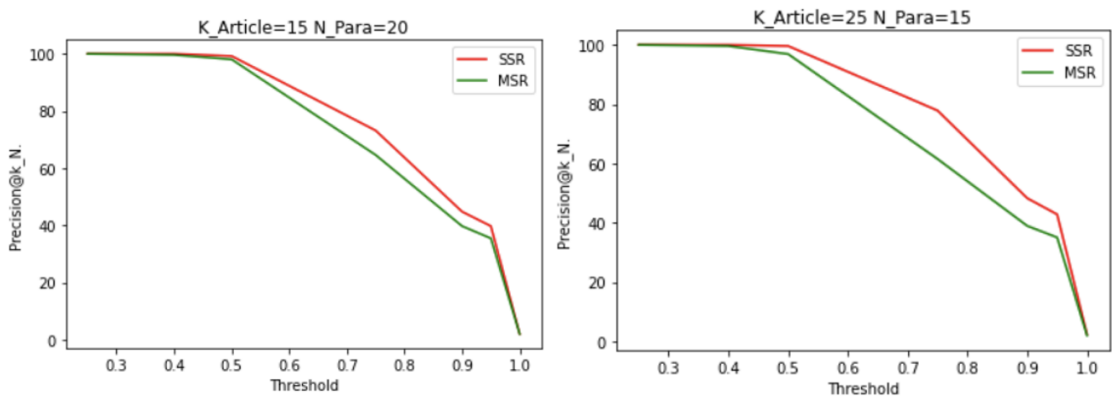

FIGURE 5.1: Retriever output for No of articles = 15 & No of paragraphs = 20 (left) and No of articles = 25 & No of paragraphs = 15 (Right)

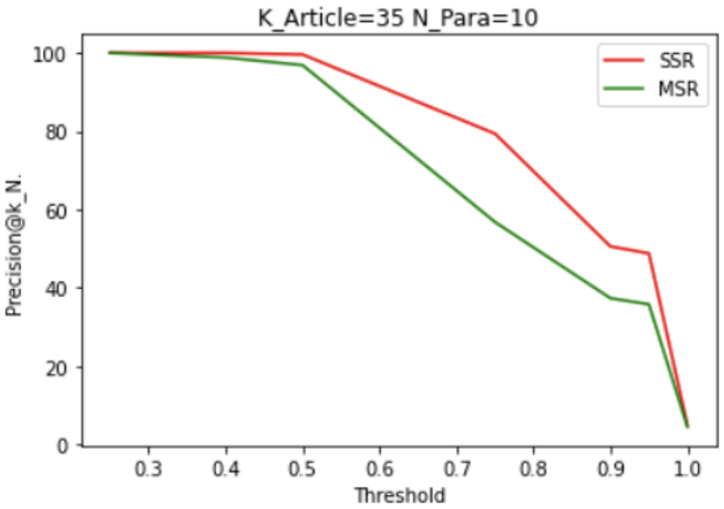

FIGURE 5.2: Retriever output for No of articles = 35 & No of paragraphs = 10.

5.1.3.1 Results interpretation

Form the above results, it can be noticed that:

1. SSR could retrieve 50% of passages that are exactly identical to SQuAD contexts at a higher threshold value (roughly equivalent to 1), whereas MSR could retrieve 40% of identical SQuAD contexts. This drop is the result of error propagation from the SSR to MSR module.

2. For MSR, greater K values yield better results. This is due to the fact that MSR will have more options for extracting the most relevant paragraphs.

3. A sanity check revealed that an information retrieval system with a higher k value and a lower N value delivers faster and more efficient outcome.

5.1.3.2 Drawbacks of Retriever

Although, SQuAD proved to be an ideal dataset for evaluating retriever models in this project as discussed earlier(see Section 5.1.1). Practically, some of the significant disadvantages are:

1. SQuAD passages are extracted from English Wikipedia dump of 2013-01-10 which is almost a decade ago. Since Wikipedia is editable, some of the paragraphs of Wikipedia may have changed/ deleted after SQuAD was constructed. Some of the gold passages were discovered to be missing from Wikipedia after several sanity

checks on the SQuAD train and development passages of the dataset. This reduces Precision@K_N because the returned passages are not identical to the missing gold passage.

2. The indexing methodology followed in this project differs from that of SQuAD. The passage extraction in SQuAD follows Project Nayuki internal page ranking system [Keraron et al., 2020], which is computationally much expensive and time consuming.

BM25 depends on bag of words (BoW). Unlike advanced retriever models like DPR and word2vec, it does not capture semantics and co-occurrence information. This leads to decreased accuracy while extracting the relevant articles for a given question. This error propagates to MSR and is replicated since another BM25 is employed in MSR ranking function to sort paragraphs.

As discussed earlier, given the title name, the paragraphs of Wikipedia are extracted using Wikipedia API. It was discovered that sometimes, Wikipedia API does not retrieve contexts of the page even though page exists. This might lead to missing context in SSR and this error also propagates to MSR while retrieving the final top_K_N paragraphs. One of the solution to this problem is to index every paragraphs of Wikipedia dump, which is computationally very expensive and time consuming (see Section 3.4) for computing resource.

5.2 Reader Evaluation

This section will discuss the empirical evaluations of traditional neural network based BiLSTM (DrQA) and self-attention based BERT-base language models on SQuAD machine comprehension datasets.

5.2.1 Experimental setup

The following methods are considered to examine how effectively a model can predict a span of answer for a given question and associated paragraph. The pretrained DrQA (see Section 4.3.1) and fine-tuned BERT-base (see Section 4.3.3) are evaluated on both SQuAD V1.0 and SQuAD V2.0 for comparability and associated discussion.

The evaluation methodology adopted in this project is the same as [Chen et al., 2017] and is also been used in subsequent publications. This helped in enhancing good comparison of model's behaviour with other state-of-the-art models, as illustrated in the table (see

Section 5.2.2.2). The evaluation is performed on SQuAD evaluation/development set which splits 10 % of the entire SQuAD dataset. The statistic of the dataset is summarised in the table. (see Section 4.2.1.1). The evaluation metrics used to asses the model performance are Exact Match (EM) score and F1 score as discussed in literature. (see Section 2.7.2.1)

5.2.2 Reader Results

The results of DrQA and BERT-base models are individually evaluate on both SQuAD V1.0 & V2.0 training datasets as illustrated in Figures 5.3, 5.4, 5.5, 5.6 bellow. At the end of every epoch, training loss is calculated to determine how well the model is fitting on the training data and validation loss indicates how well the model fits new unseen data. The cross validation is performed using the SQuAD development dataset by calculating EM and F1-Score. In the Figures 5.3, 5.4, 5.5, 5.6, below, the left column indicates the train and validation loss, while the right column represents the metrics score of the best model after each epoch.

For consistency, each set of results will be reported as follows:

1. DrQA model on SQuAD V1.0 dataset.
2. DrQA model on SQuAD V2.0 dataset.
3. BERT-base on with SQuAD V1.0 dataset.
4. BERT-base on with SQuAD V2.0 dataset

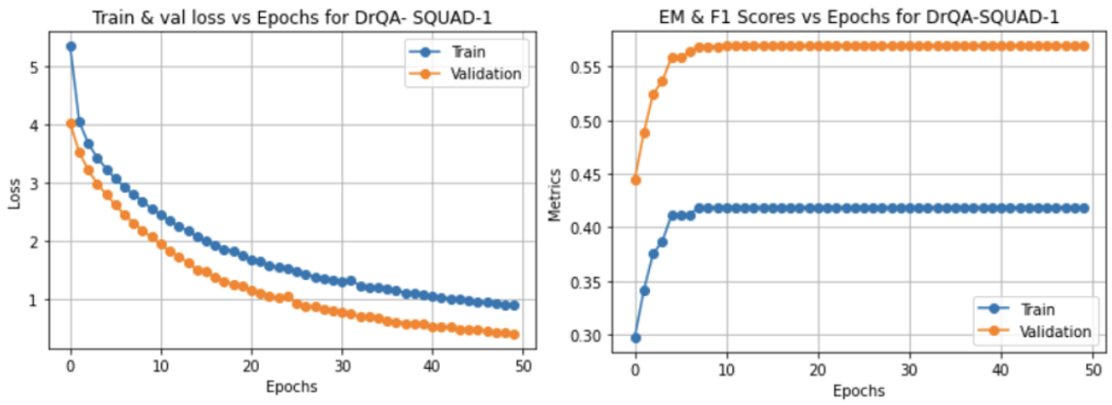

FIGURE 5.3: DrQA model results on SQuAD V1.0 dataset.

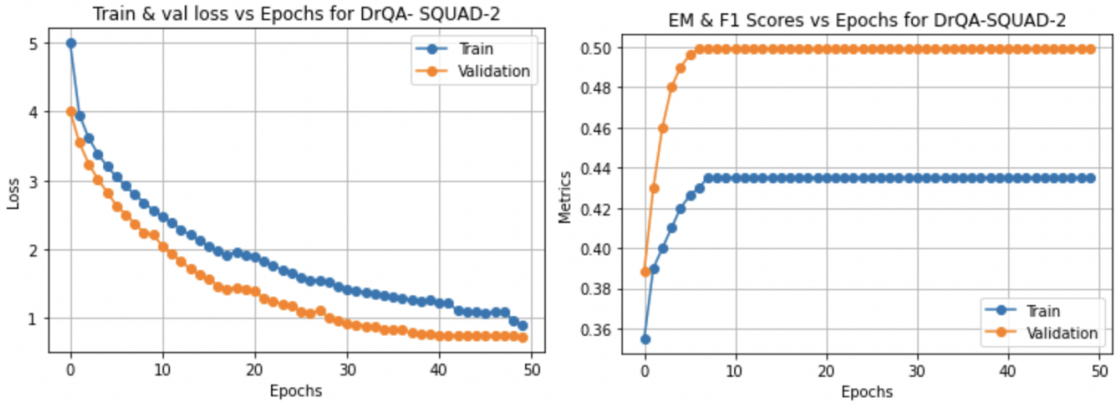

FIGURE 5.4: DrQA model results on SQuAD V2.0 dataset.

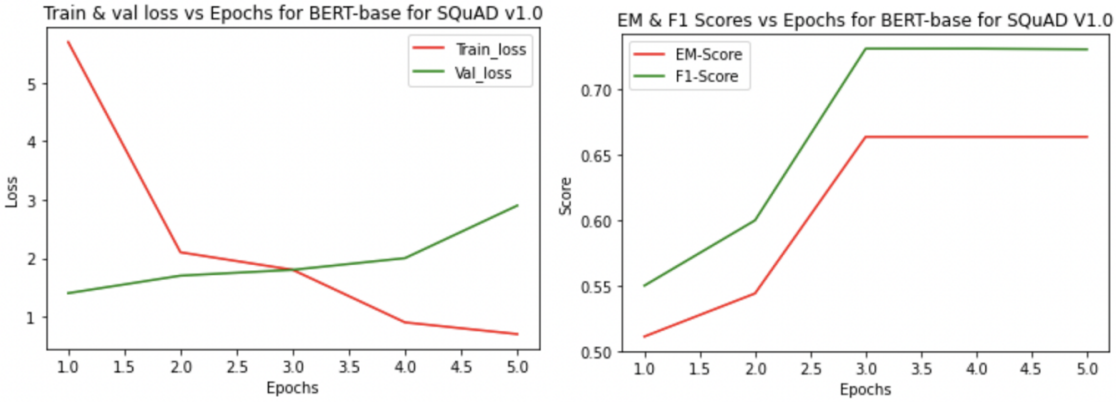

FIGURE 5.5: BERT-base model results on SQuAD V1.0 dataset.

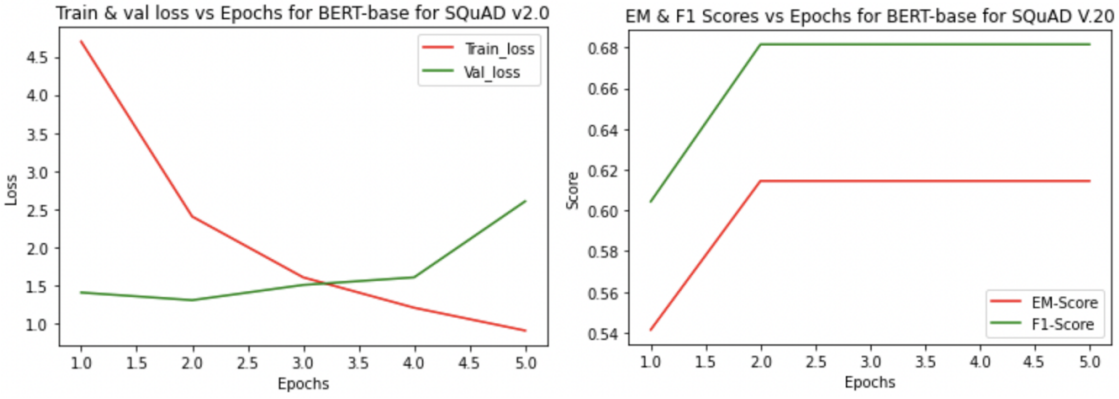

FIGURE 5.6: BERT-base model results on SQuAD V2.0 dataset.

5.2.2.1 Results Interpretation

This section will discuss some key pointers and observations of DrQA and BERT-base model behaviour from the results illustrated above (see Section 5.2.2)

1. **DrQA results analysis**
   The results from DrQA model shows that, the train and validation loss decreases whereas, the Em and F1-Score remains constant after certain epochs. This is because SoftMax function is used in the final classification layer and the output is rounded. For example: if the models predicts a vector (0.7, 0.7, 0.3), after weight optimisation, the prediction might change to (0.996, 0.994, 0.06). In both the cases the algorithm predicts same label but with lower loss.

   The validation loss is continuously smaller than the training loss and the difference between them is around the same magnitude while the training loss fluctuates. During DrQA model training, dropout penalises model variance by randomly freezing neurons in hidden layers. Dropout is only applicable during the training process and has an impact on training loss and not on validation loss. This causes circumstances where validation loss is less than training loss.

2. **BERT-base results analysis:**
   The results show that, BERT-base model starts Over-fitting after 3 epochs for both SQuAD V1.0 and V2.0. The gap between train and validation loss increases drastically after 3 epochs which can be seen in the left column of the Figures 5.5, 5.6 above.

   Validation loss is initially lower than training loss, but gradually reaches equal or higher levels. Although training loss is computed throughout each epoch, validation loss is computed at the end of each epoch with a lower batch size. As a result, each epoch will have multiple backpropagations for a smaller batch size and each step significantly improves the model performance, especially in the early epochs when the weights are still relatively untrained. This causes a reduction in validation loss in the initial few epochs.

5.2.2.2 Comparison of models.

The performance of the transformer-based BERT-base model outperforms the classic Recurrent Neural Network-based DrQA model on both versions of datasets given the same resource because BERT is deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. The fundamental limitation of BiLSTM

is the inability to effectively consider both left and right contexts of the target word, rather is simply concatenates the right-to-left and left-to-right information. This indicates that the representations could not benefit from both left and right contexts at the same time. Transformers on the other hand are based allow parallelization enabling a bidirectional flow of information while training, which is critical when working with big volumes of data. Other advantages of BERT are discussed in literature (see Section 2.3.3.2).

Another consistent behaviour of DrQA and BERT-base model found in Table 5.1 is that the EM and F1-Score for SQuAD V2.0 is lower than SQuAD V1.0, which is expected since the upgraded version is much more complected than the previous version like addition of unanswerable questions. (For challenges in SQuAD V2.0, see Section 5.1.1.1.) The main results are illustrated in the Table 5.1 bellow.

TABLE 5.1: Evaluation score of reader models on both SQuAD V1.0 and V2.0.

2* **Models**	**SQuAD V1.0**	& **SQuAD V2.0**		
	F1-Score	**EM**	**F1-Score**	**EM**
DrQA Original	78.83	69.51	–	–
BiDAF	81.12	73.31	–	–
DrQA - (This project)	**57.5**	**42.3**	**50.12**	**43.54**
BERT-Serini	46.1	38.6	–	–
BERT-base -Original	88.5	81.2	75.73	72.35
BERT-large -Original	91.281	84.328	80.23	83.0
Distil-BERT- Original	85.8	77.7	68.1	64.88
T5- base	85.71	76.95	81.2	77.64
BERT-base (This project)	**73.21**	**66.66**	**68.42**	**64.16**

5.2.2.3 Drawbacks of Reader models

DrQA:

1. Authors in [Chen et al., 2017] have undisclosed hyper-parameter values such as dropout_rate for RNN state and gradient_clipping for achieving the SOTA.

    - Dropout_rate is a powerful regularisation method that temporarily ignores randomly selected neurons during each forward propagation. This prevent over-fitting of recurrent neural network models. A good range, according to Panagiotakopoulos et al. [2021], is between 0.4 and 0.8, where 1 denotes no

dropout and 0 implies no output from the layer. Employing this, the value of 0.5 is chosen for both the RNN and word embedding dropout in this project.

- grad_clipping is another vital component that prevents vanishing gradients in recurrent neural networks. Its the process of reducing or increasing the gradient values (element-by-element) if they go outside the intended range. As reported by Stevens et al. [2022], an ideal range of clipping is between 0 to 10, which implies if a norm of 1.0 is provided, a gradient's vector norm will be re-scaled to 5.0 if it is greater than 5.0. Following this theory the grad_clipping in this project is set to 10.

2. Tokenizers are crucial for model's performance because an effective tokeniser helps a model learn raw text more efficiently and accurately. The tokeniser used in [Chen et al., 2017] is core_nlp_tokenizer, and it is a text tokeniser that could not be installed in this project(for library issues, see Section 6.3.2). As a result, an alternative sentence tokeniser called "gensim" was adopted since it was computationally quicker, supported Glove embedding, and worked well with huge datasets with data streaming. However, it lacks complex processing units (such as pos-tagging) and flexibility as compared to core_nlp_tokenizer since it splits texts into sentences without considering the semantic structure.

3. The new Pytorch version are not compatible for some of the dependencies listed in [Chen et al., 2017] (Ex: pexpect==v.4). DrQA model in this project was built on the latest CUDA equipped Pytorch version.

4. The maximum train and validation batch size permitted in this project was 64 (see Section 6.3.1 for memory issues), whereas the [Chen et al., 2017] reports SOTA on batch size of 128. Therefore the learning rate was also adjusted to maintain the optimal generalisation. (see Section 5.2.2.3)

BERT-base:

The ratio of batch size and learning rate hyper-parameters is crucial for effective generalisation in deep neural networks. Because of the reduced batch size (16) and learning rate while training, SOTA outcomes (EM & F1-Score) were not achieved in BERT-base. The general rule is that the smaller the batch size, smaller the learning rate. According to Feurer and Hutter [2019], if the batch size is divided by k, the learning rate should be divided by the square root of k (sqrtk) at the same time to maintain the gradient expectation constant. Following this theoretical premise, the learning rate was reduced to 1e-5 while the other hyper-parameters remained constant/ default.

Chapter 6

Discussion and conclusion

6.1 Error Analysis

The error analysis is performed on language model using SQuAd V2.0 dataset where questions are randomly selected based on EM and categorised them into 5 classes of error modes as illustrated bellow. Here, the error analysis is performed only on DrQA model trained on SQuAD V2.0 this is because of the complexity of this version of the dataset as discussed in Section 5.1.1.

1. **Imprecise answer boundaries:** This is an error discovered when the model was not precisely extracting the start and end boundaries of the answer, resulting in EM-Score=0 even though the model predicted the right answer. In the following example, It can be seen that the model predicts 'article 1 to 7', but the true answer is '1 to 7'.

   Example:-
   Context: "The key rules on equitable treatment of employees are spelled forth in articles 1 to 7 of the Free Movement of Workers Regulation."
   Question: What's the range of Free Movement of Workers Regulation."
   Predicted: article 1 to 7 **Answer**: 1 to 7

2. **Paraphrase problems:** This error is detected when the model is asked a paraphrased questions. Since the model is not trained to answer the paraphrased queries during linear transformation, it predicts the incorrect answer. In the example below, the model predicted 'three-tired,' when the true answer is 'secondary school/high schools.'

   Example:-
   Context: The three-tiered system of education in Australia is generally used, with

primary education (primary schools), secondary education (secondary schools/high schools), and tertiary education (universities and/or TAFE institutions) being the last two tiers.

Question: What, under the Australian system, is the second system of education?

Predicted: three-tiered **Answer**: secondary schools/high schools

3. **External knowledge:** This error was identified when the model is asked a question that required more knowledge to answer than the context provided. The model predicts an answer whereas it is intended to avoid answering such unanswerable queries. In the following example, the model predicts an inaccurate answer to question where the right answer is not included in the context.

   Example:-

   Context : "On June 4, 2014, the NFL stated that the custom of designating Super Bowl games with Roman numerals, a custom started with Super Bowl V, would be temporarily halted and that the event will be known as Super Bowl 50 rather than Super Bowl L.

   Question: Which Roman numeral would have been used if the 50th Super Bowl's name had been given in that format?

   Predicted: "Super Bowl 50" **Answer**: "L"

4. **Multi-sentence:** The majority of errors (EM=0) were discovered in multi-sentence predictions. In the example below, when the query needs more than one entity to answer (each entity separated by a comma), the model extracts only the first entity, leaving out the rest.

   Example:-

   Context: "In the years that followed, the network was improved to support host to host interactive connections in addition to host to host batch connections (remote job submission, remote printing, batch file transfer), interactive file transfer, gateways to the Tymnet and Telenet public data networks, X.25 host attachments, gateways to X.25 data networks,hosts with Ethernet connections, and eventually TCP/IP and additional public universities in Michigan. This all prepared the ground for Merit's involvement in the NSFNET project beginning in the middle of the 1980s."

   Question: "What made the Merits position in NSFNET possible?"

   Prediction: "hosts with Ethernet connections"

   Answer: "hosts with Ethernet connections, and eventually TCP/IP and additional public universities in Michigan"

6.2 Incomplete Experiments

Some optional aspects of the project was not performed. Although they were potentially good ideas, they added little value to the dissertation and demanded additional time. This also connects to the high expectations described in the requirements analysis section, thus certain optional elements of the project had to be abandoned and placed aside for future development. This section will go through some experiments that might have been performed if given more time.

6.2.1 Model ablations

Model ablations are techniques in which certain components of a fully connected pipeline are eliminated each time in order to gain a better understanding of the components behaviour and its influence. This section will go over some of the model ablation experiments that can be performed in this project.

Model ablations for DrQA:

1. **Effect of paragraph embeddings:** The feature vector of paragraph tokens can be subjected to ablation analysis. This can be carried out without the aligned question embedding by retaining only the word embeddings and few manual features.

2. **Effect of question embeddings**: Similarly feature vectors of question tokens can be analysed by eliminating aligned paragraphs embeddings.

3. **No attention layer:** Attention is a technique integrated into the BiLSTM that allows model to focus on specific elements of the input sequence while predicting a specific part of the output sequence, enabling faster and higher-quality learning. This simple attention layer in DrQA can be evaluated to understand the behaviour of the attention flow of this model by employing the dynamic evaluation approach suggested by Bahdanau et al. [2014], which involves removing the context to question attention flow and substituting the question vector U with the average of the output vectors in the contextual embeddings of question.

Model ablations for BERT-base: It would be interesting to analyse the reason behind the BERT's strength. Some of the BERT ablation studies that can be performed are:

1. **Effect of transformers architecture:** The BERT architecture created a breakthrough in the NLP area in many tasks and enabled the use of transfer learning. This technique allowed unsupervised training with massive unlabeled datasets and altering the model's final layers to fit it to specific tasks. Then, by fine-tuning the parameters, many scientists generated new state-of-the-art results for a variety of datasets just for few epochs. This can be tested by removing the self attention mechanism of BERT-base and using a simple dynamic/ static attention layer suggested by Bahdanau et al. [2014] .

2. **Feature based approach:** The results presented in this project were obtained after finetuning technique in which a simple classification layer (referred to as QA-head) was built on top of a pre-trained BERT-base model. In feature based ablation study, this classification layer can be eliminated without providing contextual word embeddings i.e randomly initializing the weights. By this its possible to discover weather the BERT's gain is either from the pre-training on large corpus (i.e BookCorpus) or from fine-tuning on SQuAD contexts.

3. **Effect of model size:** The effect of the model size can be evaluated by removing some of the hidden layer of the BERT-base model. Since larger BERT models lead to strict accuracy improvements over the dataset.

6.2.2 Data ablations

Data Ablation is the investigation of data change and its consequences on neural network performance. This can be performed by systematically splitting the SQuAD dataset into different ratios and evaluating the models performance.

6.3 Problems and issues

Several complications arose throughout this project, causing project schedule slippage and hindering project progress. Some of the major issues encountered are discussed in this section.

6.3.1 Memory issues

Memory problems were a major hindrance that caused project timeline delays. Google colab Pro+ had a maximum memory of 32 GB (for computing resource, see Section 3.4). This is one of the reasons why the SOTA results could not be replicated, since it was

not possible to load a large batch size of dataset into the model while training. This often terminated the training process in between if the memory consumption exceeds the limit.

The indexing of Wikipedia dump, on the other hand, was an expensive procedure. As a result, due to restricted memory on the local system the indexing method was separated into three components before drawing the final index . During this time, no progress was made using other applications because the Indexing process consumed all the available memory in the system.

6.3.2 Library issues

Several library problems were encountered in this project. The core_nlp_tokeniser, in particular, was incompatible with the most latest versions of Pytorch. The previous Pytorch version doesn't support Google colab Pro and other requirements such as pexpect == V.4. Debugging this problem took a considerable amount of time.

During indexing, the nltk stopwords and stem words libraries were non-functional. An alternative technique was used, in which the most frequently occurring stop words and stem words were downloaded into local files and each page was iterated through the files during pre-processing. As a result, indexing was a time-consuming process, as seen in Table. **??**

6.3.3 Run time limitations

Indexing an 89 GB Wikipedia was a time consuming process. The task took 6 days, and there was no progress on the project throughout that period. Training a large scale language model, on the other hand, is also a time-consuming and expensive process. Google Colab Pro+ was expected to have a run time restriction of 50 hours. However, an idle session is restricted to 5 hours if not interacted frequently with the server. As a result, the run time would be disconnected in between the training process. This was a great barrier while obtaining the results as it required continuous monitoring throughout the training process. Access to vast resources with infinite GUP acceleration would have been preferable. The Table 6.1 summarises the time consumed for each process.

Task	Time taken (roughly)
Indexing **89 Gb** wikipedi dump	6 days
Evaluating: SSR & MSR	7 hours & 10 hours
Training BERT-base	40 hours x 2
Training DrQA	22 hours x 2

TABLE 6.1: Time consumed for each process.

6.3.4 Debugging and understanding DrQA and Pytorch

To begin working on the Pytorch and DrQA, significant amount of effort & time was spent trying to understand and debug the code because of insufficient prior knowledge. Hence, the code in this project is built to be more readable and re-producible compared to original paper [Chen et al., 2017] as it is not wrapped with the chatbot and other unnecessary components.

6.4 Future work

1. **Full system training:** To overcome the problem of missing paragraphs, one solution to consider could be integrating end-to-end training throughout the Document Retriever and Document Reader pipelines rather than employing separate systems (Information retriever and Machine reading comprehension). By this, end-to-end system can be evaluated to asses the efficiency of fully connected pipeline.

2. **Multiple passages training:** Currently, the reader model has been trained on a single passage by encoding a single query and corresponding single paragraph. The answers are predicted from each of the extracted paragraphs. The future work can be incorporated by encoding multiple passages and to predict an optimal span of answer from all those encoded passages.

3. **Conversational Aspects:** To enable the conversational aspect in this project i.e the ability of a system to appropriately comprehend a query in consideration of previous turns, QuReCC [Anantha et al., 2020] dataset can be utilized to train the language models. This dataset is intended to provide challenging benchmark for end-to-end conversational question answering, which includes the separate subtasks of question_rewriting, passage_retrieval, and reading_comprehension.

4. **Testing SQuAD dataset:** As previously discussed in retriever drawbacks (see Section 5.1.3.2), passages in the SQuAD dataset appear to be missing from the latest Wikipedia dump. This may be evaluated by filtering out paragraphs that

are partially or completely missing. This calculates the percentage of Wikipedia passages that exist/remaining in the most recent Wikipedia dump.

5. **Using Dense Passage Retriever (DPR) for information retrieval system**: Recently, DPR for ODQA was introduced by Karpukhin et al. [2020], outperforms TF-IDF & BM25 retrievers as they are machine learning based models that uses dense vectors to encode the semantic meaning as discussed in literature (see Section 2.4.1.2)

6. **Grid search for DrQA:** Grid search technique can be performed to find the optimal values of the undisclosed grad_clipping and dropout_rate hyper-parameters (see Section 5.2.2.3) to help reproduced the SOTA results in [Chen et al., 2017]. This could be accomplished by altering the value of unknown parameter with all possible values while keeping the other hyper-parameter values constant/default while training.

6.5 Conclusion

This project demonstrated the challenge of machine reading comprehension at a large scale by using Wikipedia as the primary knowledge source for an end-to-end Open Domain Question Answering (ODQA) system. This was performed by firstly indexing the whole Wikipedia dump and implementing the Wikipedia information retrieval system and secondly, Machine Reading Comprehension models was re-implemented to extract answers for open domain questions from the retrieved passages. Although it was not feasible to reproduce state-of-the-art results in this project for various reasons described previously (see Section 5.2.2.3), the results illustrates that the model's performance of both DrQA and BERT-base is much better than anticipated, and the project's major objectives was accomplished.

The results in this project show that transformer-based models (BERT-base) outperformed Recurrent neural network based BiLSTM model (DrQA) for the task of open domain question answering, trained on large scale machine comprehension dataset (SQuAD). It can be concluded that for the task of ODQA on large scale datasets, transformer based models (like BERT) are the best choice over Recurrent Neural Network based models (BiLSTM). This is by no means to undermine LSTM models since performance of a model dependents on various factors such as task and datasets. As the literature reviewed that, for certain datasets, LSTM models outperformed transformer based model while taking less time to develop and tweak models [Ezen-Can, 2020]. So, this answers the research question: Before deciding which models to employ, it is critical to first examine the dataset and the task requirements.

Appendix A

Reader Hyper-parameters

A.1 DrQA hyper-parameters

Hyper-parameter	Value	Character
No of Epochs	50	Default and Early stopping trigger.
Learning rate	1e-3	Non-default.
Training batch size	64	Non-default
Validation batch size	64	Non-default
Dropout rate for BiLSTM	0.5	Non-Default
grad-clipping	10	Non-Default
No of layers (context- BiLSTM)	3	Default
No of layers (query- BiLSTM)	3	Default
Max Question token length	40	Default
Max Context token length	512	Default

Table A.1: DrQA training hyper-parameters.

A.2 BERT-base hyper-parameters:

Below are some of the definitions of hyperparameters of BERT for Question Answering extracted from the hugging face documentation are [Sanh et al., 2019]:

1. input ids (torch.LongTensor of shape (batch size, sequence length)): This parameter determines input sequence tokens in vocabulary.

2. attention mask (torch.FloatTensor of shape (batch size, sequence length), optional, defaults to None): This parameter is used to avoid paying attention to padding

token indices. The values are binary with 1 indicating not masked and 0 indicating masked.

3. head mask (torch.FloatTensor of shape (num heads,) or (num layers, num heads), optional, defaults to None): Selective heads of the self attention modules can be nullified using a this parameter. The values are binary with 1 indicating head not masked and 0 indicating head is masked.

4. inputs embeds (torch.FloatTensor of shape (batch size, sequence length, hidden size), optional, defaults to None): Embedded representations can be directly fed as an alternative to input ids. This is a good option to get greater control over how input ids indices are converted into associated vectors than the model's internal embedding lookup matrix provides.

5. output attentions (bool, optional, defaults to None): When this parameter is set true, the tensors of all attention layers' attentions are returned.

6. start positions (torch.LongTensor of shape (batch size,), optional, defaults to None): For determining the token classification loss, the labels for position (index) indicates the start of the labelled span. The length of the sequence (sequence length) is used to constrain positions, any position outside of the sequence are ignored while calculating the loss.

7. end positions (torch.LongTensor of shape (batch size,), optional, defaults to None): For determining the token classification loss, the labels for position (index) indicates the end of the labelled span. The length of the sequence (sequence length) is used to constrain positions, any position outside of the sequence are ignored while calculating the loss.

A.2.1 BERT-base hyper-parameter values

Hyper-parameter	**Value**	**Character**
No of Epochs	5	Non-default.
Learning rate	1e-5	Non-default.
Training batch size	16	Non-default
Validation batch size	1	Non-default
Dropout rate for BiLSTM	0.1	Default
Max Question token length	15	Default
Max Context token length	512	Default
adam_beta 1, 2, epsilon	0.9, 0.999, 1e-08	Default

Table A.2: BERT-base training hyper-parameter values.

Appendix B

Interactive Session

B.1 Fully connected pipeline interactive session

Figures below depict a sample interactive session of an End-to-end fully connected Open Domain Question Answering System. The default hyper-parameter is Number of paragraphs (N = 10) and Number of Article titles (K=10). The best answer is found with the label "Top Predicted Answer". The Wikipedia search engine takes about 50 seconds to load (depending on RAM), and each query takes about 10 seconds to generate an answer.

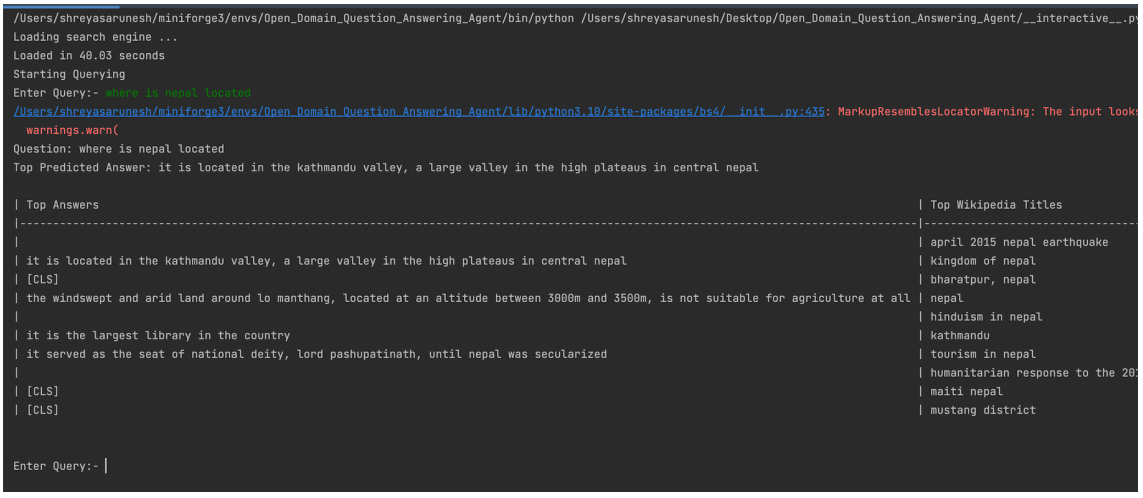

FIGURE B.1: First interactive sample of ODQA for N=10 K=10.

```
Question: where is mysore located
Top Predicted Answer: industries in mysore district are mainly concentrated around the cities of mysore and nanjangud

| Top Answers                                                
```

Appendix C

GitHub/ MACS-GitLab Project repository

C.1 Project code and repository.

Self written code follows the British Computer Society codes of conduct with citations properly referenced. Also the entire project is made readable with sufficient comments for every block of code that summarises the purpose and logic implemented in that class and function.

This project is made open source on GitHub and MACS GitLab platform given access to Dr. Arash Eshghi and Dr. Nick Taylor. Readers are encouraged to refer the README.md file in order to better understand the project structure and follow the instructions to reproduce the results achieved.

Link to GitHub repository:

Open Domain Question Answering Agent (ODQA) - GitHub Repository

Link to GitLab repository:

Open Domain Question Answering Agent (ODQA) - GitLab Repository

The models, DrQA and BERT-based, trained on SQuAD V1.0 and SQuAD V2.0, as well as the Wikipedia Index files, may be downloaded from Google Drive using the link below.

Link to trained models and Wikipedia Index files:

Language models and Wikipedia Index- Google-Drive

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Adel, H. and Shi, S. (2021). Proceedings of the 2021 conference on empirical methods in natural language processing: System demonstrations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Anantha, R., Vakulenko, S., Tu, Z., Longpre, S., Pulman, S., and Chappidi, S. (2020). Open-domain question answering goes conversational via question rewriting. *arXiv preprint arXiv:2010.04898*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, X. (2018). Text classification based on lstm and attention. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pages 29–32. IEEE.

Bdour, W. N. and Gharaibeh, N. K. (2013). Development of yes/no arabic question answering system. *arXiv preprint arXiv:1302.5675*.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Chollet, F. (2016). Building autoencoders in keras. *The Keras Blog*, 14.

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

EnWiki (2022). English$_w$*ikipedia*.

Ezen-Can, A. (2020). A comparison of lstm and bert for small corpus. *arXiv preprint arXiv:2009.05451*.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2013). Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618.

Fang, H. and Zhai, C. (2009). Evaluation of the default similarity function in lucene.

Feurer, M. and Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham.

Floridi, L. and Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.

Geva, M., Schuster, R., Berant, J., and Levy, O. (2020). Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Green Jr, B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224.

Gupta, P. and Gupta, V. (2012). A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4).

Harabagiu, S. M., Maiorano, S. J., and Paşca, M. A. (2003). Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Keraron, R., Lancrenon, G., Bras, M., Allary, F., Moyse, G., Scialom, T., Soriano-Morales, E.-P., and Staiano, J. (2020). Project piaf: Building a native french question-answering dataset. *arXiv preprint arXiv:2007.00968*.

Kim, M. Y. (1996). Qa builder: a visual toolkit for building multimedia knowledge-based systems with immediate feedback. In *Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems*, pages 269–273. IEEE.

Kupiec, J. (1993). Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190.

Lewis, D. D. and Jones, K. S. (1996). Natural language processing for information retrieval. *Communications of the ACM*, 39(1):92–101.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Mager, M., Astudillo, R. F., Naseem, T., Sultan, M. A., Lee, Y.-S., Florian, R., and Roukos, S. (2020). Gpt-too: A language-model-first approach for amr-to-text generation. *arXiv preprint arXiv:2005.09123*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mo'minov, B. (2017). Algorithms and methods for indexing and compression in intelligent data search. , (1):92–105.

Pan, Y., Zhang, Y., and Chiu, K. (2008). Hybrid parallelism for xml sax parsing. In *2008 IEEE International Conference on Web Services*, pages 505–512. IEEE.

Panagiotakopoulos, T., Kotsiantis, S., Kostopoulos, G., Iatrellis, O., and Kameas, A. (2021). Early dropout prediction in moocs through supervised learning and hyperparameter optimization. *Electronics*, 10(14):1701.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Reddy, S., Chen, D., and Manning, C. D. (2019). Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Saini, A. and Yadav, P. (2017). A survey on question–answering system. *International Journal of Engineering and Computer Science*, 6(3).

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Shi, H., Gao, S., Tian, Y., Chen, X., and Zhao, J. (2021). Learning bounded context-free-grammar via lstm and the transformer: Difference and explanations. *arXiv preprint arXiv:2112.09174*.

Soutner, D. and Müller, L. (2013). Application of lstm neural networks in language modelling. In *International Conference on Text, Speech and Dialogue*, pages 105–112. Springer.

Stevens, T., Ngong, I. C., Darais, D., Hirsch, C., Slater, D., and Near, J. P. (2022). Backpropagation clipping for deep learning with differential privacy. *arXiv preprint arXiv:2202.05089*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Svore, K. M. and Burges, C. J. (2009). A machine learning approach for improved bm25 retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1811–1814.

Vargas-Vera, M. and Lytras, M. D. (2010). Aqua: A closed-domain question answering system. *Information systems management*, 27(3):217–225.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Visa, S., Ramsay, B., Ralescu, A. L., and Van Der Knaap, E. (2011). Confusion matrix-based feature selection. *MAICS*, 710:120–127.

Voorhees, E. M. et al. (1999). The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. *arXiv preprint cmp-lg/9704004*.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018). R 3: Reinforced ranker-reader for open-domain question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, J., Zhang, Z., and Zhao, H. (2020). Multi-span style extraction for generative reading comprehension. *arXiv preprint arXiv:2009.07382*.

Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., and Lin, J. (2019). End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Yu, L., Hermann, K. M., Blunsom, P., and Pulman, S. (2014). Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Zhang, J., Mishra, A., Patwardhan, S., Agarwal, S., et al. (2022). Can open domain question answering systems answer visual knowledge questions? *arXiv preprint arXiv:2202.04306*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., and Chua, T.-S. (2021). Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.