

# QA4R: A QUESTION ANSWERING SYSTEM FOR R PACKAGES

by

Ganesh Babu

July, 2022

Director of Thesis: Dr. Venkat Gudivada, PhD

Major Department: Computer Science

There is a massive amount of data from various sources available today, and querying meaningful information from those datasets would be valuable. Question Answering Systems (QAS) implement information retrieval (IR) and Natural Language Processing (NLP) that can automatically answer the questions posed in a natural language. There are three different types of QAS as Open Domain, Closed Domain, and Restricted Domain. Following are the various types of questions: fact-based, definition, how, why, hypothetical, semantically constrained, and cross-lingual. R is a dynamic programming language widely used for statistical computing that combines functional and object-oriented programming. The R development community maintains thousands of R packages through its Comprehensive R Archive Network CRAN. However, while websites like [rdrr.io](http://rdrr.io), [rseek.org](http://rseek.org), and [search.r-project.org](http://search.r-project.org) provide search results for R packages, no intelligent question-answering system is currently available for R.

This study examines Question Answering Systems (QAS), current developments and academic research areas in the QAS field, and QAS implementations. In this research, we propose a prototype question answering system for R packages that returns R packages relevant to the user query in natural language. We created a question-answering dataset (QAD4R) for R packages using web scraping and developed a question generation model. Pre-trained BERT-based language models were used to create the question-answering system for R. All the code files are available publicly

at this GitHub location <https://github.com/GanB/QA4R-A-Question-Answering-System-for-R-Packages>.



QA4R: A QUESTION ANSWERING SYSTEM FOR R PACKAGES

A Thesis

Presented to The Faculty of the Department of Computer Science  
East Carolina University

In Partial Fulfillment of the Requirements for the Degree  
Master of Science in Computer Science

by

Ganesh Babu

July, 2022

Copyright Ganesh Babu, 2022

QA4R: A QUESTION ANSWERING SYSTEM FOR R PACKAGES

by

Ganesh Babu

APPROVED BY:

DIRECTOR OF THESIS:

---

Dr. Venkat Gudivada, PhD

COMMITTEE MEMBER:

---

Dr. Nic Herndon, PhD

COMMITTEE MEMBER:

---

Dr. Rui Wu, PhD

CHAIR OF THE DEPARTMENT  
OF COMPUTER SCIENCE:

---

Dr. Venkat Gudivada, PhD

INTERIM DEAN OF THE  
GRADUATE SCHOOL:

---

Dr. Kathleen T Cox, PhD

## Table of Contents

LIST OF TABLES . . . . .	vi
LIST OF FIGURES . . . . .	vii
1 INTRODUCTION . . . . .	1
1.1 Motivation . . . . .	1
1.2 Contribution . . . . .	5
1.3 Thesis Outline . . . . .	6
2 QUESTION ANSWERING SYSTEMS . . . . .	7
2.1 Types of Question Answering Systems . . . . .	7
2.1.1 Closed Domain . . . . .	7
2.1.2 Open Domain . . . . .	8
2.1.3 Restricted Domain . . . . .	8
2.2 Types of Questions . . . . .	9
2.3 Related Work: Survey and Trends in QAS . . . . .	9
3 QUESTION ANSWERING SYSTEMS ARCHITECTURE . . . . .	14
3.1 Architecture . . . . .	14
3.1.1 Information Extraction and NLP based Architectures . . . . .	14

3.1.2	BERT (Bidirectional Encoder Representations from Transformers) . . . . .	17
3.2	QAS Datasets . . . . .	20
3.2.1	Stanford Question Answering Dataset (SQuAD) . . . . .	20
3.2.2	Situations With Adversarial Generations (SWAG) . . . . .	20
3.2.3	Microsoft MACHine Reading Comprehension (MS MARCO) . . . . .	21
3.2.4	Google’s Natural Questions . . . . .	21
3.2.5	DeepMind . . . . .	21
3.2.6	Other Major Datasets . . . . .	22
3.2.7	Custom Domain-Specific Datasets . . . . .	22
4	QUESTION ANSWERING SYSTEM FOR R . . . . .	24
4.1	QA4R Prototype: Design and Implementation . . . . .	24
4.2	Data Acquisition - Corpus . . . . .	25
4.3	QAD4R: Question Answering Dataset for R . . . . .	26
4.4	Question Answering Model . . . . .	28
5	RESULTS EVALUATION . . . . .	30
6	FUTURE WORK AND CONCLUSIONS . . . . .	35
	BIBLIOGRAPHY . . . . .	36



## LIST OF TABLES

2.1	Question types . . . . .	10
3.1	Question Answering System Datasets. . . . .	23

## LIST OF FIGURES

1.1	An Overview of a Question Answering System. . . . .	2
1.2	2017 Data Science Survey Results from Rexer Analytics. . . . .	4
3.1	Chen et al. DrQA - Architecture . . . . .	15
3.2	Srihari et al. - Textract IE System Architecture . . . . .	16
3.3	A brief history of natural language processing. . . . .	17
3.4	An overview of BERT-based Question Answering System. . . . .	19
4.1	QA4R Prototype: System Design . . . . .	24
4.2	R packages corpus extracted using web scraping. . . . .	25
4.3	Structure of question answering dataset for R . . . . .	27
4.4	QAD4R: question answering dataset for R . . . . .	27
5.1	Annotated training dataset . . . . .	30
5.2	Evaluation results for the BERT Model . . . . .	31
5.3	Evaluation results for the DistilBERT Model . . . . .	32
5.4	Evaluation results for the last 10 runs . . . . .	33
5.5	Answer prediction by the model based on the context and question . . . . .	34

## Chapter 1

### Introduction

#### 1.1 Motivation

Question Answering Systems focus on automatically extracting a brief, direct answer to questions posed in a natural language. Information retrieval (IR) and Natural language processing (NLP) are critical components of a QAS. Recent advancements in NLP have evolved QAS from basic text pattern matching to more advanced computational models based on statistics, machine learning, and deep learning. BASEBALL and LUNAR are two of the earliest question-answering systems. BASEBALL was built to answer questions about the US baseball league for one year. LUNAR was built to answer questions related to the geological analysis of lunar rocks based on data collected from the Apollo moon mission. These earlier systems concentrated on closed domains where every query must be about the specific domain, and the answer text must be from a restricted vocabulary hand-written by experts. *Figure 1.1* shows a high-level overview of a question-answering system.

Search engines are a good example of open domain question answering systems. In December 2019, Google search rolled out *BERT* based search algorithm to better understand the search queries in natural language<sup>1</sup>. *BERT* stands for Bidirectional Encoder Representations from Transformers, is an open-sourced neural network-based

---

<sup>1</sup><https://www.searchenginejournal.com/google-bert-rolls-out-worldwide/339359/>

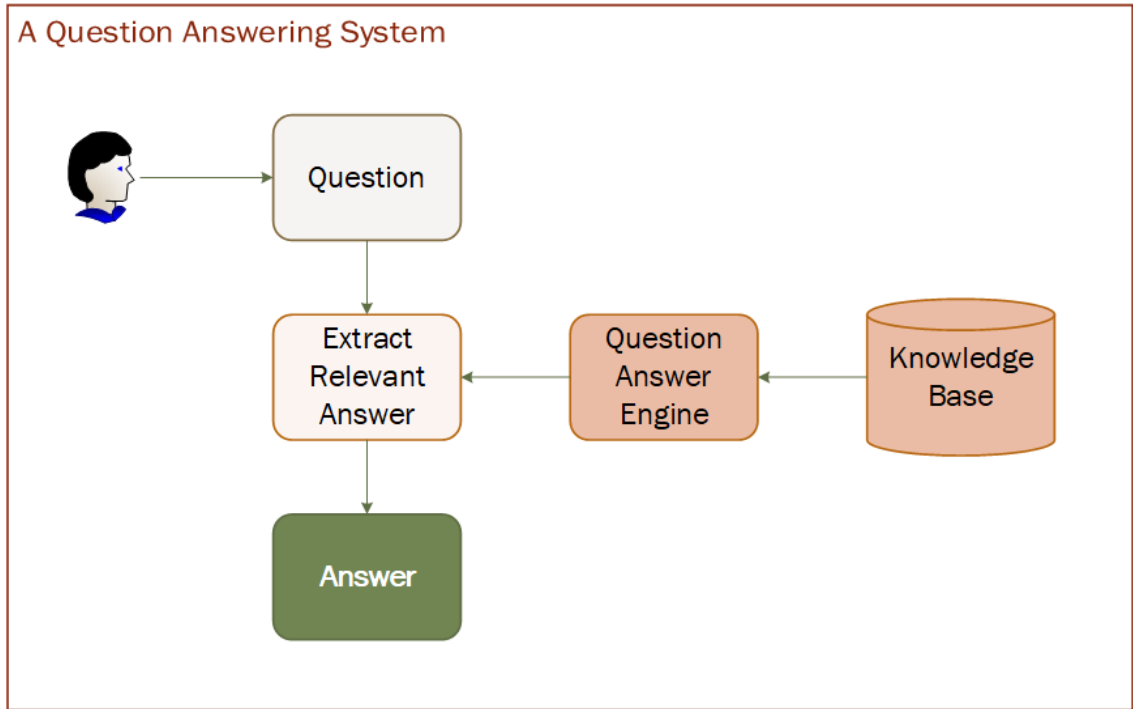


Figure 1.1: An Overview of a Question Answering System.

NLP pre-training. One of the biggest challenges in NLP is the availability of the training data and the computational capacity. With recent developments in cloud computing, for example, Microsoft Azure cognitive service QnA Maker<sup>2</sup>, question-answering systems are offered as a service.

R is an open-source(GPL) programming language widely used among statisticians and data miners for developing statistical software for data analysis. R is a sophisticated computer programming language and environment for complex statistical computing and graphics. R is cross-platform and runs on Windows, Macintosh, and Linux. The capabilities of R are extended through user-created packages, which are a collection of R functions, compiled code, and sample data. The Comprehensive R Archive Network (CRAN) is R's central software repository, currently features

---

<sup>2</sup><https://www.qnamaker.ai/>

18,350 available packages, and growing every day <sup>3</sup>. R is distributed with fourteen base packages: base, compiler, datasets, grDevices, graphics, grid, methods, parallel, splines, stats, stats4, tcltk, tools, and utils. In addition, there are fifteen recommended packages from CRAN, which are included with binary distributions of R: KernSmooth, MASS, Matrix, boot, class, cluster, codetools, foreign, lattice, mgcv, nlme, nnet, rpart, spatial, and survival. *Figure 1.2* shows the increased usage of R programming language along with other languages and tools based on a 2017 data science survey results from *Rexer Analytics*<sup>4</sup>, we can see that 65% to 73% of people in the Data Scientist group (Corporate, Consultants, Academics, and NGO / Government) report using R. Additional languages (e.g., Python, SQL, and Java), along with visualization tools (e.g., Tableau), and platforms (e.g., Hadoop / Hive / Pig) were included in the survey. R is also ranked highly in the TIBOE Popularity index and the PYPL (PopularitY of Programming Language) index<sup>5</sup>.

One of the biggest challenges with the R programming language is identifying the right package for the task. Even though CRAN contains a massive population of R packages, no intelligent question-answering system exists for R other than open domain search engines listing out packages based on the search key. This research aims to develop a prototype for a closed domain question answering system focused on R packages.

---

<sup>3</sup><https://cran.r-project.org/web/packages/>

<sup>4</sup><https://www.rexeranalytics.com/>

<sup>5</sup><https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7063554/>

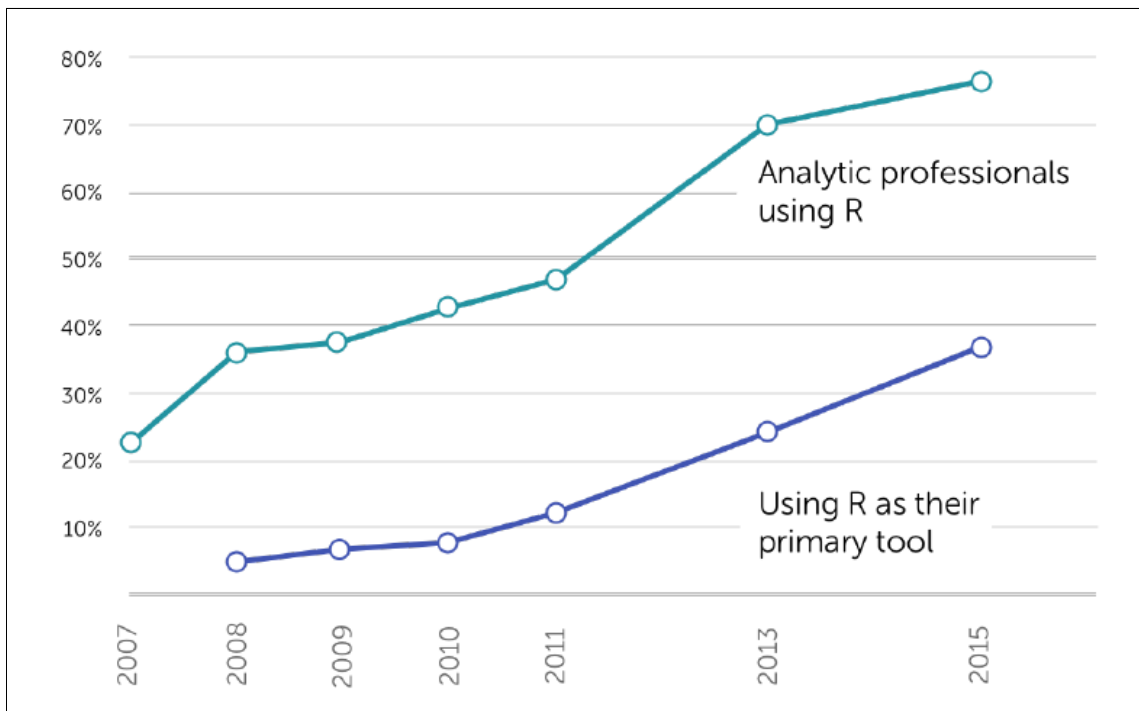


Figure 1.2: 2017 Data Science Survey Results from Rexer Analytics.

## 1.2 Contribution

This research focuses on the following areas:

- Introduction to question answering systems.
- Latest research trends and implementations of question answering systems.
- Design and implementation of a prototype question answering system for R packages.

We start with an introduction to question-answering systems and cover various types of questions and question-answering systems based on domain. We then focus on the architecture and current research areas in the QAS, along with the publicly available datasets. We then discuss designing and implementing a QAS and the technology needed with this prerequisite understanding. This research focuses on creating a question-answering dataset for R packages using web scraping the CRAN website and generating questions from the corpus. We used an open-source library developed on pre-built transformers for question generation<sup>6</sup> using the seq-2-seq model. We developed a python code to create the question-answering dataset for R in a format that the pre-trained BERT-based language models will use to train and test the question-answering system. We used the Simple Transformers<sup>7</sup> library to train and evaluate the Question Answering model, an open-source Natural Language Processing (NLP) library built on the Hugging Face Transformers library.

---

<sup>6</sup>[https://github.com/patil-suraj/question\\_generation](https://github.com/patil-suraj/question_generation)

<sup>7</sup><https://github.com/ThilinaRajapakse/simpletransformers>

### 1.3 Thesis Outline

This thesis is organized as follows: In chapter 2 we are starting with the background information on question answering systems like what a QAS is, different types of QAS, recent research trends, and developments in QAS. We then analyze different architecture and technology used to develop a QAS in chapter 3. In chapter 4 we build upon the previous knowledge about QAS and the problem domain for R packages and develop a closed-domain question answering system prototype for R packages. We also review all the technology components used in this prototype in detail. In chapter 5 we evaluate the results, and finally, we discuss the future work and developments in chapter 6.



## **Chapter 2**

### **Question Answering Systems**

A question-answering system can be compared to a reading comprehension, where the answer is extracted from a large piece of text(data). In the case of computers, this is achieved by using information retrieval and natural language processing. We are familiar with web search engines, which return a list of relevant references to a search query based on the search index ranking. Many more complexities are involved in a search engine, like identifying the keywords, contents, freshness of the webpage, and user engagement. The goal of a question-answering system is to extract a brief exact answer.

#### **2.1 Types of Question Answering Systems**

Question answering systems can be broadly classified based on the closed domain, open domain, and restricted domain.

##### **2.1.1 Closed Domain**

Closed-domain questions belong to a specific domain area. For example, finance, health care, medicine, automobile, mathematics, Programming, legal[15], etc. These systems exploit domain-specific knowledge by using a model trained on a domain-specific dataset maintained in a core database. BASEBALL, LUNAR, Wolfram AI-

pha, and Stack Overflow are good examples of close domain QA systems.

BASEBALL [10] was one of the oldest closed domain QAS that answered questions about the baseball game. The question was keyed in punch cards, a dictionary was used as the corpus to look up the definition, and the final answer was generated based on the syntactic analysis.

### **2.1.2 Open Domain**

Open-domain questions can be from any domain, such as health care, finance, technology, sports, etc. These systems are designed to answer questions from any domain. They are independent and can be based on a large collection of datasets. Examples are internet search engines, Wikipedia, IBM Watson, and intelligent virtual assistants like Amazon Alexa, Google Home, Microsoft Cortana, and Apple Siri.

### **2.1.3 Restricted Domain**

Restricted-domain questions are an amalgamation of the closed domain (domain-specific) and open domain (deep reasoning and reliable accuracy) systems. Restricted domain QAS depends on the domain information that can improve the system's accuracy. Restricted domain QAS can be developed as a complete end-to-end application that meets current needs in specialized domains like medicine, biology, construction, law, etc.

## 2.2 Types of Questions

Questions can be broadly classified into the following categories:

- *Non-factoid question* : Non-factoid questions are open-ended questions that require complex answers, like descriptions, opinions, or explanations, which are mostly passage-level texts. These questions usually require multiple sentences as answers, and these answers come from a particular paragraph in a document. Thus, the context of a sentence plays an important role in retrieving the relevant answer. Here are some examples of non-factoid questions.
  - *Definition Type Questions*: Questions that has following pattern - ("What" + aux + [Noun]). for example,
    - \* what is the process of photosynthesis?
    - \* what is a computer?
  - *Descriptive Type Questions*: Questions that begin with - Why, How, and What and require the answers to be in a few sentences or a paragraph to get thorough information about a topic. for example,
    - \* how can I reset my iCloud email password?
    - \* why is the sky blue?
- *Factoid Type Questions*: A factoid question provides concise facts that look for the precise answer in one or two words. These questions usually start with who, what, when, or where. For example, a reading comprehension passage contains information related to a specific question. Here are some examples of factoid questions.
  - who is the third president of the united states?
  - when was the declaration of independence signed?
  - when did ww2 end?

Table 2.1 shows different question types.

## 2.3 Related Work: Survey and Trends in QAS

Bouziane et al. [3] studied survey of various QAS based on the type of data sources: structured databases, unstructured free text, and pre-compiled semantic knowledge

<b>Definition</b>	<b>Descriptive</b>	<b>Factoid</b>
What	Why	Who
	How	When
	What	Where
		What
		Which

Table 2.1: Question types

bases. In this study, the authors summarize the features and techniques used by QAS for Latin languages, ontology-based QAS, and text-based QAS. They also compare the performance results of various QAS. Results indicate that the success rate of ontology-based QAS varies between 49% and 89%. The authors discuss a project implementing QAS for the natural Arabic language in this study. This QAS converts the request into SPARQL and extracts the answer from an Arabic RDF-linked data source.

A study by Calijorne et al. [4] provides a systematic literature review of various QAS, metrics, and performances based on precision and recall. Per this study, researchers have focused more efforts on natural language processing, knowledge base, information retrieval paradigms, and open domain QAS. In this study, the authors used precision, recall, accuracy, and F1 score to evaluate the performance of various methods.

Mishra et al. [16] surveyed various QAS to provide the current status and future scope of the research. In this study, the authors provide a detailed classification of various QAS from previous studies and research work. In this study, authors have classified QAS based on various criteria like - types of questions, types of data sources, types of processing, types of a retrieval model, and so on. This study shows that the

performance of a QAS is highly dependent on the quality of the corpus and well-formalized user requirements. This paper also indicates the major challenges faced in developing QAS, like understanding the natural language questions and knowledge inferred from various data sources.

Biswas et al. [2] propose a model that uses NLP tools to extract the exact and precise answer for the given question from a large dataset. This framework contains four modules: Question Processing Module, Document Processing Module, Paragraph extraction module, and Answer extraction module. This paper classifies questions into three different types: Definition, Descriptive, and Factoid type of questions, and describes the question pattern for identifying the types and the related algorithms. The authors used agriculture as the restricted domain in this study and created their dataset. This model uses keywords and a headword-based approach to extract the answer. Definition type questions returned exact answers 92% of the time. The performance of descriptive-type questions was the same as a search engine result. The performance of factoid-based questions was not good and can be improved using better ontologies and domain-specific dictionaries.

Gupta et al. [11] provide a broad overview of different QAS and various methods and techniques used in current QAS. In this study, the author discusses various research papers involving different types of QAS and their issues. This study proposes a general architecture of QAS into three main steps: process question - get critical information like identifying the type of the question, extract keywords and focus; process document: documents relevant to the question are retrieved and searched; process answer: apply algorithms for extracting relevant answers from the dataset.

Mollaj et al. [17] provide a historical perspective on QAS and an overview of the current methods and applications used in restricted domain QAS. A restricted domain system should meet the following characteristics:

- it should be circumscribed
- it should be complex
- it should be practical

In this study, authors[17] propose the following design considerations while developing a QAS:

- domain query system analysis: different ways how users can ask for information
- domain knowledge selection: selecting the relevant domain knowledge resources
- domain knowledge acquisition and representation: model selection for domain knowledge to encode and represent the domain knowledge base
- system interface design: Natural Language interfaces to communicate between the users and the system
- technological requirements selection: decisions on the specific technology and methods will be taken based on the domain, type of questions and model encoding used. Restricted domain QAS answers complex questions compared to open domain QAS using ontological knowledge and complex inferences.

Several researchers have studied the evolution of R packages.

Decan et al. [8] studied how R packages are developed and distributed on various repositories (such as *BioC*, *R-Forge*, *GitHub*) and the evolution of R package ecosystem.

Claes et al. [7] proposed a web-based dashboard to understand the package dependencies and provide visual tools for the package developers. Strzalkowski et al. [23] studied how document content can be represented as a collection of terms like words, phrases, and names which could then be weighted to indicate their importance

within the document. We will use this approach in our *Natural Language Processing* algorithm. We will also perform text analysis using a machine learning model and tokenizing the package details. Lunn et al. [14] studied how *Web-Scraping* and *Natural Language Processing* can be utilized to extract information. Wang et al. [27] describes how a *Question Answering System* can be developed using *Natural Language Processing*. However, the results were not entirely successful; we will improvise the approach in this research.

## Chapter 3

### Question Answering Systems Architecture

#### 3.1 Architecture

Open-domain question answering systems typically follow the approach of retrieving the context from an external knowledge base for the question posed in natural language and extracting the answer from the selected context based on ranking. Chen et al. [5] proposed a document retriever question answering (*DrQA*) system using this approach. The authors used a component-based search using *bigram* hashing and *TF-IDF* matching with multi-layer RNN (recurrent neural network) in this framework. In this study, the authors compared the performance of various datasets like SQuAD, CuratedTREC, WebQuestions, and WikiMovies with the DrQA system. They observed that DrQA performed 70.0% exact matches and 79.0% F1 scores on the test set on the SQuAD dataset.

##### 3.1.1 Information Extraction and NLP based Architectures

Textract is a QAS developed by Srihari et al. [22] using information extraction (IE) and natural language (NL). In this study, the authors compare various web-based QAS and propose a sophisticated QAS based on IE. *Figure 3.2* shows the system architecture of the Textract QAS prototype based on named entity (NE) tagging. This architecture returns answers based on correlated entities and open-ended general



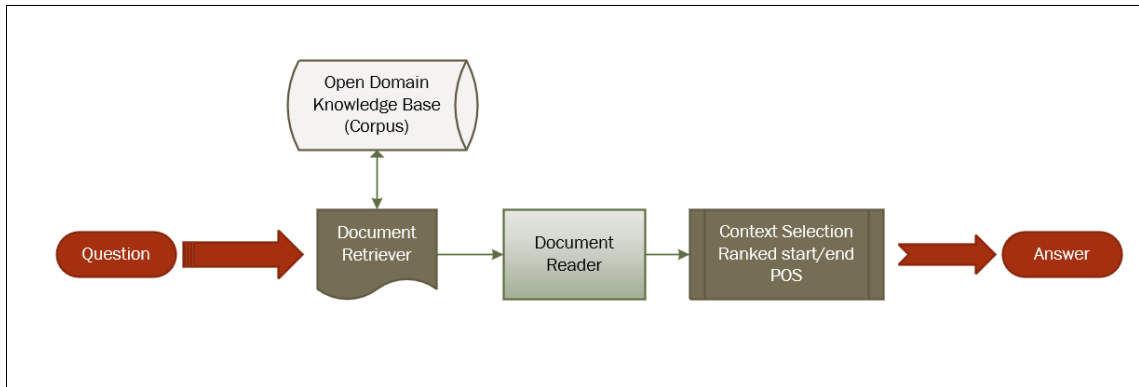


Figure 3.1: Chen et al. DrQA - Architecture

events. In this study, the authors propose multi-level IE along with relationships between the events and CE from the database as future work.

The following is the general algorithm used by Textract for QAS:

- Process Question
  - Shallow parse question
  - Determine asking point
  - Question expansion (using words lists)
- Process Documents
  - Tokenization, POS tagging, NE indexing
  - Shallow parsing
- Text Matcher
  - Intersect search engine results with NE
  - Rank answers

Walter et al. [26] propose a layered pipeline-based architecture for processing natural language questions called BELA, which can take a natural language question as input and produce a SPARQL query and the corresponding answer as output. In this study, the authors benchmarked the results of BELA based on the Question Answering over Linked Data (QALD-2) dataset using precision, recall, and F-measure

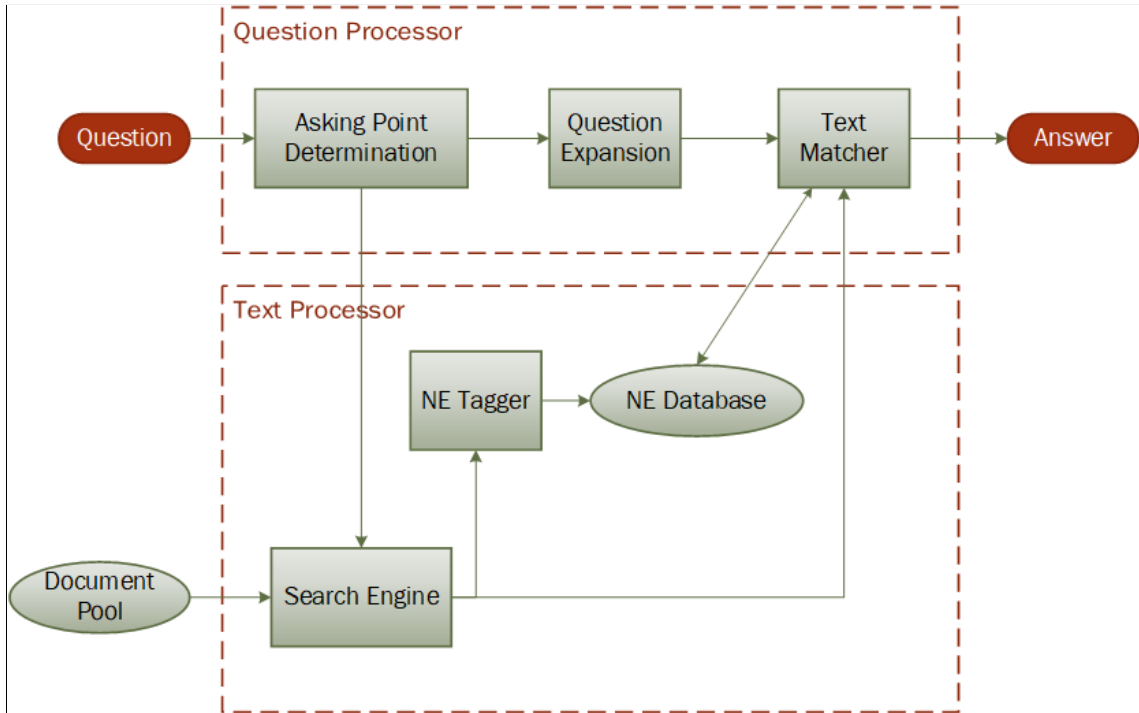


Figure 3.2: Srihari et al. - Textract IE System Architecture

and compared the performance against the various state-of-the-art systems like WolframAlpha. Walter et al. propose that the performance of this architecture can be improved by adding additional lexical knowledge to the system to bridge the gap between the semantic structure of the natural language question and the structure of the dataset.

- Parsing and template generation - Lexical Tree Adjoining Grammars (LTAG) based parser
- Inverted index lookup - data was extracted from DBpedia
- String similarity computation
- Lexical expansion
- Semantic similarity computation - using Explicit Semantic Analysis (ESA)

Sucunuta et al. [24] propose a three-layered architecture for developing a question-answering system. The first layer analyzes the question entered in natural language

and represents a model. Relevant documents are selected in the second layer, content analysis is performed in the last layer, and the exact answer is extracted.

Pre-trained language models can significantly improve the performance of question answering systems. Petroni et al. [18] studied the influence of retrieved relevant context in generative language models. Authors found that Augmenting search queries with relevant contexts improved the performance of the pre-trained language models in an unsupervised setting. BERT's next sentence prediction effectively removed the noisy and irrelevant context. *Figure 3.3* Shows a evolution of natural language processing.

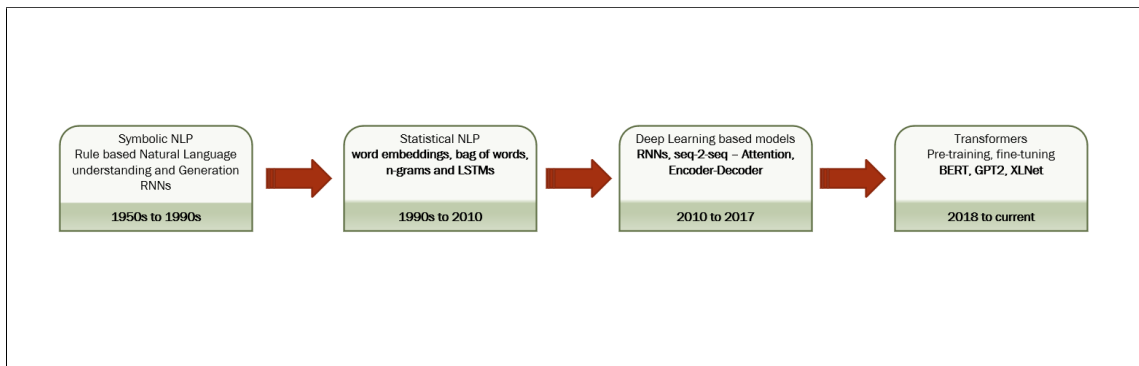


Figure 3.3: A brief history of natural language processing.

### 3.1.2 BERT (Bidirectional Encoder Representations from Transformers)

BERT is a Transformer[25] (deep learning) based machine learning technique used for natural language processing pre-training developed by Google [9]. Language models before BERT like Transformers and LSTM (Long short-term memory) can read only sequentially, either from left to right or right to left. BERT was designed to read in both directions (bi-directional architecture), which made it possible to achieve two critical NLP pre-training tasks,

- *Masked Language Modeling (MLM)*: masking the actual word and predicting the masked word in a sentence based on the context.
- *Next Sentence Prediction (NSP)*: the capability to predict whether the two given sentences are related, logically connected, or random.

The main objective of BERT is to understand the natural language, predict the next or missing word in the pre-training phase, and fine-tune BERT in the next phase for the specific application. *Figure 3.4* shows an overview of BERT based question answering system. BERT was trained on the Wikipedia dataset (2.5B words) and Google's BooksCorpus dataset (800M words). BERT is currently used in applications like Google search, question answering systems, conversational AI, sentiment analysis, neural machine translation, and text summarization.

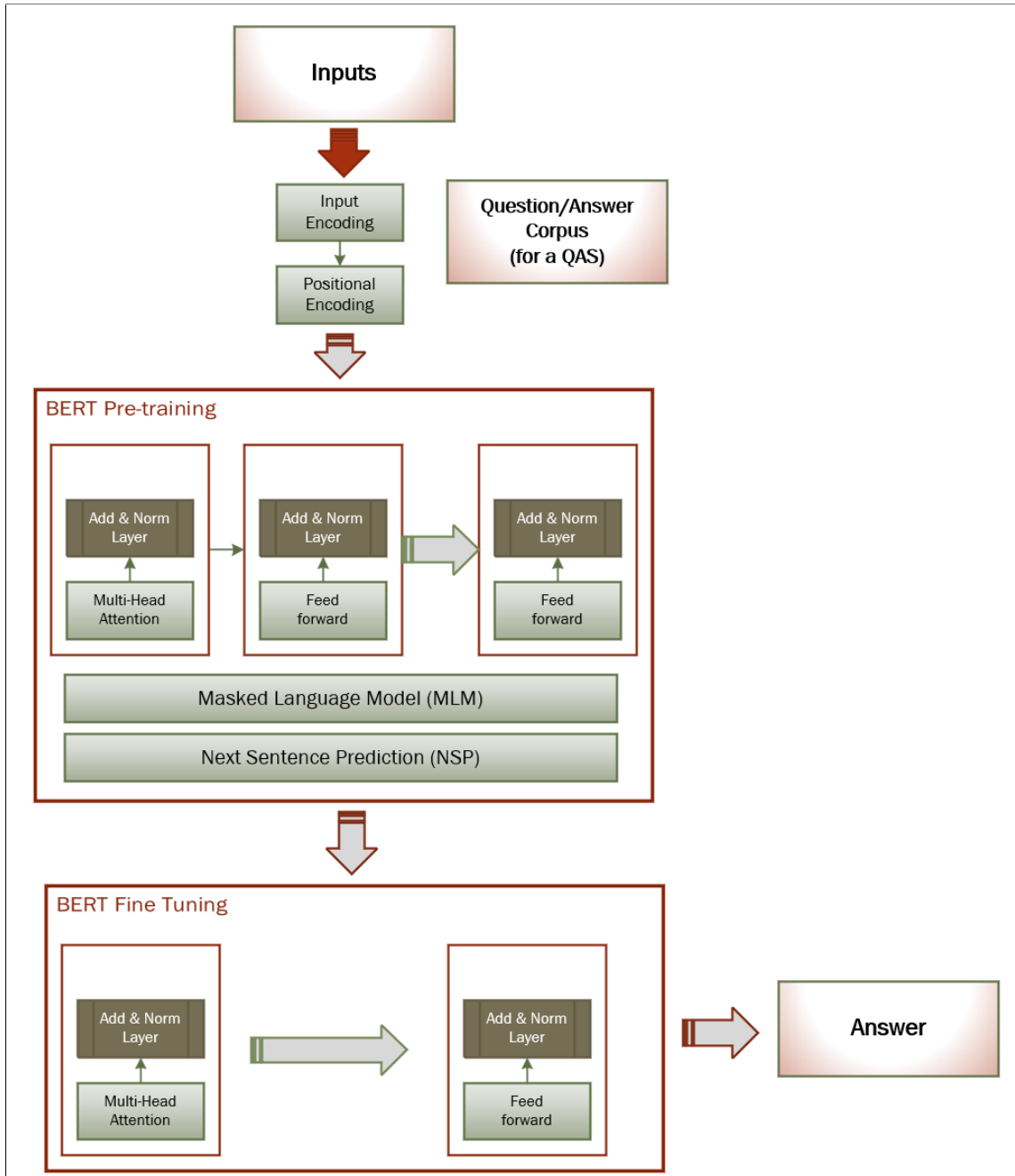


Figure 3.4: An overview of BERT-based Question Answering System.

## 3.2 QAS Datasets

Following are some of the large question-answering datasets containing questions and answers for use in Natural language processing tasks.

### 3.2.1 Stanford Question Answering Dataset (SQuAD)

The Stanford Question Answering Dataset (SQuAD) is a new reading comprehension dataset comprised of 100,000+ questions posed by crowdworkers on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding reading passage. SQuAD2.0 introduced tests to test the ability of a system not only to answer reading comprehension questions but also to abstain when presented with a question that cannot be answered based on the provided paragraph.

Rajpurkar et al. [19] developed a QAS model using the SQuAD dataset to understand the types of reasoning required to answer the questions, leaning heavily on dependency and constituency trees. The authors also built a robust logistic regression model, which achieves an F1 score of 51.0%, a significant improvement over a simple baseline (20%). However, human performance (86.8%) is much higher, indicating that the dataset presents a good challenge problem for future research.

### 3.2.2 Situations With Adversarial Generations (SWAG)

A large scale dataset with natural language inference that leverages commonsense knowledge and reasoning. This dataset consists of more than 100,000 multiple choice questions about grounded situations [28] [6].

### 3.2.3 Microsoft MACHine Reading Comprehension (MS MARCO)

A large-scale machine reading comprehension dataset created by Microsoft AI & Research. This dataset consists of more than 1,000,000 anonymized questions from Bing's search query logs, each with a human-generated answer and 182,669 completely human-rewritten answers [1].

### 3.2.4 Google's Natural Questions

Kwiatkowski et al. [13] present Google's Natural Questions corpus, a question-answering dataset consisting of aggregated real-world anonymized queries issued to the Google search engine. An annotator was presented with a question along with a Wikipedia page from the top 5 search results and annotated a long answer (typically a paragraph) and a short answer (one or more entities) if present on the page or marked null if long/short answer was not found. The public release consisted of 307,373 training examples with single annotations, 7,830 examples with 5-way annotations for development data, and a further 7,842 examples 5-way annotated sequestered as test data. It also presented experiments validating the quality of the data. This dataset also described the analysis of 25-way annotations on 302 examples, giving insights into human variability on the annotation task. The authors introduced robust metrics to evaluate question-answering systems, demonstrated high human upper bounds on these metrics, and established baseline results using competitive methods drawn from related literature.

### 3.2.5 DeepMind

DeepMind [20] [21] is a large-scale extendable dataset consisting about 2 million of mathematical question and answer pairs from various question types like algebra,

arithmetic, calculus, comparison, measurement, polynomials and probability at a school-level difficulty. This dataset tests the mathematical learning and algebraic reasoning skills of learning models.

### **3.2.6 Other Major Datasets**

- The bAbI project is a rich collection of datasets for automatic text understanding and reasoning developed by Facebook AI Research <sup>1</sup>.
- DeepMind Q&A Dataset - Question/Answer datasets from CNN and Daily Mail [12].

### **3.2.7 Custom Domain-Specific Datasets**

Domain-specific datasets can be used, for example, an image repository or archive of video library to train a CNN.

---

<sup>1</sup><https://research.facebook.com/downloads/babi/>



<b>Name</b>	<b>Description</b>	<b>Dataset Link</b>
SQuAD	Stanford Question Answering Dataset	<a href="https://rajpurkar.github.io/SQuAD-explorer/">https://rajpurkar.github.io/SQuAD-explorer/</a>
SWAG	Situations With Adversarial Generations	<a href="https://github.com/rowanz/swagaf/tree/master/data">https://github.com/rowanz/swagaf/tree/master/data</a>
MS MARCO	Microsoft MACHine Reading Comprehension	<a href="https://microsoft.github.io/msmarco/">https://microsoft.github.io/msmarco/</a>
Natural Questions	from Google AI	<a href="https://github.com/google-research-datasets/natural-questions">https://github.com/google-research-datasets/natural-questions</a>
The bAbI project	from Facebook AI Research	<a href="https://github.com/facebookarchive/bAbI-tasks">https://github.com/facebookarchive/bAbI-tasks</a>
DeepMind	Mathematical Question and Answer pairs	<a href="https://github.com/deepmind/mathematics_dataset">https://github.com/deepmind/mathematics_dataset</a>

Table 3.1: Question Answering System Datasets.

## Chapter 4

### Question Answering System for R

#### 4.1 QA4R Prototype: Design and Implementation

This prototype was developed using google colab and python language. *Figure 4.1* shows the steps followed in developing this prototype. Colab notebook is available in the project GitHub repository<sup>1</sup>.

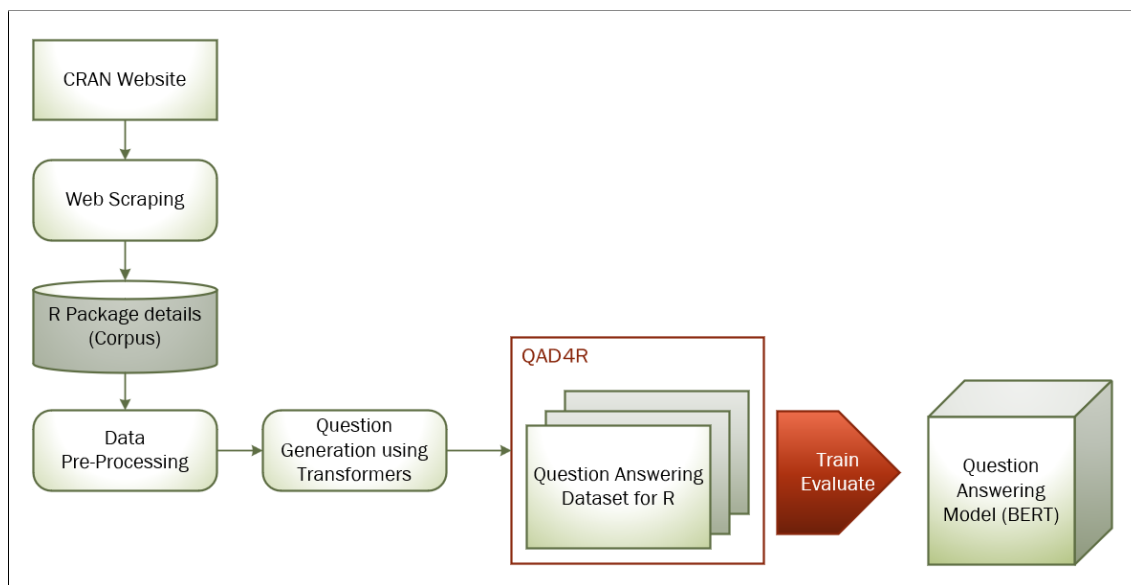
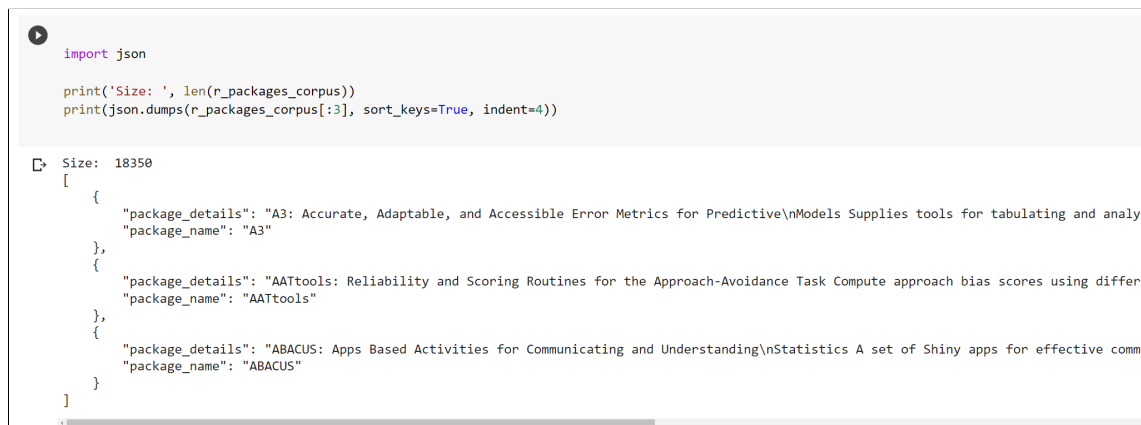


Figure 4.1: QA4R Prototype: System Design

<sup>1</sup><https://github.com/GanB/QA4R-A-Question-Answering-System-for-R-Packages>

## 4.2 Data Acquisition - Corpus

Data acquisition is one of the critical activities for this research. Developing the corpus for R packages is the key to successfully building the question-answering system. R packages data were extracted from the CRAN website <sup>2</sup>. We have extracted the complete list of 18,350 R packages from the CRAN website and created the corpus *Figure 4.2*. This file contains the package name, title, and description of the package extracted from the url of each of those packages. Following python packages were used to extract the corpus - pandas, requests, beautifulsoup4. This process ran for 4 hours to extract the corpus details.



```
import json

print('Size: ', len(r_packages_corpus))
print(json.dumps(r_packages_corpus[:3], sort_keys=True, indent=4))
```

```
Size: 18350
[
  {
    "package_details": "A3: Accurate, Adaptable, and Accessible Error Metrics for Predictive\nModels Supplies tools for tabulating and analy",
    "package_name": "A3"
  },
  {
    "package_details": "AATtools: Reliability and Scoring Routines for the Approach-Avoidance Task Compute approach bias scores using differ",
    "package_name": "AATtools"
  },
  {
    "package_details": "ABACUS: Apps Based Activities for Communicating and Understanding\nStatistics A set of Shiny apps for effective comm",
    "package_name": "ABACUS"
  }
]
```

Figure 4.2: R packages corpus extracted using web scraping.

---

<sup>2</sup><https://cran.r-project.org/>

### 4.3 QAD4R: Question Answering Dataset for R

R packages corpus was then pre-processed to drop special characters and other invalid data conditions. An Open source question generation library based on transformers was used to develop the questions for R packages based on the corpus<sup>3</sup>. The questions were generated using answer-aware question generation and multitask question-answer-generation using the T5 model. Python was used to perform data pre-processing and calling the question generation module to generate the dataset in the format needed for the question answering model. Following python packages were used to extract and create the dataset - transformers 3.0.0, nltk, downloader punkt, pipeline (from the library), UUID, regex, and pickle. Pickle is a python package used to serialize and deserialize python objects to persistent storage like files and databases. We used the pickle library to store the dataset as a binary object to a file since it is heavy GPU-dependent processing and ran for about 6 hours. *Figure 4.3* shows the structure of the dataset. *Figure 4.4* shows question-answering dataset for R.

---

<sup>3</sup>[https://github.com/patil-suraj/question\\_generation](https://github.com/patil-suraj/question_generation)

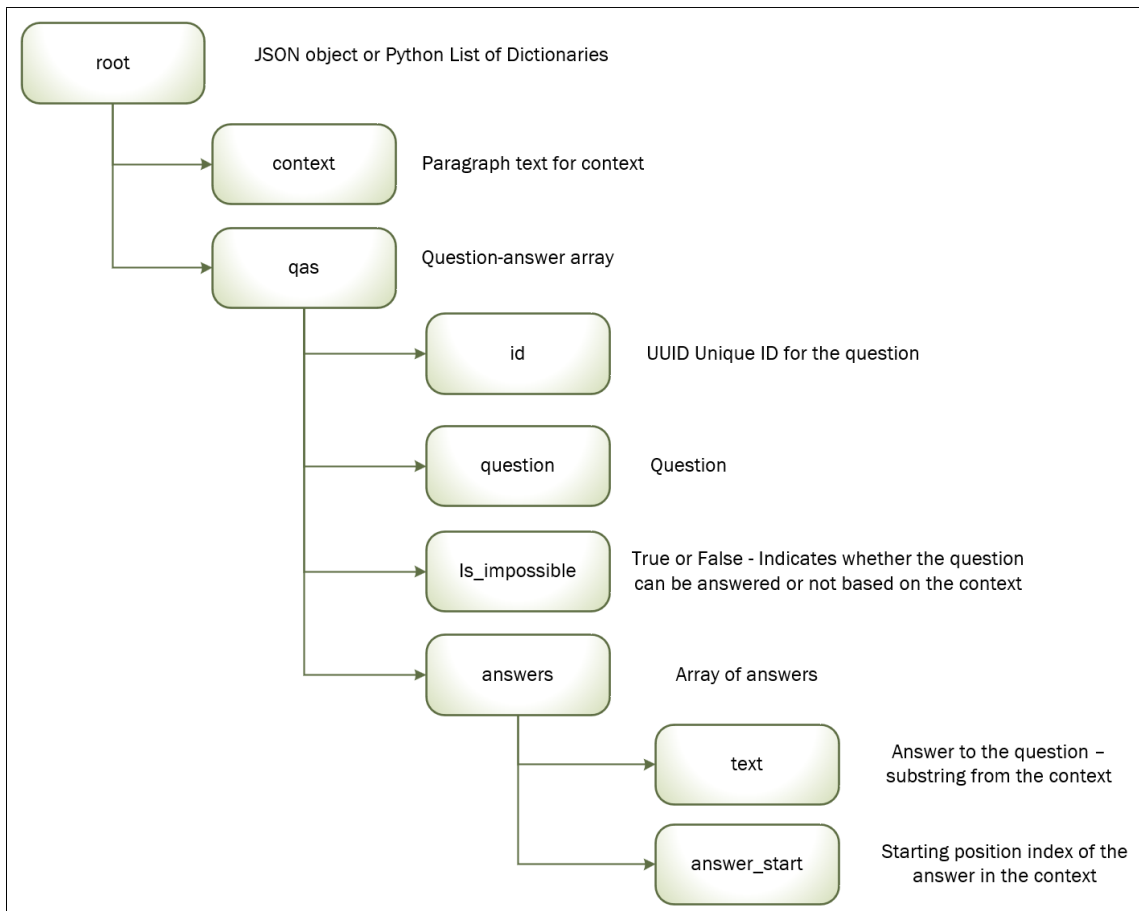


Figure 4.3: Structure of question answering dataset for R

```

{
  "context": "abe: Augmented Backward Elimination Performs augmented backward elimination and checks the stability of the obtained model. Augment
  "qas": [
    {
      "answers": [
        {
          "answer_start": 7,
          "text": "Alpha and Beta Diversity Measures"
        }
      ],
      "id": "8cbf1849-bd63-494f-9da9-438c4e471fe6",
      "is_impossible": false,
      "question": "What are abdiv's measures for measuring ecological diversity?"
    },
    {
      "answers": [
        {
          "answer_start": 145,
          "text": "alpha diversity measures the diversity within a single site or sample, and beta diversity measures the diversity across
        }
      ],
      "id": "9fe116dd-dd62-4f54-925f-a5faa3ee4728",
      "is_impossible": false,
      "question": "What does beta diversity measure the diversity across two sites?"
    }
  ]
}
  
```

Figure 4.4: QAD4R: question answering dataset for R

#### 4.4 Question Answering Model

We used the Simple Transformers open-source question answering model library to train and evaluate the question answering system for R using the dataset. Following are the steps followed,

1. initialize the model
2. set up train test datasets
3. train the model using `train_model()`
4. evaluate the model using `eval_model()`
5. make predictions on unlabelled data using `predict()`

BERT and DistilBERT models were used in this prototype. Below hyper parameters were used for the training.

1. `n_best_size = 1`
2. `train_batch_size = 16`
3. `evaluate_during_training = True`
4. `num_train_epochs = 5`
5. `max_seq_length = 128`
6. `evaluate_during_training_steps: 1000`
7. `eval_batch_size = 64`

The weights & Biases<sup>4</sup> machine learning platform was used to track the iteration, evaluate model performance, and visualize results. Following python packages were used simpletransformers and wandb to run the question-answering model.

---

<sup>4</sup><https://wandb.ai/site>

## Chapter 5

### Results Evaluation

We can successfully extract the high-level information about 18,350 R packages from CRAN and create the corpus. We created the R question-answer dataset using the corpus and generated questions using the question generation model. The quality of questions generated using the model is unsatisfactory, so we annotated the dataset with custom questions to train and evaluate the model.

*Figure 5.1* shows the annotated training dataset for the question answering model. We can notice that training loss dropped after six epochs and remained consistent. We can also observe that DistilBERT performed slightly better than BERT.

```
[{'context': 'A system for declaratively creating graphics, based on The Grammar of Graphics. ggplot2 is used to Create Elegant Data Visualisations Using the Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.', 'qas': [{'id': '00001', 'is_impossible': False, 'question': 'What is the R package for data visualization?', 'answers': [{'text': 'ggplot2 is used to Create Elegant Data Visualisations Using the Grammar of Graphics', 'answer_start': 81}]}, {'id': '00002', 'is_impossible': False, 'question': 'What is the R package for grammar of graphics?', 'answers': [{'text': 'ggplot2 is used to Create Elegant Data Visualisations Using the Grammar of Graphics', 'answer_start': 81}]}], {'context': 'Shiny is an R package that makes it easy to build interactive web apps straight from R. You can build standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions. Shiny comes with a variety of built in input widgets. With minimal syntax it is possible to include widgets in your apps. Shiny is an R package that makes it easy to build interactive web apps straight from R. You can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.'}]
```

Figure 5.1: Annotated training dataset



Figure 5.2, Figure 5.3 and Figure 5.4 shows the evaluation results for BERT and DistilBERT models.

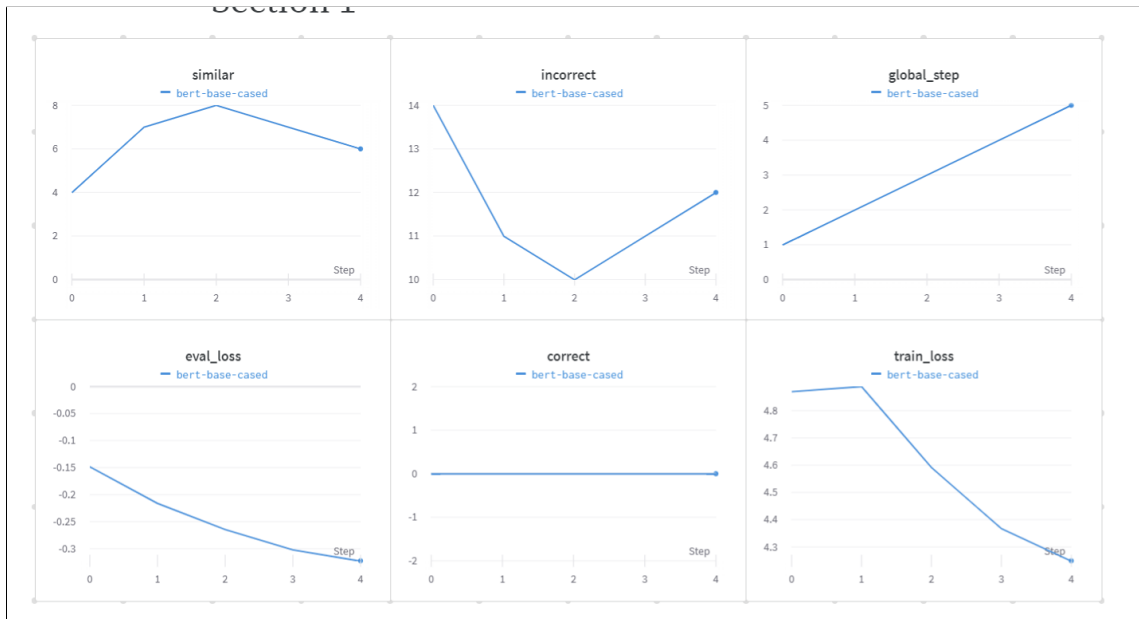


Figure 5.2: Evaluation results for the BERT Model

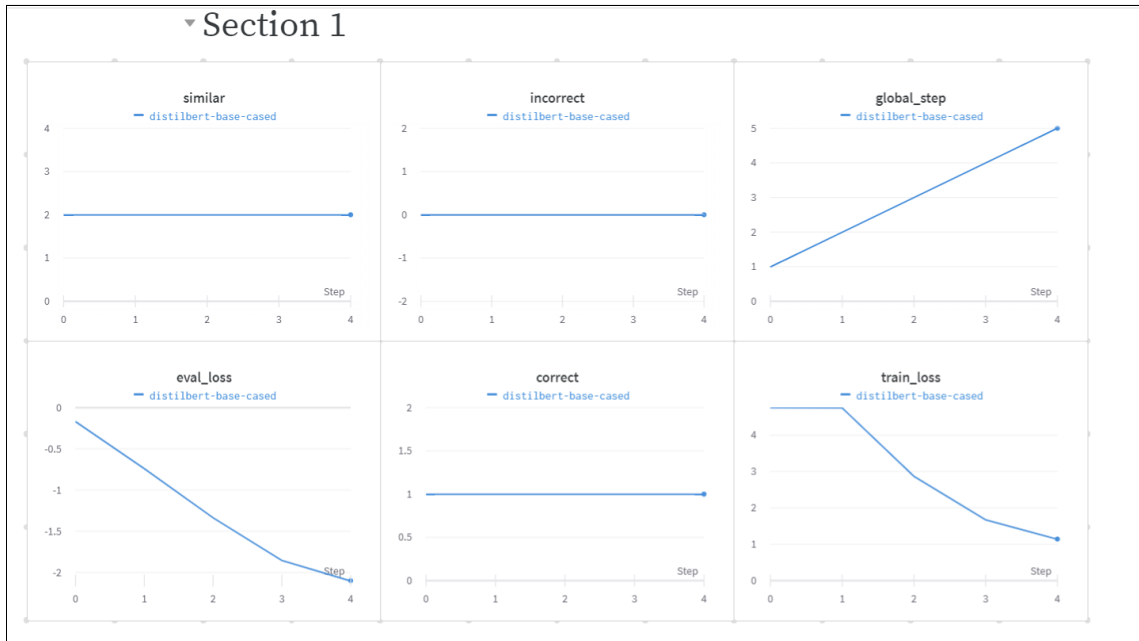


Figure 5.3: Evaluation results for the DistilBERT Model

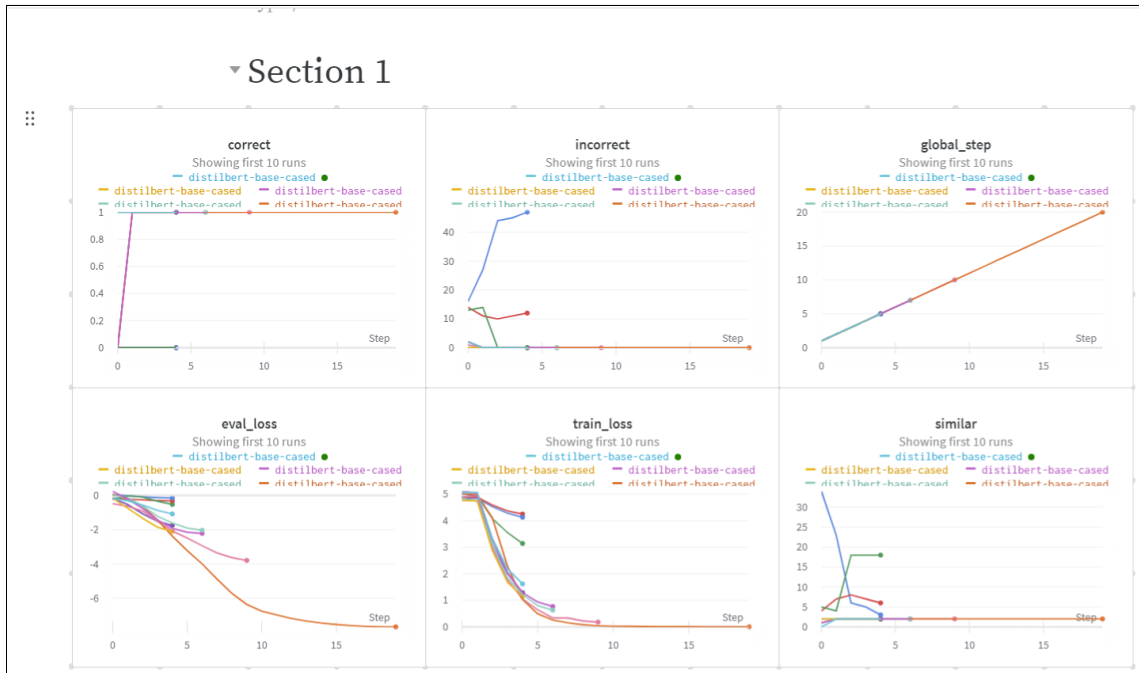


Figure 5.4: Evaluation results for the last 10 runs

Figure 5.5 shows the answer predicted by the question answering model using the BERT language model type for a given context and question.

```
[28] # Make predictions with the model
to_predict = [
  {
    "context": "gtrendsR is the R package to perform and display Google Trends queries. An interface for retrieving and displaying the info",
    "qas": [
      {
        "question": "what is package for google trends?",
        "id": "0",
      }
    ],
  }
]

answers, probabilities = model.predict(to_predict)

print(answers)

print(probabilities)
```

```
convert squad examples to features: 100% | ██████████ | 1/1 [00:00<00:00, 148.18it/s]
add example index and unique id: 100% | ██████████ | 1/1 [00:00<00:00, 7913.78it/s]
Running Prediction: 100% | ██████████ | 1/1 [00:00<00:00, 22.41it/s]
[{'id': '0', 'answer': ['gtrendsR is the R package to perform and display Google Trends queries. An interface for retrieving and displaying the info', 'gtrendsR is the R package to perform and display Google Trends queries. An interface for retrieving and displaying the info']}]
```

Figure 5.5: Answer prediction by the model based on the context and question

## Chapter 6

### Future Work and Conclusions

We developed a prototype to demonstrate the question-answering system for R packages. This work can be further expanded by including more sources like *stackoverflow* and *reddit* to develop the corpus. The question-answering dataset developed for R packages can be improved with more meaningful questions in the context of R packages. Furthermore, other language model types like ALBERT, ELECTRA, RoBERTa, and XLNet can be used to train the question-answering system and compare the performance.

## BIBLIOGRAPHY

- [1] BAJAJ, P., CAMPOS, D., CRASWELL, N., DENG, L., GAO, J., LIU, X., MAJUMDER, R., MCNAMARA, A., MITRA, B., NGUYEN, T., ROSENBERG, M., SONG, X., STOICA, A., TIWARY, S., AND WANG, T. MS MARCO: A Human Generated MACHine Reading COMprehension Dataset. *arXiv:1611.09268 [cs]* (Oct. 2018). arXiv: 1611.09268.
- [2] BISWAS, P., SHARAN, A., AND MALIK, N. A framework for restricted domain Question Answering System. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)* (Feb. 2014), pp. 613–620.
- [3] BOUZIANE, A., BOUCHIHA, D., DOUMI, N., AND MALKI, M. Question Answering Systems: Survey and Trends. *Procedia Computer Science 73* (2015), 366–375.
- [4] CALIJORNE SOARES, M. A., AND PARREIRAS, F. S. A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University - Computer and Information Sciences 32*, 6 (July 2020), 635–646.
- [5] CHEN, D., FISCH, A., WESTON, J., AND BORDES, A. Reading wikipedia to answer open-domain questions, 2017.
- [6] CHEN, M., D’ARCY, M., LIU, A., FERNANDEZ, J., AND DOWNEY, D. CO-DAH: An Adversarially Authored Question-Answer Dataset for Common Sense. *arXiv:1904.04365 [cs]* (July 2019). arXiv: 1904.04365.
- [7] CLAES, M., MENS, T., AND GROSJEAN, P. Maintainer: A web-based dashboard for maintainers of cran packages. In *Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution (USA, 2014)*, ICSME ’14, IEEE Computer Society, pp. 597 – 600.
- [8] DECAN, A., MENS, T., CLAES, M., AND GROSJEAN, P. On the development and distribution of r packages: An empirical analysis of the r ecosystem. In

*Proceedings of the 2015 European Conference on Software Architecture Workshops* (New York, NY, USA, 2015), ECSAW '15, Association for Computing Machinery.

- [9] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding, 2018.
- [10] GREEN, B. F., WOLF, A. K., CHOMSKY, C., AND LAUGHERY, K. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference* (New York, NY, USA, 1961), IRE-AIEE-ACM '61 (Western), Association for Computing Machinery, pp. 219 – 224.
- [11] GUPTA, A., AND CHAUHAN, M. P. Survey: Restricted Domain Question Answering System Using Semantic. *Restricted Domain Question Answering* 5, 5 (2016), 3.
- [12] HERMANN, K. M., KOČISKÝ, T., GREFENSTETTE, E., ESPEHOLT, L., KAY, W., SULEYMAN, M., AND BLUNSOM, P. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)* (2015).
- [13] KWIATKOWSKI, T., PALOMAKI, J., REDFIELD, O., COLLINS, M., PARIKH, A., ALBERTI, C., EPSTEIN, D., POLOSUKHIN, I., KELCEY, M., DEVLIN, J., LEE, K., TOUTANOVA, K. N., JONES, L., CHANG, M.-W., DAI, A., USZKOREIT, J., LE, Q., AND PETROV, S. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics* (2019).
- [14] LUNN, S., ZHU, J., AND ROSS, M. Utilizing web scraping and natural language processing to better inform pedagogical practice. In *2020 IEEE Frontiers in Education Conference (FIE)* (2020), pp. 1–9.
- [15] MARTINEZ-GIL, J. A survey on legal question answering systems, 2021.
- [16] MISHRA, A., AND JAIN, S. K. A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences* 28, 3 (2016), 345 – 361.
- [17] MOLLÁJ, D., AND VICEDO, J. L. Question Answering in Restricted Domains: An Overview. *Computational Linguistics* 33, 1 (2007), 41–61.
- [18] PETRONI, F., LEWIS, P., PIKTUS, A., ROCKTÄSCHEL, T., WU, Y., MILLER, A. H., AND RIEDEL, S. How context affects language models’ factual predictions, 2020.

- [19] RAJPURKAR, P., ZHANG, J., LOPYREV, K., AND LIANG, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv:1606.05250 [cs]* (Oct. 2016). arXiv: 1606.05250.
- [20] SAXTON, D., GREFENSTETTE, E., HILL, F., AND KOHLI, P. Analysing Mathematical Reasoning Abilities of Neural Models. *arXiv:1904.01557 [cs, stat]* (Apr. 2019). arXiv: 1904.01557.
- [21] SAXTON, D., GREFENSTETTE, E., HILL, F., AND KOHLI, P. Analysing mathematical reasoning abilities of neural models, 2019.
- [22] SRIHARI, R., AND LI, W. A question answering system supported by information extraction. In *Proceedings of the sixth conference on Applied natural language processing - (Seattle, Washington, 2000)*, Association for Computational Linguistics, pp. 166–172.
- [23] STRZALKOWSKI, T. Document indexing and retrieval using natural language processing. In *Intelligent Multimedia Information Retrieval Systems and Management - Volume 1* (Paris, FRA, Oct. 1994), RIAO '94, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, pp. 131–143.
- [24] SUCUNUTA, M. E., AND RIOFRIO, G. E. Architecture of a question-answering system for a specific repository of documents. In *2010 2nd International Conference on Software Technology and Engineering* (2010), vol. 2, pp. V2–12–V2–16.
- [25] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need, 2017.
- [26] WALTER, S., UNGER, C., CIMIANO, P., AND BÄR, D. Evaluation of a layered approach to question answering over linked data. In *The Semantic Web ISWC 2012*, vol. 7650. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 362–374. Series Title: Lecture Notes in Computer Science.
- [27] WANG, W., AUER, J., PARASURAMAN, R., ZUBAREV, I., BRANDYBERRY, D., AND HARPER, M. P. A question answering system developed as a project in a natural language processing course. In *ANLP-NAACL 2000 Workshop: Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems* (2000).
- [28] ZELLERS, R., BISK, Y., SCHWARTZ, R., AND CHOI, Y. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *arXiv:1808.05326 [cs]* (Aug. 2018). arXiv: 1808.05326.



