Unveiling the Past: AI-Powered Historical Book Question Answering

by

Hari Thapliyal

MBA, MCA, MS(DS), PGDFM, PMP, PMI-ACP, PRINCE2, SCT, MCT

Student Number: 59028

Thesis

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfilment

Of the Requirements

For the Degree

GLOBAL DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

November, 2023

Unveiling the Past: AI-Powered Historical Book Question Answering

by

Hari Thapliyal

APPROVED BY

____

Dr. Iva Buljubasic, Ph.D, Chair

____

Dr. Ljiljana Kukec, Ph.D, Committee Member

____

Dr Hemant Palivela, Ph.D, Research Supervisor

Contents

List of Tables

List of Figures

DEDICATION

वाक्यकारं वररुचिं भाष्यकारं पतञ्जलिम्। पाणिनिं सूत्रकारञ्च प्रणतोस्मि मुनित्रयम् ॥

”I bow to the three great sages: Vararuchi, the composer of sentences; Patanjali, the author of commentaries; and Panini, the composer of the Sutras.” These three Maharishi are the north stars of the Sanskrit language.

yenAkSharasamAmnAyam adhigamya maheshvarAt | kRutsnaM vyAkaraNaM proktaM tasmai pANinaye namaH ||

येनाक्षरसमाम्नायं अधिगम्य महेश्वरात् | कृत्स्नं व्याकरणं प्रोक्तं तस्मै पाणिनये नमः ||

Salutations to Panini, who, having obtained the instructions of Maheshvara, has expounded the entire grammar by means of the alphabets.

To Maharshi Panini (पाणिनि), the greatest known grammarian of the world of all time. His work Ashtaadhyai (अष्टाध्यायी) forces us to understand that the Language is not only about alphabets, words, sentences, and grammar but a complex set of mathematical rules. The Computer Programmers of the 20th century and Machine Learning Experts of the 21st century want machines should learn these rules from the letters, and words (corpus) of scripts (which come from various languages).

ACKNOWLEDGMENT

We cannot achieve anything of high value alone, without the support of people and the systems around us. Sometimes these supports are direct and other times these exist around us in the form of a system or an ecosystem. Without these, I couldn't even have tried it or it could have taken a very long time for me to complete this work.

Thanks for OpenAI's ChatGPT. It was my first-hand research guide for my countless questions on programming, history, machine learning, philosophy, grammar, education system, learning process, and how to proceed with research in a systematic way. which my University guide could not have answered because of time constraints or other limitations. Many times answers were hallucinating but a good researcher knows how to pick and what to pick.

Thanks for Kaggle and Google: Kaggle's kernel is an excellent free tool for deep learning work. Google's Colab, is excellent for organizing the flow of deep learning work in the Colab notebook. They both provide free GPU and TPU resources. I do not have a GPU machine and it is expensive to buy this, that too when I already have a good machine with me, discarding my working laptop is not possible. Without Kaggle's Kernel and Google Colab's support, it was impossible to conduct experiments on the finetuning of deep learning models. Even inferencing from the deep learning models using a CPU machine is painful or almost impossible work.

Thanks to Overleaf for giving the community a fanatic research writing online

tool. Initially, it was painful to write even a single line of code in LaTex. But slowly when I got a grip on this I felt more comfortable writing any pdf document in Overleaf and avoided using any Word document.

Thanks to GitHub for version management. Without this, my confidence that my code is always available from other machines and I can retrieve older versions at any time was not possible.

Thanks to Paperwithcode for their leaderboard. They track the performance of the model proposed by each great AI paper. Without them, it was impossible to conduct a literature survey.

Thanks to Mendeley for their free research paper management tool. Without this, it is impossible for me to track the contributions of earlier researchers. Neither it was possible to complete the literature survey.

Thanks to Huggingface for providing a free cloud of pretrained models. Without this, it could have been very painful to look at zeroshot models, their performance, and how to finetune these.

Entire open-source community: I have used many open-source frameworks in my work like pandas, numpy, seaborn, pytorch, sentence transformer, maintainer of python language, and many others. Without their continuously building reliable libraries, it was impossible for me to write every line of code.

Thanks to the upGrad founders for creating this opportunity to learn and pursue my DBA program at this age when most of my friends and cousins are looking for retirement or thinking about what to do after retirement.

Thanks to Sanskrit Bharati. They inspired me to learn Sanskrit and ultimately I became passionate about human language and finally turned towards NLP.

Thanks to Prof. Anil Vipulla of IIIT Hyderabad: He inspired me to learn and use of LaTex and Overleaf for research work.

Thanks to Bhaskara Reddy Sannapureddy for pushing new stuff every day in our AI WhatsApp group. He keeps pushing AI/ML articles, ideas, books, links, and conferences in this group. Many times it becomes overwhelming but this is how the AI landscape is expanding and changing. This inspires me and keeps me on my toe every day.

Thanks to Mr. Ramesh Rawat, he is an old person, a fatherly figure, and my neighbor but his passion for learning and knowing is contagious. Whenever he finds some interesting AI article in a newspaper he will read it thoroughly and then share that with me. I don't know that he ever missed learning more and sharing more about this new technology at an age when most people feel they cannot handle their smartphones.

Thanks to my traditional Advaita Vedanta Guru Swami Parmarthananda Ji, who inspired me to look at the meaning of life, and define everything in life. Understand what is awareness, consciousness, compassion, intelligence, meaning of life from the Upanishadic literature. This inspired me to dive deeper with the help of linguistics and mathematics, and finally ended up in computational linguistics, NLP, NLU, NLG, AI and AGI (Artificial General Intelligence).

Thanks to all my friends and relatives who didn't distract me during my DBA journey.

Last, but very important my research guide on this important academic journey, Dr. Hemant Palivala, Most of the time he was busy but when I needed him badly he never stopped himself to come forward and encouraged me to come on the track and finish this work.

LIST OF ABBREVIATIONS

1. AGM - Automatic Generation Method
2. AGS - Answer Generation System
3. ALBERT - A Lite BERT
4. API - Application Programming Interface
5. AQG - Automatic Question Generation
6. BART - Bidirectional Auto-Regressive Transformers
7. BERT - Bidirectional Encoder Representations from Transformers
8. BFQ - Blank space-fill Questions
9. BIG - BIG Benchmark
10. BLEU - Bilingual Evaluation Understudy
11. BLOOM - BigScience Large Open-science Open-access Multilingual Language Model
12. CAI - Conversational AI
13. CDQA - Closed Domain Question Answering
14. CTRL - Conditional Transformer Language Model

15. DEBERTA - Decoding-Enhanced BERT with Disentangled Attention

16. DRS - Document Retrieval System

17. ELECTRA - Efficiently Learning an Encoder that Classifies Token Replacements Accurately

18. ERNIE - Enhanced Representation through Knowledge Integration

19. FAQ - Frequently Answered Question

20. FIB - Fill in the Blank

21. GAN - Generative Adversarial Network

22. GLUE - General Language Understanding Evaluation

23. GLUECoS - General Language Understanding Evaluation Code-Switched

24. GPT - Generative Pretrained Transformer

25. GRU - Gated Recurrent Unit

26. IPR - Intellectual Property Right

27. ITS - Intelligent Tutoring Systems

28. KMLM - Knowledge Masked Language Modeling

29. KNN - K Nearest Neighbour

30. LLM - Large Language Model

31. LLaMA - Large Language Model Meta AI

32. LSTM - Long Short-Term Memory

33. MCQ - Multi Choice Question

34. METEOR - Metric for Evaluation of Translation with Explicit ORdering

35. MTL - Multitask Learning

36. NER - Named Entity Recognition

37. NLG - Natural Language Generation

38. NLI - Natural Language Inference

39. NLP - Natural Language Processing

40. NLTK - Natural Language Toolkit

41. ODQA - Open Domain Question Answering

42. OOV - Out of Vocabulary

43. OPT - Open Pre-trained Transformer

44. PEFT - Parameter-efficient Fine-tuning

45. POS - Parts of Speech

46. PaLM - Pathway Language Model

47. QAGS - Question Answer Generation System

48. QA - In our work, QA refers to Question Answering, and not Quality Assurance, which is general understanding.

49. QAS - Question Answering System.

50. QG - Question generation

51. RAAGS - Retrieval Augmented Answer Generation System

52. RAG - Retrieval Augmented Generation

53. REALM - Retrieval-Augmented Language Model Pre-Training

54. RGF - Retrieve-Generate-Filter

55. RNN - Recurrent Neural Network

56. ROGUE - Recall-Oriented Understudy for Gisting Evaluation,

57. RoBERTa - A Robustly Optimized BERT Pretraining Approach

58. SOTA - State-of-the-Art

59. SQuAD - Stanford Question Answering Dataset

60. STT - Speech-to-Text

61. T5 - Text-to-Text Transfer Transformer

ABSTRACT

Unveiling the Past: AI-Powered Historical Book Question Answering

Hari Thapliyal

2023

Dissertation Chair: Dr. Iva Buljubasic

Co-Chair: Dr. Ljiljana Kukec

In the recent past, we have seen a number of approaches, datasets, models, and even large language models (LLM) used for question-answering. The results of these initiatives are very encouraging, but generating questions from the given document, and generating a correct answer to a given question from a given document, is still challenging. It becomes more challenging when the text is historical, and it is translated from another language and a different script is used to write the translation than the script of the original text. This problem is because of the spelling of nouns in the original script. Secondly generating description answers and evaluating the correctness of descriptive answers is a challenge. Thirdly, if n number of question-answer pairs are generated from a certain corpus then how do you measure the performance of this model? Finally, if we have a large body of text and some questions in our mind then without giving the context how to get the answer? In this work, we are exploring techniques for creating questions and answers for a book corpus using ChatGPT and other available techniques, we call this QAGS. Secondly, we are finetuning the t5, flan-t5 model for creating an answer generation system, we call this AGS. Thirdly we are retrieving a relevant document that can answer a question in our hand, we call this DRS. Finally, we are creating and evaluating a system that can answer a history question without any context, we call this RAAGS. In this work, we are using The Mahabharata book as a corpus. To evaluate the different sub-systems we have used different metrics like BLEU, ROUGE, Accuracy, Recall, R@n, P@n, F1@n, Cosine. We have used SentenceTransformer for text embedding. Our approach does not depend upon the domain, script, or language of the text document. Neither, does it depend upon the Era when the text was written. It doesn't need any manual feature engineering of the text. We have explored SOTA transformers like T5, distilBERT, RoBERTa, Bloom, BERT, BigBird from huggingface for zeroshot learning. The cosine between answer & question, answer & chunk is 0.91. In DRS MRR metric is 0.55, and MAP is 0.25. In AGS cosine between the reference answer and predicted answer is 0.827. In RAAGS cosine between the reference answer and the predicted answer is 0.763

Keywords: Question Answering with NLP, Historical Books Question Answering, Question Answering Generation, NLP Transformer Models

Chapter 1

INTRODUCTION

How many facts you know, and how many questions you can answer at the right time are signs of how knowledgeable you are and your presence of mind. But the greatest sign of your intelligence is whether can you ask a deeper question at the right time that no one asked.

In a professional setup, questions can be asked by an interviewer, one who is asking questions is called the interviewer, and one who is responding to these questions is called the interviewee. In an exam setup, the question can be asked by a machine or examination body and the examinee needs to answer those questions. In a judicial setup, questions are asked by a judge or advocate and answers can be given by a witness or accused or petitioner. In a family setup, questions can be asked by parents or children; answers can be given by children or parents. In a school setup, a question is asked by a teacher or student and an answer can be given by the student or teacher. In a crime investigation setup, police ask questions, and a criminal or accused answers those questions. In a business setup, the boss or subordinate or client can ask questions and the subordinate or client or boss can answer those questions. when you are alone and puzzled over a question, you ask a question to yourself, think deeper, and get the answer from your own thinking, and intuition. Thus we see questioning and answering are everywhere in human life and in fact if we say questions play a critical role in the evolution of human civilization that won't be any hyperbole. For the purpose of simplicity in this whole work, one who is asking a question will be referred to as the questioner and one who is responding to the question will be referred to as the answerer.

The number of classical literature in India is quite substantial, spanning multiple languages, scripts, and genres. It is challenging to provide an exact count due to the vastness and complexity of India's literary heritage. According to Bibek Debroy, (Chairman, Economic Advisory Council to the Prime Minister and Member, NITI Aayog) in his article in Tribune India, (Debroy & Chauhan 2022) says National Mission for Manuscripts (NAMAMI) was set up in 2003, it has listed just 3.5 million manuscripts out of the estimated 40 million in India. And mind the word listed, it is not translated. This body of Indian literature encompasses a wide range of texts, including epics, religious scriptures, history, poetry, philosophy, drama, science, and more. The classical literature of India is a huge treasure of the rich cultural and intellectual history of the Asian region. Scholars, researchers, and enthusiasts worldwide keep studying, translating, digitizing, and sharing this knowledge treasure.

For the namesake, a few classical literatures are Mahabharat which has 100K shlokas, Valmiki Ramayana (24K shlokas), Bhagwatam (18K shlokas), Arthashastra (6K), Shakuntalam (1800), Raghuvamsha (1800), there are 18 Puranas of various lengths, folktale stories, etc. These kinds of literature have been translated into various other world languages. Authors have translated these texts according to their own intellectual rigor, ideological and philosophical alignment, and understanding of the context. To impart the lessons from the stories, we need to narrate these stories in different languages to the audience. Traditionally, these stories are narrated by grandpa and grandma to their children. Professional storytellers called Kathakars or Vyasacharya, also narrate these stories on different occasions for different audiences. Apart from these, other ways of telling the stories were Nautanki, drama, and monologue. Nowadays, with the advancement of technologies, the new formats of telling these stories are podcasts, TV shows, new articles, blog posts, interactive discussions, and many other evolving formats.

Primarily there are three objectives of these stories, the first one is entertainment, the second is character building and the third is to give knowledge. For character-building and acquiring knowledge, it is important we interpret these stories correctly. For entertainment, we need to understand the context and meaning of these stories. These stories are part of the school course books and our day-to-day cultural life.

One key challenge after telling the story is to validate what people learned after listening, reading, or watching the story. To validate that the best format is asking questions and comparing the answers. However, generating questions, and answers with references is a very tedious process and expensive process. And, if we have to scale QA work to millions of people then humanly it is impossible but with machine it can be done effortlessly.

1.1 Statement of Problem

The older way of teaching history is through narrating stories. Now audio, video, podcast, and conversation have become new norms. There are two challenges in all these approaches.

1. How do you know the listener has understood the topic?

2. How to make the learning process interactive?

To address the above two challenges we can insert question-answer sessions in the learning process. These questions should be short one or two line answers. We can use these questions during storytelling or post-storytelling. But to introduce a question-answer module we need to have ready-made questions and answers. To create an exhaustive set of questions and answers is a painful process.

In this research, we are exploring AI technologies to generate questions from a given Indian history book. And, generate answers to those questions with the correct reference. While doing this research, we don't have any benchmark database of question-answer on Indian history books. Therefore, we decided to conduct this research with one of the most voluminous historical works, the Mahabharat. The Mahabharat was written many centuries back by Maharishi Vedvyasa. Originally it was in Sanskrit Language and Brahmi script [1]. But nowadays, we can find hundreds of translations of this book in hundreds of world languages and different scripts. For this research purpose, we are using the English language, and Roman script, and considering translation done by Kisari Mohan Ganguli (famously known Ganguly), published between 1883 and 1896. In this work we are maintaining a sharp distinction between language and script. Language is that which is spoken, script is that in which we write words of any language. A language can be English and we can write this in Devanagari script, or language can be Hindi and we can write that in Devanagari script.

The Ganguli English translation of the Mahabharat is available at Ganguli 1896. It is the only complete English translation, copyright free, in text format in the public domain. Mahabharat Books comprises 18 volumes, these volumes are referred also as books. Books 1-4 were proofed at Distributed Proofing (Juliet Sutherland, Project Manager), from page images scanned at sacred-texts.com. Books 5-7 and 12-15 were proofed at sacred-texts.com by John Bruno Hare. Books

[1] There is no definitive answer about the script but most of the historians agree this

8-11 and 16-18 were proofed by Mantra Caitanya.

A good Indian historical textbook, even if they are written in English is full of Sanskrit words (Nouns). These nouns generally are names of people, places, events, festivals, plants, and non-translatable philosophical terms. When you write a translation of Sanskrit work like Ramayana in English it will have lots of spelling mistakes. This you will notice when you listen to a non-native person reading those English stories or you translate that English work into any other Indian language. [2]

We want to do our work on Mahabharat text and at the time of doing this work We are not aware of any dataset of QA on Mahabharat.

1.2 Significance of Study

This research work has many interdisciplinary promises some of those are.

1. Evaluation of human learning from historical books at a minimal or zero cost.
2. Historical text question-answer generation will become time efficient.
3. Expending existing historical books in the form of Question Answer Books.
4. Bringing some insightful uncommon questions is easier otherwise we need human great scholars for this work.
5. History learning will be more interactive and fun.

[2] In NLP, before we ask a machine to do some NLP task we need to create word embedding of the text. For creating word embedding we need an embedding model. For two reasons, 1- Normal Western English doesn't have those nouns, which are part of Indian history work, 2- Because of the missing corresponding alphabets between Indian scripts and Roman scripts, the spelling of words goes wrong in the translation process. Thus we cannot rely on the embedding model which was created using Western English language text. We need a different embedding model. Either create an embedding model with a multi-lingual text corpus or fine-tune the existing embedding model using our own corpus. Otherwise, we will have ”out of vocabulary” (OOV) problem.

6. It can be extended to other fields like earlier time Agriculture, Ayurveda, Mining, Construction, etc.

7. Creating topic-specific, chapter-specific lighter QA models can be used on mobile devices for fun and gaming.

8. Archaeological sites can use the content created by this system.

9. Schools and Colleges can use the content created by this system.

10. Developer can create apps for tour guides to engage with their clients on the historical site.

11. With correct transliteration of nouns, QA set of one language can be used for Translation into other Indian Languages.

In summary, this research holds significance in terms of preserving historical knowledge and making the learning process fun, entertaining, engaging, and insightful. This technological showcasing can be integrated into the humanities, enhancing education, addressing information retrieval challenges, and paving the way for future advancements in AI-powered historical analysis.

1.3 Research Questions

1. What is the cost-effective way of generating high-quality question-answer dataset from a historical text?

2. What is the cost-effective way of generating high-quality answers from historical text-based Questions?

3. What different NLP/NLU/NLG technologies are available for Question Answering tasks in general?

4. Under 30 minutes, is it possible to generate one thousand high-quality questions and answers with reference from translated Mahabharat text using freely available hardware resources?

5. How to create a system that can answer any question from a historical book without any reference QA paired dataset?

6. What are the linguistic nuances of Indian Historical text that impact the quality of the translation of questions and answers between the Indian Language?

1.4 Hypothesis

1. Some methods are more cost-effective than others in generating high-quality questions from a historical text.

2. Some methods are more cost-effective than others in generating high-quality answers from historical text-based questions.

3. There are more than 10 NLP/NLU/NLG technologies in general Question Answering tasks.

4. It is possible to generate one thousand high-quality questions and answers from translated Mahabharat text in under 30 minutes using freely available hardware resources.

5. It is possible to create a system that can answer any question from a historical book, even if that question was not part of QA paired dataset.

6. There are linguistic nuances in Indian historical texts that impact the quality of translation of questions and answers between Indian languages.

1.5 Assumptions

The effectiveness of a Question Answering (QA) system is intrinsically linked to the quality and attributes of the source text from which it is developed. It also depends upon the availability of resources and technologies. To investigate this relationship, we make the following assumptions:

1. The sourcebook used to train the QA system must meet specific criteria, including correctness, accurate translations (if needed), and clarity in sentence and paragraph meaning. It is assumed that these books are easily and freely available.

2. The source text is easily available in either HTML or plain text format, as this work does not involve the processing of content from images, video, audio, or tables.

3. We need not employ Optical Character Recognition (OCR) technology in this study.

4. It is assumed that human experts require more time to comprehend and craft high-quality questions and answers compared to automated machine processes.

5. Cost-Effectiveness of Methods: It is assumed that there are different methods available for generating high-quality questions and answers from historical texts, and some of these methods may be more cost-effective than others.

6. Availability of NLP/NLU/NLG Technologies: It is assumed that there are multiple NLP/NLU/NLG technologies available for general Question Answering tasks.

7. Hardware Resources for Rapid Generation: It is assumed that there are freely available hardware resources capable of generating one thousand high-quality questions and answers from translated Mahabharat text in under 30 minutes if such methods exist.

8. Development of Question-Answering System: It is assumed that it is possible to create a system that can answer any question from a historical book without a reference question-answer dataset, although the methods for achieving this may vary.

9. Linguistic Nuances in Indian Historical Texts: It is assumed that Indian historical texts contain linguistic nuances that can impact the quality of translation of questions and answers between Indian languages, although the specific nature and extent of these nuances may vary.

1.5.1 Limitations

Limitations refer to factors or aspects of research that are constraints or restrictions that could potentially impact the validity, reliability, or generalizability of findings. These are characteristics or conditions that exist but are not under your control as a researcher. There are the following limitations of this work.

1. Availability of Historical Texts: We have used a specific version of an Indian historical textbook. We used Mahabharat translated by GM Ganguly. Due to time and cost constraints, we are picking specific chapters of this work to conduct our research.

2. Quality of ChatGPT Generated Answers: We don't have a benchmark dataset for this work. Creating a human-annotated dataset is time-consuming and costly. In fact, that is the reason we are doing this research. Therefore, we

are using ChatGPT to generate 1000+ questions from selected chapters of the Mahabharat book. We will do random quality checks of questions and answers. We will keep good quality QA pairs for our model training work.

3. We didn't read GM Ganguli English translation of Mahabharat. If there is some wrong translation then the context will be wrong and hence question answer will be wrong. This has nothing to do with QA generation technology.

4. Generalization to Other Languages: We have used only the English translation of the Mahabharat book. We know there are spelling errors when we translate Sanskrit words between different Indian languages.

5. Hardware and Computational Resources: We need GPU machines for this work. At this point in time Kaggle Kernel and Google Colab provides limited capability GPU machines free of cost. We are going to work only with those given resources.

6. ChatGPT is commercial and it is expensive to generate questions, and answers using ChatGPT. To generate better quality questions, we need a larger context window and it is more costly. We are going to use free account services. Our objective is to go away from paid services to generate the best quality question answers.

7. Ethical Considerations: Mahabharat is a historical, religious and political work. There are different translations and different interpretations of the original work. With good intentions, we only want to know the capability of AI for question-answer generation. We don't own the wrong interpretation of the original text.

8. Quality of Ground Truth Data: We rely on answers and references generated by ChatGPT. We are assuming that the answer generated and the reference

provided is correct. We are further expanding our work based on this assumption.

9. Subjectivity in Quality Assessment: If a system is generating 1000 benchmark questions then all questions and answers will not be of the same quality. We are not deploying any sophisticated system to measure the quality of questions and quality of answers. Hence no way to know what is bad quality. Therefore, based on the random sampling if we find some bad question we will remove it. We are not using any other technique to detect an remove bad-quality questions. Although we will use cosine similarity metric but we will not use this to include or exclude QA pairs.

10. Time Constraints: This is self-funded, limited-time, academic research, and the outcome of this research can be improved for commercial work.

11. Dependency on Existing Tools: At the time of conducting this research we notice that there is a flood of NLP/NLU/NLG technologies in the market and it is changing at a very fast pace. We are going to research, use, and build on what is available. We are not developing new architecture, hardware, algorithms for this work.

12. Question Type: We are focused only on generative questions and not doing any work for other type of questions.

13. Reference Time Frame: For historical work, we need to be careful about the timeframe we are taking to generate questions and answers. We are picking one book, which covers a certain timeframe of history. We are not doing a fact-check of this work with other available history books.

14. Data Format: It can work on other data formats, but we built and tested it only on .txt format data.

1.5.2 Delimitations

Delimitations are choices made by the researcher to intentionally narrow the focus or scope of the study. This work has the following delimitations.

1. Text and Language Limitation: We are picking only Mahabharat books and English language translations.

2. Historical Period Limitation: No reference check with other works of the same era. Whatever is written in Mahabharat book's English translation, is the gospel truth for this work.

3. Limited Historical Expertise: We personally have read Mahabharat and understand that works fairly well. We are not using any historical expert.

4. Question and Answer Format: We are using generative descriptive answers and not any other format of QA.

5. Quality Threshold: We have not used any technique to know the quality of questions. However, We will use cosine similarity between questions and answers.

6. Ethical and Cultural Factors: During this work, we have avoided using controversial chapter/sections/topic but readers may still find some controversy in the content generated with the system. We will use few selected chapters for this work.

7. Legal and Copyright Considerations: We own the copyrights of questions-answers generated by the system. We own the copyrights of finetuned models and approach.

8. Evaluation Metrics: We are using metrics like BLEU, ROUGE, Precision, Recall, P@n, R@n, F1@n, Cosine and MRR to access the performance of the

specific aspect of the system.

1.6 Definition of Terms

1. AI-Powered Historical Book Question Answering: Refers to four technologies that generate question answers from historical texts using artificial intelligence techniques. A- technology-driven process of automatically generating question-answer pairs. B- A finetuned model which can answer any question of the historical book when context is supplied along with question. C- Retrieving a right document/section/chapter/page which can answer the question. D- Generating an answer to a question from the book, just with a question but without giving any context to the system.

2. Question Answering System (QAS): A computer-based system that uses natural language processing and machine learning to generate relevant answers to user-provided questions.

3. Text Mining: The process of extracting valuable information and knowledge from large volumes of textual data through computational techniques.

4. Historical Texts: Written materials from the past that provide insights into historical events, cultures, societies, and narratives. In our case we are referring to pre-Islamic history of India. The reasons for that are A- It has huge volume. B- It is in Sanskrit language. C- Translation of this work is available in English and other languages. D- It shapes psyche of Indian.

5. NLP (Natural Language Processing): A branch of artificial intelligence that focuses on enabling computers to understand the text and perform NER, Sentiment analysis, POS tagging.

6. NLG (Natural Language Generation): A branch under NLP which generate text, the generated text may be summary, writing a new story, generating questions and answers from the given text.

7. NLU (Natural Language Understanding): A branch under NLP which is used to understand the meaning of text. In comprehending the meaning of text, it is not only the words but the in what context those words has been used.

8. Deep Learning: A subset of machine learning that employs neural networks with multiple layers to learn complex patterns and representations from data.

9. Fine-Tuning: The process of adapting a pre-trained model to perform specific tasks or domains by updating its parameters on task-specific data.

10. Question Generation: The task of automatically creating questions from a given text or context, often used in educational and information retrieval applications.

11. Knowledge Representation: The process of structuring and encoding information in a way that can be easily understood and processed by computers. Generally we use different embedding techniques and generate floating number vectors for the text.

12. Latent Knowledge Retrieval: The technique of uncovering implicit or hidden knowledge from textual data using information retrieval methods.

13. Historical-Domain: The specific subject area or period of history to which the research and question generation process are focused.

14. Task-Agnostic: Refers to a model or system that is not designed for a specific task and can be applied to various tasks without significant modifications.

15. Contextual Embedding: A technique in NLP where words or phrases are represented as dense vectors that capture their contextual meaning within a given context. It is related to knowledge representation.

16. Question Classification: The process of categorizing questions based on their type, structure, or intent to facilitate accurate answer generation.

17. Data Source: The origin of textual content, such as historical books, documents, or manuscripts, from which the system generates question-answer pairs. This is used as a text corpus.

18. Archival Material: Historical documents, records, manuscripts, or artifacts that are preserved and provide valuable insights into the past. It may be scanned text from manuscripts or uneditable text.

19. Language Paradigm: The linguistic framework, rules, and structures that govern how languages are organized and function.

20. Answer Extraction: The process of identifying and extracting relevant portions of text as answers to user questions.

21. Unsupervised Learning: A machine learning approach where a model learns patterns from unlabeled data without explicit guidance.

22. Cross-Lingual Question Answering: The task of generating answers to questions posed in one language using information from documents in another language.

23. Historical Context: The background information, circumstances, and events surrounding historical texts that contribute to their interpretation.

24. Ethical Guidelines: Established principles and rules that govern the responsible and ethical use of historical content and AI technologies.

25. Inference: The process of deducing or deriving conclusions from given information, often used in question answering to determine correct answers.

26. Benchmarking: The practice of evaluating the performance of a system or model by comparing its results against established standards or datasets.

27. Modularity: The property of a system or model that allows different components or modules to be developed and updated independently.

28. Prolific Research Avenues: Emerging and promising areas of research that are expected to yield significant contributions and advancements in the field.

29. Evaluation Metrics: Quantitative measures used to assess the performance and effectiveness of the AI-powered historical book question-answering system. In NLP work popular metrics are BLEU, ROUGE, METEOR, Recall, Precision, Cosine, R@n, P@n, F1@n.

30. ChatGPT-FE - Using ChatGPT via Front End

31. ChatGPT-API - Using ChatGPT via Application Programming Interface (API). We are using python script for this.

32. ChatPDF - An LLM based application. It has API services.

1.7 Background

1.7.1 Why People Ask Question?

Human is one of the highly evolved intelligent life in the known cosmos. Asking questions is a sign that humans think and human is intelligent. No single human being knows the answer to all the questions. But whatever answer we know, it is because we asked questions in our past. People ask questions for various reasons.

In a blog article 15 Reasons To Ask Questions [3] author states the following reasons for asking questions.

1. To acquire knowledge
2. To eliminate confusion
3. To cause someone else to feel special
4. To guide a conversation in a direction
5. To demonstrate humility to another
6. To enable a person to discover answers for themselves
7. To gain empathy through better understanding another's view
8. To influence/alter someone else's opinion/view
9. To begin a relationship
10. To strengthen a relationship
11. To humbly show we have knowledge on a specific topic
12. To stimulate creativity and idea generation
13. To gain a person's attention
14. To solve a problem
15. To reach agreement or to "agree to disagree" with clarity

[3] https://qbq.com/15-reasons-to-ask-questions

1.7.2 Why Question Answering of Historical Text?

(Bevir 2000) highlights historians might examine how beliefs and texts are established, neglected, and promoted within public discourses and social practices. Historical text is of high interest to almost all people. Whether someone is literate or not. Whether someone has an interest in science or philosophy or not, all have an interest in stories. People want to know about the past of their ancestors, they want to understand the current problem with the history and origin of the problem. They want to understand the past solutions for similar problems. But the issue is the original old text is in a language that most people don’t understand. Therefore, we translate the text into a language that today’s people can read and speak. In the process of text translation of any Indian language, there are two main problems.

> 1- Wrong spellings are used to write the original nouns. Nouns are names of the people, places, festivals, and concepts in the story. And these nouns are part of the culture.
>
> 2- When we do reverse translation or use the translated text to translate into another language, then writing the original noun correctly is not possible.

For example, the original text contains a noun "कर्ण". When we do the translation of this word into English, then we write this noun as ”Karna”. Now, we want to use English translation to translate further into Kannada then we should get noun **ಕರ್ಣ** but we will get **ಕರ್ನ**

1.7.3 AI-NLP Capabilities

Human capabilities around text are the proof of human intelligence. Since the early time of AI development, AI researchers have been trying to make machines that can do those tasks that humans can do with the text. In the recent past, we have added many capabilities. As of today, these capabilities can be summarized as follows.

1. Text Classification and Categorization: It involves tasks like Sentiment Analysis (positive, negative, neutral), Topic Classification, Intent Detection, Document Classification, Spam Detection, sarcasm detection,

2. Named Entity Recognition (NER): This tasks involves Identifying and classifying entities (e.g., names, locations, dates) in text

3. Part-of-Speech or PoS Tagging: It means tagging each word as noun, verb, adjective, adverb etc to words in a sentence.

4. Dependency Parsing: Analyzing grammatical structure and relationships between words in a sentence

5. Machine Translation: Translating text from one language to another

6. Language Generation: It has tasks like Text Summarization, Text Generation (e.g., chatbots, content creation), Language Style Transfer

7. Question Answering (QA): Extractive QA (Answering questions by selecting text spans from a document), Generative QA (Answering questions with coherent and relevant responses).

8. Textual Entailment and Paraphrasing: Determining if one sentence logically follows from another, Generating paraphrased versions of sentences

9. Text Completion and Prediction: Auto-completing sentences or predicting the next word in a sequence

10. Document Summarization: Condensing longer documents into shorter summaries

11. Text Segmentation and Chunking: Dividing text into meaningful segments or chunks

12. Text Similarity and Semantic Search: Measuring semantic similarity between text documents or sentences. Conducting semantic search in large text corpora

13. Coreference Resolution: Identifying when different words or expressions refer to the same entity

14. Sentiment Analysis and Emotion Detection: Determining the sentiment or emotion expressed in a piece of text

15. Text-to-Speech (TTS) Synthesis: Converting written text into spoken language

16. Speech-to-Text (STT) Transcription: Converting spoken language into written text

17. Dialogue Management: Managing multi-turn conversations and interactions with users

18. Language Detection and Identification: Detecting the language of a given text

19. Morphological Analysis: Analyzing the structure and formation of words

20. Multilingual NLP: Handling and processing text in multiple languages

21. Cross-lingual Tasks: Translating, transferring, or adapting NLP models across languages

22. Humor and Sarcasm Detection: Identifying humorous or sarcastic content in text

23. Legal and Medical Text Analysis: Analyzing legal documents or medical records for relevant information

24. Code Generation from Natural Language: Converting natural language descriptions into code (e.g., programming)

25. Fake News Detection: Identifying and classifying fake or misleading news articles

26. Commonsense Reasoning: Inferring and applying common-sense knowledge to text understanding

27. Text Analytics for Social Media and Customer Feedback: Analyzing social media posts, reviews, and customer feedback for insights

1.7.4 Different Kinds of Answers

For example, the question from Mahabharat is ”Who was Yudhishther?” The answer from the book could be more than one. Potential answers are.

1. Yudhishther was king of Hastinapur.

2. Yudhishther was the eldest son of Pandu and Kunti.

3. Yudhishther was the son of Dharamaraj god from Kunti.

4. Yudhishther was the younger brother of Karna.

5. Yudhishther was the eldest husband of Draupadi.

6. Yudhishther was the eldest among Pandavas.

7. The first son of Pandu was Yudhishther.

All answers are correct. Sometimes you can combine these answers and create another answer that also is the correct answer. Sometimes answers are extracted from the context, for example, ”King of Hastinapaur”. Sometimes the answer is

generated from the text based on the context understanding and the exact text is not available in the book. For example, ”Yudhishther was the elder brother of that who killed Duryodhana in Mace fight.”
Sometimes we find the answer to the question is in one or two paragraphs, but it is not a continuous chunk of words from the context. But the keywords are spread across the paragraph(s).

1.7.5 What are the Challenges of Generating Questions and Answers?

There is a flood of SOTA question-answering models available, but the problems are

1. The cost of generating questions automatically from a large corpus is high.

2. The cost of generating answers to questions is high. You need to search all the context documents and then shortlist the potential documents that can have an answer, and then within the selected document look for the correct answer and its position in the context document.

3. Time required to generate the questions from a specific corpus. If the text corpus size is big, and we want to generate, say 5000 questions it is time-consuming work.

4. Using the correct spelling of nouns if questions or answers are being translated from one language to another language

5. The cost of generating answers with the reference, especially when the answer is spread in the context paragraph, is high.

6. Generally, we have one reference answer to compare the generated answer. Generating multiple possible correct answers, as mentioned earlier is expensive.

1.7.6 Time Investment to Create QA

To create good-quality questions and answers from any text you need to understand the text thoroughly and you need to have a deeper understanding of the domain. The time to create question-answers depends upon

1. Complexity of the Content: If the book contains a long chain of relationships and complex or technical information, the history expert may need to spend more time comprehending the material before generating questions and answers. In texts like Mahabharat, we find there are thousands of people and their complex relationships with each other over hundreds of year.

2. Familiarity with the Topic: If the history expert is already well-versed in the topic, they might be able to generate questions and answers more quickly compared to a topic that is new to them.

3. Writing Speed: The speed at which the history expert can write and formulate questions and answers will influence the overall time required. Nowadays, because of STT (Speech to Text) technology, we can ignore this factor in languages like English.

4. Research and Verification: Depending on the level of detail required for accurate answers, the history expert may need to verify facts, dates, and events, which could add to the time.

5. Cognitive Load: Generating a large number of questions and answers can be mentally taxing, and the history expert's cognitive load may affect their

speed and accuracy.

There is no benchmark about how much time and money it takes to create questions but we can guess it. We know the average reading speed of a human is 200 words per minute. One page has an average of 500 words. So you need approx 2 min to read a page. To read a book of 200 pages you need approx 400 min. But it is difficult for any normal human being to read continuously for 400 min without deteriorating the speed and focus. To draft 200 questions and answers from this 200-page book you need almost four times[4] of the reading time i.e. 800 minutes, It includes the second reading, creating the question, extracting answers, typing, plus validating answers. Thus we need 1200 minutes time to write 200 questions. But this 1200 min is not 1200/60 = 20 hours. Being a creating work, it is going to be approx 5 days (one week). So need to wait for one week, and pay the compensation of one week to an expert to generate 200 questions and answers. Let's assume an expert is hired in India at INR 1,00,000/ month. We need to pay 25000 to generate 200 questions. This is just a theoretical, no human, even in India can generate 200 QA pairs from a 200-page book at this cost. If we use AI to generate these questions then we do the same work with human-level quality within a few minutes and with a few hundred rupees. Those human experts can be used to review this work or for any value addition.

1.8 Structure of the Study

The structure of the study is as follows.

2.2 Type of Questions

2.3 Evolution of Question Answering Systems

[4] I didn't find any benchmark study for this but I personally have been involved in QA systems development for a long time therefore mentioning based on my personal experience

Chapter 2

LITERATURE REVIEW

2.1 Introduction

As of 2023, there are around 214 NLP Tasks from BIG Benchmark [1].

Question-answer generation is one of the very complex tasks in NLP from this list. Chatbot System, Question Answering System (QAS), and Conversational AI (CAI) are related terms. Over the period of time when the industry needed a more complex system researchers kept refining the name. In the evolution process, Chatbot is a primitive name, followed by QAS, and finally conversational AI systems. Chatbots are CAI, but some of them are rule-based. CAIs are not rule-based. FAQ systems deal with a fixed set of questions and answers. From the text corpus, we can manually pick the most frequently asked questions and find the answer (either manually or using AI). We store this question-answer pair and then based on the need answer the question with a standard answer from this standard FAQ dataset. Like chatbots, FAQs can be interactive or you can publish them on a webpage or in a PDF file.

FAQ is not effective in

[1] 214 NLP Tasks from BIG Benchmark.

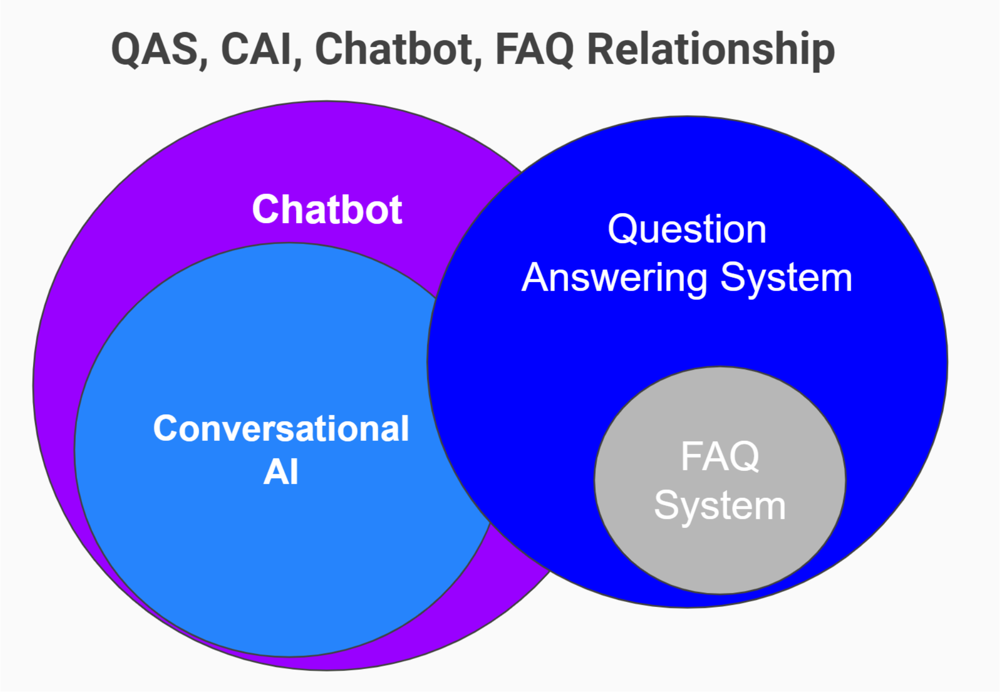

QAS, CAI, Chatbot, FAQ Relationship
Chatbot
Conversational AI
Question Answering System
FAQ System

Figure 2.1: QAS, CAI, Chatbot, FAQ Relationship, Source: Author's own illustration

A. Live dataset,

B. When you don't know what question can be asked, and

C. When the number of questions cannot be determined,

D. When a dataset is huge.

In computer systems, NLP (natural language processing) deals with human language. Analyzing text, predicting the next word, sentiment analysis, and translation can be done using a statistical approach or a rule-based approach. But it faces problems like (a) analyzing large volumes of data and discovering all the rule is not humanly possible. (b) With more volume or complicated text rules become complex. (c) It is not practical to write those all rules and change them whenever we analyze some new rule in the text. Therefore NLP also needs a machine learning approach where models are learned from the data during training.

Interactive QAS is one where a dialogue interface enables follow-up and clarification questions, (Quarteroni et al. 2007). The goal of a chatbot, the QAS and CAI is the same. They all want to answer user's questions and engage users in such a way that the learning journey or exploration is interesting. In the chatbot system & CAI, humans interact with machines. But in a questions-answering system, a consumer of information may be human or another system or process. When a chatbot or CAI needs to answer complex questions or deal with huge data it needs the help of QAS. Therefore, to build a CAI we need a sophisticated question-answering system and for that, we need a large domain-related corpus. Chatbot's primary focus is to understand the intent of the question and extract the question's answer from the knowledge base or generate that answer from the knowledge base.

Unlike search engines, QA systems focus on delivering specific requested information rather than entire documents. With the proliferation of information, obtaining precise fragments of information even for simple queries has become resource-intensive. In their work "Question answering system on education acts using NLP techniques", (Lende & Raghuwanshi 2016*b*), outlines diverse methodologies and implementation specifics for a general language QA system, and introduces a closed domain QA system tailored for educational acts sections.

We consider Google Search, Wolfram Alpha, IBM Watson as question-answer or search tools. And Amazon Alexa, Apple Siri, Google Assistant are considered as chatbot. A recent add in this category are ChatGPT, GPT4, and Bard. As AI technology continues to evolve, the lines between QAS, chatbots, and CAI are blurring.

(Palasundram et al. 2021) in their work "SEQ2SEQ++: A Multitasking-Based Seq2seq Model to Generate Meaningful and Relevant Answers" mentions Question Answering Systems are implemented using either rule-based or machine learning-based systems. Machine learning-based chatbots are more scalable. Sequence-to-sequence (Seq2Seq) learning is one of the machine learning-based approaches. It has shown remarkable progress in recent times. However, chatbots based on Seq2Seq learning show a weakness in that. It tends to generate answers that can be generic and inconsistent with the questions, thereby becoming meaningless and, therefore, may lower the chatbot adoption rate. This weakness can be attributed to three issues: a) question encoder overfit, b) answer generation overfit, and c) language model influence. Several recent methods utilize multitask learning (MTL) to address this weakness. However, the existing MTL models show very little improvement over single-task learning.

2.1.1 The Purpose of Question

1. Evaluation Questions: Teachers use evaluation questions to help students. Supervisors use questions to evaluate new team members' knowledge, to make value judgments, or to anticipate future events or outcomes when their leaders do not provide them the complete information. These questions require information organization and analysis.

2. Interview Questions: These questions are used in interviews to gather information about a person's background, skills, experiences, or opinions.

3. Survey Questions: Survey questions are used in research and data collection to gather specific information from respondents.

2.2 Type of Questions

”A framework for restricted domain question answering system using advanced NLP tools and software” by (Rami & Pawar 2017) is important work in NLP. They mention that the main challenge of the QAS is to give a correct answer to a question to the user. As per (Rami & Pawar 2017) QAS can be classified into three categories, they are (a) open domain, (b) closed domain, and (c) restricted domain. Before we build QAS we can look at the existing question-answering setup from a different perspective. As per our analysis, we can classify QAS in the following different ways. In reality, at the time of building a question-answering system, all these come together and make business problems more complex.

2.2.1 Classification Based on the Knowledge Required

Based on the input needed to answer the question, we can have the following categories.

1. Historical/Current-Affair Based Question: The answerer needs to be aware of historical facts and current affairs to answer these types of questions.
   (a) Who is the current Prime Minister of India?
   (b) Where is the headquarter of the United Nation?
   (c) Who was the prime minister of India during the Indo-China war?
2. Commonsense Question:
   (a) If you were facing the sun and you turned left which direction you are facing now?
   (b) If I refer to X as my son then what will X call me?
3. Awareness-based Question: The answerer needs to be aware of the surroundings and self to answer these types of questions.
   (a) What is the time right now?
   (b) Where are you right now?

   (c) What is your phone number?

   (d) Do you know, you are on the 65th floor at this time?

4. Formula-Based Question: The answerer needs to use some physics, mathematics, statistics formulas, or business rules to answer these types of questions.

   (a) If I am coming to meet you and your house is 50 km away from my current location. My car is moving at an average speed of 40 km/hour. How many minutes it will take me to reach you?

   (b) I am 38 years old, I live in Mumbai, and want to buy ABC health insurance, How much premium do I need to pay?

5. Guess-Based Question: The answerer needs to be aware of the surroundings, postulate some hypothesis, and then calculate or count to answer the question.

   (a) How many crows are there in your city?

   (b) Next week Tuesday there is a political rally for Narendra Modi in Nagpur, How many people will join the rally?

6. Subject Specific Question:

   (a) A patient is having some symptoms and his blood report is before the doctor, doctor can tell the answer, what is the disease, and what treatment is needed, etc.

   (b) A red rose has white spots on its leaves what disease is that and what is the treatment?

   (c) If we keep increasing the temperature of the liquid within an airtight closed steel container, how long we can keep increasing the temperature? How much temperature it can bear?

If you think deeper you may realize there are other categories of knowledge-based questions. But we need to keep in mind that to answer these questions an answerer needs a deeper understanding of different types of knowledge bases, and the ability to think and analyze. Before we build any QA system we need to keep this in mind.

2.2.2 Classification Based on Scope

(Nimavat 2010) in their work ”chatbots-An overview Types, Architecture, Tools and Future Possibilities” mentions QAS can be of two types as below.

1. Closed Domain: Such QASs are focused on a particular knowledge domain and might fail to respond to other questions. Sales domain QASs won’t be able to reply to finance domain questions.

2. Open Domain: Such bots can talk about general topics and respond appropriately. They are primarily built for engagement purposes.

2.2.3 Classification Based on the Input Modality

1. Text: QA systems can accept only text as input and respond in text format.

2. Image and Text: QA systems can accept image context and text questions as input and generate text-based answers.

3. Tabular Data: QA systems can accept structured data in the form of tables or CSV files, along with text questions, and produce text-based answers.

4. Speech: QA systems can accept audio recordings (context) and text questions as input and generate text-based answers.

5. Multi-Modal: QA systems can handle multiple types of input as context, including text, tables, images, and voice, along with text questions, and generate text-based outputs.

2.2.4 Classification Based on the Input Language

Below D denotes the main document or context document. Q denotes question. A denotes the answer. 1 denotes language 1, 2 denotes language 2, and 3 denotes language 3.

1. D1+Q1+A1 One Language: Takes the main document in one language say English, questions in the same language, and generates output in the same language.

2. D1+Q1+A2 Two Language-1: Takes the main document and question in one language say English, and generates output in a different than the input language.

3. D1+Q2+A2 Two Language-2: Takes the main document in one language say English. Questions and answers are in another language say Hindi.

4. D1+Q2+A3 Three Language: Takes the main document in language 1 say Tamil, question in second language say English, and generates output in a third language say Hindi.

2.2.5 Classification Based on the Input-Output Length

1. Long question One-word answer: The question along with context may be in one paragraph, approximately 200 words. The answer is one or two words.

2. Short question long answer: The question is short, say 10 words long. The answer is one paragraph.

3. Short question short answer: The question is short, say 10 words long. The answer is one or two words.

2.2.6 Classification Based on the Approach Used for Answering

1. Calculated Output: An answer needs to be calculated using the input context and question.

2. Extracted Output: An answer needs to be extracted from the input context.

3. Generated Output: An answer needs to be generated, by searching different sections of the context document, and stitching those words (sometimes different words but semantically same) together.

4. FAQ: Searching paired question answer bank to answer a question.

2.2.7 Classification Based on the Domain

1. Pure One Business Domain: Question - Answer from the document of only one domain say Banking or Insurance or Pharma.

2. Pure One Business Domain with Simple Answer: Question from the document of only one domain say Banking or Insurance or Pharma or Legal but the answer should not be in that domain's technical jargon. The answer should be generated in a language that a common person outside of that domain can understand. For example, a common citizen can ask a question from a document ”Constitution of India” but the answer should be in simple English.

3. Multiple Domains: Two or more than two domain's documents are input. Questions can be mixed from these domains. To get the answer all the domains need to be searched.

2.2.8 Classification Based on the Goal

1. Providing factual information
2. Generating a humorous or entertaining answer
3. Engaging users for a short or long conversation
4. Telling answers through generated stories
5. Showing empathy
6. Solving a specific relationship or business problem

You can add many goals based on the need or problem you have before you.

2.2.9 Classification Based on the Question Style

Based on the Question Style, purpose of the question, environment where the question can be asked, and length of answer we can classify the question as below. An Indeed.com blog article [2] discusses 15 types of questions on their blog. Another blog article at dasarpAI[3] discusses 24 types of questions. These classifications help us understand the challenges in creating QAS. As per dasarpAI blog 24 question types are as follows. When we create a question it may have multiple characteristics from the types mentioned below.

1. Closed-ended Questions (Multi Choice Questions - MCQ): These questions have a limited set of predefined answer options, often requiring a specific choice from a list.
2. Closed-ended - Yes/No Questions: These questions are designed to elicit a binary response of ”yes” or ”no.”

[2] An Indeed.com blog article

[3] dasarpAI : Types of Questions

3. Open-ended Questions: Open-ended questions allow for a wide range of responses and encourage more detailed and expressive answers.

4. Factual Questions: These questions seek specific, verifiable information. The answers are based on concrete facts.

5. Descriptive Questions: Descriptive questions ask for descriptions, explanations, or details about a topic or situation.

6. Comparative Questions: Comparative questions involve comparing two or more items, entities, or concepts to identify similarities and differences.

7. Cause-and-Effect Questions: Cause-and-effect questions inquire about the reasons behind events or actions and their outcomes.

8. Inferential Questions: Inferential questions require respondents to draw conclusions or make inferences based on the provided information.

9. Opinion-based Questions: Opinion-based questions solicit subjective viewpoints, beliefs, or preferences from respondents.

10. Hypothetical Questions: Hypothetical questions ask respondents to consider imaginary scenarios and provide answers based on those scenarios.

11. Rhetorical Questions: Rhetorical questions are posed for rhetorical effect and do not necessarily require an answer. They often emphasize a point or provoke thought.

12. Wh-Questions or Information Questions: Wh-questions typically start with question words like ”who,” ”what,” ”when,” ”where,” ”why,” and ”how.” They seek specific details or explanations.

13. Tag Questions: Tag questions are short questions added at the end of a statement to confirm or seek agreement (e.g., ”It’s a nice day, isn’t it?”).

14. Direct Questions: Direct questions are straightforward inquiries that directly ask for information or clarification.

15. Indirect Questions: Indirect questions are phrased as statements but are used to ask for information indirectly (e.g., ”I wonder where the park is?”).

16. Interview Questions: These questions are used in interviews to gather information about a person’s background, skills, experiences, or opinions.

17. Explanatory Questions: Explanatory questions are posed to seek further clarification or elaboration on a previously provided response.

18. Clarifying Questions: Clarifying questions seek additional details to better understand a situation, statement, or concept.

19. Leading Questions: Leading questions subtly guide respondents toward a particular answer, often by presenting a biased or persuasive viewpoint.

20. Probing Questions: Probing questions are used to delve deeper into a topic, encouraging respondents to provide more extensive answers.

21. Divergent Questions: Divergent questions have no right or wrong answers but rather encourage open discussion. While they are similar to open questions, divergent questions differ in that they invite the listener to share an opinion, especially one that relates to future possibilities.

22. Application Questions: Application questions ask learners, students, or new employees to apply an idea or principle in a new context to demonstrate higher-level learning.

23. Survey Questions: Survey questions are used in research and data collection to gather specific information from respondents.

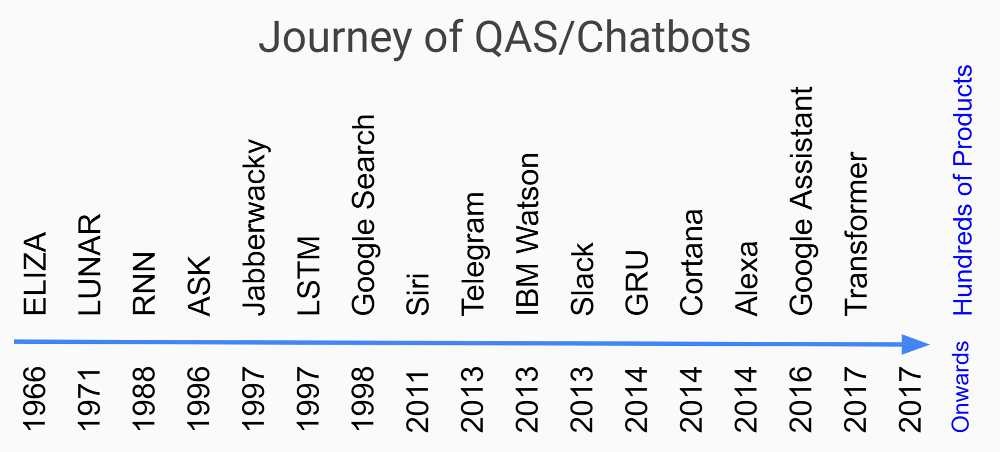


Figure 2.2: Evolution of Question Answering System, Source: Author's own illustration

24. Affective Questions: Affective questions seek to learn how others feel about the information they're learning. These responses can help the speaker affirm the listener's feelings or clarify information.

2.3 Evolution of Question Answering Systems

The evolution of QAS is not only about developing a chatbot product, discovering a new approach, using other technology to solve QAS problems, creating a new algorithm, or integrating QAS in a completely different way, it is a mix of all. We know the evolution of human life is around the evolution of our questions. What different questions, how deep those questions were, how important those questions were, they brought us here, where we stand as a civilization. Similarly in AI, the evolution of QAS is not different from the evolution of AI. They both are tightly integrated. From the figure below you will notice there are several key milestones that brought us here, where we are. Among thousands of other important work done by thousands of researchers, and corporations these milestones are nothing less than a sum of leaps.

The history of AI-based question-answering systems (QAS) can be traced back to the early 1960s, when researchers began to explore the use of computers to answer questions posed in natural language. One of the earliest QA systems was ELIZA, developed by (Weizenbaum 1966) at MIT. ELIZA was a rule-based very basic Rogerian psychotherapist chatbot system that simulated a Rogerian therapist, and it was able to hold conversations with users about their problems.

In the 1970s, there was a growing interest in QA systems, and a number of new systems were developed. One of the most notable systems was LUNAR, developed by (Winograd 1971) at Stanford University. LUNAR was able to answer questions about lunar geology, and it used a more sophisticated understanding of natural language than ELIZA.

In the 1990s, the development of QA systems was accelerated by the rise of the internet. This was because the internet provided a vast amount of text data that could be used to train QA systems. One of the most successful QA systems of the 1990s was Ask Jeeves, which used a statistical approach to answer questions. Ask was developed by (Gruener & Warthen 1996)

In 1997, (Jabberwacky 1997) in their work ”Jabberwacky AI” mentions while speaking to you their QAS uses just that learned material, and borrows a little bit of your intelligence as it learns more. With no hard-coded rules, it relies entirely on the principles of feedback. This is very different from the majority of chatbots, which are rule-bound and finite. If you speak a foreign language it will learn it, and respond appropriately if it has enough to go on. It can be taught slang English, word games, jokes, and any other form of identifiable language trait.

In the 2000s, the development of QA systems was further accelerated by the development of deep learning techniques. Deep learning techniques allow QA systems to learn from large amounts of data without being explicitly programmed with rules. This has led to a new generation of QA systems that are more accurate and capable than ever before.

Some of the most advanced AI-based QA systems today include Google Search, Microsoft Bing, and Amazon Alexa. These systems are able to answer a wide range of questions, including factual questions, open-ended questions, and even hypothetical questions.

As of 2023, there are following popular QAS, chat systems, and CAI around us. Some of these are commercial, some are open source.

1. Aivo: Multilingual bots
2. Chatfuel: Easy-to-use bot for your Facebook Messenger and Instagram customers.
3. Dialogflow: uses NLP to create conversational interfaces for websites, apps, IoT devices, and messaging platforms. It is also optimized for Google Assistant for a better user experience.
4. Drift chatbot: It offers a live chat, chatbots, and email marketing solution, as well as a video communication tool. You can create multiple inboxes, add internal notes to conversations, and use saved replies for frequently asked questions.
5. Engatic: It supports analytics, uses machine learning, and integrates with third-party tools for CRM, marketing, and more.
6. Genesys DX: AI chatbot platform that assists the sales and customer support teams. It will give you insights into your customers, their past interactions, orders, etc., so you can make better-informed decisions. The bot also pinpoints areas for improvement and optimization. chatbot platform comes with an SDK tool to put chats on iOS and Android apps.
7. Imperson: It supports text, audio, video, VR, and AR. This can help you create unique experiences for the client and achieve their goals.

8. MobileMonkey: This is an omnichannel platform that lets you integrate SMS messaging, Instagram, Facebook, and Webchat for better customer service.

9. Octane AI: Its Product Recommendation Quiz is used by Shopify on the official Shopify Hardware store. It is also GDPR & CCPA compliant to ensure you provide visitors with a choice in their data collection.

10. Pandorabots: You can publish your bot on the web and on mobile apps. You can also publish it on messaging channels, such as LINE, Slack, WhatsApp, and Telegram. So, you can add it to your preferred portal to communicate with clients effectively.

11. Tidio: Tidio offers a live chat solution with chatbots to help you improve customer support.

12. Wit.ai: This chatbot development platform is open source, and you can use it for much more than bot creation. You can use Wit.ai on any app or device to take natural language input from users and turn it into a command.

13. Telegram: Telegram was developed by (Durov & Durov 2013) in 2013.

14. IBM Watson: In 2013 IBM Watson is developed by (Ferrucci 2013) under DeepQA project.

15. Slack: Slack was developed by (Butterfield et al. 2013) is 2013

16. Bard: Bard is Conversational AI developed by Google. It was launched in March 2023. Initially, it was powered by LaMDA, and by May’2023, it upgraded to PaLM 2, which significantly enhanced its performance.

17. ChatGPT: ChatGPT is a conversational AI, developed by OpenAI and launched Nov’2022. It enables users to refine and steer a conversation towards a desired length, format, style, level of detail, and language used.

Thus we notice these QAS are different from each other in some sense or other and offer different benefits. They are different from each other on parameters like

A. Channels of communication supported in terms of getting information and giving what a consumer is requesting,

B. It works on live data or existing knowledge base,

C. Simultaneously how many languages it can engage in,

D. Modality they can support,

E. Ethics/compliance/government rules supported,

F. Business domain for which they are made,

G. Department they can support,

H. Creating a new knowledge base while conversation,

I. SDK and API support,

J. Device supported, and

K. OS supported.

2.4 Inclusion Criteria

There are many dozens of AI tasks. There have been many datasets and research papers around these tasks in the last decade. We are considering only those research papers, blog articles, and books that help to understand and go deeper around question-answering tasks. Although We are exploring only those technology papers which are written between 2017 and today and using Transformer and GAN architectures, if something insightful comes out from older papers or even non-technology papers we may refer to those to understand the problem and various solutions around.

However, we are interested in Historical Book question-answering from the perspective of correct transliteration of nouns. But in this work, we consider English translated version of a Historical book called ”The Mahabharat”. For this purpose, we are using a widely available, well-respected, old translation of Mahabharat called (Ganguli 1896) translation.

We will use Mendeley and Google Scholar to fetch the papers and refer only to those papers that are freely available for reading. Most of these papers are published at ACM Digital Library, Springer, Elsevier, aclweb (Association of Computational Linguistics), arXiv, IEEE Xplore, and some papers are referred from other well-respected publishers.

We are not restricting ourselves to any particular methodology, approach, or tools. Whatever different approach researchers have adopted in question-answering tasks for text analysis, text cleaning, question generation, answer generation, and model evaluation we will look into those. Our work is not having any benchmark database so we need to create one and evaluate the quality of generated questions and generated answers. We are assuming the ChatGPT tool used to create a benchmark dataset will produce good results.

To evaluate the model performance we will use metrics like the BLEU (Bilingual Evaluation Understudy) Score, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score, Cosine, Accuracy, Recall, R@n, and P@n.

2.5 Transformers for NLP Tasks

Transformers are complex architecture. In 2017, in their paper ”Attention is all you need” (Vaswani et al. 2017) introduced the term ”Transformer” in the machine learning world. This architecture process and understand input sequences in a different way. The term ”Transformer” signified a departure from traditional sequential models and introduced a more parallelizable and scalable architecture. The core innovation of the transformer model was the self-attention mechanism, which enabled the model to focus on the different parts of the input sequence while processing each token. There are many components and sub-components of this architecture. Based on the task in hand we can put those components together and we can create a new transformer. Examples of transformers are BERT, CTRL, GPT, T5, etc. Important components of transformer architecture are as follows.

1. Input Embedding: This component converts input tokens (e.g., words or subwords) into numerical vectors. It often includes learned embeddings and positional encodings to represent the tokens in a continuous vector space.

2. Position Encoding: Position encoding is a method to represent the position or order of tokens in a sequence using fixed numerical values.

3. Position Embedding: Position embedding, on the other hand, refers to the learned representations of token positions within a sequence. Unlike position encoding, which uses fixed mathematical functions, position embedding is part of the model’s learnable parameters.

4. Self Attention & Multihead Self Attention: Self-attention mechanisms allow the model to weigh the importance of different parts of the input sequence when processing a particular token. Multi-head attention computes multiple sets of attention weights to capture various relationships within the sequence.

5. Position-wise Feedforward Network: After attention layers, position-wise feedforward networks are applied to each token’s representation

independently. These networks typically consist of one or more fully connected layers with non-linear activation functions.

6. Residual Connection and Layer Normalization: Residual connections, often combined with layer normalization, help mitigate the vanishing gradient problem during training and enable more efficient training of deep networks.

7. Encoder Layers (for Encoder Transformers): Encoder layers are stacked to process and encode the input sequence. Each layer typically includes self-attention mechanisms and feedforward networks.

8. Decoder Layers (for Encoder-Decoder Transformers): Decoder layers are used in sequence-to-sequence tasks and include self-attention mechanisms and feedforward networks. They also connect to the encoder's output to incorporate context information.

9. Masking (for Autoregressive Models): Causal masking is applied to ensure that each token prediction depends only on previously generated tokens, maintaining a causal order in the output.

10. Contextual Output: The final output from the transformer is a contextualized representation of the input sequence. This can be used for various downstream tasks, depending on the model's purpose.

11. Output Heads: Depending on the task, transformers can have multiple output heads, each tailored to a specific prediction or classification task. For example, a language model may have a classification head for sentiment analysis and a language generation head for text generation.

12. Initialization Parameters

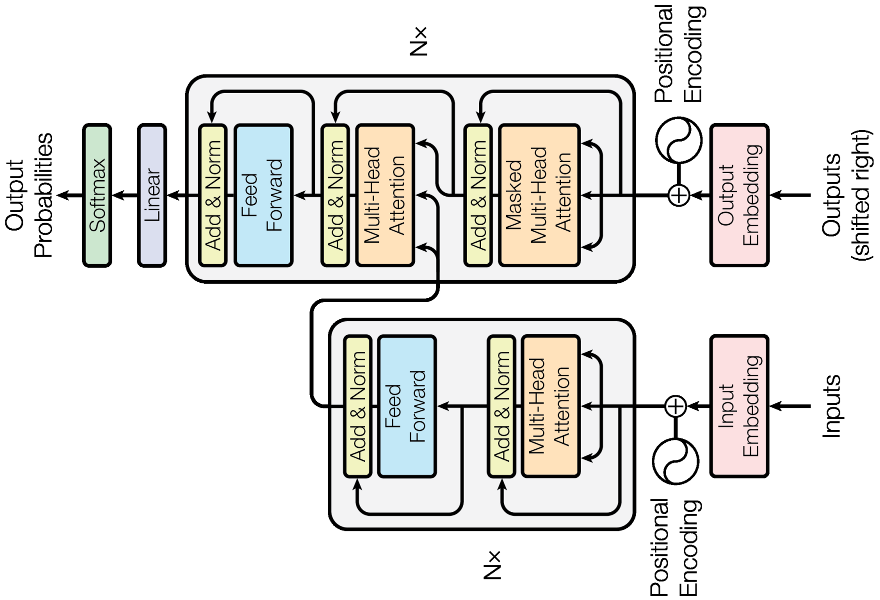


Figure 2.3: Transformer Architecture Source: Vaswani et al. (2017)

13. Loss Function

14. Optimization Algorithm

Using these components we can create different classes of transformers. Broadly there are the following classes of transformers.

Transformer Type	Goal	Training Data	Common Tasks	Examples
Encoder-only	To encode the input sequence into a numerical representation that captures its meaning and context	Labeled or unlabeled sequences	Language understanding, text classification, sentiment analysis, question answering	BERT, ALBERT, RoBERTa

Autoencoder	An Encoder type transformer. To reconstruct the input sequence from a compressed representation	Unlabeled sequences	Dimensionality reduction, anomaly detection, denoising	Autoencoding Variational Inference for Topic Models (AVITM), VAE, AE
Autoregressive or Decoder-only	A decoder type transfromer. To generate the next token in the output sequence based on the previous tokens and the input sequence	Labeled sequences	Text generation, language modeling, machine translation, text summarization	GPT, GPT-2, GPT-3, XLNet, ChatGPT
Encoder-decoder	To encode the input sequence and decode it into an output sequence that may have a different length or vocabulary	Labeled sequences	Machine translation, text summarization, image captioning, speech recognition	T5, BART, Transformer (original)

Table 2.1: Transformer Types, Source: Author's own illustration

2.6 SOTA Transformers for QAS

Here we are discussing some state-of-the-art (SOTA) transformers, that produced outstanding results on question-answering tasks. Depending upon text language, text length, type of question, business domain, and source of the data for the dataset these architectures, created by different researchers, have given the best results. We are discussing these in the chronology of their development.

A research paper ”Pre-training of Deep Bidirectional Transformers for Language Understanding” by Devlin et al. (2019) introduces a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is a deep bidirectional transformer model that is pre-trained on a massive dataset of unlabeled text. This pre-training allows BERT to learn to represent words and phrases in a way that is aware of their context, both left and right. BERT has been shown to be very effective for a variety of natural language processing tasks, including question-answering. In the paper, the authors show that BERT outperforms previous language models on the SQuAD question-answering benchmark by a significant margin. SQuAD is a dataset of question-answer pairs that are derived from Wikipedia articles.

(Raffel et al. 2019) in their work ”Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” introduced a unified framework for text processing tasks called the ”Text-to-Text Transformer” (T5). T5 is able to handle a wide range of NLP tasks using unified architecture. T5 is pretrained on a massive corpus of text data in a self-supervised manner. During pretraining, it learns to predict the target text (output) from the source text (input). This unsupervised training helps the model learn language understanding and generation abilities. T5 can be fine-tuned on specific downstream tasks. Fine-tuning involves providing labeled data for the target task, such as question-answering pairs for a question-answering task. The model’s parameters are adjusted to perform well on the task-specific data. T5’s performance is evaluated based on standard metrics specific to the task, such as accuracy for classification tasks, BLEU score for translation, or F1 score for question answering.

On the other hand, FLAN-T5 is an extension of T5 that has been fine-tuned on more than 1000 additional tasks, covering multiple languages. FLAN-T5 is designed to be better than T5 in terms of performance and versatility. It can be fine-tuned to answer questions in a conversational manner, making it ideal for customer service and support. FLAN-T5 can also be used for machine translation, making it suitable for multilingual content creation and localization. If you are looking for a powerful and versatile model for question-answering tasks, both T5 and FLAN-T5 are good choices. (Longpre et al. 2023) mentions, Flan-T5 requires less finetuning to converge higher and faster than T5 on single downstream tasks-motivating instruction-tuned models as more computationally efficient starting checkpoints for new tasks.

”ERNIE: Enhanced Representation through Knowledge Integration” focuses on knowledge integration. ERNIE by (Sun et al. n.d.) is a novel representation learning model designed to improve the performance of question-answering tasks and other natural language understanding tasks. ERNIE leverages external knowledge sources, such as knowledge graphs and text corpora, to enhance its understanding of language. This integration of external knowledge helps ERNIE better comprehend and answer questions that require world knowledge. ERNIE introduced an enhanced version called Knowledge Masked Language Modeling (KMLM). KMLM involves masking not only words but also entities, phrases, or concepts related to external knowledge sources. This allows ERNIE to acquire knowledge and reasoning abilities during pretraining. ERNIE can be fine-tuned on specific downstream tasks, such as question answering. Fine-tuning adapts the model to the target task by providing labeled data and task-specific objectives. ERNIE is evaluated using accuracy, F1 score, or BLEU score, depending on the specific task requirements. ERNIE achieves competitive results in question-answering tasks, particularly those that demand a deep understanding of world knowledge and context.

(Keskar et al. 2019) introduces a conditional language model (CTRL), in their work ”CTRL: A Conditional Transformer Language Model for Controllable Generation” This focuses on controllable text generation. While CTRL is not primarily designed for question answering, it provides a flexible framework that can be used to generate answers to questions and tailor them to specific requirements. CTRL introduces a conditional generation setup, allowing users to specify control codes to influence the generated text's style, content, and attributes. For question answering, this means that you can condition the model to generate answers in a particular format, style, or tone. You can use control codes to instruct the model to provide concise answers, elaborate on answers, or follow a specific writing style. The model can be fine-tuned on specific datasets or tasks, including question-answering datasets. Fine-tuning adapts CTRL to generate answers that align with the desired task requirements, such as providing accurate and relevant responses to questions. CTRL's performance in question answering or other text generation tasks can be evaluated using standard metrics like accuracy, F1 score, BLEU score, or human evaluations, depending on the specific task and goals.

DistilBERT is introduced by Sanh et al. (2019) in ”DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter”. DistilBERT is created through a process called knowledge distillation, where a larger, pre-trained model (in this case, the original BERT) is used to teach a smaller model (DistilBERT). This process helps transfer the knowledge from the large model to the smaller one. DistilBERT is significantly smaller than the original BERT model, with fewer layers and parameters. This reduction in size makes it more lightweight and faster to train and deploy, making it suitable for resource-constrained environments. Despite its reduced size, DistilBERT is designed to retain much of the performance of the larger BERT model, particularly in tasks like question answering. It achieves this by optimizing the knowledge transfer process during distillation. DistilBERT can be fine-tuned on specific question-answering datasets. Fine-tuning adapts the model to the specific requirements of the task, such as providing accurate and contextually relevant answers to questions. f1 score and accuracy are used to measure the performance of this model. From a question-answering perspective, DistilBERT offers a balance between model size and performance, making it a cost-effective and efficient choice for question-answering tasks, especially in scenarios where computational resources are limited or efficiency is a priority.

Liu et al. (2019) in their paper ”RoBERTa: A Robustly Optimized BERT Pretraining Approach” introduced a highly optimized variant of the BERT model, with a focus on its application to question answering and other natural language processing tasks. like BERT, RoBERTa undergoes a pretraining phase where it learns to predict masked words in a large corpus of text. This pretraining helps the model capture general language understanding and contextual information. It uses larger batch sizes, more training data, and longer training times, resulting in improved model performance. These optimizations enhance RoBERTa's ability to understand and generate natural language text. Standard metrics like accuracy and F1 score are used to evaluate RoBERTa. From a question-answering perspective, RoBERTa offers enhanced language understanding and performance, making it a robust choice for tasks that require accurate and contextually relevant answers to questions. Its optimizations in data collection, training, and hyperparameter tuning contribute to its superior performance in various NLP benchmarks.

”ALBERT: A Lite BERT for Self-supervised Learning of Language Representations” introduces a more efficient and compact variant of the BERT model for self-supervised learning of language representations. Lan et al. (2019) in their work ALBERT achieves model efficiency by using parameter reduction techniques, such as shared parameter factors and cross-layer parameter sharing. These techniques significantly reduce the number of parameters compared to BERT while maintaining model performance. Like BERT, ALBERT is pretrained in a self-supervised manner on a large corpus of text data. During pretraining, it learns to predict masked words in sentences, allowing it to capture contextual information and semantic understanding. ALBERT can be fine-tuned on specific downstream tasks, such as question answering. Fine-tuning adapts the model to the target task by providing labeled data and task-specific objectives. ALBERT offers a balance between model size and performance, making it a valuable choice for tasks that require accurate and contextually relevant answers to questions. Its parameter reduction techniques enable it to achieve competitive results with significantly fewer parameters, improving efficiency and resource utilization.

”Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism” discusses the training of extremely large language models using a technique called model parallelism. While the paper doesn’t focus specifically on question answering, Megatron-LM has the potential to enhance question-answering models. Shoeybi et al. (n.d.) in this work ”Megratron-LM” pushes the boundaries of model size, enabling the creation of extremely large language models. For question answering, this can lead to improved performance by capturing more nuanced language patterns and domain-specific knowledge. The paper emphasizes the use of model parallelism, which divides the model into smaller components that can be trained on multiple GPUs. This technique enables the training of exceptionally large models that wouldn’t fit on a single GPU, potentially leading to better question-answering models with increased capacity for understanding and generating answers.

A research paper ”Transformer-XL: Attentive Language Models for Long Sequences” by Dai et al. (2020) introduces a new language model called Transformer-XL, which is an extension of the Transformer model.

Transformer-XL is able to learn long-range dependencies between words in a sequence, which makes it well-suited for tasks such as question answering. In this paper, the authors show that Transformer-XL outperforms previous language models on a variety of question-answering benchmarks, including SQuAD and TriviaQA. SQuAD is a dataset of question-answer pairs that are derived from Wikipedia articles, while TriviaQA is a dataset of questions that are answered by facts from Wikipedia. Transformer-XL, is able to learn long-range dependencies between words in a sequence.

”BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension” is presented by Lewis, Liu, Goyal, Ghazvininejad, Mohamed, Levy, Stoyanov & Zettlemoyer (2020). BART is a sequence-to-sequence pre-training model, which means it learns to map input sequences to output sequences. In the context of question answering, it can be fine-tuned to take a question as input and generate a corresponding answer as the output. BART is trained using a denoising objective, where portions of the input text are masked or corrupted, and the model learns to reconstruct the original text. This training strategy encourages BART to capture semantic meaning and contextual relationships in text, which is valuable for understanding and generating answers to questions. BART is designed to perform multiple NLP tasks, including text generation, translation, and comprehension. It can be fine-tuned for specific tasks like question answering, making it a versatile choice for applications that involve answering questions based on textual information.

”Big Bird: Transformers for Longer Sequences” is designed to handle longer sequences of text efficiently. Processing and understanding longer sequences of text, which is often required in question answering and other natural language processing tasks, is handled by Zaheer et al. (2020) in their work BigBird. Traditional Transformer models, including BERT and GPT, have limitations when it comes to processing very long sequences due to quadratic complexity. Longer texts are often truncated or split into smaller parts, which can result in a loss of context and hinder the performance of question-answering models. Big Bird introduces a *global attention mechanism* that allows the model to focus on relevant parts of the input text, even for long sequences. This reduces the computational complexity of the attention mechanism and enables efficient processing of longer texts. Big Bird employs *sparse attention patterns*, where the model attends to only a subset of tokens in the input sequence. This reduces memory requirements and computational costs, making it practical to handle much longer text inputs. This model is pretrained on large corpora and can be fine-tuned on specific question-answering datasets. Fine-tuning adapts Big Bird to the task of answering questions, allowing it to generate contextually relevant responses.

DIALOGPT is presented by Zhang et al. (2020) in their work ”DIALOGPT: Large-Scale Generative Pre-training for Conversational Response Generation”. DIALOGPT is primarily designed for generating responses in conversational contexts. While this is different from traditional question answering, it shares similarities as both tasks involve generating coherent and contextually relevant text responses. DIALOGPT is trained on a massive corpus of conversational data, enabling it to capture the nuances of human conversation. This excels at understanding the context of a conversation, which is essential for generating accurate responses. This ability is also valuable in question-answering tasks where context plays a crucial role in understanding and addressing the question. DIALOGPT can be fine-tuned on specific datasets or tasks, including question answering. Fine-tuning adapts the model to the target task by providing labeled data and task-specific objectives. This paper discusses extending DIALOGPT to handle multimodal inputs, such as text and images.

”ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators” introduces the ELECTRA model. This paper doesn’t focus exclusively on question answering, Clark et al. (2020) proposes a unique pre-training method that has potential implications for question-answering tasks. Unlike traditional pre-training models like BERT, which use a generative pre-training approach (i.e., predicting masked words), ELECTRA adopts a discriminator pre-training approach. In this method, the model learns to discriminate between ”real” and ”fake” tokens in a sentence. This approach encourages the model to understand and capture contextual relationships more effectively. ELECTRA is designed to be computationally more efficient than traditional generative pre-training models like BERT. This efficiency can be advantageous in question-answering tasks, especially when processing large datasets and long passages of text.

”DEBERTA: Decoding-Enhanced BERT with Disentangled Attention” paper of He et al. (2021) is a variant of the BERT model that incorporates decoding enhancements and disentangled attention. This paper doesn’t exclusively focus on question answering. DEBERTA incorporates decoding enhancements to improve the quality of the generated text. This can be particularly relevant in question-answering tasks where generating contextually relevant and coherent answers is crucial. DEBERTA introduces disentangled attention, which helps the model to focus on different aspects of the input text separately. This enhanced attention mechanism can be valuable for capturing relevant information in questions and passages during the question-answering process.

”Comparative Analysis of Transformer-based Models for Question Answering” by (Rawat & Samant 2022) undertakes a comparative analysis of several prominent Transformer models, including BERT, ALBERT, RoBERTa, XLNET, DistilBERT, Electra, and Pegasus, within the context of Question-Answering (QA) systems using the SQUAD2 dataset. The author mentions, While traditional methods like ”Bag of Words” were time-consuming and limited in handling large databases, modern approaches, such as BERT-based models, have revolutionized QA by effectively extracting answers from extensive documents.

In ”Deep Learning for Natural Language Processing” (Patel & Patel 2021) conducts a comprehensive review of deep learning architectures, encompassing CNN, RNN, and Attention models, highlighting their applications in the realm of natural language processing.

Reading Comprehension and question-answering (QA) task in NLP has advanced from statistical methods to neural networks and deep learning. (Krishnamoorthy 2021) surveys these developments, highlighting Attention and Transformer models like BERT. It also explores Open Domain Question Answering (ODQA), discusses frameworks like Haystack, and delves into recent trends such as Multi-Hop QA and evaluation using Adversarial Networks.

2.7 Question Answering with NLP Technology

(Rami & Pawar 2017) presented a paper ”A Framework for Restricted Domain Question Answering System Using Advanced NLP Tools and Software”. Its primary goal is to precisely extract accurate answers from extensive datasets in response to user questions. The system comprises four distinct modules: Question Processing, Document Processing, Paragraph Extraction, and Answer Extraction. Additionally, the paper proposes distinct algorithms catering to Definition, Descriptive, and Factoid question types, ensuring efficient extraction of optimal answers from large datasets within restricted domains.

QA comprises three distinct modules: answer extraction, information retrieval, and question classification, each with core and auxiliary components. Question classification categorizes questions based on type, a crucial step in QA systems. Information retrieval is essential for locating answers, as without relevant content in a document, further processing is futile. Answer extraction aims to pinpoint responses to user queries. (Sarkar et al. 2023) in ”The Task of Question Answering in NLP: A Comprehensive Review” offers an extensive review of diverse QA methods, evaluation criteria, and benchmarking tools commonly employed by researchers.

Question Answering (QA) system in information retrieval is a task of automatically answering a correct answer to the questions asked by humans in natural language using either a pre-structured database or a collection of natural language documents. (Lende & Raghuwanshi 2016*a*), presents only the requested information instead of searching full documents like a search engine. This paper presents the implementation methods and experimental results with an analysis of the closed-domain QA System which handles only documents related to the education system to retrieve more precise answers using NLP techniques.

(Paranjape et al. 2022) in their work ”Retrieval-guided Counterfactual Generation for QA” mentions Deep NLP models have been shown to be brittle to input perturbations. Recent work has shown that data augmentation using counterfactuals (minimally perturbed inputs) can help address this weakness. They focus on the task of creating counterfactuals for question answering, which presents unique challenges related to world knowledge, semantic diversity, and answerability. To address these challenges, they developed a Retrieve-Generate-Filter (RGF) technique to create counterfactual evaluation and training data with minimal human supervision. Using an open-domain QA framework and question generation model trained on original task data, they created counterfactuals that are fluent, semantically diverse, and automatically labeled. Data augmentation with RGF counterfactuals improves performance on out-of-domain and challenging evaluation sets over and above existing methods, in both the reading comprehension and open-domain QA settings. Moreover, we find that RGF data leads to significant improvements in robustness to local perturbations

Discourse connectives are the words that connect two coherent clauses, phrases, or sentences. The identification of discourse connectives signals the semantic relation between the text fragments. This is explored by clustering the semantically similar Thirukkurals (Tamil literary work). (Anita & Subalalitha 2019) in their work ”An Approach to Cluster Tamil Literatures Using Discourse Connectives” proposes using K-Means Clustering using discourse connectives as the predominant features. These clusters of Thirukkural can form a basis to identify efficient indices for many Natural Language Processing (NLP) applications, such as Information Retrieval Systems, Summary Generation Systems, and Question Answering Systems.

(Huang & Li 2021) in their work ”Translating Sentimental Statements Using Deep Learning Techniques” mentions the application of NLP to translate statements from negative to positive sentiment while retaining semantic similarity has not been explored extensively. It begins by creating datasets of negative-to-positive sentimental statements for training a sentiment translation model. Deep learning methods are then employed to construct the sentiment translation model. The model's effectiveness is evaluated through perplexity, bilingual evaluation understudy, and human assessments, yielding favorable results.

”Collaborative Writing Support Tools on the Cloud” discusses managing writing activities and providing feedback to students is very labor intensive and academics often opt out of including such learning experiences in their teaching. iWrite provides tools for managing collaborative and individual writing assignments in large cohorts. (Calvo et al. 2011) developed an architecture for a new collaborative writing support environment used to embed such collaborative learning activities in engineering courses. It outsources the writing tools and the storage of student content to third-party cloud-computing vendors.

Many Natural Language Processing and Computational Linguistics applications involve the generation of new texts based on some existing texts, such as summarization, text simplification, and machine translation. (Chen et al. 2018) in their paper ”A Semantic QA-Based Approach for Text Summarization Evaluation” addresses a persistent challenge in NLP and Computational Linguistics regarding the accurate assessment of text generation applications. The focus is on pinpointing content differences between two text passages, particularly for longer texts such as articles or books. The proposed approach treats a text passage as a small knowledge base and poses numerous questions to comprehensively identify its content points. This method is distinct from existing approaches and shows promising results in an experiment using the 2007 DUC summarization corpus, indicating potential for precise content differentiation in large text passages.

(Berger et al. 2022), in ”Generation of English Question Answer Exercises from Texts using Transformers-based Models” explores the NLP techniques for generating question/answer exercises from English texts. The research focuses on creating beginner-level exercises for teaching English as a Second Language (ESL) to children. The proposed approach comprises four stages: 1- pre-processing, including co-reference resolution; 2- answer candidate selection through semantic role labeling; 3- question generation using a Transformer-based language model; and 4- post-processing to refine question format. The question generation model achieved comparable results to previous works on a benchmark evaluation.

Using NER and BERT (Acharya et al. 2022) introduces an interactive question-answering system designed for effective communication. ”Question Answering System using NLP and BERT” focuses on simplifying daily tasks through enhanced communication. The proposed system operates by generating a personalized dataset for users, prioritizing significant answers using NER. This dataset is continually updated based on user input. Subsequently, the system employs BERT and CDQA (closed domain question answering) techniques to provide optimal answers when users inquire, resulting in an interactive and responsive platform.

(Singh et al. 2021) in their work ”Question Answering Chatbot using Deep Learning with NLP” focuses on constructing a closed domain, factoid Question Answering system. The approach utilizes NLP methods such as pattern matching and information retrieval to generate an answer candidate pool. By mapping questions and answers into a feature space, the model employs distributional representations of words and sentences, capturing lexical, semantic, and syntactic facets. A convolutional neural network ranks candidate answers based on their representations. The model learns both input question-answer sentence representations and a matching function through supervised learning from training data.

In their review work ”Ontology-Based Approach to Semantically Enhanced Question Answering for Closed Domain: A Review” (Arbaaeen & Shah 2021) conducts a literature review of ontology-based methods that enhance closed-domain QA through semantic means. The review encompasses papers published between 2000 and 2020 and identifies 83 out of 124 studies that recognize and suggest the development and evaluation of QA approaches using various methods. While the majority of the reviewed studies focus on open domains, this study uniquely explores the application of ontology-based enhancements in the context of closed-domain QA, highlighting the potential for advancing semantic understanding and improving QA performance.

2.8 NLP for History-Related Work

”Normalisation of historical text using context-sensitive weighted Levenshtein distance and compound splitting” by (Pettersson et al. 2013) mentions natural language processing for historical text imposes a variety of challenges, such as dealing with a high degree of spelling variation. Furthermore, there is often not enough linguistically annotated data available for training part-of-speech taggers and other tools aimed at handling this specific kind of text. In this work, they used a Levenshtein-based approach to normalization of historical text to modern spelling. In their basic version, no annotated historical data is needed. The experiments were carried out on Swedish data they got promising results.

”Novel Event Detection and Classification for Historical Texts” by (Sprugnoli & Tonelli 2019) discusses the importance of event processing in Natural Language Processing for historical texts, highlighting the gap in resources and systems for this domain. The authors propose new annotation guidelines comprising 22 classes of event mentions and types for historical texts. They apply these guidelines to annotate a historical corpus and compare two approaches for automatic event detection and classification. The study contributes to enhancing the understanding of events in historical texts, providing new annotation guidelines, a corpus, and models for automatic annotation, thereby addressing a relatively unexplored aspect of Temporal Information Processing.

In their work ”Harnessing Biomedical Natural Language Processing Tools to Identify Medicinal Plant Knowledge from Historical Texts”, (Sharma et al. 2017) addresses the challenge of cataloging historical medicinal plant uses from digitized texts and proposes a systematic approach for identifying potential plant-based therapies. It highlights the limited use of informatics methods in this domain and investigates the feasibility of employing biomedical terminologies and natural language processing techniques. The study's preliminary results demonstrate the potential of informatics approaches to extract therapeutic plant information from historical biodiversity literature, identifying both successful aspects and areas for enhancement in utilizing digitized resources for medicinal plant knowledge retrieval.

(Brown & Shackel 2023), in their paper ”Coal Mining, Labor History, Natural Language Processing, Oral Histories, Text Mining” explores the potential of text mining and natural language processing techniques to enhance historical archaeology and oral history research. It acknowledges the current limitations of text mining methods, which are primarily designed for contemporary big data, and discusses their limited relevance in historical and archaeological contexts. Using oral histories from the anthracite coal mining region of Pennsylvania, the study demonstrates the application of text analysis to historical texts and proposes ways to better align general text mining approaches with the specific demands of working with historical and location-based texts, aiming to bridge the gap between these methodologies and the requirements of historical research.

(Ramli & Noah 2016) in their work ”Building an Event Ontology for Historical-Domain to Support Semantic Document Retrieval” addresses the growing concern about ontology's role in explaining data semantics across diverse data sources. It acknowledges challenges in ontology development, especially in historical contexts with vast and complex information. While prior efforts have focused on technical aspects like language representation and inference mechanisms, the study emphasizes a gap in practical application methods. The paper contributes by discussing the experience and processes of ontology creation within the context of history, specifically centered around retrieving historical documents based on events. This highlights the need for a balanced approach that combines technical refinement with effective application strategies in historical ontology development.

In their work ”International Journal on Advanced Science, Engineering and Information Technology”, (Ramli et al. 2020) addresses the challenges of effectively retrieving historical documents from large archives using traditional Information Retrieval (IR) methods. It highlights the limitations of conventional models, like Bag-of-Words (BOW), in accurately retrieving historical documents due to the unique nature of historical data. The study introduces an ontology-based approach for indexing and ranking historical documents, using the Vietnam War as a case study. By extending an existing ontology and semantically mapping it to relevant concepts in the documents, the proposed approach improves retrieval precision and recall. A novel ontology-based weighting mechanism is also proposed and evaluated, demonstrating that the ontological approach yields better results compared to the BM-25 probabilistic retrieval model. The findings suggest that incorporating ontological semantics can enhance historical document retrieval, making it competitive with traditional keyword-based approaches.

(N & B 2022) in their work ”A Convolutional Autoencoder based Keyword Spotting in Historical Handwritten Devanagari Documents” suggests the preservation and processing techniques for historical Devanagari documents by digitizing them. It acknowledges the need for automation and information extraction from handwritten documents, emphasizing the importance of efficient retrieval of relevant information. The study proposes an alternative approach to accurate transcription, using a keyword spotting method. This method aims to create digital libraries for quick searching and browsing of old manuscripts, contributing to the preservation of cultural heritage. The proposed technique involves a convolutional neural network (CNN) architecture based on Autoencoder representation, specifically designed for word spotting in historical handwritten Devanagari manuscripts. The CNN method demonstrates superior accuracy and favorable outcomes for this purpose.

We know that there are some challenges related to text comprehension. In their work ”Supporting historical understandings with refutation texts”, (Donovan et al. 2018) focuses on the challenge of readers’ misconceptions affecting their comprehension of texts and knowledge acquisition. It introduces the concept of refutation texts, which correct misconceptions by identifying and correcting inaccuracies within the text. While refutation texts have been mainly studied in scientific contexts, the study investigates their effectiveness in history topics. Two experiments involving participants with misconceptions about Rosa Parks’ actions during the Civil Rights Movement were conducted. The results indicate that refutation texts led to better learning outcomes, improved recall, and more accurate questionnaire responses compared to non-refutation texts. Additionally, participants processed refutation content faster, suggesting a potential for enhanced comprehension. The findings suggest that refutation texts could be useful in history learning to address readers’ misconceptions and promote accurate understanding.

(Peng et al. 2021) in their work ”Summarising Historical Text in Modern Languages” introduces the task of historical text summarization, which involves summarizing documents in historical forms of a language into the corresponding modern language. This task is crucial for historians and digital humanities researchers but hasn’t been automated before. The authors create a high-quality dataset for text summarization, containing historical German and Chinese news summarized in modern German or Chinese. They propose a summarization model using cross-lingual transfer learning techniques that don’t require cross-lingual parallel data. The model is benchmarked against state-of-the-art algorithms, showing that their transfer learning approach performs well and is distinct from standard cross-lingual summarization tasks, highlighting the significance of their dataset and approach for historical language summarization.

”Comparative Study of Layout Analysis of Tabulated Historical Documents” by (Liang et al. 2021) discusses multimedia retrieval systems and their significance in efficiently retrieving information from documents. Specifically, it explores image processing techniques like layout analysis and word recognition for the transcription of printed or handwritten documents into digital data, known as document digitization. The study arises from an industrial need to transcribe a vast number of Swedish historical documents into digital format. The research evaluates various approaches including public tools, traditional image processing techniques, and machine learning methods. Results indicate that existing OCR tools struggle with layout analysis for Swedish handwritten historical documents. While traditional image processing techniques show some success in extracting table layouts, machine learning techniques enhance accuracy. The study aims to apply the most effective method in future document mining research to develop resource-efficient systems for big data analytics.

(Moruwawon 2010) in their work ”Translating African Proper Names in Literary Texts” explores the analysis of proper names within Yoruba culture and their translation in three African literary texts. It highlights the significance of African names, emphasizing their non-arbitrary nature through historical and textual analysis. The ethical dimension of translating these texts into another language is addressed, necessitating translation strategies that preserve the subversive nature of these cross-cultural works. The study concludes that examining African proper names provides foreign readers with insights into the African worldview and the socio-cultural importance of these names, contributing to a deeper understanding of African cultures.

2.9 Rule Based QAS

Observing the text corpus available, looking into the problems in hand and identifying patterns, discovering rules that can help us extract the answer from the corpus is a common ways to solve QAS tasks. There are many kinds of rules, some of which are given below.

1. Grammar Rules: Rule-based NLP often involves the application of grammatical rules, which define the structure and syntax of a language. These rules dictate how words, phrases, and sentences are constructed. For example, parts-of-speech (POS) tagging, parsing, and syntactic analysis involve using grammar rules to understand sentence structure.

2. Pattern Matching: Rule-based systems can use pattern matching to identify specific sequences of words or phrases in text. Regular expressions (RegEx) are a common tool for pattern matching in NLP. For instance, identifying email addresses, phone numbers, or URLs in text using predefined patterns.

3. Named Entity Recognition (NER): NER is a rule-based technique used to identify and classify named entities in text, such as names of people, organizations, locations, dates, and more. These rules are often based on dictionaries or lists of known entities.

4. Sentiment Analysis Rules: Sentiment analysis systems may use predefined rules and heuristics to determine the sentiment or emotion expressed in a text. For example, detecting positive or negative sentiment based on specific keywords or phrases.

5. Text Classification Rules: Rule-based text classification involves assigning predefined categories or labels to text based on specific rules or criteria. For

example, classifying news articles into topics like sports, politics, or entertainment based on keywords or patterns.

6. Question Answering Rules: In question answering, rule-based approaches can involve creating rules to identify question types, extract relevant information from text passages, and generate answers based on predefined patterns or templates.

7. Chatbots and Dialogue Systems: Rule-based chatbots and dialogue systems use predefined rules and patterns to understand user inputs, identify the intent, and generate responses. These systems often have predefined scripts for handling specific user queries or intents.

8. Rule-Based Generation: Some rule-based NLP systems focus on text generation, such as generating templates for news articles or reports. These rules dictate the structure and content of the generated text.

9. Spell Checking and Correction: Spell-checking and correction systems use linguistic rules to identify and correct misspelled words based on dictionary lookups, phonetic similarity, and other linguistic patterns.

Rule-based QAS are very fast and we can run them on commodity hardware. For simpler problems, smaller, simple, and clean text corpus we should always prefer rule-based QAS over deep learning-based or LLM-based solutions.
(Panchal et al. 2021) in their work ”Automatic Question Generation and Evaluation” mentions by employing NLP-based mechanisms like tokenization, Named Entity Recognition (NER) tagging, and Parts of Speech (POS) tagging, the system processes input, checks context, and avoids generating invalid questions. Automatic Question Generation (AQG) system, includes a machine learning-based Fill in the Blank (FIB) generator with multiple-choice options and a rule-based

approach for Wh-type questions. The generated Wh questions are evaluated using the BLEU score to measure their similarity to human-generated ones. This system finds applications in education, self-assessment, and chatbot-based applications, enhancing understanding and evaluation. The approach taken proves effective, particularly for simpler sentences, showcasing its potential in diverse contexts.

In their work ”Automated Question Generator System Using NLP”, (Gumaste et al. 2020), outlines a standardized approach for automated question generation, encompassing both semantic and syntactic aspects of sentences. Utilizing NLP libraries like Spacy and NLTK, tokenization, stemming, lemmatization, and punctuation are employed to process input text. Part of Speech tagging, a crucial NLP component, aids in identifying appropriate tags for the given text. Leveraging these NLP tools, the proposed system exhibits effective performance when provided with text paragraphs as input. This system finds utility in educational institutions, enabling the generation of question papers by schools, colleges, and coaching centers, as well as aiding students in self-assessment.

Code-switching refers to using multiple languages within the same conversation or utterance. Multilingual contextual embedding models, which are trained on various monolingual text collections, have shown promise for cross-lingual and multilingual tasks. (Khanuja et al. 2020) in this paper ”GLUECoS: An Evaluation Benchmark for Code-Switched NLP” introduces an evaluation benchmark called GLUECoS, focusing on code-switched languages like English-Hindi and English-Spanish. The benchmark covers diverse NLP tasks such as Language Identification, POS tagging, Named Entity Recognition, Sentiment Analysis, Question Answering, and a new task, Natural Language Inference, specifically for code-switching. The study compares results from cross-lingual word embedding models and multilingual models and demonstrates that fine-tuning multilingual models on code-switched data yields the best performance, highlighting the potential of optimizing multilingual models for code-switching tasks.

2.10 Earlier Neural Network Based QAS

Before the Transformer age, RNN, LSTM, and GRU were extremely successful in performing any task in NLP. They were tried for question-answering tasks as well. They are deep learning-based architectural solutions. The problem is because of their architecture, they are slow. But the good thing is we don't need expensive hardware to train these models and inference from these models. If it is possible to produce the kind of results that we get with today's LLM and transformer-based architecture from these earlier approaches then that would save more economic and energy saving. For this reason, researchers still keep trying new approaches using LSTM, RNN, and GRU.

(Vincentio & Suhartono 2022) in their paper ”Automatic Question Generation Monolingual Multilingual pre-trained Models using RNN and Transformer in Low Resource Indonesian Language” demonstrates the potential of advanced models for AQG tasks in Indonesian. Prior AQG studies in Indonesian utilized RNN-based architectures like GRU and LSTM. Performance was compared to mBART, mT5, IndoBART, and IndoGPT models. IndoBART, after fine-tuning, outperformed BiGRU and BiLSTM on SQuAD, and showed better performance on TyDiQA.

2.11 Question Generation Approaches

An Automatic Blank space-fill Multiple Choices Question Generation method is one of the research fields that aims to support the increasing demand for specialized educational systems and active learning. (Jadhav & Laddha 2017) in their paper ”An Automatic Gap Filling Questions Generation using NLP” proposes an automatic blank space-fill generation method can proposed to form blank space-fill questions (BFQ) with multiple choices (one correct answer and three choices). The crafting of such type of questions is time-consuming for teachers because making the BFQ from external source material like textbooks and other electronic texts is a very tedious task. It can be generated in three parts, selecting the Descriptive sentence to ask the question, choosing the blank space of the resulting selected sentence, and searching for the choices that distract the learner from the correct answer of the question. Natural language Processing (NLP) techniques like Tokenization, Part-of-Speech tagging, and Name Entity Recognition are applied to each of these sentences. The advantage of this automatic generation method (AGM) is to provide the services that make it easy for teachers to generate the BFQ and many other competitive exams in which the evaluation can be done through conducting multiple choice questions test (Quiz test).

In their work ”Automatic generation of short-answer questions in reading comprehension using NLP and KNN”, (Riza et al. 2023) tell that evaluations of question and answer creation demand significant time investment. Automatic question generation research aims to streamline this process, serving as a tool to generate questions and answers and saving time. This study specifically focuses on utilizing Natural Language Processing (NLP) and K-Nearest Neighborhood (KNN) techniques to automatically generate short answer questions in reading comprehension. The questions are generated from grammatically reliable news articles, with machine learning methods applied to maintain question quality through training on existing questions. The research involves stages such as sentence extraction, problem classification, question sentence generation, and assessing candidate questions against training data for eligibility. Experimental results indicate that the software achieved a notable average of 63.23% across various parameters, making it a promising alternative for automated reading comprehension question creation.

The COVID-19 pandemic has spurred a heightened interest in self-care, driving more individuals to seek disease-related knowledge online. Consequently, medical question-answering and question-generation tasks have gained prominence within natural language processing (NLP). However, the availability of medical question-answer pairs is limited, and existing question-generation systems may not fully cater to the needs of non-professionals seeking medical information. To address this, (Zhou & Zhang 2021) introduces a BERT-based medical pretraining model, leveraging GPT-2 for question augmentation and T5-Small for topic extraction. The model in the work ”DATLMedQA: A Data Augmentation and Transfer Learning Based Solution for Medical Question Answering”, calculates the cosine similarity of the extracted topic and employs XGBoost for prediction. Through GPT-2 augmentation, the proposed model surpasses the state-of-the-art (SOTA) prediction accuracy. Experimental results exhibit impressive performance in medical question answering and generation, showcasing the model's potential to tackle various biomedical question-answering challenges.

The process of creating question papers for exams is traditionally time-consuming and resource-intensive, involving manual efforts from teaching staff. The aim of (Hemanth* et al. 2020) in their work ”Random Multiple Choice Questions Generation using NLP” is to automate question paper generation using Machine Learning and Natural Language Processing (NLP). By leveraging these technologies, the model processes textual data, generating selective questions from a given paragraph and creating multiple-choice options through a unique approach. The implementation of automation through ML and NLP technologies aims to streamline and simplify the question paper creation process, optimizing resource utilization and reducing manual exhaustion. This advancement aligns with the progress of new technologies and their potential to transform educational practices.

(Wang et al. 2022) in their work ”Towards Process-Oriented, Modular, and Versatile Question Generation that Meets Educational Needs” discusses Automatic Question Generation (QG) powered by Natural Language Processing (NLP) offers a promising avenue for educators to save time and enhance student learning. Despite the potential, QG systems have not been widely integrated into classrooms. This study seeks to identify barriers and enhance the usability of automatic QG techniques in education by understanding how instructors create questions and improving the underlying NLP models. Through an in-depth study involving 11 instructors from 7 universities, the research uncovers instructors' thought processes and needs during question creation. While instructors express interest in using NLP systems for question design, none have practically employed such tools. Instructors draw on various information sources, including domain knowledge and students' misconceptions, which are not addressed in current QG systems. The study advocates for the development of human-NLP collaborative QG systems that prioritize instructor control and explainability, highlighting the importance of process-oriented support, modular design, and handling diverse input sources for real-world adoption.

”Few-Shot Question Generation for Personalized Feedback in Intelligent Tutoring Systems (ITS)” by (Kulshreshtha et al. 2022) mentions existing work on generating hints in ITS focuses mostly on manual and non-personalized feedback. In this work, they explored automatically generated questions as personalized feedback in an ITS. Personalized feedback can pinpoint correct and incorrect or missing phrases in student answers as well as guide them towards correct answers by asking a question in natural language. This paper combines cause-effect analysis to break down student answers using text similarity-based NLP Transformer models to identify correct and incorrect or missing parts. They trained a few-shot

Neural Question Generation and Question Re-ranking models to show questions addressing components missing in the student's answers which steers students toward the correct answer. Their model outperforms both simple and strong baselines in terms of student learning gains. Finally, they show that our personalized corrective feedback system has the potential to improve Generative Question Answering systems.

”Transformer-based End-to-End Question Generation” paper from (Lopez et al. 2020*b*) demonstrates that robust question generation can be achieved using transformer-based fine-tuning methods with a single pretrained language model. This approach surpasses the performance of intricate RNN-based Seq2Seq models, achieving notable increases in METEOR and ROUGE scores. Interestingly, the proposed method achieves similar results to Seq2Seq models incorporating additional mechanisms, even though it relies solely on a single model. The study explores the impact of factors like input formatting, context paragraph length, and answer-awareness on model performance, while also investigating failure modes and their underlying reasons.

”Towards Answer-unaware Conversational Question Generation” is a Conversational question generation is a novel area of NLP research that has a range of potential applications. This paper by (Nakanishi et al. 2019) presents a framework for conversational question generation that is unaware of the corresponding answers. To properly generate a question coherent to the grounding text and the current conversation history, the proposed framework first locates the focus of a question in the text passage and then identifies the question pattern that leads the sequential generation of the words in a question.

(Cho et al. 2019) in their work ”Mixture Content Selection for Diverse Sequence Generation”, suggests as following. Generating diverse sequences is important in many NLP applications such as question generation or summarization that exhibit semantically one-to-many relationships between the source and the target sequences. They suggested separating diversification from generation using a general plug-and-play module (called SELECTOR). For diverse content selection, the diversification stage uses a mixture of experts to sample different binary masks on the source sequence. The generation stage uses a standard encoder-decoder model given each selected content from the source sequence. Due to the non-differentiable nature of discrete sampling and the lack of ground truth labels for a binary mask, they leverage a proxy for a ground-truth mask and adopt stochastic hard-EM for training. In question generation (SQuAD) and abstractive summarization (CNN-DM), this demonstrates significant improvements in accuracy, diversity, and training efficiency, including state-of-the-art top-1 accuracy in both datasets.

(Riza et al. 2019) in the work ”Question Generator System of Sentence Completion in TOEFL Using NLP and K-Nearest Neighbor” are leveraging computer technology for creating high-quality TOEFL sentence completion questions. The proposed system involves several stages, including data collection from reputable English grammar news sites, Natural Language Processing (NLP) preprocessing, Part of Speech (POS) tagging, question feature extraction, sentence selection, feature determination, categorical data conversion, blank position word classification using K-Nearest Neighbor (KNN), heuristic rule determination by human experts, and distraction selection based on heuristic rules. Through experimentation with 10 news articles, the system produced 20 questions that were evaluated as having very good quality (81.93%) by human experts. Additionally, 70% of the blank positions in the generated questions matched historical TOEFL question data. This suggests that the generated questions align with historical training data and offer high-quality distractions based on expert heuristics.

Question generation (QG) involves training a model to create questions based on input text. While recent approaches often use sequence-to-sequence models with additional features and mechanisms, these can increase complexity and reliance on unavailable auxiliary data. (Lopez et al. 2020*a*) in their work ”Simplifying Paragraph-level Question Generation via Transformer Language Models” proposes a more straightforward approach using a single unidirectional Transformer-based language model, fine-tuned from GPT-2 Small. The model outperforms paragraph-level QG baselines on the SQuAD dataset by 0.95 METEOR points. Human evaluators found the generated questions easy to answer, contextually relevant, and naturally phrased. Additionally, the study introduces baseline scores for the RACE dataset, a novel use for QG tasks. Further research is recommended to explore different model capacities and datasets, specifically those with non-identification type questions, to confirm the robustness of pretrained Transformer-based language models as effective question generators.

2.12 Answer Generation Approaches

Machine reading comprehension methods that consider multiple passages for generating answers have gained significant attention in AI and NLP. However, existing methods overlook the relationships among these passages during answer generation, missing potential answer candidates linked by correlated topics. (Nakatsuji & Okui 2020) in their work ”Answer Generation through Unified Memories over Multiple Passages” proposes an approach to Neural Answer Generation through Unified Memories over Multiple Passages (GUM-MP). Their dataset has questions, positive passages (which are assigned to the question), and negative passages (which are not related to the question). This approach addresses this issue as follows: (1.) Identify tokens in positive passages matched with the question. (2.) Identify tokens in negative passages matched with the question. (3.) Encode these matching tokens into unified memories within passage encoders and utilize an encoder-decoder model with a multiple-pointer-generator mechanism for answer sequence learning. GUM-MP effectively generates answers by pointing to crucial tokens spanning passages, resulting in notably improved accuracy compared to current models, as demonstrated through evaluations.

2.13 Ethical Considerations in QA Systems

When students of social science courses communicate, socialize, shop, learn, and work online. They are asked to collect data for course projects they are often drawn to social media platforms and other online sources of textual data. There are many software packages and programming languages available to help students collect data online, and there are many texts designed to help with different forms of online research, from surveys to ethnographic interviews. However, there is no textbook available that teaches students how to construct a viable research project based on online sources of textual data such as newspaper archives, site user comment archives, digitized historical documents, or social media user comment archives. (Ignatow & Mihalcea 2018) in their work ”An Introduction to Text Mining” gives a better starting point for undergraduates and first-year graduate students interested in collecting and analyzing textual data from online sources, and will cover the most critical issues that students must take into consideration at all stages of their research projects, including ethical and philosophical issues; issues related to research design; web scraping and crawling; strategic data selection; data sampling; use of specific text analysis methods; and report writing.

In their work ”The great Transformer: Examining the role of large language models in the political economy of AI”, (Luitse & Denkena 2021) mentions large language models (LLMs) like GPT-3 are boasting human-like language generation capabilities, have sparked concerns about biases, environmental impact, and societal implications. The article examines LLMs as integral components in the political economy of AI, closely tied to the business strategies of tech giants. It highlights the transformative impact of the underlying architecture, the Transformer, on LLM development and underscores how corporate interests can lead to monopolization and dependency in AI research through strategies such as exclusive licensing and paid API access.

2.14 QAS on Edge Device

(Sun et al. 2020) introduces ”MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices”, MobileBERT, a compressed and accelerated version of the BERT model, specifically designed for resource-limited mobile devices. MobileBERT maintains the task-agnostic nature of BERT and achieves a balance between self-attentions and feed-forward networks. The training process involves a teacher model and knowledge transfer, resulting in a model that is 4.3× smaller and 5.5× faster than BERTBASE. Despite its compact size, MobileBERT achieves competitive performance on various NLP benchmarks, including GLUE and SQuAD question-answering tasks. For instance, it achieves a GLUE score of 77.7 (0.6 lower than BERTBASE) and a dev F1 score of 90.0/79.2 (1.5/2.1 higher than BERTBASE) on SQuAD v1.1/v2.0 tasks, while maintaining a low latency of 62 ms on a Pixel 4 phone.

2.15 Large Language Model (LLM) Based QAS

Large Language Models or LLMs are recent developments in Machine Learning. There is no established definition of what model is LLM and what not, but generally speaking, ”models with tens or hundreds of billions of parameters and trained on terabytes of text data are considered LLM”[4]. With the same architecture, a transformer model with a different number of layers, different embedding sizes, different sizes and number of dense layers, and different sizes of data can have different hyperparameters and different sizes.

Zhang et al. (2022) introduces OPT in their paper “OPT: Open Pre-trained Transformer Language Models”. This is a suite of decoder-only pre-trained transformers, ranging from 125M to 175B parameters, that are open and accessible to researchers. The paper shows that OPT-175B is comparable to GPT-3 in performance while requiring only 1/7th of the carbon footprint to develop. The paper also provides a logbook detailing the infrastructure challenges faced during the training process, along with code for experimenting with the models. The paper demonstrates that OPT models can perform well on various natural language processing tasks, including question answering, using a unified text-to-text format.

In their paper “BLOOM: A 176B-Parameter Open-Access Multilingual Language Model” Workshop et al. (2022) introduces BLOOM, a large language model (LLM) that can process and generate text in 59 languages, including 13 programming languages. The paper describes the ROOTS corpus, a dataset of hundreds of sources that was used to train BLOOM, and the challenges and solutions involved in creating it. The paper shows that BLOOM achieves competitive performance on various natural language processing (NLP) tasks, such as text generation, text classification, and question answering, using a unified text-to-text format. The

[4] What are LLMs, and how are they used in generative AI? Accessed on 10-Sep-23

paper also demonstrates the benefits of multitask-prompted finetuning, a technique that adapts BLOOM to specific tasks using natural language instructions and examples. The paper releases BLOOM and its code under the Responsible AI License, aiming to democratize access to LLMs and foster ethical and inclusive research and applications.

“LLaMA: Open and Efficient Foundation Language Models” work of Touvron et al. (2023) presented LLaMA. It is a collection of foundation language models ranging from 7B to 65B parameters, trained on trillions of tokens from publicly available datasets. The paper shows that LLaMA models achieve state-of-the-art performance on various natural language processing tasks, including question answering, using a unified text-to-text format. The paper releases all LLaMA models to the research community under the Responsible AI License, aiming to democratize access to large language models and foster ethical and inclusive research and applications.

In their work ”Visconde: Multi-document QA with GPT-3 and Neural Reranking”, (Pereira et al. 2023) presents a question-answering system named Visconde, designed to address questions that require evidence from multiple documents. The system utilizes a three-step approach involving decomposition, retrieval, and aggregation. Initially, the question is broken down into simpler ones using a few-shot large language model (LLM). Next, a search engine retrieves candidate passages from a document collection for each decomposed question. Lastly, the LLM in a few-shot setting aggregates information from the passages to form the final answer. The system is evaluated on various datasets, indicating that the effectiveness of the system depends heavily on the retriever’s performance, and human-level performance is achievable when relevant passages are provided. The paper also highlights the system’s improved effectiveness when prioritizing explanations before answering questions.

(Rahaman et al. 2023), in their work ”From ChatGPT-3 to GPT-4: A Significant Advancement in AI-Driven NLP Tools” investigates the advancements in Natural Language Processing (NLP) driven by powerful language models such as ChatGPT, Google’s BARD, and Ernie. It particularly focuses on the potential of the latest iterations of the GPT model, GPT-4 and GPT-5, in tackling complex language tasks. The research employs a narrative analysis of existing literature, gathering secondary data from various sources including articles, websites, and blogs. The findings indicate that GPT-4 enhances training data, computation speed, answer quality, and overall performance. It surpasses GPT-3.5 in tasks like language translation, question answering, and sentiment analysis. The study lays a foundation for further advanced NLP tools like GPT-5 and offers insights for AI, LLM researchers, NLP developers, and academics. Since it’s the first of its kind, the researchers suggest future quantitative research to validate these findings.

(Witteveen & Andrews 2019) presents a useful technique in ”Paraphrasing with Large Language Models”. As per the authors, large language models such as GPT-2 have shown themselves to be extremely adept at text generation and have also been able to achieve high-quality results in many downstream NLP tasks such as text classification, sentiment analysis, and question answering with the aid of fine-tuning. They suggest a useful technique for using a large language model to perform the task of paraphrasing a variety of texts and subjects. This approach is demonstrated to be capable of generating paraphrases not only at a sentence level but also for longer spans of text such as paragraphs without needing to break the text into smaller chunks.

2.16 Parameter-efficient Fine-tuning (PEFT)

PEFT stands for Parameter-efficient Fine-tuning. It is a technique used in Natural Language Processing (NLP) to improve the performance of pre-trained language models on specific downstream tasks. Transfer learning plays a crucial role in the development of large language models such as GPT-3, T3, Llama, BERT. It is an ML technique in which a model trained on a certain task is used as a starting point for a distinct but similar task. Fine-tuning is one of the most popular methods used in transfer learning. It involves adapting a pre-trained model to a particular task by training it on a smaller set of task-specific labeled data.

However, with the parameter count of large language models reaching trillions, fine-tuning the entire model has become computationally expensive and often impractical. In response, the focus has shifted towards in-context learning, where the model is provided with prompts for a given task and returns in-context updates. However, inefficiencies like processing the prompt each time the model makes a prediction and its poor performance at times make it a less favorable choice. This is where Parameter-efficient Fine-tuning (PEFT) comes in as an alternative paradigm to prompting. PEFT aims to fine-tune only a small subset of the model's parameters, achieving comparable performance to full fine-tuning while significantly reducing computational requirements1. It offers an efficient way to fine-tune Large Language Models (LLMs) on downstream tasks. (Mangrulkar & Paul 2023) in their article ”PEFT: Parameter-Efficient Fine-Tuning of Billion-Scale Models on Low-Resource Hardware” writes as models get larger and larger, full fine-tuning becomes infeasible to train on consumer hardware. In addition, storing and deploying fine-tuned models independently for each downstream task becomes very expensive, because fine-tuned models are the same size as the original pretrained model.

PEFT approaches only fine-tune a small number of model parameters while freezing most parameters of the pretrained LLMs. Thus, greatly decreasing the computational and storage costs. This also helps in overcoming the issues of catastrophic forgetting, a behavior observed during the full finetuning of LLMs. But, we need to keep in mind PEFT is a training time concept. If you have finetuned any LLM using PEFT it does not mean for inference purposes you need lesser grade hardware than you used for that LLM originally.

LoRA, or Low-Rank Adaptation, is a technique used to fine-tune large language models (LLMs) in a more efficient and parameter-efficient way. LoRA is a kind of PEFT. It works by adding a low-rank weight matrix to each layer of the LLM, rather than updating all of the weights in the model. This can significantly reduce the number of parameters that need to be updated and can also lead to faster training times and better performance on downstream tasks. LoRA has been shown to be effective for fine-tuning LLMs on a variety of tasks, including text classification, sentiment analysis, question answering, and summarization.

2.17 RAG Based QAS

Lewis, Perez, Piktus, Petroni, Karpukhin, Goyal, Küttler, Lewis, Yih, Rocktäschel, Riedel & Kiela (2020) presented a work ”Retrieval-augmented generation for knowledge-intensive NLP tasks” (RAG), The paper presents a novel approach to question answering and other knowledge-intensive NLP tasks by combining retrieval and generation techniques. Knowledge-intensive NLP tasks, including question answering, often require access to external knowledge sources like databases or documents. This work has two components Retrieving relevant information and generating coherent answers. The first component, of the RAG model selects relevant passages or documents from a knowledge source based on the input question. This retrieval step is crucial for obtaining the necessary information to answer the question accurately. The second component of RAG, after retrieving relevant content, the model employs a generation component, typically based on a transformer architecture, to generate the answer. This component ensures that the answer is coherent, contextually relevant, and grammatically correct. The retrieval-augmented generation model is trained on datasets containing questions, passages, and corresponding answers. During training, it learns to retrieve the most relevant passages and generate accurate answers based on the input questions. Accuracy and F1-score are used to evaluate this model's performance.

"REALM: Retrieval-Augmented language model pretraining" paper from (Guu et al. 2020), proposes an approach called Retrieval-Augmented Language Model pre-training (REALM) to enhance language model pre-training with a latent knowledge retriever. This retriever enables the model to retrieve and attend to documents from a large corpus like Wikipedia during pre-training, fine-tuning, and inference. The knowledge retriever is pre-trained in an unsupervised manner using masked language modeling and retrieval steps, which allows the model to capture modular and interpretable world knowledge. REALM is evaluated on Open-domain Question Answering (Open-QA) tasks and compared to state-of-the-art models for both explicit and implicit knowledge storage. The results demonstrate significant performance gains (4-16% absolute accuracy) on multiple benchmarks, while also providing advantages such as interpretability and modularity.

2.18 Application of AI-Based QAS

(Lekova et al. 2022) in their work "Making humanoid robots teaching assistants by using natural language processing (NLP) cloud-based services” focuses on harnessing humanoid robots as teaching and social assistants, considering the high expectations children have for human-like interactions with these robots. The researchers propose a comprehensive model that utilizes Generative Pre-trained Transformer (GPT) language models within the Internet of Things framework for natural language understanding in human-robot interaction. The physical presence of the robot aids in refining the GPT model through context-driven prompts and subsequent robot actions, enhancing the embodiment comprehension of GPT-generated outputs. The model integrates web or cloud-based Natural Language Processing (NLP) services for tasks like text generation and question answering, leveraging local or cloud-based speech recognition for verbal input processing. The GPT-J model is employed and customized, with options to control text randomness, and the robot utilizes Text-to-Speech (TTS) cloud services to render human-like responses audibly. The model's applicability was assessed through evaluations of outputs from different NLP and GPT-J services, with the intention of aiding speech-language therapy practice using a humanoid NAO-type robot, while also holding potential for diverse robotic platforms and contexts.

”Conversational AI for Natural Language Processing: A Review of ChatGPT” by (Goar et al. 2023) reviews ChatGPT's advantages like accuracy and flexibility over traditional NLP tools, outlines its limitations, and emphasizes the importance of addressing ethical concerns. The potential for ChatGPT in NLP tasks, such as question-answering and dialogue generation, is highlighted, underscoring the need for further research and development in these domains.

(Furlan et al. 2021) in their work ”A Natural Language Processing–Based Virtual Patient Simulator and Intelligent Tutoring System for the Clinical Diagnostic Process: Simulator Development and Case Study” mentions, the shortage of human resources, rising educational costs, and the need for social distancing due to the COVID-19 pandemic have underscored the importance of remote clinical training methods. Virtual patient simulators (VPSs) can partially address these challenges. This study aims to develop a VPS for clinical diagnostic reasoning, integrating natural language interaction and an intelligent tutoring system (ITS). The VPS, named Hepius, employs a Siamese long short-term memory network for anamnesis and NLP algorithms coupled with the Systematized Nomenclature of Medicine (SNOMED) ontology for generating diagnostic hypotheses. The ITS in Hepius offers real-time feedback and topic recommendations to bridge knowledge gaps. A short-term learning test conducted on undergraduate medical students revealed improved test scores and core question scores post-simulation, indicating the potential of Hepius to enhance diagnostic reasoning training. This approach is especially valuable during the pandemic when students have limited access to clinical settings.

”A Turkish Question Answering System Based on Deep Learning Neural Networks” by (GEMİRTER & GOULARAS 2021) addresses challenges by introducing a Question Answering (QA) system for the Turkish language. The system’s objective is to generate high-quality answers, considering both content and length, from questions based on a collection of documents related to the banking sector. To achieve this, the system employs algorithms like BERT and extensive datasets. They fine-tuned BERT for machine reading in the question-answering (MRQA) task. Multiple experiments are conducted, incorporating original and translated datasets, to overcome challenges arising from the morphologically complex nature of Turkish. Their system demonstrates performance that surpasses other Turkish-language QA systems, offering a methodology applicable not only to Turkish but also adaptable to various languages for diverse NLP tasks.

(Agrawal et al. 2019) in their work ”Automatic Fill in the Blank Question with

Distractor Generation Using NLP”, selected a paragraph from which relevant sentences are selected. These sentence serves as potential options to create fill-in-the-blank questions. Only sentences containing substantial information are chosen for question generation. The questions will include appropriately placed blanks that can be filled with one correct choice out of four provided options, similar to multiple-choice questions. Each question will feature a single correct choice, while the remaining three options will serve as incorrect choices or distractors.

”Objective Type Question Generation using Natural Language Processing” presents objective-type questions, facilitated by Information and Communication Technology (ICT) and Intelligent Tutoring Systems (ITS), offering a valuable approach to assessing students’ understanding. However, crafting a suitable set of questions can be time-consuming for teachers, and using external question sources might not align well with the covered content. (Deena & Raja 2022) employs Natural Language Processing (NLP) techniques to generate familiar types of objective questions, such as True or False, ’Wh’ questions, Fill in the Blanks, and match the following questions, directly from the provided study material. The approach employs different rules for generating True/False and ’Wh’ questions, utilizing a dependence parser for the latter. The system’s efficacy is demonstrated using a Grade V Computer Science textbook as input, showing promising results in generating a substantial amount of objective-type assessment questions.

Given two sentences, a hypothesis sentence and a premise sentence, NLI predicts whether these two sentences are entailed, contract, or neutral. Currently, NLI focuses only on text data, surprisingly structured data is not used much for NLI problems. To address this, (Wang et al. 2019) presents a combination of techniques that harness external knowledge to improve performance on the NLI problem in the science questions domain. In their work ”Improving Natural Language

Inference Using External Knowledge in the Science Questions Domain” they present the results of applying our techniques on text, graph, and text-and-graph-based models; and discuss the implications of using external knowledge to solve the NLI problem. Our model achieves close to state-of-the-art performance for NLI on the SciTail science questions dataset.

(Soni et al. 2019) in their work ”Automatic Fill-in-the-blank Type Questions and Answer Generation from Text Using NLP” outlines the detailed process used to generate fill-in-the-blanks type questions with multiple-choice options by extracting significant and informative sentences from a text document.

In their work ”Learning to Automatically Generate Fill-In-The-Blank Quizzes”, (Marrese-Taylor et al. 2018) have formalized the problem of automatic fill-on-the-blanks quiz generation using two well-defined learning schemes: sequence classification and sequence labeling. They have also proposed concrete architectures based on LSTMs to tackle the problem in both cases.

(Jacquemart & Zweigenbaum 2003) represents an evolution in search engines by integrating Information Retrieval (IR), Natural Language Processing (NLP), and Knowledge Engineering. In their work ”Towards a medical question-answering system: A feasibility study” they explore the feasibility of a French QA system in the healthcare domain, focusing on oral surgery. The research examines two key aspects: identifying the appropriate keywords in questions for effective IR, and aligning the question content with a conceptual model for efficient NLP processing.

2.19 Summary

Thus, we see many researchers have tried to solve the question-answering task in different domains using different techniques. Our focus here was to understand what has been done in general and around historical text QA. In recent times researchers are moving towards transformers, retrieval augmented generation, and large language models to solve this problem. To evaluate the model performance they have used various metrics like accuracy, f1 score, BLEU (Bilingual Evaluation Understudy), GLUE (General Language Understanding Evaluation), ROGUE (Recall-Oriented Understudy for Gisting Evaluation), METEOR (Metric for Evaluation of Translation with Explicit ORdering), Accuracy, Recall, R@n, P@n, F1@n. We didn’t find any benchmark history dataset which can be used for our work. Recent LLM-based techniques need expensive hardware to train and deploy the models.

In this extensive literature survey on AI-powered historical book question answering (QAS), a wide array of crucial facets were examined. Topics encompassed various question types, knowledge levels, scopes, input modalities, languages, answer generation methods, historical domains, and ethical considerations. The survey also delved into the evolution of QAS systems, inclusion criteria for historical texts, the advent of transformer-based models, state-of-the-art models, and the role of NLP in historical research. Furthermore, it explored rule-based and earlier neural network-based QAS, strategies for question and answer generation, as well as the application of QAS on edge devices. Large Language Models (LLMs) and the Retrieval-Augmented Generation (RAG) approach were explored, along with practical applications of AI-based QAS in education, research, and knowledge dissemination, making this survey a comprehensive resource for researchers and practitioners in the field of historical text analysis and QAS systems.

Chapter 3

METHODOLOGY

3.1 Introduction

Question Answering is one of the most important tasks in NLP. Depending upon data type (tabular, text, image, etc), data size, context availability (the question is related to the general domain for which the answer is available on websites like Wikipedia or Quora), or whether it is a business-related or domain-specific question, length of context, various models have been built and tested on various QA datasets. Recently, after 2021, we have seen a surge of LLMs (Large Language Models). And performance of these LLMs and non-LLMs can be seen on various leaderboards. Some of the popular leaderboards are

1. Metrics at Huggingface
2. SOTA Paperwithcode
3. Question Answering Paperwithcode
4. BLEU Score
5. GLUE Score

6. Open LLM Leaderboard
7. AllanAI Leaderboards
8. AI Google NaturalQuestions/leaderboard
9. Eval AI
10. QuAC Question Answering in Context

Based on our desired metrics can we select the best-performing model for our project from these leaderboards and complete our question-answering work? But no single LLM or non-LLM can answer all the questions of our work.

Without training, there is no model. We create models using different training techniques. For training we need data, some algorithm/architecture, and some approach using which we play with different hyperparameters to create a model. The actual performance test of the model happens when the model predicts using unseen data. And this happens in a production environment. If the consumer/user/human is satisfied with the predictions then we can say a model is good, model is predicting good. And, we keep that model as long as it satisfies the user.

Prediction can be a number, information classification, information generation, or information extraction. Sentiment analysis, text summarization, sentence completion, story writing, question answering, translation, reasoning, etc., all are different kinds of prediction tasks in NLP. The performance of these prediction models depends upon the quality of data used, prepossessing steps, architecture/algorithm used, and hyperparameters used for model training.

ChatGPT is a conversational model. It can be used for question generation, but for this, we need to create an appropriate prompt. It can also be used for answer generation, for this we need different kinds of prompts. Whatever question an AI Model can answer it depends upon the data used to create those models.

No matter how big is an LLM model, it cannot answer those questions for which it has not seen the corpus. If some data is private to me then we cannot use ChatGPT to answer those questions which are not part of ChatGPT corpus. Creating ChatGPT kind of LLM model is an extremely expensive and time-consuming task, and it is not worth spending this much time and money to answer the questions of a book or to generate question answers based on certain books. Evening finetuning any bulky LLM is expensive. Although there are different techniques that can be used to finetune an LLM for our task, it is expensive and complex enough to be a deterrent to be used by common business users. Can we use AI Model to generate questions from data/text which that AI Model has not seen? Similar to answer generation this is also not possible. AI model cannot be used to generate questions for which it has seen the context.
To generate questions or answers around some data using AI model there are two ways

1. Either we need to train the model on that data. Then anyone who is asking questions from the model need not bother about the input text. For example, you can ask AI Model (ChatGPT) ”How many planets are there in our solar system” and you will get the correct answer. Or you can ask ChatGPT ”Can you ask three questions about our solar system” and you will get three questions as a response. During this conversation, you need not bother about, where is text or book on the planetary system. So that you can train the AI model on that. ChatGPT is already created using this kind of commonly available data. We only need to pass the prompt or question without any other data and the model generates the response.

2. Second way, we need to pass the data/context along with a question. Using prompt engineering we can combine the question, instruction, and context in human understandable way. Now, We put the request to the AI model. In

this situation, the AI Model needs not to look anywhere to answer the question but look into the context. For example, you can put your corporate communication policy along with a request/question and ask the AI model whether particular communication violates the policy or not. In request, you have clubbed the text regarding which you want to ask a question. No LLM can answer this kind of question without context. And, even with context, if a model has not seen the dataset, it cannot perform the desired task.

Keeping the nitty gritty of question types aside, which we discussed earlier, for a moment, from our work perspective broadly there are 4 types of questions

1. You can give context and ask questions for which answer is Yes or No. For example: Whether Kunti is the mother of Duryodhana?. This is a classification problem.

2. You can give context and ask questions for which answers need to be extracted from the text. For example: Who was the second son of Kunti?. This is a text extraction problem.

3. You can give context and questions along with options and the answer is extracted from the given options. For example: Who is the father of Karna? Options: Surya, Bhisma, Indra, Kashyap. This is multi choice problem.

4. You can give context and ask questions but the answer can be generated in a creative way, For example: Who was the charioteer of Arjuna in Kurushetra? The answer could be ”Devaki’s eight’s son”. Or ”Balram’s younger brother” or ”Arjuna’s charioteer in Kurushetra was Krishna”. All three are correct answers. In fact, if we consider human relationship then answer one and two are the same. If we understand English grammar and remove all grammatical text around the third answer then the extracted text is

”Krishna”. However, evaluating these answers is not that easy. In our work, we are interested in these kinds of questions.

In this work, we are not only interested in the answer generation of a question from the context. But we also want to generate questions from the context. Broadly there are 4 parts of our work.

1. **Question Answer Generation System (QAGS**): In this, we are interested in generating unique questions from the given text. We also want to generate the answer and reference of those answers for the corresponding questions. We call it **HBQADataset**. For this giving an entire book in one go is not an efficient way nor today’s hardware and software technology can handle those big-size context windows. Therefore, we need to break the entire book into small chunks and use those chunks to generate questions and answers. So we need the ”Chunk Dataset” of the entire book. We will call this **HBQAChunkDataset**. Corresponding to each chunk we will create some questions and answers, and store that in **HBQADataset**. If the chunk size is big it can have more questions, otherwise, it will have one QA pair at least.

2. Document Retrieval System (DRS): In this, we are interested in identifying specific chunk(s)/document(s) that can answer the question.

3. **Answer Generation System (AGS)**: In this, we are interested in creating a fine-tuned answer generation system. Neither it is possible nor viable to draft all sets of the possible questions from all the given chunks of a book. So we can generate one or two representative questions from each chunk. Using **HBQADataset** and **HBQAChunkDataset** we can create a **finetuned model**. Which has seen the entire book in the form of chunks to generate questions and answers. Now we can use that model to answer those questions that were not available during the finetuning. The answer generated by the

system can be extractive in nature. In that case, it is easy for the machine to verify the answer using metrics that are based on unigram, bigram comparison. But if the answer is creative or stitched together after mixing different portions of the learned chunk, as mentioned earlier, then we can look for the semantic meaning of the answer. While developing this system we will try zeroshot learning with t5, flan-t5, RoBERTa, Bloom, DistilBERT and then select the best zeroshot model and finetune that model for our work.

4. **Retrieval Augmented Answer Generation System (RAAGS)**: Answering questions using AGS is fine when context is known. But when we don't know the context then AGS will not work. If a model is created using voluminous text with a limited set of questions then it is not possible to tell how this model will perform in the production environment. Performance depends upon how creatively people are asking book-related questions, it also depends upon how big was model size and on what hardware the model is hosted. To address this problem we need a vector dataset for chunk, we call it **HBQAChunkVectorDS**. Using HBQAChunkVectorDS and **QuestionVector** we can identify the closest chunk, which can answer the question. Now using these selected chunks and a given question, we can get answers from a fine-tuned model. We can retrieve n chunks to get n answers from the fine-tuned model. Finally, we can combine all the answers and give that as a context to the finetuned model to predict the final answer. We also need a dataset for questions and answers, we called it **HBQAVectorDS**. HBQAVectorDS dataset helps us identify how different models' answer is from a predicted answer.

3.2 Research design

3.2.1 Research Strategy

In our work, we want to build a system that can do the following

- Generate questions and corresponding answers from any history book. However, keeping the volume and varieties of history work in mind we will restrict the corpus size. We will only deal with Indian History Books. And, in this research work, all experiments will be restricted to a very popular historical work ”The Mahabharat”. In the Indian culture, we refer to History as ”Itihaas”

- Generate answers to any question which can be answered using a history book. This will also be restricted to ”The Mahabharat” book.

Question-answer generation from a given book is a one-time work. It is like creating a fixed FAQ. This is useful for printing books, creating FAQs on a given book, and making some ideas popular when people do not know much about that book. So, FAQ does make sense when people have not heard the stories so through a small FAQ, one-liner or two-liner question answers people become interested in the subject.

But, when people have read the book and they have some questions then they do not ask those fixed questions that are part of the FAQ. In fact, they ask more deeper questions. In that situation, we need a system that can look into different sections of the book and extract the answer or stitch the different parts of the book and generate an answer. For this, we ”Answer Generation System”

HBQA (Historical Book Question Answering) system has 4 four components :

1. QAGS : Question Answer Generation System

2. AGS : Answer Generation System
3. DRS : Document Retrieval system
4. RAAGS : Retrieval Augmented Answer Generation System

Why Mahabharat Book?

We are choosing the Mahabharat book because of multiple reasons.

- This is the single most popular book that defines the psyche of any Indian. Irrespective of faith they practice, there is too much in this book that people don’t know, and don’t speak but they behave as per that undercurrent which is set by this book. It defines the social physiology of Indian culture and this continent.
- It has multiple translations and into multiple languages. Although we will work with only one translation of this book. In the future, we want to work with other translation work.
- It has thousands of stories on leadership, family values, spirituality, human struggle, morality, etc. And we want to learn from these kinds of stories in a different way. Question answering, and quizzing is other ways of learning.
- It is a quite voluminous book. This makes it an interesting case for machine learning.
- We are picking the Ganguli edition of Mahabharat because it was published in the late 1800s and is now available online for free (as they’re out of copyright).

We found that BORI (the Bhandarkar Oriental Research Institute), took up the project to compile a ‘Critical Edition’ of Mahabharat in 1919. This edition would be a consolidation of all the available manuscripts, across all vernacular languages, sourced from different states of India. It was a massive project, and many scholars were involved in distilling the true Mahabharat, without the later additions, regional interpolations, and other unnecessary factions. (The version being referred to by most writers and scholars before this was called the Vulgate edition and it was from the 17th century).

The most popular translations of Mahabharat are

- Kisari Mohan Ganguli (1883-1896): Ganguli’s translation is one of the oldest and most complete English translations of the Mahabharat. It is a faithful rendering of the Sanskrit text, but it can be difficult to read due to its archaic language.

- Romesh Chunder Dutt (1898): Dutt’s translation is a poetic condensation of the Mahabharat, focusing on the main themes and characters. It is a more accessible read than Ganguli’s translation, but it sacrifices some of the accuracy and detail of the original text.

- P. Lal (2005-present): Lal’s translation is a verse-by-verse rendering of the Critical Edition of the Mahabharat, which is considered to be the most scholarly and accurate edition of the text. Lal’s translation is still in progress, but it is already considered to be one of the best English translations of the Mahabharat available.

- Bibek Debroy (2010-2014): Debroy’s translation is a complete and unabridged translation of the Critical Edition of the Mahabharat. It is written in a clear and modern style, making it a very accessible read. Debroy’s translation is also highly praised for its accuracy and scholarship.

Apart from the above translations, there are hundreds of other translations of

Mahabharat in English. We need to keep in mind Mahabharat is translated into Hundreds of Indian and non-Indian languages.

3.2.2 Data Collection

Mahabharat's original text is around 100 thousand Sanskrit sloka. The entire work has 18 Parvas (like chapters and sometimes referred to as a book). Each chapter has a specific name, like Shanti Parva, some authors refer to it as ”The Book on Peace”. This way there are 18 Books. To avoid the confusion of referring to them with confusing English names like chapter or book, we will call them by their original name ”Parva”. Each Parva has hundreds of sections. Some authors refer to them as chapters. In the original book, they are referred as Adhyaya or Up-Parva. To avoid the confusion of multiple names we will refer to these as Sections.

So Mahabharat book has 18 Parva, and each Parva has hundreds of Sections there are a total 2100 sections in this book.

We downloaded the Ganguli translation of Mahabharat book from Mahabharataonline.com. All the 18 Parvas available here [1]. Each link on this webpage leads to separate English translated Parva. Each Parva has hundreds of section. We downloaded all 2100 section of this book using python script and python selenium library.

After download, we checked whether any special character issue in the text downloaded. This text is clean utf8 format. There are no images, not tabular data in this text. After downloading these files on local machine, we successfully loaded these files into pandas dataframe in csv format with utf8 formatting, with this we assume there is not problem in the text and there is no unicode formatting issue.

[1] https://www.mahabharataonline.com/translation/

3.2.3 Data Analysis

Using python we created one csv file which contains metadata about all the downloaded content. This metadata includes number of letters, number of words, number of paragraph in each section and Parva, number of section in Parava. We checked data distribution of these columns. It helps us understanding how skewed or symmetric our dataset is, how many unique words are used. We created a word cloud and word frequency plot, to understand what kind of words and what letters in the text.

3.2.4 Key Activities

From data collection to analysis we need to do following activities.

1. Establish, how many books and which book should be included in our work.
2. Establish, should we go beyond English translation of Historical work to other languages?
3. Explore, how many translation of the work are available, their credibility, how good they are written.
4. Any copyright or legal issue related to content
5. From where we can download this free content
6. write python script to download
7. Clean/merge data
8. Data Analysis

3.2.5 Challenges and Limitations

We understand getting free of cost, popular, useful, copyright free, modern style, clean content for our work is difficult. In our work we know we have compromised on the style aspect, Ganguli translation is old style English translation and will lot of transliteration issues. We will try to understand the transliteration issue few examples

- yudhishthira युधिष्ठर should be yudhishthara
- bharata भारत should be bhaarata
- govardhan गोवर्धन should be goverdhana
- ritwija रित्विज should be ritvija
- madri - माद्री should be maadri
- devaraja - देवराज should be devaraaja
- Section number in roman number like dclxxviii

Mahabharat is a text of ancient Bharat, it is possible that at some places some bias towards some gender, profession, or geography. But, to our knowledge, we did not notice any such bias.

3.2.6 Validation and Reliability

To make sure that our work is reliable and can be taken forward by any other researcher or organization we are following these steps.

- Traceability of every document from source to analysis conclusion.
- Use Python source code, rather than analysis in Excel. It ensures reproducibility.

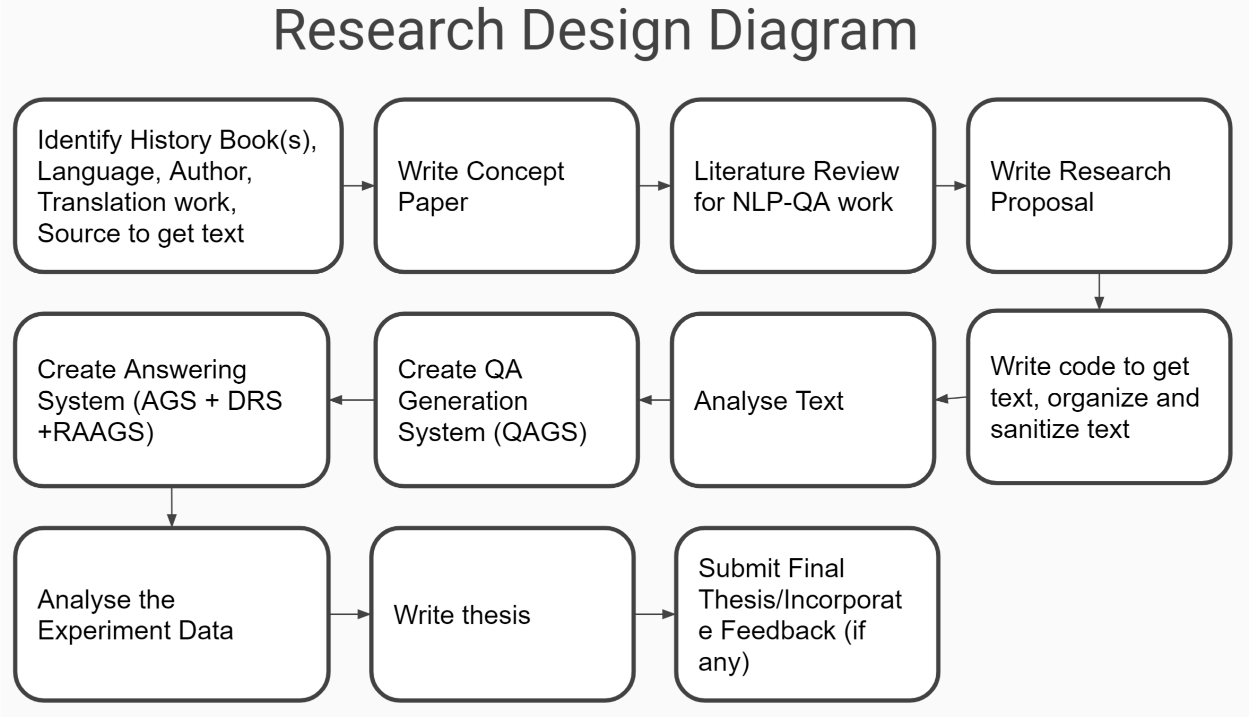

Research Design Diagram
Identify History Book(s), Language, Author, Translation work, Source to get text
Write Concept Paper
Literature Review for NLP-QA work
Write Research Proposal
Write code to get text, organize and sanitize text
Analyse Text
Create QA Generation System (QAGS)
Create Answering System (AGS + DRS +RAAGS)
Analyse the Experiment Data
Write thesis
Submit Final Thesis/Incorporat e Feedback (if any)

Figure 3.1: High-Level Research Design, Source: Author's own illustration, Source: Author's own illustration

- Have a reliable configuration management system in place
- Respect the IPR of everyone and proper citation. For this purpose, we use Mendeley and Overleaf,

3.2.7 Research Design Diagram

3.3 Population and Sample

3.3.1 Name of Mahabharat Parvas / Books

- Book01 : Adi Parva
- Book02 : Sabha Parva
- Book03 : Vana Parva

- Book04 : Virata Parva
- Book05 : Udyoga Parva
- Book06 : Bhishma Parva
- Book07 : Drona Parva
- Book08 : Karna Parva
- Book09 : Shalya Parva
- Book10 : Sauptika Parva
- Book11 : Stri Parva
- Book12 : Santi Parva
- Book13 : Anusasana Parva
- Book14 : Aswamedha Parva
- Book15 : Asramavasika Parva
- Book16 : Mausala Parva
- Book17 : Mahaprasthanika Parva
- Book18 : Svargarohanika Parva

3.3.2 Mahabharat Book Summary

Table 3.1: Mahabharat Book Summary, Source: Author's own illustration

Statistics of Mahabharat Book	Value
Letters	
Minimum Number of Letters in any Parva	16,840 (16.8 thousands)
Average Number of Letters in any Parva	777,699 (780 thousands)
Maximum Number of Letters in any Parva	2,650,870 (2.6 millions)
Words	
Minimum Number of Words in any Parva	2,866 (2.8 thousands)
Average Number of Words in any Parva	132,808 (133 thousands)
Maximum Number of Words in any Parva	454,079 (450 thousands)
Paragraphs	
Minimum Number of Paragraphs in any Parva	35
Average Number of Paragraphs in any Parva	623
Maximum Number of Paragraphs in any Parva	2,279 (2.2 thousands)
Parva	
Total Number of Parva	2100
Total Number of Letters in All Parvas	13,998,590 (13 millions)
Total Number of Word in All Parvas	2,390,561 (2.4 millions)
Total number of paragraph in All Parvas	11,217 (11.2 thousands)

3.3.3 Mahabharat Book - Parvawise Summary

*Sec : Number of Sections, *Words : Words Count In Parva, *Para : Paragraphs Count In Parva, *WordL : Avg Word Len In Parva, *WordC: Avg Word Count Section, *ParaC: Avg Para Count Section

Table 3.2: Mahabharat Book - Parvawise Summary, Source: Author's own illustration

	Sec*	Words*	Para*	WordL*	WordC*	ParaC*
Book01	236	226,993	1725	5.89	962	7.31
Book02	79	73,308	421	5.95	928	5.33
Book03	312	321,351	1285	5.79	1030	4.12
Book04	72	60,134	338	5.84	835	4.69
Book05	198	187,212	719	5.84	946	3.63
Book06	124	147,486	651	5.97	1189	5.25
Book07	198	245,217	815	5.97	1238	4.12
Book08	96	137,646	331	5.94	1434	3.45
Book09	64	90,217	274	5.94	1410	4.28
Book10	18	21,038	154	5.87	1169	8.56
Book11	25	21,048	88	5.83	842	3.52
Book12	363	454,079	2279	5.85	1251	6.28
Book13	167	275,073	1334	5.74	1647	7.99
Book14	92	80,554	487	5.94	876	5.29
Book15	39	30,243	159	5.9	775	4.08
Book16	8	7,636	38	5.83	955	4.75
Book17	3	2,866	35	5.89	955	11.67
Book18	6	8,460	84	5.9	1410	14.00
Mean	117	132,809	623	6	1103	6

3.3.4 Mahabharat Book - Summary of Sections

Table 3.3: Mahabharat Book - Summary of Sections, Source: Author's own illustration

Mahabharat Book - Summary of Sections	Value
Total Sections in Book	2,100
Letters	
Minimum Number of Letters in any Section	514
Average Number of Letters	6,665 (6.5 K)
Maximum Number of Letters in any Chapter	84,950 (85 K)
Words	
Minimum Number of Words in any Section	90
Average Number of Words	1,138
Maximum Number of Words in any Chapter	14,850 (15 K)
Paragraphs	
Minimum Number of Paragraphs in any Section	2
Average Number of Paragraphs	5.4
Maximum Number of Paragraphs in any Chapters	86

3.3.5 Word Length Analysis

Word Length Distribution in Book

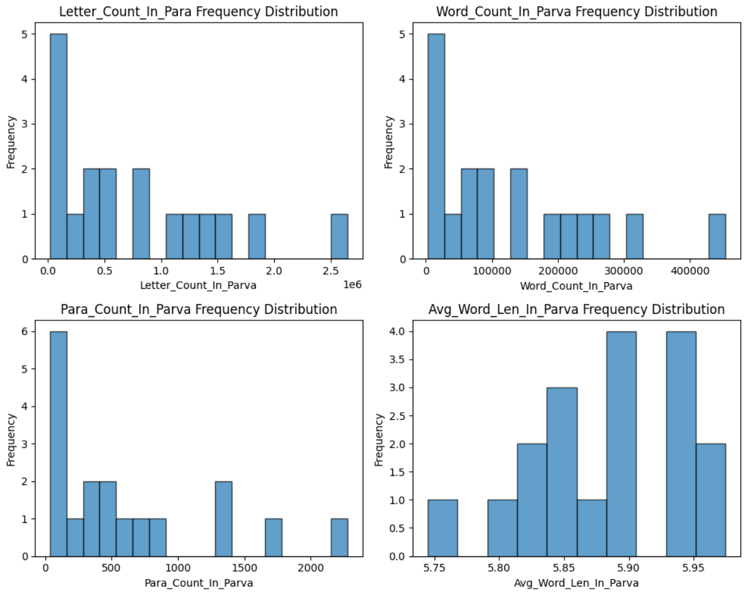


Figure 3.2: Word Length Distribution in Book, Source: Author's own illustration, Source: Author's own illustration

Word Length Distribution in Sections of Book

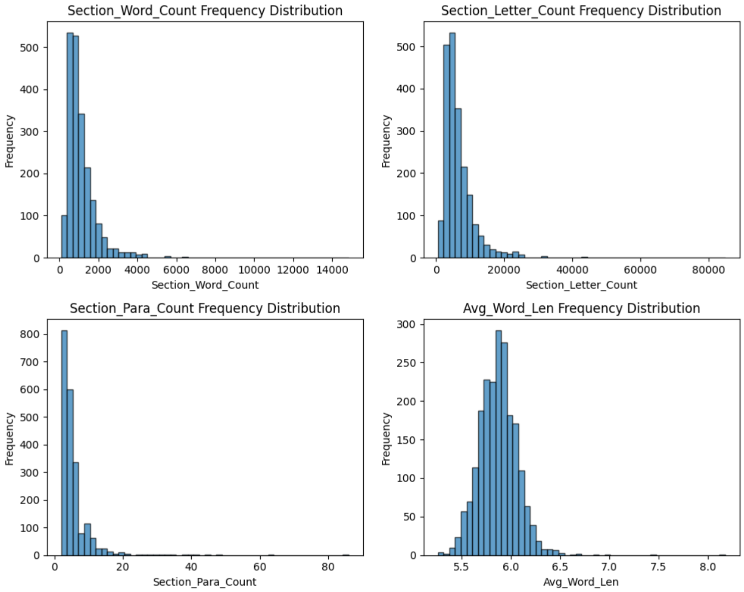


Figure 3.3: Word Length Distribution in Sections of Book, Source: Author's own illustration

3.3.6 Word Frequency Analysis

Top 50 Words After merging all the Parava and sections of Mahabharat book, data cleaning, removing stop words we found it has approx 30K unique words. Frequency of the top 50 words can be noted from the diagram ”Top 50 Most Common Words” **Repetition of Words** Any book which has more words but their frequency of usage is less, that book is not easy to read. It challenges the reader every time because the reader is reading the next word as a new word every time. Imagine a book that has 100K words and no word is repeated. The vocabulary size of any book decides the quality of a book, but how frequently those words are used decides how challenging or interesting it may be to read. Out of 30K unique words, less than 2000 words have having frequency of more than 100 times, remaining 28000 words have a frequency of more than 100 times. This translated book is a very interesting and tough-to-read kind of book.

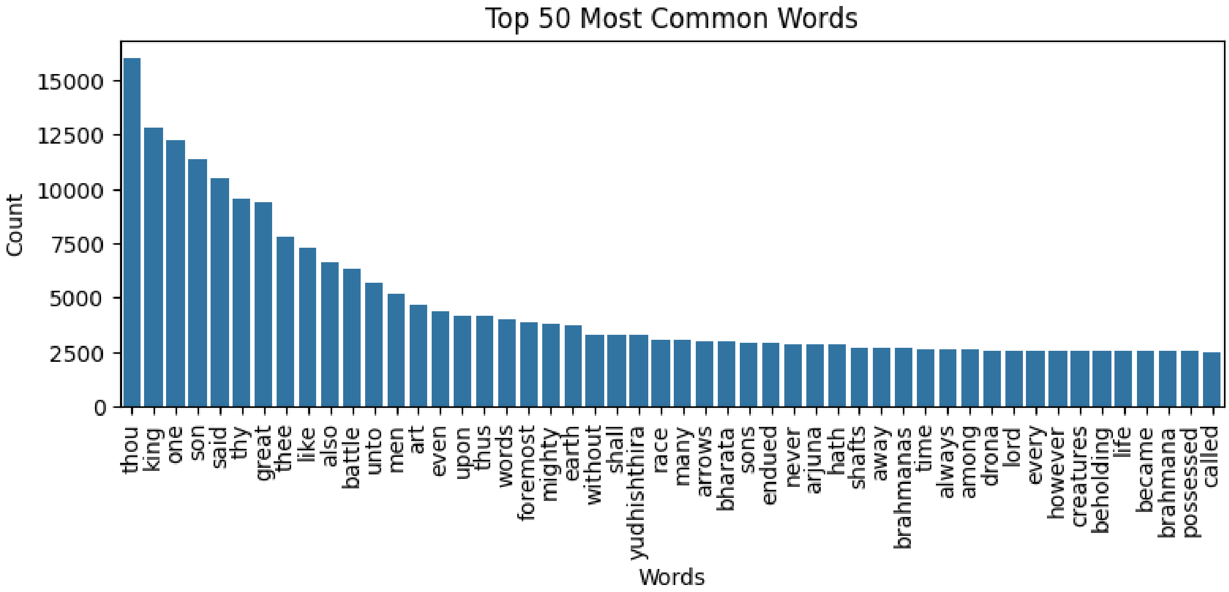


Figure 3.4: Frequency of top 50 words, Source: Author's own illustration

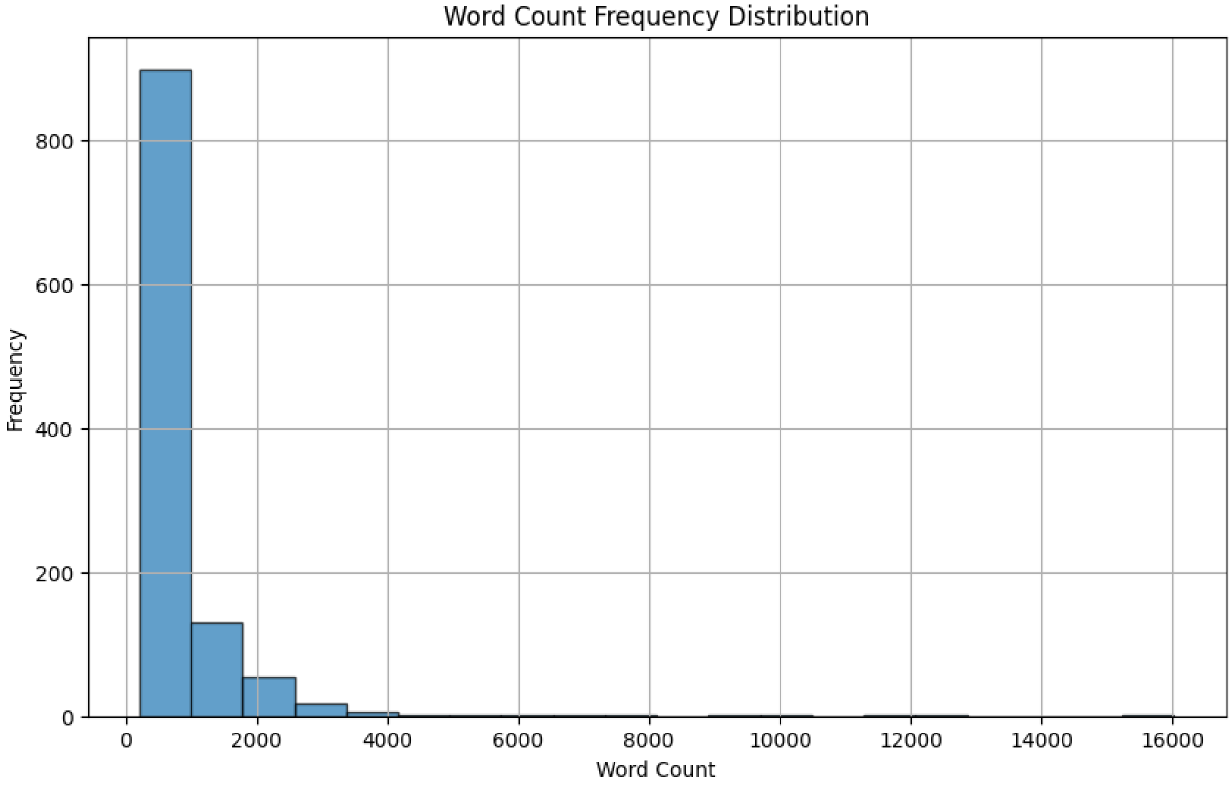


Figure 3.5: Word Frequency Distribution in Book, Source: Author's own illustration

3.3.7 Text Data Summary

Mahabharat Book Chunk Statistics

What is a chunk?

Because of today’s hardware, software architecture, NLP technologies we cannot process the entire Parva of Mahabharat at one time on a single piece of hardware. Nor it is economically viable. Due to this reason, we need to break text into smaller groups. These groups are called chunks. Depending upon the context chunk is also referred to as ”context” or ”document”. We will use these chunks to define the context to create QA and get answers to questions.

The entire book has 18 Parvas and 2100 sections. The length of text in these books and sections is not equally distributed. Processing entire text for our research work is expensive and time-consuming. And, keeping the goal of our research it is not necessary that we process all the Parva of the book. Therefore we selected some Paravas and some sections for our work.

Choosing the Right Chunk Size

Choosing the right size chunk is a challenging task and a matter of experiments. Therefore we need to look at what max token a prediction model/LLM can handle. In our work, we are using SentenceTransformer-based sentence embedding. Each embedding can handle certain maximum tokens in a sequence. For our work we have used 5 sentence embeddings, you can refer to them in table 3.9. If the chunk size is larger than the max size then the text is truncated. To avoid this we can use the window technique. However, in our work, we have not used the window technique. Suppose the max token size a model can handle is 512 tokens and our input text has 2,000 tokens then 2000/512 = 3 vectors of 512 words embedding and 1 vector of 464 token+48 padded tokens will be averaged to create sentence embedding vector of 2,000, in this process we may lose important information. Smaller chunks reflect more accurate semantic meaning after creating embedding. But, sometimes might lose the bigger picture and might sound out of context, making it difficult for the LLM to properly answer the user's query with limited context per chunk.

From 2100 sections, 2746 chunks were created, and dirty chunks were removed. We used langchain's textsplitter for this work with the following setting. chunk_size=8000, chunk_overlap=300. Chunks which has only page numbers, chapter name, and length <= 50 characters were considered dirty chunk. We can control the chunk size but using this library we cannot control what text will go in which chunk. During the splitting process when the program reaches towards the end of the text, whatever text is left is assigned a chunk number and this may be a dirty chunk. 1049 chunks were filtered out from our project work. The reason for that is those chunks are very long or very short. Even after we set the chunk_size=8000, we got chunks that were more than 8000 characters, it happens because of longer paragraphs. We used only those chunks which has more than 100 words and less than 1000 words in length. Finally, we have only 1697 chunks in our chunk dataset. Some of these chunks were selected to generate questions and answers. How many QA can be created from a given chunk of a certain size? This is not an easy question. However, we decided we would generate only one QA from a chunk that has a length between 100 and 400 words. If the chunk is longer than 400 then the number of questions = size of the chunk in words / 150. For example, if a chunk has a size of 780 then 780/150 = 5 (approx) questions will be created.

Summary:

- Sections: 2,100
- Chunk Created (after dirty chunk removed): 2,746

- Smaller (<100 words), larger (>1000 words) chunks removed: 1,049

- Chunks available for project: 1,697

- The mean length of a chunk in letters is 3,753 letters.

- The mean length of a chunk in words is 640 words.

Chunks Used for Question Answering Work - QAGS-FE

Following is statistics of Mahabharat Book's Chunks Used for Question Answering Work.

Table 3.4: Chunk Used for QA Work, Source: Author's own illustration

Statitics	Value
Number of Chunks	331
The mean number of words in chunk (median= 605)	592
The mean number of tokens in a chunk (approx)	788
The mean number of letters in a chunk	3453
Letters used to create QA	1,143,265
Words used to create QA	196,179
Approx tokens used to create QA	260,926

Chunks Used Parvawise

Parva Used with Front End of ChatGPT: Parvawise chunks used for QA Generation. These chunks are used for Manual Prompt Engineering. We created the HBQADataset dataset from these chunks. QAs of these chunks are used for model finetuning.

Table 3.5: Parvawise Chunk Used: With Front End Prompt Engineering, Source: Author’s own illustration

Parva Number	Number of Chunks
Book01	33
Book02	11
Book03	236
Book04	32
Book05	9
	321

Parva Used with ChatGPT API: These chunks are used with ChatGPT API Prompt Engineering. Parvawise chunks used for QA Generation. We have not used QAs of these chunks for model finetuning.

Table 3.6: Parvawise Chunk Used: With ChatGPT API Prompt Engineering, Source: Author’s own illustration

Parva Number	Number of Chunks
Book01	139
Book02	10
Book03	25

Parva Used with ChatPDF API: While generating QA using ChatPDF API we used only Book01 and 334 chunks of this Parva. Parva 01 has 334 pages and each page is passed as context (chunk) to API to create questions. We have not used QAs of these chunks for model finetuning.

3.4 Data Collection and Instrumentation

3.4.1 Folder Structure, Accounts, Software Used

Due to security, cost of services, services for which we already have at the organization level, the need for different kinds of resources for this project, it looks like files are spread all over different places. Keeping the following constraint in mind we created a folder structure, which decides what to be kept where.

- Overleaf: This online application is used for thesis writing. It is an extremely flexible, powerful tool for thesis writing where you can take care of TOC, tables, diagrams, appendix, references, footnotes, page layouts, report layouts, orientations, bibliography, hyperlinking, and multiple language/script text in a fully controlled way. To achieve all this in MS Word, Google Docs, or LibreOffice is much more complicated and unsustainable. Overleaf is free in cloud software for limited use.

- Google Workspace: This product of Google allows to create presentation, collaboration/sharing work, and data analysis. We already paid for these services. We are using it for creating presentations, images, and flow diagrams. We are using google-sheet to clean the data before we put it in our LaTex report.

- Kaggle Kernel and Google Colab(Pro): For finetuning model/ running jupyter notebooks we need GPU resources and for that, we are relying on Kaggle Kernel and Google Colab. If we need more resources we can buy them quickly at an economical price from Google. sometimes Google Colab account doesn’t allow GPU when it is exhausted or for some other reason, so we need to use other accounts if needed. In that situation, Google Drive files which are accessible in the first Colab account’s Google Drive folder should be available in another account’s Google Drive. We tried to share those folders and files between Google Drive but no luck. Therefore, if we have to temporarily use GPU from another account we need to manage this by loading those files manually again in this Google Colab account.

- GitHub is for source code version management and collaboration.

- Documents kept in GitHub cannot be edited in Google Workspace. They cannot be accessed either. Therefore all the project documents are kept in Google Docs and GitHub is not used for this purpose.

- Documents kept in GDrive cannot be versioned the way they can be in GitHub. We are working on these documents in Notepad++ and using markdown language. But they are not versioned. But notepad++ takes care of the backup of these files.

- GDrive Mapped to a local folder, for documentation purposes.

- GDrive Folder is used to keep code-generated outputs. If the source code needs some data or configuration information that is also kept in GDrive. It is not kept in GitHub, because then it is challenging to GitHub files and manipulate them from Colab.

3.4.2 Purpose and Type of Content in Folders

From a process perspective, if we want to know which process or for which task we are using what folder and where it is in the cloud is mentioned below.

1. D:\github\HBQA\HBQA: This is the source code folder and it is under GitHub. Visual Studio Code IDE is used for coding in the local access and versioning. The code is versioned in this folder. Colab is used for GPU-related experiments and versioning for that is maintained by Google Colab.

2. H:\My Drive\HBQA\Data: These folders need to be accessible for reading/writing from colab and local machine. This is part of official Google Workspace. We may need a paid version of colab for some experiments. It is easier to make access from this location. It is easier to make payments via an office account.

3. D:\github\HBQA\HBQA\DBA□SSBM: For SSBM-supplied templates, it is under GitHub.

4. D:\github\HBQA\DBA: Documentation folder but this is not under GitHub. It is synced to Google Drive. Notepad++ is used to create and update files in this folder. No version control is needed. The following kinds of files are under this folder. Files in this folder are idea or WIP files. We don't want to share these notes/documents with anyone. This folder is mapped to gdrive for sync purposes. An office Google account is used for this work. Therefore I can use Google Workspace features which are available for any corporate account. The types of content in the folder are as follows.

    - Thesis backup
    - PPT
    - Images created
    - Notes related to thought/ideas
    - WIP, before overleaf or ppt or idea.
    - Research papers downloaded for bib and reading purposes.
    - Research papers not available on Mendeley are downloaded from the internet and uploaded into Mendeley from here.

5. H:\My Drive\HBQA: All the final models are kept in this folder, During model building, generated model files are bulky. They can be easily saved on the official gdrive account. It needed to pay for 100GB for a couple of months. These models need to be accessible from other Google-Colab using other Google accounts for inference.

6. H:\My Drive\HBQA\Data : All the files created at different stages in this project are kept in this folder. It is under gdrive.

7. H:\My Drive\HBQA\CorpusBooks: Mahabharat book text corpus is under this folder. It is under gdrive.

8. H:\My Drive\config\hbqa.txt : This is configuration file. It is under gdrive. All API and project-specific constant variables are kept in this file. It is secured and not available to unauthorized person

3.4.3 Process Code, Files Generated

All the files generated by our project are available in gdrive. Process code, process description, and generated file information is as below.

Process 01 : Data Collection Process generates file 01.0_Mahabharat_ Metadata.csv

Process 02 : Section File Merging into 18 Parva Files Mahabharata_01.pdf, mahabharata_02.pdf, mahabharata_03.pdf, mahabharata_04.pdf, mahabharata_05.pdf, mahabharata_06.pdf, mahabharata_07.pdf, mahabharata_08.pdf, mahabharata_09.pdf, mahabharata_10.pdf, mahabharata_11.pdf, mahabharata_12.pdf, mahabharata_13.pdf, mahabharata_14.pdf, mahabharata_15.pdf, mahabharata_16.pdf, mahabharata_17.pdf, mahabharata_18.pdf

Process 03 : Creating chunks from section files

- 03_Chunked_Book.csv
- 03_Chunked_Book_Incl_Small_Chunk.csv
- 03_Chunked_Book_All.csv
- 03_Chunked_Book_Excluded.csv

Process 04 : Creating Prompt from Chunked Data

- 04.1_Chunk _with _Prompts.csv

- 04.1_Chunk_with_Prompts_Incl_Small_Chunk.csv
- Embedding of HBQADataset with Different Embedding Model
- 04.3_HBQA_Chunk_Vector_all _distilroberta_v1.csv
- 04.3_HBQA_Chunk_Vector_all_ MiniLM_L6_v2.csv
- 04.3_HBQA_Chunk_Vector_all_ roberta_large_v1.csv
- 04.3_HBQA_Chunk_Vector_multi_ qa_distilbert_cos_v1.csv
- 04.3_HBQA_Chunk_Vector_multi_ qa_mpnet_base_dot_v1.csv

Process 05 : This is manual process of executing prompts on ChatGPT, copying and pasting ChatGPT response to 04.2_Chunk_ with_Prompts +ChatGPT_Response.csv

Process 06 : Cleaning ChatGPT response and creating Question, Answer, Response. Assign question id, chunk id to questions answer

06_HBQA_Manual_ with_Chunk.csv

Process 08.1 : Results of Document Retriever System (DRS)

- 08.1_Predict_DocumentId_ for_Ques_all_distilroberta_v1.csv
- 08.1_Predict_DocumentId_for_Ques_all _MiniLM_L6_v2.csv
- 08.1_Predict_DocumentId_for_Ques_all _roberta_large_v1.csv
- 08.1_Predict_DocumentId_for_Ques_multi _qa_distilbert_cos_v1.csv
- 08.1_Predict_DocumentId_for_ Ques_multi_qa_mpnet_base_dot_v1.csv

Process 08.2 : Metrics of Document Retriever System (DRS).

- 08.2_Predict_DocumentId_for_Ques_Metrics_all_distilroberta_v1.csv
- 08.2_Predict_DocumentId_for_Ques_Metrics_all_MiniLM_L6_v2.csv

- 08.2_Predict_DocumentId_for_Ques_Metrics_all_roberta_large_v1.csv
- 08.2_Predict_DocumentId_for_Ques_Metrics_multi_qa_distilbert_cos_v1.csv
- 08.2_Predict_DocumentId_for_Ques_Metrics_multi_qa_mpnet_base_dot_v1.csv

Process 09.11 : Answer predicted by every model and embedding of the answer by SentenceTransformer is kept in these files.

- 09.11_T5Predicted_Ans_E1.csv
- 09.11_T5Predicted_Ans_E2.csv
- 09.11_T5Predicted_AnsVector_E2.csv
- 09.11_T5Predicted_AnsVector_E1.csv

Process 09.12 : Calculated Score of Each Model's Prediction are kept in these files.

- 09.12_T5Predicted_Ans_Score_E2.csv
- 09.12_T5Predicted_Ans_Score_E1.csv

Process 10-RAAGS: All the RAAGS process documents are kept in this folder.

Process 21 : Question Answer Generation using API.

- 21.1_ChatPDF_Mahabharata_01_QA.txt
- 21.1_HBQA_ChatPDF_QA.xlsx
- 21.2_ChatGPT_API_Response_QA_Final.csv
- 21.2_QA_Final_ChatGPT_API.csv

Process 22 : Cleaning API Generated Response and creating QA dataset 22_HBQA_Auto_with_Chunk.csv

Process 31 : Analysis of Data Process. Some files generated during this process are as below.

- 31.1_Book_Statistics.md
- 31.2_Book_Section_Statistics.md
- 31.3_Chunk_Statistics.md
- 31.4_QA_Statistics.md
- 31.5_Words_Statistics
- 31.6_All_Model_QA_Cosine
- 31.6_All_Model_QA_Cosine
- 31.7_Cosine_Score

3.4.4 Thesis Report Files

Overleaf is used for thesis writing. There are different folders in Overleaf to keep different kinds of files.

- Appendix: All appendices in this folder
- Backup: Chapter backup is maintained in this folder
- Chapters: All 5 chapters of this are in this folder
- FrontPages: All front-page material like title, dedication, toc, table, figure, and acknowledgment goes in this folder.

- hbqa.bib: All the bibliography links are in this file. These are taken from Mendeley

- Images: All workflow + images created/ referred are kept in this folder.

- main-page.tex: This is the main tex file. It links all the chapters and loads all necessary packages.

3.4.5 Source Code Files

All the code we wrote for this project is available in the Github. Github project code is HBQA. All api keys (secured keys) or configuration details are kept in a separate file which is at different gdrive folder. We are reading the configuration from that file during code execution. First two letter of the code file refers to the process name.

Table 3.7: Source Code File and Purpose, Source: Author's own illustration

Source Code File	Purpose
01-Extract_Book_From_URL.ipynb	Source code for web-scrapping
02-Merge-Textfiles-into-chapters.ipynb	Merging section text files, creating a final pdf file with page number
03-Text-Splitting.ipynb	Creating chunk
04-QAPrompt-Generation.ipynb	Creating Prompt using Chunk
06-Create-QA-Dataset_from_Manually _CorrectedQA.ipynb	ChatGPT Response Cleaning and creating QA dataset
07_HBQA_Embedding.ipynb	Creating Chunk Embedding
08_Document_Retrieval.ipynb	Document Retrieval System
09_1_HBQA_T5_E1.ipynb	T5 Model Finetuning code
09_2_HBQA_GPT2_Medium.ipynb	GPT2 Model Finetuning code
21_1_QAGen_HBQA_ChatPDF.ipynb	QA generation using ChatPDF API
21_2_QAGen_HBQA_ChatGPT.ipynb	QA generation using ChatGPT API
22-Create-QA-Dataset.ipynb	Creating QA Dataset from ChatGPT API Reponse
31_HBQA_Analysis.ipynb	EDA and Other Data Analytics

3.5 Procedure

3.5.1 Overall Approach

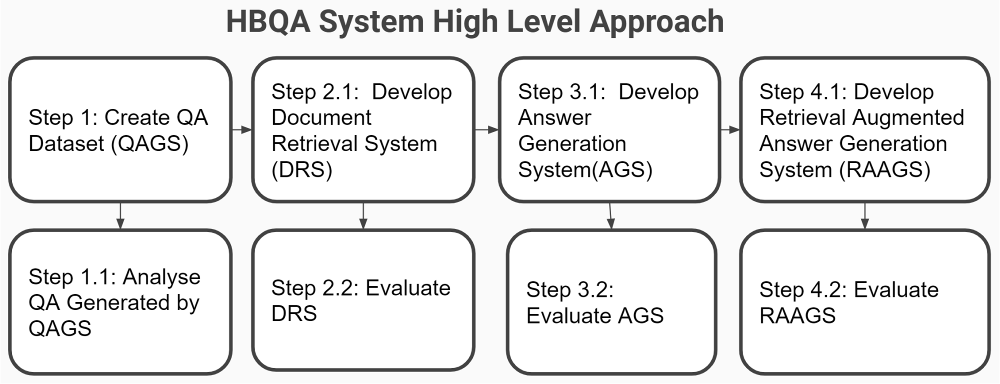

HBQA System High Level Approach
Step 1: Create QA Dataset (QAGS)
Step 2.1: Develop Document Retrieval System (DRS)
Step 3.1: Develop Answer Generation System(AGS)
Step 4.1: Develop Retrieval Augmented Answer Generation System (RAAGS)
Step 1.1: Analyse QA Generated by QAGS
Step 2.2: Evaluate DRS
Step 3.2: Evaluate AGS
Step 4.2: Evaluate RAAGS

Figure 3.6: HBQA System High-Level Approach, Source: Author’s own illustration

Overall Approach for Developing HBQA System

1. **QA Dataset Creation**: Using ChatGPT or other Alternative software Generate Question Answers from a Historical Book.

2. **Deliverable**: 1000+ QA Dataset on a Historical Book called Mahabharata

3. **Model Development:** Explore Zeroshot models, Developing multiple fine-tuned models for answer generation

4. **Deliverable**: The best answer generating finetuned model.

5. **Model Evaluation**: Compare Model Generated answer to the Reference Answer.

6. **Deliverable**: Performance of model on various quality metrics.

7. **Search Answer: Search Answer of Question in the absence of context**

8. **Deliverable**: An approach, how to search for answers to questions from a big book of many books.

3.5.2 Building Question Answer Generation System (QAGS)

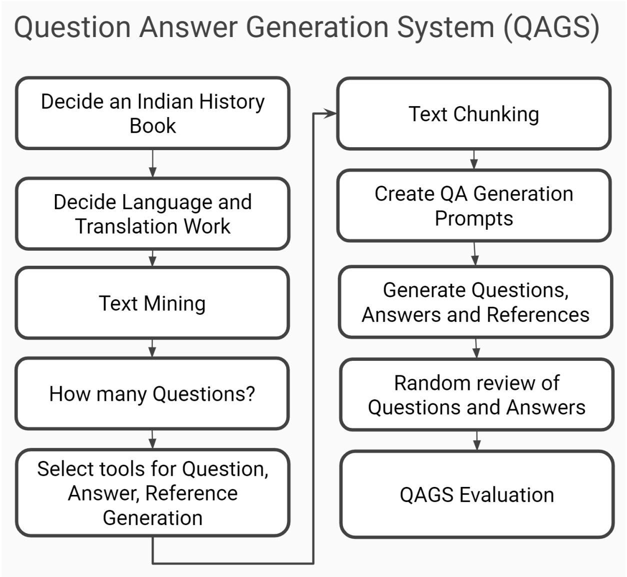

Question Answer Generation System (QAGS)
Decide an Indian History Book
Decide Language and Translation Work
Text Mining
How many Questions?
Select tools for Question, Answer, Reference Generation
Text Chunking
Create QA Generation Prompts
Generate Questions, Answers and References
Random review of Questions and Answers
QAGS Evaluation

Figure 3.7: QA Dataset Creation, Source: Author's own illustration

Steps in QAGS

1. **Decide an Indian History Book**: The Mahabharata Selected

2. **Decide Language and Translator**: English Language Translation of KM Ganguli Selected.

3. **Text Mining**: Web Scraping of 18 Parvas, 2100 Sections: Download 2100 text files of total 14MB Size.

4. **How many Questions?**: Decide how many/ what kinds of questions are needed. Select a chapter to create Questions from. Selected Book 3, all the sections of this Parva are taken, apart from this smaller section of Parva 1, 2,4,5 237 also taken. We need a minimum of 1000 descriptive questions.

5. **Select tools for Question, Answer, and Reference Generation**: Decide what tool to use for creating questions, answers, and references. Decided to use ChatGPT, ChatGPT API, ChatPDF.

6. **Text Chunking**: Used Langchain Library for splitting the Text.

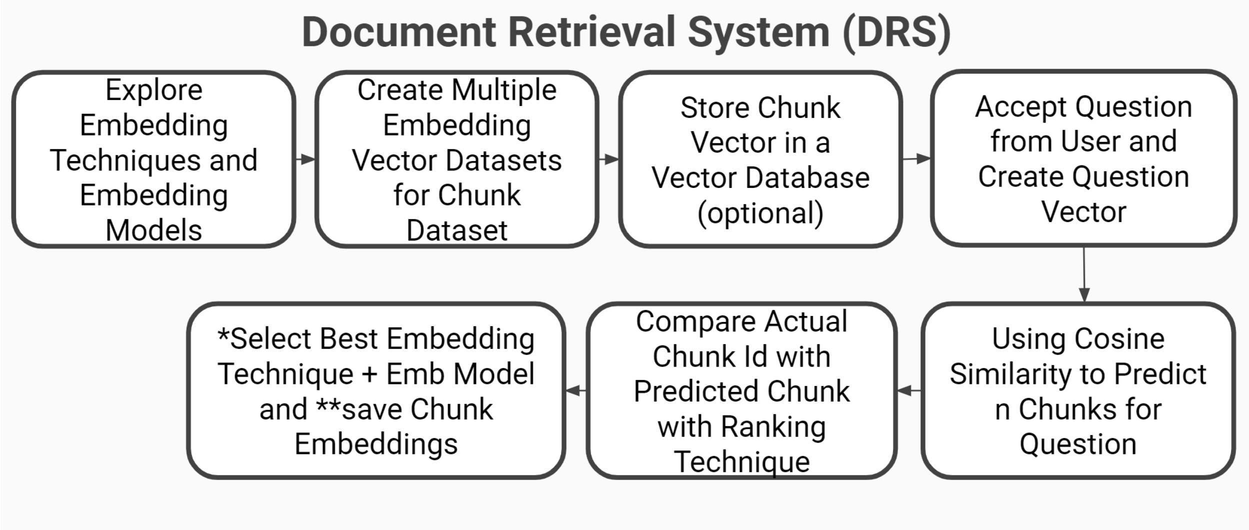
7. **Create Question Prompts**: Create Queries (ChatGPT Prompts) for QA. Created a Prompt dataset.

8. **Generate Questions, Answers, and References**: Use selected tools to Create Questions, Answers, and References from earlier created Prompts. Mahabharata-HBQA Dataset Created.

9. **Random review of Questions and Answers**: Remove duplicate, ambiguous, contextless, meaningless questions. Mahabharata-HBQA Dataset Updated.

10. **Train Test Split**: Create Train and Test split on Mahabharata-HBQA Dataset

3.5.3 Building Document Retriever System (DRS)


Document Retrieval System (DRS)
Explore Embedding Techniques and Embedding Models
Create Multiple Embedding Vector Datasets for Chunk Dataset
Store Chunk Vector in a Vector Database (optional)
Accept Question from User and Create Question Vector
Using Cosine Similarity to Predict n Chunks for Question
Compare Actual Chunk Id with Predicted Chunk with Ranking Technique
*Select Best Embedding Technique + Emb Model and **save Chunk Embeddings

*In RAAGS, we will use this embedding model/technique to embed the question and search in chunk database.
*In AGS, we will also use this embedding to embed the predicted answer and reference answer. This helps us understanding the difference between predicted answer and reference answer.
**In the production environment use Vector Database like Pinecone to store this.

Figure 3.8: Document Retriever Model, Source: Author's own illustration

Steps in DRS

1. **Explore Word and Sentence Embedding Models:** Selected SentenceTransformer architecture and explored following embedding (1) multi-qa-distilbert-cos-v1, (2) all-MiniLM-L6-v2, (3) all-distilroberta-v1, (4) all-mpnet-base-v2, (5) all-roberta-large-v1

2. SentenceTransformer has dozens of Embedding Models, they are different in size, dimension, normality, and corpus used to create embedding.

3. **Create Multiple Embedding Vector Datasets**: Using multiple embedding models, **create multiple Vector Embeddings for Question, Reference Answer, and Chunk (Context).**

4. **Predict Chunk for Question**: Using **cosine similarity** between the question vector and chunk vector select top 10 chunks for a given question.

5. **Select the best Embedding Model**: We compared the Predicted Chunk Id and Actual Chunk Id and select the embedding ”all-mpnet-base-v2” model.

3.5.4 Building Answer Generation System (AGS)

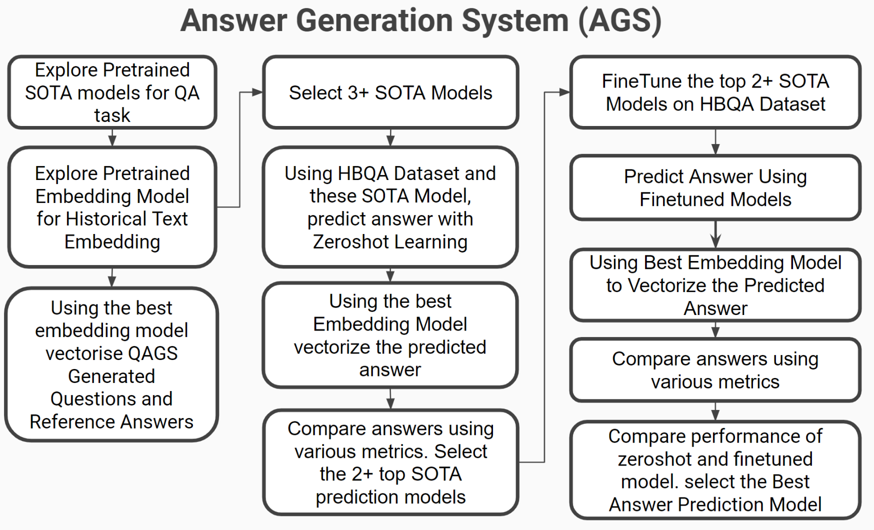

Answer Generation System (AGS)
Explore Pretrained SOTA models for QA task
Explore Pretrained Embedding Model for Historical Text Embedding
Using the best embedding model vectorise QAGS Generated Questions and Reference Answers
Select 3+ SOTA Models
Using HBQA Dataset and these SOTA Model, predict answer with Zeroshot Learning
Using the best Embedding Model vectorize the predicted answer
Compare answers using various metrics. Select the 2+ top SOTA prediction models
FineTune the top 2+ SOTA Models on HBQA Dataset
Predict Answer Using Finetuned Models
Using Best Embedding Model to Vectorize the Predicted Answer
Compare answers using various metrics
Compare performance of zeroshot and finetuned model. select the Best Answer Prediction Model

Figure 3.9: Answer Generation Model, Source: Author's own illustration

Steps in AGS

1. **Identify SOTA models for QA tasks:** Selected T5, BERT, DistilBERT, FlanT5, RoBERTa, GPT2, Bloom models

2. **Identify Embedding Model** : Used the same embedding that was used in DRS.

3. **Experiment Zeroshot Learning**: Using QAGS-generated Questions and Chunks predict answers with SOTA models (without any finetuning)

4. **Evaluate Zeroshot Model**: Using BLEU score, ROUGE score, Precision, Recall metrics computed the quality of a predicted answer.

5. **Select SOTA model for Finetuning**: Selected T5 and Flan-T5.

6. **FineTune each SOTA Model**: Each selected model is finetuned on the QAGS generated Dataset.

7. **Generate Answer**: Generate each question's answer with Every FineTuned Model

8. **Create Vector Embedding**: Create embedding for each Predicted answer using the Embedding model.

9. **Compute the cosine** distance between the predicted answer and the reference answer

10. **Compute Answer Correctness**: Using the BLEU score, ROUGE score, Precision, Recall, and Cosine metrics computed the quality of a predicted answer.

11. **Compare Model Performance**: Compare model performance against BLEU, ROUGE, Precision, Recall, and Cosine metrics and select the answer generation model.

Answer Prediction Models

These transformer-based models are used in our work.

For Zeroshot

1. T5 and Flan-T5: Sequence-to-sequence models

2. DistilBERT and RoBERTa: Encoder-only models

3. BERTSquad: Encoder-only model specifically designed for question-answering tasks

4. BigBird: Sequence-to-sequence model with long-range dependencies

5. Bloom: Decoder-only model

For Finetuning

1. T5 or Text-to-Text Transfer Transformer, is a sequence-to-sequence transformer language model developed by Google AI.
2. Flan-T5 : Flan (Finetuned Language) T5 is finetuned version of T5 from Google. It was pre-trained on a massive dataset of text and code, and then fine-tuned on a mixture of tasks, including question answering, summarization, and translation.

What are the three types of Transformers?

- **Sequence-to-sequence models**: These models are designed to take a sequence of input tokens and generate a sequence of output tokens. They are commonly used for tasks such as generative question answering, machine translation, and text summarization. Example: T5, FlanT5, BART
- **Encoder-only models**: These models are designed to encode a sequence of input tokens into a representation that can be used for downstream tasks such as classification and extractive question answering. They read sentences from both directions. Example: BERT, RoBERTa.
- **Decoder-only/ Autoregressive models**: These models are designed to generate a sequence of output tokens from a latent representation. They are commonly used for tasks such as text generation and code generation. They read sentences from only one direction. Example GPT

Limit of Length of Generated Answer by Models

Below is the list of popular models which can handle question answering task. But we need to keep in my the maximum token limit of input text, which these models can handle for this task. If chunk or input text becomes more than the maximum token limit then text is truncated and it cannot be used for answering questions.

Table 3.8: Token Limits of Transformers, Source: Author's own illustration

LanguageModel	Token-Limit
BERT	512
T5	512
Open Assistant (Pythia family)	2048
Dolly	2048
GPTJ6B	2048
GPTNeoX20B	2048
ChatGPT (GPT3.5Turbo)	4096
StableLMAlpha	4096
OpenLLaMA	2048
GPT4	32K
MPT7B	84k
Claude	100k

To take care of this problem there are techniques available but those make the system slow.

Techniques for optimizing LLM performance

There are two popular techniques that can be used to optimize the performance of LLMs and ensure that they are generating accurate and relevant responses.
One, **fine-tune the model on a specific dataset or task**. This involves taking a pre-trained LLM and further training it on a specific set of data to improve its performance on that task. By fine-tuning the model on a specific dataset or task, developers can improve the accuracy and relevance of the model's responses for that particular use case.
Two, **To use prompt engineering**. This involves crafting high-quality prompts or questions that are tailored to the specific task at hand. By providing the model with clear and relevant prompts, it can more easily generate accurate and relevant responses.
Prompt engineering can involve techniques such as

- adding context to the prompt,

- using multiple prompts to provide more diverse input to the model,
- and incorporating additional constraints or requirements into the prompt.

Challenges During Model Optimization

These points apply to all model training whether classical ML model deep learning model or LLM

1. Overfitting the model to the training data. Overfitting occurs when the model becomes too closely aligned with the training data and is not able to generalize to new or unseen data. To avoid overfitting, it is important to use diverse and representative training data and to regularly update and retrain the model with fresh data.

2. Biased data or prompts. If the training data or prompts are biased toward a particular demographic or viewpoint, the model's responses may also be biased or discriminatory. To avoid bias, it is important to use diverse and representative training data and prompts and to test the model's responses for fairness and accuracy.

3. LLMs can sometimes generate responses that are factually incorrect or misleading. To avoid these issues, it is important to carefully evaluate the model's responses and to incorporate additional sources of information or fact-checking mechanisms into the process.

4. It is important to be mindful of ethical considerations when developing and deploying LLMs. This includes ensuring that the model is being used for ethical purposes and that it is not being used to spread misinformation or to harm individuals or groups.

3.5.5 Retrieval Augmented Answer Generation System (RAAGS)

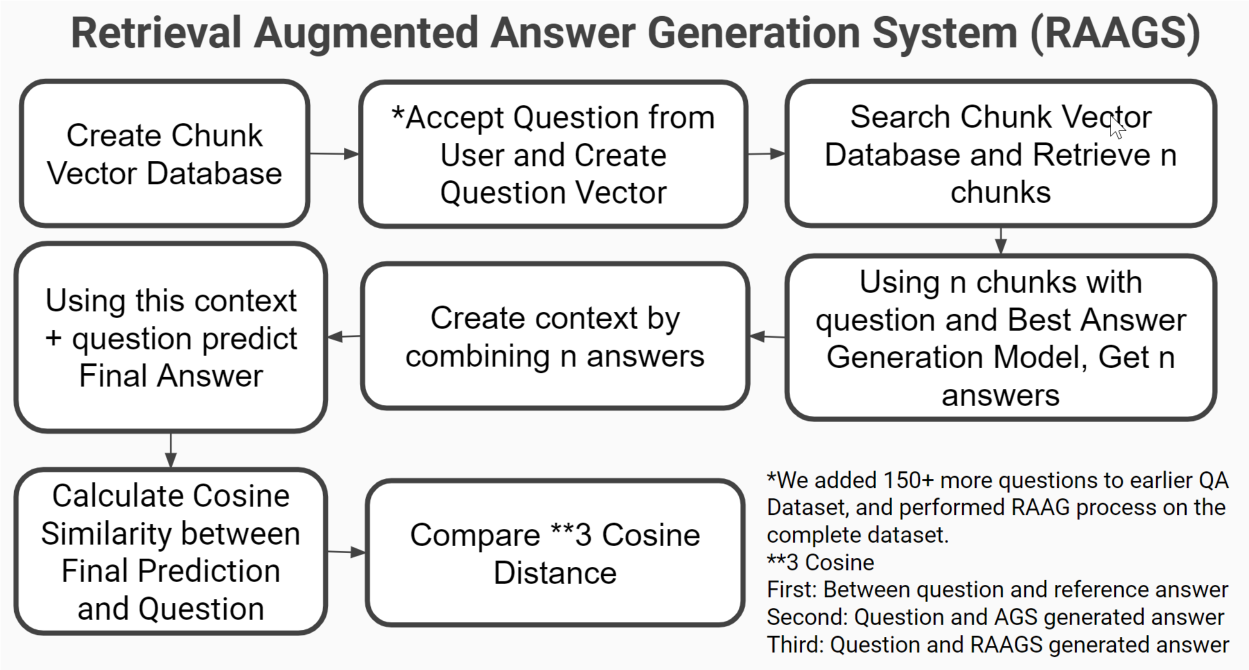

Retrieval Augmented Answer Generation System (RAAGS)
Create Chunk Vector Database
*Accept Question from User and Create Question Vector
Search Chunk Vector Database and Retrieve n chunks
Using n chunks with question and Best Answer Generation Model, Get n answers
Create context by combining n answers
Using this context + question predict Final Answer
Calculate Cosine Similarity between Final Prediction and Question
Compare **3 Cosine Distance
*We added 150+ more questions to earlier QA Dataset, and performed RAAG process on the complete dataset.
**3 Cosine
First: Between question and reference answer
Second: Question and AGS generated answer
Third: Question and RAAGS generated answer

Figure 3.10: Retrieval Augmented Answer Generation System (RAAGS), Source: Author's own illustration

Steps in RAAGS

1. Create Chunk Vector Database: Chunk Vector Database using SentenceTransformer and all-mpnet-base-v2 embedding model

2. Accept Question from User and Create Question Vector: Create embedding vector using all-mpnet-base-v2 embedding model.

3. Search Chunk Vector Database and Retrieve n chunks: n=5, Five chunks were predicted for every question.

4. Using n chunks with a question and Best Answer Generation Model, Get n answers: T5 Finetuned Model which was finetuned over 30 epochs was used.

5. Create context by combining n answers: Chunk corresponding to predicted chunk id is concatenated in the same order they were predicted.

6. Using this context + question predict Final Answer: Question along with newly created chunk was given to T5 finetuned model to predict the answer. It gives us the final answer.

7. Calculate the Cosine Similarity between the Final Prediction and the Question

8. Compare *3 Cosine Distance

   (a) First: Between question and reference answer

   (b) Second: Question and AGS generated answer

   (c) Third: Question and RAAGS generated answer

3.5.6 LLM security

: Although these points apply to all kinds of models just because LLMs are dealing with extremely huge data compared to other models, it doesn't mean they are giving balanced output. Secondly during training when all kinds of text are fed to LLM it is extremely important not to give that data which you don't want to see in the model-generated output. When we are creating a model on historical text we need to understand no history is unchallenged and uncontroversial. Therefore consider who the users of the model are, and what historical textbooks are used to generate the answer. LLM should mention in their responses that as per which author or book a particular answer is generated. Otherwise, like historians and politicians technocrats and technology companies will also have court cases for ideological fights. Keep in mind the following points.

1. Whoever can give the input to the LLM can control the output.

2. The only way to keep an LLM from returning training data is to not train on it. Use external data/retrieval augmented generation for private data.

3. Guardrails are best used for content moderation, not security.

4. Pay attention to logging: logs are both a privacy and a data security risk. If you log communication between the user and LLM then there is one kind of problem and if you don't log then it is a different kind of problem.

5. Stateless interactions (no conversation history fed back into prompts) are generally safer, but users can get boring replies.

6. User education is important: even with RAG, LLMs can make stuff up. Users need to be reminded of this.

3.5.7 Risk

Any language model especially LLM makes business and personal life easy for us. But it poses many threats. When we are creating any Language Model we need to keep in mind the following points.

- It should not over-represent or underrepresent some viewpoints, some characters, some geography.
- Is should not be stereotypes
- Text/corpus does not contain personal information
- Generated QAs are not unintentionally or without your knowledge:
  - Hateful, abusive, or violent language
  - Discriminatory or prejudicial language

  - Content that may not be appropriate for all settings, including sexual content
  - Make errors, including producing incorrect information as if it were factual
  - Generate irrelevant or repetitive outputs

To mitigate these risks, avoid these risks whatever needs to be done should be done in advance. Typical risk management action items can be as follows.

- Ask yourself, what is the purpose of creating this QA system?
- What are the different textbooks available on this topic?
- What are the different translations available of this work?
- What are the key controversies around that historical work?
- Mention the name of the book and author along with the answer.
- Consider using multiple viewpoints and get answers from all the different textbooks.
- Avoid summarising multiple viewpoints into one, summarise easy points individually.
- Ask yourself, who is the audience/user of this QA system, take care of hate, abuse, and violence. History is about the truth, there is no wisdom in hiding important hate crimes done. Neither there is any wisdom in putting bombs under the carpet. People know the facts from different sources. Striking a balance is important.
- Just because AI is used to generate QA it doesn't mean QAs are correctly representing the organization or book author. Human review of each question and answer by the expert is a must.

3.5.8 Metrics to Evaluate QA System

Evaluating whether a generated answer is correct or not, how much correct it is, is a highly challenging open task in NLP as of today. Even for a human who knows the data points on their tips and knows the text by heart, sometimes it becomes difficult to answer a question in a satisfying way. One of the reasons for that is, that it is necessary that the answer be factually correct but should be articulated in such a way that it appeals to the listener/reader/receiver/questioner. Same facts when presented differently different people respond differently. Human biases because of ideology, philosophy, and culture are always there in giving answers and accepting answers. It has become prominent in history, philosophy, personal life, and cultural work.
For example:

Patient : Why I am gaining so much weight?

Doctor:

Ans1: Don't worry sometimes it is hereditary.

Ans2: You are eating too much than you are burning.

Ans3: You are lazy and not doing exercise.

Ans4: You are too busy and not able to give time to your health.

Ans5: You are eating excessive junk food.

Ans6: You are giving too much time to work and there is no work-life balance.

Ans7: You are real Karma Yogi, you don't have time for your health. You have given everything of your life to the people around you.

A doctor knows, due to what reason weight increases. He knows the context of the patient. His answers from the knowledge and context can be multiple. Which one patient will like?

Let's assume all answers are correct. Do you think the patient will accept those answers with equal seriousness? Humans have their own biases, whatever hurts their ego they will not accept it, even if a doctor is saying it. Whatever boosts their ego they will accept it even if they are listening to it from odd corners. When it is difficult for humans then for machines it will remain difficult. It is not only about what answer is given, it is also about whether the questioner liked the answer or not. On top of this nowadays LLM generates answer that hallucinates human but many times humans are not able to understand that the answer is incorrect. Sometimes even after reading or listening correct answer, which is against their belief system, they want to check it from multiple sources. Finally, accept the answer from that one source that is aligned with their belief and reject dozens of different but similar answers because it is against their belief.

Let's understand the difficulty in answer evaluation with the following examples.

- This is a close-ended question. Fact-based extractive question and answer. There is no subjectivity in this answer.
  - Q: What is the color of the sky in the day?
  - A: Blue
- Fact-based True False question and answer. There is no subjectivity here.
  - Q: Kunti is Mother of Arjuna?
  - A: Yes
- Open-ended question.
  - Q: Kunti is the mother of whom?

  - The answer can be ”Pandavas” or name of any of 5 Pandava brother or any of their other names like Arjuna is also called Dhananjay.

It is difficult to evaluate open-ended questions even with ROUGE, BLEU, and METEOR metrics. So how to evaluate these types of questions?
We can use:

- Use a Human Evaluator and humans can evaluate the answer on some given scale.
- Have multiple reference answers to the question. Compare the predicted answer using EM or ROUGE metrics with multiple reference answers of that question and whatever is the highest consider that ROUGE metric.

Both of these techniques are expensive and time-consuming. In the ”human evaluator” technique there is a subjectivity element that comes because of human interpretation, as mentioned earlier.
Whatever technique we use we need to check the answer, we need to apply the following aspects of an answer. For this also we need human intervention.

- Relevance: Are the model’s answers relevant to the question?
- Completeness: Do the model’s answers provide all of the information that is requested in the question?
- Accuracy: Are the model’s answers factually accurate?
- Fluency: Are the model’s answers well-written and easy to understand?

It is important to note that no single metric can perfectly capture the performance of a question-answering system. Therefore, it is typically recommended to use a combination of metrics to get a more comprehensive view of the system’s performance.

Unlike a classification model where we measure the model's performance using a confusion matrix, in NLP there is one more layer of complexity. That complexity is to understand whether a particular prediction is correct or not. For this, we can use metrics like exact match (EM). But this metric is useful for one or two-word answers. For longer answers or generative answers, it does not work. Because the correct answer may be worded differently. These differences can be because of word order in the sentence, verb forms used, noun plurality, synonyms, and words of different languages used to communicate the answer.

In classical machine learning, we use terms like target column, ground truth, and already available response, in NLP this is referred to as reference answer. To evaluate the model's performance we need to know the reference answer and predicted answer.

Some common NLP metrics used to evaluate question-answering systems are mentioned in Appendix-A.

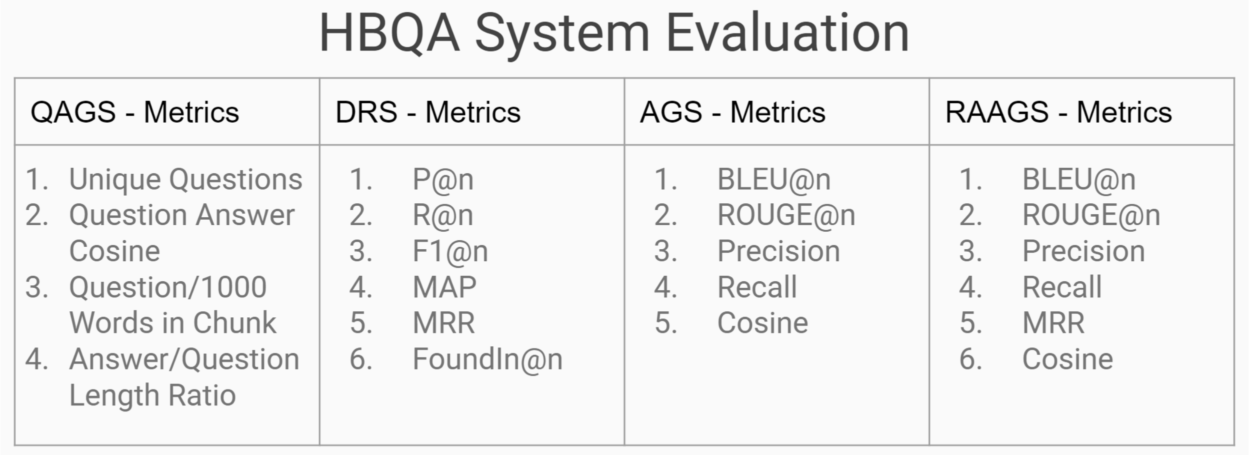
Word Overlap Metrics : Metrics like BLEU, METEOR, ROUGE, and CIDER are often used in machine translation and text generation tasks but can also be adapted for question answering. They measure the overlap of words or n-grams between the system's answer and the reference answer.

Semantic answer similarity (SAS) : This metric measures the semantic similarity between the system's answer and the gold answers. It is calculated using a variety of methods, such as word embeddings and contextual embeddings. Common metrics used for SAS include Precision at K (P@K), Recall at K (R@K), or F1 Score at K (F1@K), where K represents the number of reference answers considered.

We are using the following metrics

3.5.9 Development Tools Used

- Language: Python 3.0>



HBQA System Evaluation

QAGS - Metrics	DRS - Metrics	AGS - Metrics	RAAGS - Metrics
1. Unique Questions 2. Question Answer Cosine 3. Question/1000 Words in Chunk 4. Answer/Question Length Ratio	1. P@n 2. R@n 3. F1@n 4. MAP 5. MRR 6. FoundIn@n	1. BLEU@n 2. ROUGE@n 3. Precision 4. Recall 5. Cosine	1. BLEU@n 2. ROUGE@n 3. Precision 4. Recall 5. MRR 6. Cosine

Figure 3.11: HBQA System Evaluation Metrics, Source: Author's own illustration

- ML Libraries
  - Matplotlib
  - Seaborn
  - Pandas
  - Numpy
  - Sklearn
  - RE
  - NLTK
- Language Models and Large Language Models, Tools and Frameworks
  - ChatGPT
  - ChatPDF
  - LangChain
  - PyTorch
  - Huggingface Transformers
  - T5

  - Flan-T5
  - GPT2
  - DistilBERT
  - RoBERTa
  - BERT
  - Bloom
- Development Tools
  - VS Code
  - Google Colab
  - Kaggle
  - Github
- Tools for Literature Survey, and Research Writing
  - ChatGPT & Bard
  - Google Scholar
  - Google Workspace (sheet, slide)
  - Google Drive
  - Overleaf
  - Mendeley

3.5.10 Text Embedding Tools & Models

SentenceTransformer is a Python library for natural language processing. It provides a simple interface to fine-tune transformer models for various NLP tasks. It is built on top of the transformers library and provides pre-trained embedding models for various languages and tasks. The library uses Siamese and triplet networks to learn fixed-length sentence embeddings that can be used for various NLP tasks such as semantic similarity, clustering, and classification. The embeddings are learned by training the network on pairs or triplets of sentences, where the network learns to maximize the similarity between similar sentences and minimize the similarity between dissimilar sentences. This library provides pre-trained embedding models that can be used out-of-the-box for various NLP tasks such as sentence classification, semantic textual similarity, and question answering. Because sentence transformer embedding takes care of words in a sentence it understands the meaning of a sentence better than Word2Vec, GloVe, fastText, and BERT embedding.

Sentence Transformer Repository has pretrained 39 models (As of Oct'23) of different sizes which are trained on the different corpus. We are selecting the following 5 embedding models for our work.

Table 3.9: Text Embedding Models Used, Source: Author's own illustration

Sentence Embedding	Size	Max Sequence Length	Dimension	Normalized Embeddings
multi-qa-distilbert-cos-v1	250 MB	512	768	True
all-MiniLM-L6-v2	80 MB	256	384	True
all-distilroberta-v1	290 MB	512	768	True
all-mpnet-base-v2	420 MB	384	768	True
all-roberta-large-v1	1.36 GB	256	1024	True

3.6 Data Analysis Limitations

3.6.1 Data Collection Limitations

When we started this work. We thought the text would be readily available but very soon we realized that there is no single text that fulfills all the criteria, frankly it was difficult to establish the criteria. After analyzing data for some time we realized that every work has one or another issue. We have gone through several resources and realized the following problems.

1. Some texts in Hindi, some in Sanskrit, Kannada, or English
2. Sometimes pdf file with images.
3. Sometimes we had scanned book pdf file.
4. Some time in the format of commentary. Where every sloka is followed by some commentary.
5. One time we got a clean pdf file but it had a lot of formatting issues when we imported text into a normal text file using Python PDF libraries.
6. OCRization can be done but sometimes we have language support issues and after that text cleaning and formatting issues.
7. We know proper noun spelling problems will be there no matter how good a translation we pick.
8. When we are creating a question-answering system we need to pick the most authentic translation, otherwise whole learning via QA will be wrong.
9. Sometimes we had very good translations before us but it has an IPR issue and we cannot use that.

3.6.2 Subjectivity and Human Involvement

For data collection and data cleaning, we are using Python script. To check the quality of question answers generated by the ChatGPT system we randomly read some questions, corresponding answers, and references generated. In all the sample checks questions, answers, and references were aligned. For QA generation we have not used all the chunks of the entire book. Selected chapters and chunks are used. Our goal was to establish what it takes to create 1000+ good-quality QA datasets from a historical book and for that purpose reading the entire book was not necessary.

3.7 Ethics Related to Human Subject Participation

The goal was to use AI for question-answer generation from historical books. Therefore no human involvement was needed. We personally have a very good understanding of this book therefore while QA generation we have reviewed those questions. We understand to create a production-ready HBQA system from the historical book this is important to involve some expert to read the questions and under question quality, read answers, and check their correctness.

3.8 Summary

In this chapter, we discussed 4 components of our system namely QAGS, DRS, AGS, and RAAGS. The purpose of these individual components and how they fit into the overall HBQA system is discussed. How we zeroed down to The Mahabharat book and why we decided to use the Ganguli translation of this work was discussed. Challenges in evaluating the performance of the question-answering system, related metrics, and selected metrics were discussed. Data security, configuration management, folder structure, and tools used for this project were discussed in detail.

In the next chapter, we will discuss the following

1. Results of Question Answer Generation System (QAGS) with different

approaches used

2. Results of Document Retrieval System (DRS) with different Sentence Embedding used

3. Results of Answer Generation System (AGS) with different zeroshot and finetuned models

4. Results of Retrieval Augmented Answer Generation System (RAAGS) with Selected Sentence Embedding Model and Prediction Model.

Chapter 4

RESULT AND DISCUSSION

4.1 Introduction

We started our work with these 6 questions. To seek the answer to these questions we have conducted some experiments and logged the results of those experiments. This is the time to understand what answer we get for these 6 questions.

1. What is the cost-effective way of generating a high-quality question-answers dataset from a historical text?

2. What is the cost-effective way of generating high-quality answers from historical text-based Questions?

3. What different NLP/NLU/NLG technologies are available for Question Answering tasks in general?

4. Under 30 minutes, is it possible to generate one thousand high-quality questions and answers with reference from translated Mahabharat text using freely available hardware resources?

5. How to create a system that can answer any question from a historical book without any reference QA paired dataset?

6. What are the linguistic nuances of Indian Historical text that impact the quality of the translation of questions and answers between the Indian Language?

We will explore the answers to these questions in the order mentioned following. First, we will look at QAGS. What are the different techniques we used to generate these questions? How many questions we could generate and how much time and money it take to generate these questions? What are the strengths or weaknesses of each system used for this work?

Then we will look into DRS. Given a question, we want to know out of those hundreds of documents which document can respond to a given question correctly? For this, we created approximately 2746 chunks of the Mahabharat book, we can say these chunks are independent documents. Our DRS needs to accept the question and identify the top n documents that can answer the given question. In our case, we have taken n=10. It means we are interested in knowing, if can we correctly identify the relevant document in those top 10 documents. For this purpose, we have explored 5 sentence embedding models, and to understand which sentence embedding model can do the embedding of these documents and questions in the best way. For us, sentence embedding is the best that produces maximum mean reciprocal ranking (MRR).

Thirdly, we will look into AGS. When zeroshot models are already available then how do these models perform with our problem? It means if we don't finetune the model on our data and just give context and question then can we get the correct reply? We will also finetune our model with the best SOTA models. Finally, we will check how much score uplifting happens when we fine-tune our models.

Finally, we will look into RAAGS. Now using finetune models and the best embedding in place can we answer the question without any context? How that system is performing>?

4.2 Organization of Data Analysis

4.2.1 Question Answer Generation System (QAGS)

ChatGPT Prompt Engineering Front End

We used 321 chunks from the selected chapters of the Mahabharat book. Using these chunks we created question-answer generation prompts. Each of these chunks can generate a minimum one question, and a maximum of seven questions, depending on the length of the chunk. These prompts are kept in googlesheet, because manual copy-paste is easy, and in csv file for future processing by our systems.

We used 321 to generate 1,102 questions. A standard process is copying a chunk from googlesheet to ChatGPT Front End (FE), ChatGPT processes the prompt, and we copy the response back to googlesheet. For one prompt we can take 2-3 minutes to finish everything. On average one prompt has 3.5 questions per prompt. It means we can generate 3.5 questions in 2-3 minutes. It roughly translates to 1 question per minute. If we need 1000 question-answer pair then we need approx 1000 minutes approx 17 hours to get this work done manually. We need not to pay for API service. Of course, there is a human labor cost for running these prompts and copy-pasting the text. But once prompts are ready then following the standard process is low-skilled work. Even smaller organizations or individuals can afford to do this at their own leisure.

Table 4.1: ChatGPT-FE QA Summary, Source: Author’s own illustration

Parva Id	No. of Ques	WordsInQ	WordsInA	WordsInC	TWordsInC
Book01	33	18	51	289	9,547
Book02	11	17	47	307	3,377
Book03	934	16	47	720	672,842
Book04	115	16	47	716	82,402
Book05	9	17	57	286	2,577
Total	1,102				
*Mean		16	47	699	

WordInQ: Mean number of words in a question in this book

WordsIn: Mean number of words in an answer in this book

WordInC: Mean number of words in a chunk in this book

TWordsInC: Total number of words in chunks in this book

Total: Total number of all questions.

*Mean : $\frac{\sum(mean\_No.\_of\_Ques\_in\_the\_Book * Column_Value)}{Total\_No.\_of\_All\_Ques}$

We created chunks from the sections of the Parva of the book. Mostly one section has one chunk, but some larger section has more than one chunk. The number of words, in any chunk or section used for QA work, varies between 100 and 1000 words. The number of words in questions and answers generated by the QAGS are normally distributed. The mean number of words in question is around 15 words and the mean number of words in an answer is around 50 words with a long tail on the right side. The QAGS generated **1.43 Questions per 1000 Chunk Words**. The mean **ratio of Words in Answer/Words in Question is 2.89.**

Quality of Generated Question and Answers From a large chunk of historical text, when we create questions and answers knowing the quality of questions and answers is a complex issue. We used cosine metrics for this purpose. A higher cosine value between the generated question and the generated answer shows the

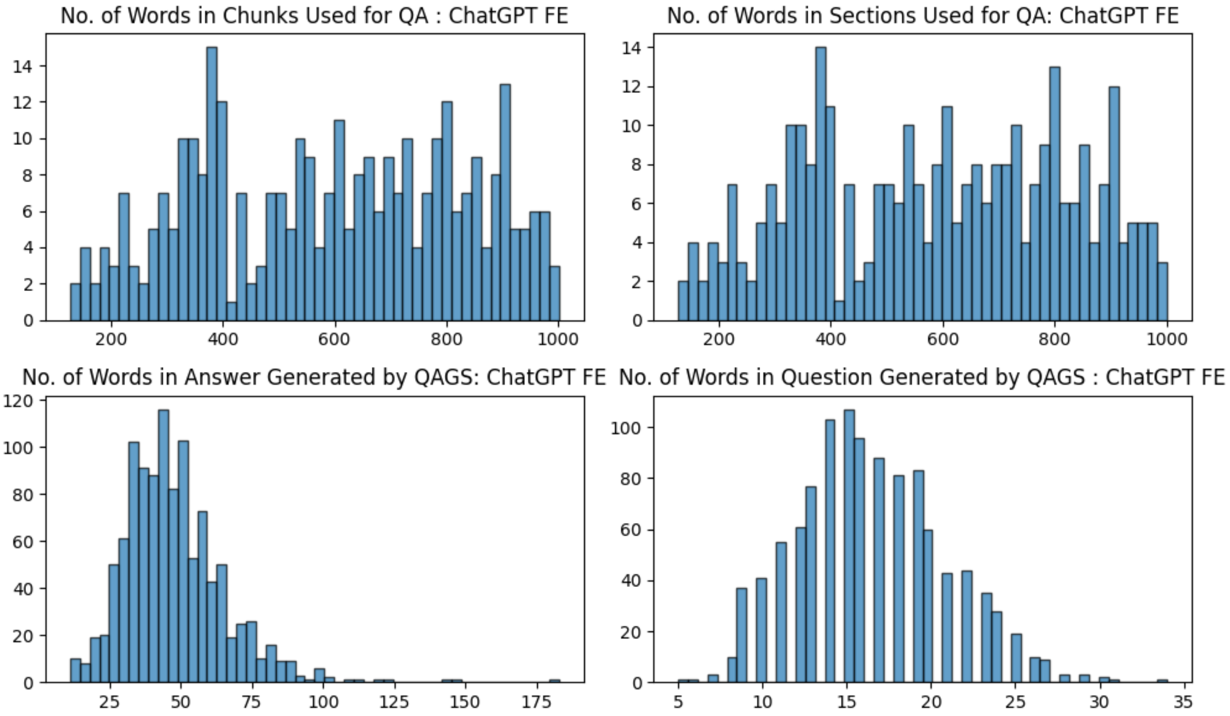


Figure 4.1: QAGS ChatGPT FE: Word Analysis, Source: Author's own illustration

question and answer are related and aligned. We can see the cosine between the answer and the chunk is 1, which means answers are picked from the related chunk and not from another unrelated chunk. We also see the cosine between question & chunk, and question & answer is the same, it is 0.91. It means questions are aligned to chunk and questions are aligned to answer.

Example of the prompt used:

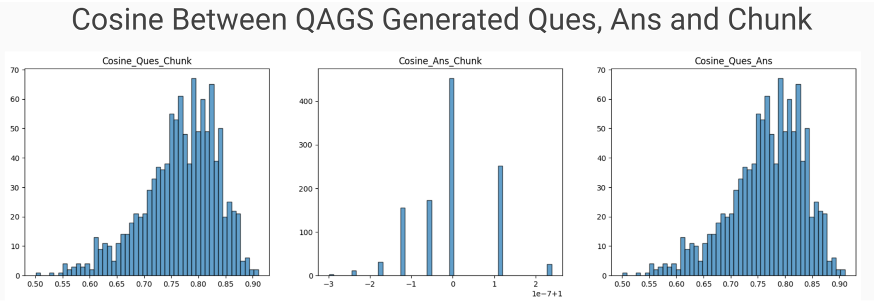


Figure 4.2: Quality of Generated Question and Answers, Source: Author's own illustration

Table 4.2: Cosine Between QAGS Generated Ques, Ans and Chunk, Source: Author's own illustration

Measure	Cosine_Ques_Chunk	Cosine_Ans_Chunk	Cosine_Ques_Ans
count	1102	1102	1102
mean	0.766	1.000	0.766
std	0.068	0.000	0.068
min	0.501	1.000	0.501
25%	0.727	1.000	0.727
50%	0.774	1.000	0.774
75%	0.816	1.000	0.816
max	0.910	1.000	0.910

```
"""Write {Questions} unique questions, corresponding answers and reference from the below context (do not repeat the context in your response)\n\nText: {text}\n\nFollow this template for your answer\n\nQuestion 1.\nAnswer 1.\nReference 1.\n\n"""
```

Here Question variable is calculated based on the length of chunk / 150. But this applies to those chunks which have a size larger than 400 words. The text variable is the chunk or context, created by splitting the sections of the book.

An Example of Questions and Answers Generated from a Chunk Make a note that this question has two questions in one. Secondly, the answer generated by ChatGPT is not simple information extraction but looking into different parts of the chunk to get relevant words, and synonyms and stitching a final answer. It is quite complex work done by ChatGPT.

Question: What predicament does Yudhishthira face, and how does he seek guidance to resolve it?

Answer: Yudhishthira faces the predicament of being unable to support the Brahmanas who are following him as he departs for the forest. He seeks guidance to resolve this dilemma by approaching his priest, Dhaumya, and inquiring about the appropriate course of action.

Chunk: Section III ”Vaisampayana said, ’Yudhishthira the son of Kunti, thus addressed by Saunaka, approached his priest and in the midst of his brothers said, ’The Brahmanas versed in the Vedas are following me who am departing for the forest. Afflicted with many calamities I am **unable to support** them. I cannot abandon them, nor have I the power to offer them sustenance: Tell me, O holy one, what should be done by me in such a pass.’ ”Vaisampayana said, ’After reflecting for a moment seeking to find out the (proper) course by his yoga powers, **Dhaumya, that foremost of all virtuous men, addressed Yudhishthira,** in these words, ’In days of old, all living beings that had been created were sorely afflicted with hunger. And like a father (unto all of them), Savita (the sun) took compassion upon them. And going first into the northern declension, the sun drew up water by his rays, and coming back to the southern declension, stayed over the earth, with his heat centered in himself. And while the sun so stayed over the earth, the lord of the vegetable world (the moon), converting the effects of the solar heat (vapours) into clouds and pouring them down in the shape of water, caused plants to spring up. Thus it is the sun himself, who, drenched by the lunar influence, is transformed, upon the sprouting of seeds, into holy vegetable furnished with the six tastes. And it is these which constitute the food of all creatures upon the earth. Thus the food that supporteth the lives of creatures is instinct with solar energy, and the sun is, therefore, the father of all creatures. Do thou, hence, O Yudhishthira, take refuge even in him. All illustrious monarchs of pure descent and deeds are known to have delivered their people by practising high asceticism. The great Karttavirya, and Vainya and Nahusha, had all, by virtue of ascetic meditation preceded by vows, delivered their people from heavy afflictions. Therefore, O virtuous one, as thou art purified by the acts do thou likewise, entering upon a file of austerities. O Bharata, virtuously support the regenerate ones.’ ”Janamejaya said, ’How did that bull among the Kurus, king Yudhishthira, for the sake of the Brahmanas adore the sun of wonderful appearance?” ”Vaisampayana said, ’Listen attentively, O king, purifying thyself and withdrawing thy mind from every other thing. And, O king of kings, appoint thou a time. I will tell thee everything in detail, And, O illustrious one, listen to the one hundred and eight names (of the sun) as they were disclosed of old by Dhaumya to the high-souled son of Pritha. Dhaumya said, ’Surya, Aryaman, Bhaga, Twastri, Pusha, Arka, Savitri. Ravi, Gabhastimat, Aja, Kala, Mrityu, Dhatri, Prabhakara, Prithibi, Apa, Teja, Kha, Vayu, the sole stay, Soma, Vrihaspati, Sukra, Budha, Angaraka, Indra, Vivaswat, Diptanshu, Suchi, Sauri, Sanaichara, Brahma, Vishnu, Rudra, Skanda, Vaisravana, Yama, Vaidyutagni, Jatharagni, Aindhna, Tejasampati, Dharmadhwaja, Veda-karttri, Vedanga, Vedavahana, Krita, Treta, Dwapara, Kali, full of every impurity, Kala, Kastha, Muhurtta, Kshapa, Yama, and Kshana; Samvatsara-kara, Aswattha, Kalachakra, Bibhavasu, Purusha, Saswata, Yogin, Vyaktavyakta, Sanatana, Kaladhyaksha, Prajadhyaksha, Viswakarma, Tamounda, Varuna, Sagara, Ansu, Jimuta, Jivana, Arihan, Bhutasraya, Bhutapati, Srastri, Samvartaka, Vanhi, Sarvadi, Alolupa, Ananta, Kapila, Bhanu, Kamada, Sarvatomukha, Jaya, Visala, Varada, Manas, Suparna, Bhutadi, Sighraga, Prandharana, Dhanwantari, Dhumaketu, Adideva, Aditisuta, Dwadasatman, Aravindaksha, Pitri, Matri, Pitamaha, Swarga-dwara, Prajadwara, Mokshadwara, Tripistapa, Dehakarti, Prasantatman, Viswatman, Viswatomukha, Characharatman, Sukhsmatman, the merciful Maitreya. These are the hundred and eight names of Surya of immeasurable energy, as told by the self-create (Brahma). For the acquisition of prosperity, I bow down to thee, O Bhaskara, blazing like unto gold or fire, who is worshipped of the gods and the Pitris and the Yakshas, and who is adored by Asuras, Nisacharas, and Siddhas. He that with fixed attention reciteth this hymn at sunrise, obtaineth wife and offspring and riches and the memory of his former existence, and by reciting this hymn a person attaineth patience and memory. Let a man concentrating his mind, recite this hymn. By doing so, he shall be proof against grief and forest-fire and ocean and every object of desire shall be his.' ”Vaisampayana continued, 'Having heard from Dhaumya these words suitable to the occasion, Yudhishthira the just, with heart concentrated within itself and purifying it duly, became engaged in austere meditation, moved by the desire of supporting the Brahmanas. And worshipping the maker of day with offerings of flowers and other articles, the king performed his ablutions. And standing in the stream, he turned his face towards the god of day. And touching the water of the Ganges the virtuous Yudhishthira with senses under complete control and depending upon air alone for his sustenance, stood there with rapt soul engaged in pranayama. And having purified himself and restrained his speech, he began to sing the hymn of praise (to the sun).' 'Yudhishthira said, ”Thou art, O sun, the eye of the universe. Thou art the soul of all corporeal existences. Thou art the origin of all things. Thou art the embodiment of the acts of all religious men. Thou art the refuge of those versed in the Sankhya philosophy (the mysteries of the 1.

Some Examples of Questions and Answers Generated by QAGS

Table 4.3: Examples of Questions and Answers Generated by QAGS, Source: Author's own illustration

Question	Ref_Answer
What is the significance of performing the Agnihotra, and what consequences are mentioned for neglecting it?	Performing the Agnihotra is considered important for eternal morality. Neglecting it, as mentioned in the text, leads to the consumption of sin. Those who do not perform the Agnihotra are said to suffer the consequences, including not waiting upon bulls and neglecting their kinsmen, guests, friends, sons, wives, and servants.

What predicament does Yudhishthira face, and how does he seek guidance to resolve it?	Yudhishthira faces the predicament of being unable to support the Brahmanas who are following him as he departs for the forest. He seeks guidance to resolve this dilemma by approaching his priest, Dhaumya, and inquiring about the appropriate course of action.
What advice does Dhaumya offer to Yudhishthira regarding his predicament?	Dhaumya advises Yudhishthira to take refuge in the sun, who is considered the father of all creatures. He explains that the sun, by his influence, transforms solar energy into food, which sustains all living beings. Dhaumya suggests that Yudhishthira should purify himself, engage in ascetic meditation, and support the Brahmanas virtuously.
How did Yudhishthira adore the sun, and what is the significance of this adoration?	Yudhishthira adored the sun by performing ablutions in the Ganges, standing in the stream with a concentrated mind, and singing a hymn of praise to the sun. This adoration is significant because it is believed to bestow various benefits, including the acquisition of prosperity, a wife, offspring, riches, memory of past lives, patience, and protection from grief and calamities.

Where did the Pandavas go after leaving the banks of the Ganges, and what was the significance of this place?	The Pandavas, along with their followers, journeyed to the Kamyaka woods, which were situated on the banks of the Saraswati. Kamyaka was known as the favorite haunt of Munis (sages) and was abundant in birds and deer. It became their chosen place of residence in the forest.
How did Sanjaya respond to Dhritarashtra's request to find Vidura?	Sanjaya expressed his approval and agreed to fulfill Dhritarashtra's request. He assured the king that he would go to the Kamyaka woods to find Vidura and bring him back without delay.

ChatGPT API Question-Answers

ChatGPT API is being called to generate these questions. How much money do we pay to generate the question? This depends upon the number of tokens our request generates for input and output. The price keeps changing from time to time, you can refer to OpenAI pricing page for this information. In 10 minutes we could finish generating these 1084 question-answer pairs. If you have a budget and doing commercial work then there is no harm in using ChatGPT API for this work but keep in mind, even if you are not happy with questions and discarding those, you have to pay the price for that. Secondly, double-check that correct prompts are being used at the API call.

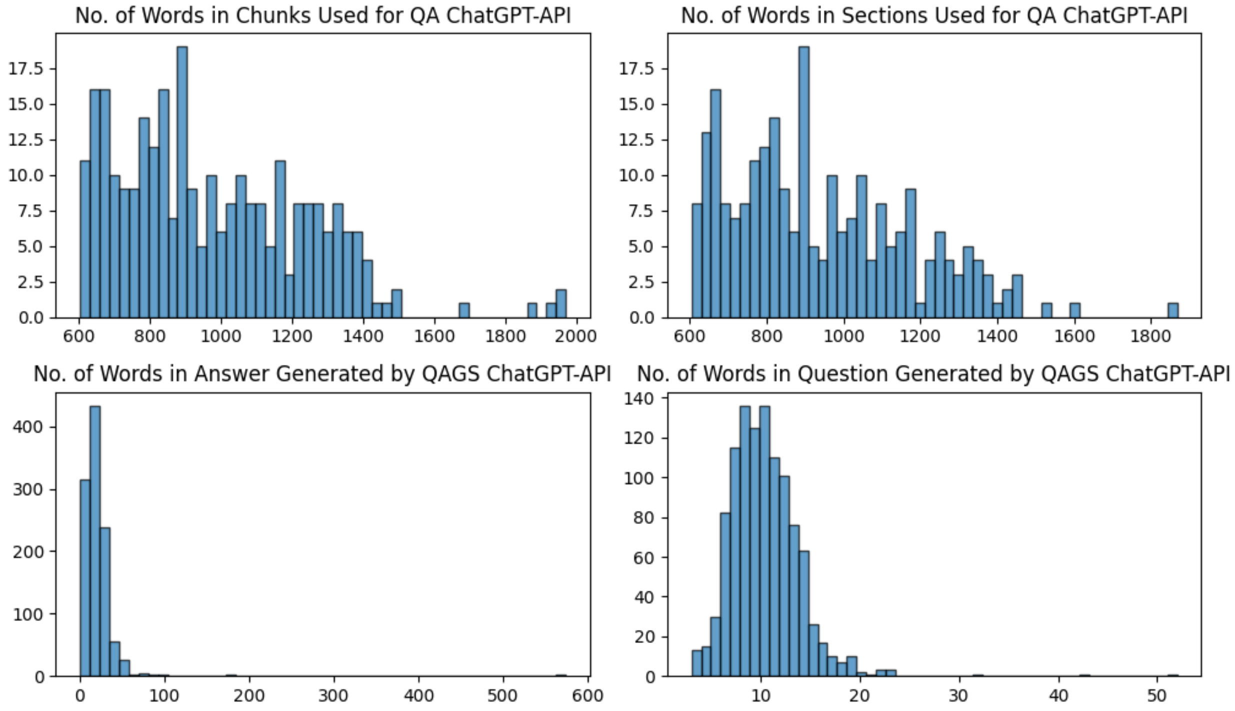


Figure 4.3: QAGS ChatGPT API: Word Analysis, Source: Author's own illustration

Table 4.4: ChatGPT API QA, Source: Author's own illustration

Parva Id	Number of Ques	Avg Words In Ques	Avg Words In Ans
Book01	470	10	22
Book02	29	11	22
Book03	53	12	24

Total 1084 questions answer pair generated by this approach. But only 552 questions are unique. **49% of the generated questions by ChatGPT API are duplicate questions.** And we need to keep in mind we are paying for every token openAI is processing for our request. Secondly overall quality and variety in the question is different from the QA pairs generated by ChatGPT FE. This we can observe from the distribution charts of the output generated.

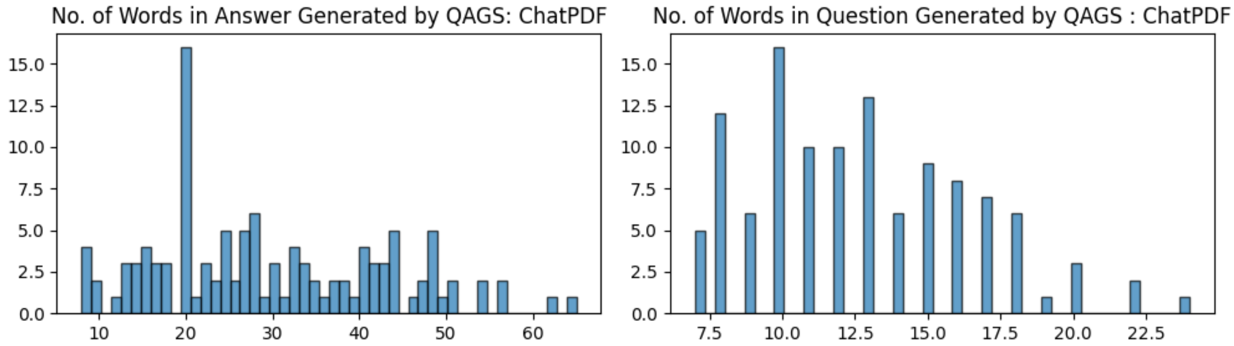


Figure 4.4: QAGS ChatPDF: Word Analysis, Source: Author’s own illustration

ChatPDF Question- Answers

Table 4.5: ChatPDF QA, Source: Author’s own illustration

Parva Id	Number of Ques	Avg Words In Ques	Avg Words In Ans
Book01	115	13	30

We used ChatPDF as another tool to generate automatic question answers. We chose Parva01 of Mahabharat to generate question answers. We tried two prompt

```
"Generate 3 questions, their corresponding answer and reference from section {section_id}"

"Generate 3 questions, their corresponding answer and reference from page {page_number}?"
```

We called these two prompts on 334 pages using ChatPDF API and generated 1023 question-answer pairs. During analysis, we found only 115 unique QA pairs. This means only 11% of unique QA pairs were generated and **89% of QA pairs were duplicates.**

The minimum length of the question generated is 7 words and the maximum length is 24 words. The median length of the question is 12 words. The minimum length of the answer generated is 8 words and the maximum length of the answer generated is 65 words. The median length of the answer is 28 words.

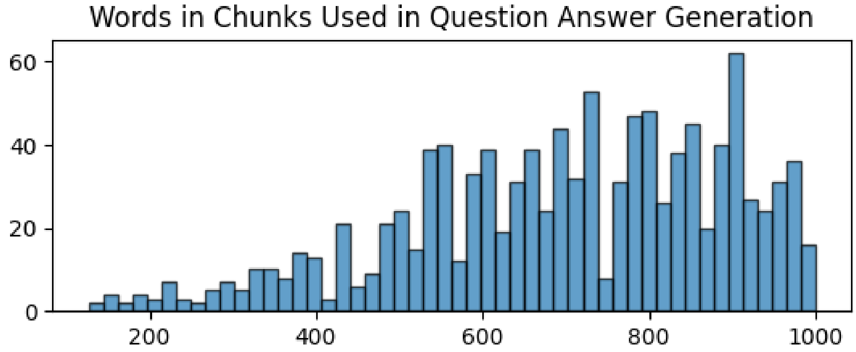


Figure 4.5: Words in Chunks Used in QA Generation, Source: Author's own illustration

Compared to ChatGPT the question length and answer length is lesser and it generates lots of duplicate question. There is no way to control this even after trying many other prompt engineering techniques.

4.2.2 Document Retrieval System (DRS)

FoundIn@n Metric

An embedding vector dataset which was created using a chunk and mpnet embedding model produces the best FoundIn@10 metric. It means if the DRS system is returning 10 chunk ids for a given question then 81% probability that one of those 10 documents is the real document.

Table 4.6: FoundIn@n Metric, Source: Author's own illustration

	distilbert	minilm	distilroberta	mpnet	roberta
FoundIn@1	0.38	0.33	0.36	0.43	0.36
FoundIn@2	0.50	0.44	0.47	0.57	0.47
FoundIn@3	0.57	0.51	0.55	0.63	0.54
FoundIn@4	0.63	0.55	0.60	0.69	0.57
FoundIn@5	0.66	0.58	0.64	0.72	0.61
FoundIn@10	0.75	0.66	0.74	**0.81**	0.71

Precision@n Metric

Number of relevant documents returned in top n documents. In our case one question has only one related chunk, so the relevant document will always remain 1. If for a given question DRS is returning 10 chunk-ids and the correct chunk-id is at 3rd location in the list then p@1/1 = 0, P@2/2 = 0, P@3/3 = 1/3 = .0.33. An embedding vector dataset which was created using a chunk and mpnet embedding model produces the best P@1 metric. Formula for calculating P@n is P@1 = Found@1/1, P@2=Found@2/2, P@10=Found@10/10.

Table 4.7: Precision@n Metric, Source: Author's own illustration

	distilbert	minilm	distilroberta	mpnet	roberta
P@1	0.38	0.33	0.36	**0.43**	0.36
P@2	0.25	0.22	0.24	0.28	0.24
P@3	0.19	0.17	0.18	0.21	0.18
P@4	0.14	0.13	0.14	0.16	0.13
P@5	0.13	0.12	0.13	0.14	0.12
P@10	0.07	0.07	0.07	0.08	0.07

Recall@n Metric

Number of Relevant Items in the Top K/ Total Number of Relevant Items in the Collection. R@n = FoundIn@n/1 (1 because , in our case any given question has only 1 relevent chunk). R@10 is the best metric with mpnet embedding.

Table 4.8: Recall@n Metric, Source: Author's own illustration

	distilbert	minilm	distilroberta	mpnet	roberta
R@1	0.38	0.33	0.36	0.43	0.36
R@2	0.50	0.44	0.47	0.57	0.47
R@3	0.57	0.51	0.55	0.63	0.54
R@4	0.63	0.55	0.60	0.69	0.57
R@5	0.66	0.58	0.64	0.72	0.61
R@10	0.75	0.66	0.74	**0.81**	0.71

F1@n Metric

$F1@n = \frac{2*R@n*P@n}{R@n+P@n}$

Table 4.9: F1@n Metric, Source: Author's own illustration

	distilbert	minilm	distilroberta	mpnet	roberta
F1@1	0.38	0.33	0.36	**0.43**	0.36
F1@2	0.34	0.30	0.32	0.38	0.32
F1@3	0.29	0.25	0.27	0.31	0.27
F1@4	0.23	0.20	0.22	0.25	0.21
F1@5	0.22	0.19	0.21	0.24	0.20
F1@10	0.13	0.12	0.13	0.15	0.13

MAP, MRR Metric

$$Mean\ Average\ Precision\ (MAP) = \frac{P@1 + P@2 + P@3 + P@4 + P@5}{5}$$

$$Mean\ Reciprocal\ Ranking\ (MRR) = \frac{1}{Index\_Location\_at\_Which\_correct\_ChunkId\_Found}$$

Table 4.10: MAP, MRR Metric, Source: Author's own illustration

	distilbert	minilm	distilroberta	mpnet	roberta
MAP	0.22	0.19	0.21	**0.25**	0.21
MRR	0.49	0.43	0.48	**0.55**	0.47

Performance of Sentence Transformers

Table 4.11: Performance of Sentence Transformers, Source: Author's own illustration

Sent. Embd.	Size	Dim	R@10	P@1	MRR
multi-qa-distilbert-cos-v1	250 MB	768	0.75	0.38	0.49
all-MiniLM-L6-v2	80 MB	384	0.66	0.33	0.43
all-distilroberta-v1	290 MB	768	0.74	0.36	0.48
all-mpnet-base-v2	420 MB	768	**0.81**	**0.43**	**0.55**
all-roberta-large-v1	1.36 GB	1024	0.71	0.36	.0.47

As we can see from above metrics, the mpnet-base-dot-v1 embedding is showing the best performance in all the metrics. Here are the findings.

- Highest MRR is .55
- Highest MAP is .25
- Highest F1@n score is at F1@3. It is .31
- Highest Recall@n score is R@10. It is .81
- Highest P1@n score is at P@1. It is ..43

Relationship between Chunk size and metrics

Table 4.12: Correlationship Between Variables in DRS, Source: Author's own illustration

	RR	MAP	F1@3	R@10	P@1	Chunk_Words
RR	1.00	0.99	0.86	0.65	0.94	-0.22
MAP	0.99	1.00	0.90	0.60	0.91	-0.21
F1@3	0.86	0.90	1.00	0.64	0.67	-0.17
R@10	0.65	0.60	0.64	1.00	0.43	-0.19
P@1	0.94	0.91	0.67	0.43	1.00	-0.20
Chunk_Words	-0.22	-0.21	-0.17	-0.19	-0.20	1

Chunk size and all important metrics have a negative relationship. The strength of the relationship between all the chunk_size and all the metrics is almost the same, it is -0.20

4.2.3 Answer Generation System (AGS)

Zeroshot Learning Models

T5: Text-to-Text Transfer Transformer

```
   from transformers import T5Tokenizer,  T5ForConditionalGeneration
   model_name = "t5-base"
   tokenizer = T5Tokenizer.from_pretrained(model_name)
   model = T5ForConditionalGeneration.from_pretrained(model_name,
      return_dict=True)
```

BERT: Bidirectional Encoder Representations from Transformers

```
   from transformers import AutoTokenizer,
      AutoModelForQuestionAnswering
   model_name = "bert-large-uncased-whole-word-masking-finetuned-squad
      "
   tokenizer = AutoTokenizer.from_pretrained(model_name)
   model = AutoModelForQuestionAnswering.from_pretrained(model_name)
```

BigBird:

```
   from transformers import BigBirdModel, BigBirdTokenizer
   model_name = "google/bigbird-roberta-base"
   tokenizer = BigBirdTokenizer.from_pretrained(model_name)
   model = BigBirdModel.from_pretrained(model_name)
```

Bloom560m: BigScience Large Open-science Open-access Multilingual

```
   from transformers import BloomTokenizerFast, BloomForCausalLM
   model_name = "bigscience/bloomz-560m"
   tokenizer = BloomTokenizerFast.from_pretrained(model_name)
   model = BloomForCausalLM.from_pretrained(model_name,
      low_cpu_mem_usage=True)
```

Bloom1.7:

```
   from transformers import BloomTokenizerFast, BloomForCausalLM
```

```
model_name = "bigscience/bloomz-1b7"
tokenizer = BloomTokenizerFast.from_pretrained(model_name)
model = BloomForCausalLM.from_pretrained(model_name,
   low_cpu_mem_usage=True)
```

distilBERT:

```
from transformers import DistilBertTokenizer,
   DistilBertForQuestionAnswering
model_name = 'distilbert-base-cased-distilled-squad'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
```

RoBERTa: Robust BERT Approach

```
from transformers import AutoTokenizer,
   AutoModelForQuestionAnswering
model_name = "deepset/roberta-base-squad2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
```

GPT2: Generative Pretrained Transformer

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel,
   TextDataset, DataCollatorForLanguageModeling
model_name = "gpt2-medium"
tokenizer = GPT2Tokenizer.from_pretrained(model_name,  padding_side
   ="left")
model = GPT2LMHeadModel.from_pretrained("gpt2-medium").to(DEVICE)
```

We tried this model but the results are just a repetition of question tokens. So this didn’t work as expected.

longformer:

```
from transformers import LongformerTokenizer, LongformerForQuestionAnswering
model_name = "valhalla/longformer-base-4096-finetuned-squadv1"
tokenizer = LongformerTokenizer.from_pretrained(model_name, padding_side="left")
model = LongformerForQuestionAnswering.from_pretrained("gpt2-medium").to(DEVICE)
```

We tried zero-shot learning on this transformer but we couldn't do because Longformer is not supported for the latest transformer.

Zeroshot Model Performance

We did the answer prediction using SOTA models with default hyperparameters and ignoring the max token size limit of the model. We got the following metrics for the answers generated by these models.

Table 4.13: Performance os Zeroshot-1 Learning Models, Source: Author's own illustration

Metrics	T5	BERT	RoBERTa	BigBird
BLEU1	0.042	0.041	0.031	0.010
ROUGE1_P	0.218	0.200	0.132	0.042
ROUGE1_R	0.222	0.198	0.122	0.046
ROUGE1_F1	0.211	0.184	0.120	0.038
ROUGEL_P	0.154	0.137	0.089	0.029
ROUGEL_R	0.158	0.134	0.082	0.031
ROUGEL_F1	0.149	0.125	0.080	0.026
Precision	0.130	0.133	0.096	0.031
Recall	0.155	0.135	0.092	0.035
Cosine	0.482	0.450	0.371	0.283

All the models are performing badly. Even T5 model which is excellent for generative QA tasks is producing bad answers.

Table 4.14: BLOOM Model Size, Source: Author's own illustration

Variant	Number of parameters
BLOOM-176B	176 billion
BLOOM-137B	137 billion
BLOOM-7B1	7.1 billion
BLOOM-3B	3 billion
BLOOM-1B7	1.7 billion
BLOOM-1B1	1.1 billion
BLOOM-560M	560 million

Table 4.15: Performance os Zeroshot-2 Learning Models, Source: Author's own illustration

Metrics	distilBERT	distilBERT512	Bloom560	Bloom1.7b
BLEU1	0.042	0.029	0.015	0.014
ROUGE1_P	0.218	0.142	0.085	0.073
ROUGE1_R	0.222	0.128	0.259	0.275
ROUGE1_F1	0.211	0.127	0.105	0.095
ROUGEL_P	0.154	0.096	0.056	0.047
ROUGEL_R	0.158	0.086	0.166	0.181
ROUGEL_F1	0.149	0.085	0.067	0.061
Precision	0.130	0.104	0.060	0.049
Recall	0.155	0.097	0.176	0.186
Cosine	0.482	0.379	0.405	0.422

We took a distilBERT pretrained model and while predicting we took care of the max token limit imposed by the model. Below ”distilBERT” column shows the results when longer than 512 tokens context is taken care. BLOOM is LLM it has the following variants.

We took BLOOM560 and BLOOM1B7 pretrained models to predict the answer. Results are shown in table, Zeroshot-2. We can see we get marginal improvement in the metrics when longer tokens are handled during the prediction or more parameter models are used. But this is not sufficient for our work.

Finetuned Models

BLOOM is LLM and to finetune this we need expensive hardware. Although techniques like PEFT, and LoRA can reduce the time, because we do need not to finetune all the parameters, we still need expensive hardware to load the model for finetuning and prediction. Therefore we are dropping LLM models that have a high number of parameters. Secondly, we noticed the performance of BLOOM is not that good which even a small model has. From the results of these zero-shot models, we took T5 and Flan-T5 for finetuning.

Hyper-parameters: T5 T5 - Text-to-Text Transfer Transformer, has 5 variants with different sizes. We used t5-small for our work.

- t5-small: 60 million parameters,
- t5-base: 220 million parameters,
- t5-large: 770 million parameters,
- t5-3b: 3 billion parameters,
- t5-11b: 11 billion parameters

```
    from transformers import T5Tokenizer,  T5ForConditionalGeneration # , T5Model , T5TokenizerFast
    model_name = "t5-base"
    tokenizer = T5Tokenizer.from_pretrained(model_name)
    model = T5ForConditionalGeneration.from_pretrained(model_name, return_dict=True)

    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    num_train_epochs=30,
save_steps=1000,
eval_steps=1000,

question, context = 1500
answer = 500

question_tokenized = tokenizer(question, context, max_length=self.
   q_len, padding="max_length", truncation=True, pad_to_max_length=
   True, add_special_tokens=True)
answer_tokenized = tokenizer(answer, max_length=self.t_len, padding
   ="max_length", truncation=True, pad_to_max_length=True,
   add_special_tokens=True)
```

Hyper-parameters: Flan-T5 Flan-T5 (Finetuned T5) model has the following 5 variants with different size. We used flan-t5-small for our work.

- google/flan-t5-small: 80M
- google/flan-t5-base: 250M
- google/flan-t5-large: 780M
- google/flan-t5-xl: 3B
- google/flan-t5-xxl: 11B

```
from transformers import T5Tokenizer, T5ForConditionalGeneration #,
   T5Model,  T5TokenizerFast
model_name = "google/flan-t5-small"
tokenizer = T5Tokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name,
   return_dict=True)
```

Finetuned Models Performance

The cosine distance between predicted answers and reference answers is .827 for t5small model which was created in 30 epochs. We get the same cosine distance even with a flan-t5 model which was created in 30 epochs. We can see, that when we train the t5 model for more epochs we get significant improvement in almost all metrics. Thus we conclude if we use more parameters models like t5-3b or t5-11b and finetune that with more epochs we can get significant improvement in these metrics. We can see for a historical book, t5, and flan-t5 both produce almost the same results.

Table 4.16: Performance of Finetuned Models, Source: Author's own illustration

Metrics	T5smal-1ep	t5small-30ep	flant5-30ep
BLEU1	0.095	0.196	0.204
ROUGE1_P	0.458	**0.513**	0.483
ROUGE1_R	0.309	0.455	0.477
ROUGE1_F1	0.342	**0.466**	0.465
ROUGEL_P	0.362	0.409	0.387
ROUGEL_R	0.242	0.367	0.386
ROUGEL_F1	0.268	0.373	0.374
Precision	0.358	**0.393**	0.357
Recall	0.243	**0.350**	0.357
Cosine	0.715	**0.827**	0.826

Distribution of Metrics

We are presenting a distribution of metrics that are related to answers predicted by the T5-30 epochs finetuned model. All the metrics value varies between 0 to 1, 1 being the best and 0 being the worst. All these metrics are normally distributed but the mean, and median of the metrics are different. The mean of the cosine metric is .82, mean of the ROUGE1_P is .483. BLEU1 score is the worst score and cosine score is the best score. But, when we compare the BLEU1 score of the T5-30ep model with other models the .204 BLEU score of T5-30ep is the best score

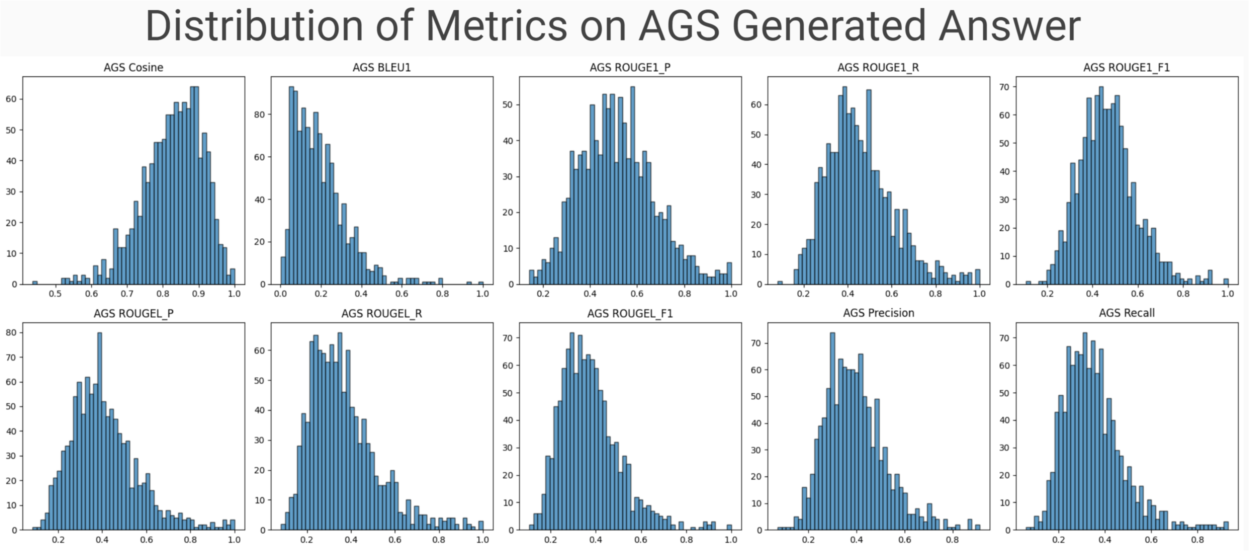


Figure 4.6: Distribution of Metrics on T5-30 Epochs Generated Answers, Source: Author's own illustration

among all the models.

Alignment Between Generated Questions and Answers

In QAGS, the cosine distance between the questions and answers generated by ChatGPT Front End is calculated after vectorization of questions and answers on 5 different SentenceTransformers, the results are in the below table. We can see multi-qa-distilbert-cos-v1 and all-MiniLM-L6-v2 both are able to capture the alignment between question and answers more accurately than other embedding models. The mean length of the question and the answer is 16 and 47 words respectively, while the mean length of the chunk is 700 words, Table 4.1, ChatGPT FE QA Summary. The best embedding model that can capture the relationship between QAGS-generated questions and answers is MiniLM. The mean Cosine distance between the generated question-answer pair is 0.78.

Table 4.17: Cosine Value Between ChatGPT Generated QA, Source: Author’s own illustration

Model-Name	Cosine
multi-qa-distilbert-cos-v1	0.779
all-MiniLM-L6-v2	0.780
all-distilroberta-v1	0.710
multi-qa-mpnet-base-dot-v1	0.766
all-roberta-large-v1	0.731

Comparing Answers Generated by Models

To get an idea about the text of a chunk created from the Mahabharat book’s section text, question+answer+reference generated by ChatGPT, we can look into one example. Using zeroshot learning models when we supplied the question and chunk to the model, each model gave a different answer. More examples of Mahabharat Book QA Appendix-B

Table 4.18: Sample Answer Generated by Different Models, Source: Author’s own illustration

Question: 7	What was the primary concern expressed by Dhritarashtra to Vidura regarding the situation after the Pandavas had gone to the forest?
Ref_Answer	Dhritarashtra expressed deep sorrow over the situation and was concerned about how to secure the goodwill of the citizens and maintain their support. He sought Vidura’s advice on what should be done for the welfare of both his sons and the Pandavas.
	Zeroshot Learning
ZSt5 _Pred_Answer	he said,–’ he said,–’ ” Vidura said,’Vidura Vidura, Vidura said, ’This race is doomed’. Vidura said, ’O king, Vidura said, ”O,. ,. let do thou Vidura said, ’

ZSbert _Pred_Answer	sorrowful. and seated at his ease the king addressed these words to the virtuous vidura of profound intelligence,'thy understanding is as clear as that of bhargava. thou knowest also all the subtleties or morality, and thou
ZSbigbird _Pred_Answer	xxx
ZSbloom _Pred_Answer	virtue as its basis. Therefore, O monarch, according to the best of thy power, cherish thou virtuously thy own sons and those of Pandu. That virtue had been beguiled by wicked souls with Suvala's son at their head, when thy sons invited the righteous Yudhishthira and defeated him in the match at dice. O king, of this deed of utter iniquity I behold this expiation whereby, O chief of the Kurus, thy son, freed from sin, may win back his position among good men. Let the sons of Pandu, obtain
ZSbloom17 _Pred_Answer	of profound intelligence, 'Thy understanding is as clear as that of Bhargava. Thou knowest also all the subtleties or morality, and thou lookest on all the Kauravas with an equal eye. O, tell me what is proper for me and them. O Vidura, things having thus taken their course, what should we do now? How may I secure the goodwill of the citizens so that they may not destroy us to the roots? O, tell us all, since thou art conversant with every excellent expedient.' "Vidura said, 'The three-fold purposes, O king (viz., profit, pleasure, and salvation), have their foundations in virtue, and the

ZSdistilbert512_Pred_Answer	sorrowful. And seated at his ease the king addressed these words to the virtuous Vidura of profound intelligence,’Thy understanding is as clear as that of Bhargava. Thou knowest also all the subtleties or morality,
ZSroberta_Pred_Answer	sorrowful. And seated at his ease the king addressed these words to the virtuous Vidura of profound intelligence, ’Thy understanding is as clear as that of Bhargava. Thou knowest also all the subtleties or morality, and thou
	Finetuned Models
t5_1ep_Pred_Answer	Dhritarashtra expressed concern about the situation after the Pandavas had gone to the forest. He said that he was not friendly to Vidura. He said that Vidura said that he would not abandon his son for the sake of the sons of Pandu.
t5_30ep_Pred_Answer	Dhritarashtra expressed his concern about the situation after the Pandavas had gone to the forest. He emphasized that the situation was not friendly to Vidura and that Vidura was not friendly to Vidura.
flant5_30ep_Pred_Answer	Dhritarashtra expressed his concern about the situation after the Pandavas had gone to the forest. He expressed his concern about the morality of the Kurus and the way in which a king should remain content with his own possessions. He mentioned that if a king consents to reign in peace with the sons of Pandu, he would not have to repent. Dhritarashtra expressed his concern about the Kurus’ morality and

From this example, we can see that

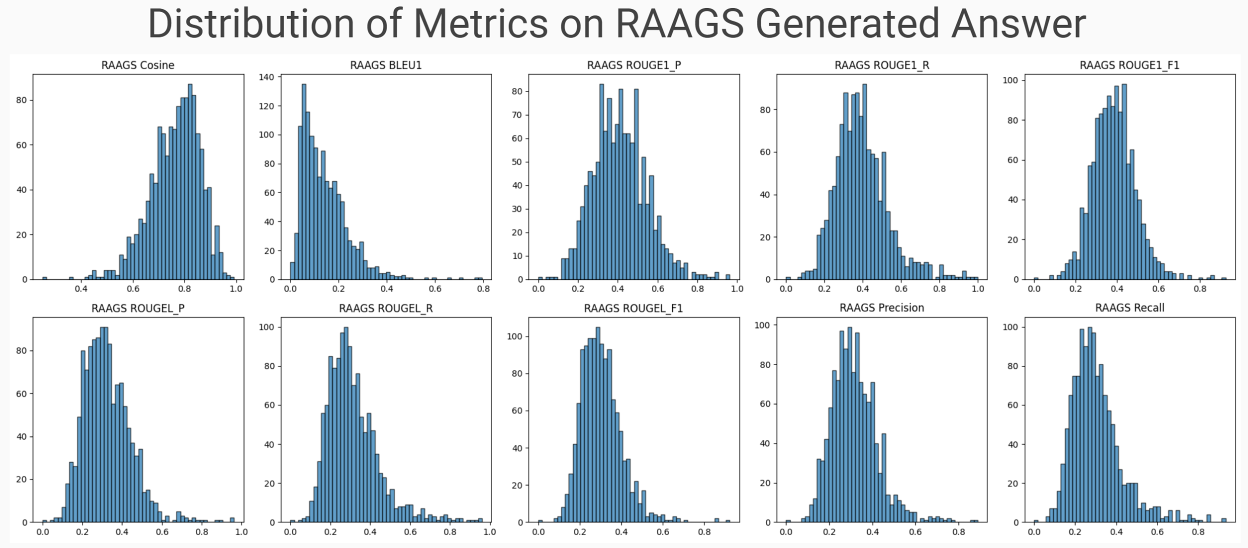


Figure 4.7: Distribution of Metrics on RAAGS-Generated Answers, Source: Author's own illustration

- When we are using SOTA Zeroshot learning models to predict the answer then some models are able to generate closer to reference answer, and some are generating xxx, i.e. I don't know. Some are throwing garbage.

- But, when we are using a fine-tuned model the answers are very close to the reference answer.

4.2.4 Retrieval Augmented Answer Generation System (RAAGS)

RAAG System Performance

RAAG system generates an answer to a given question without any context. It uses t5-30ep, a finetuned model to predict the answers. With this, we are getting the following metrics. The cosine score is 0.763, Recall is 0.304. When we physically look into the answer generated by RAAGS those are the plausible answers and very close to AGS generated answer, when context was given to generate an answer.

Table 4.19: Metrics on RAAGS-Generated Answers, Source: Author's own illustration

Metrics	t5-30ep Model
BLEU1	0.140
ROUGE1_P	0.411
ROUGE1_R	0.393
ROUGE1_F1	0.384
ROUGEL_P	0.323
ROUGEL_R	0.313
ROUGEL_F1	0.303
Precision	0.316
Recall	0.304
Cosine	0.763

Cosine Distance Between Question, Ref.Answer, and Pred.Answer

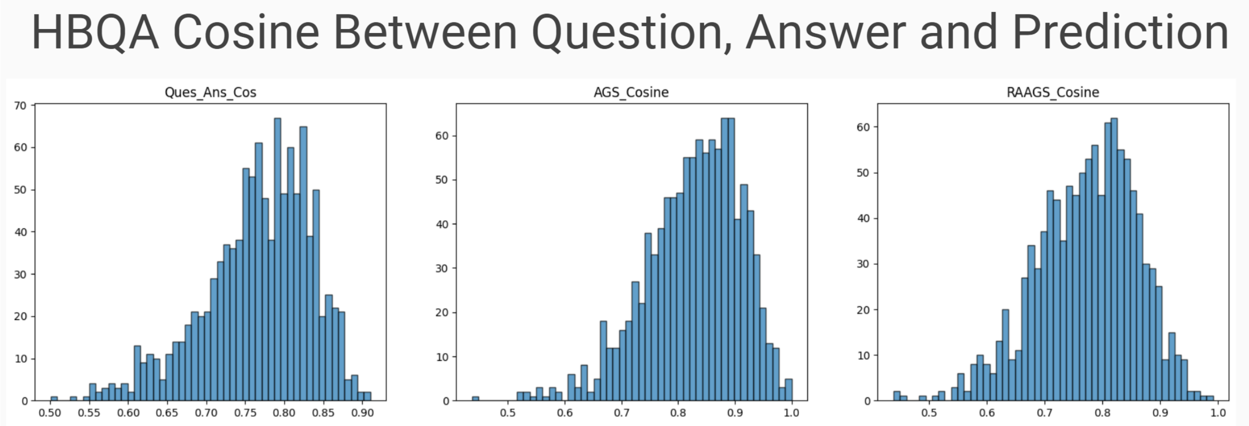


Figure 4.8: Cosine Distance Between Question, Ref.Answer, and Pred.Answer, Source: Author's own illustration

Ques_Ans_Cos is the cosine distance between a question and a reference answer. The mean cosine value is 0.77. We know cosine is a measure of similarity and not of correctness. Therefore higher value of cosine indicates questions and answers are aligned. AGS_Cosine is the cosine distance between the AGS predicted answer and reference answer. The mean AGS_Cosine is 0.83, which means there is a high alignment between the predicted answer and the reference answer.

RAAGS_Cosine is the cosine distance reference answer and RAAGS predicted

Table 4.20: Cosine Distance Between Question, Ref.Answer, and Pred.Answer, Source: Author's own illustration

Measure	Ques-Ans-Cos	AGS-Cosine	RAAGS-Cosine
count	1102	1102	1102
mean	0.77	0.83	0.77
std	0.07	0.08	0.09
min	0.50	0.44	0.44
25%	0.73	0.78	0.72
50%	0.77	0.84	0.78
75%	0.82	0.89	0.84
max	0.91	1.00	0.99

answer. The mean RAAGS_Cosine is 0.77. Although RAAGS_Cosine values are weaker than AGS_Cosine, we need to keep in mind RAAGS is predicting answers without any context. RAAGS looks at chunk vector space and picks up the relevant chunks and uses those along with a question to predict the answer. While to predict answers in AGS we need to give the context document.

4.3 Research Findings

We had 6 research questions and research findings related to each can be summarised as below.

RQ1: What is the cost-effective way of generating high-quality questions from a historical text? RQ4: Under 30 minutes, is it possible to generate one thousand high-quality questions and answers with reference from translated Mahabharat text using freely available hardware resources?

We can create a chunk from the original text. Using these chunks we can do Prompt Engineering work to create requests that can generate questions, answers, and references. Use these prompts on the ChatGPT frontend interface on the Openai website. If we put requests, with smaller text chunks then it free and ethical way of generating thousands of questions. If you have prompts ready then the following can be achieved in under 17 hours, and this can generate 1000+ questions, answers and references.

1. We copy and paste prompts to the chatGPT website
2. ChatGPT processes the prompt request and displays the results
3. We copy and paste ChatGPT response to some CSV file for further cleaning

If we don't want to spend 17 hours on OpenAI front end to generate 1000 questions then we can use ChatGPT API and within 30 min you get 1000+ question answers. But we need to keep in mind, that it generates duplicate questions and secondly, we need to pay for the API. Finally, even if you execute the wrong prompt at openai API, you have to pay for that.

RQ2: What is the cost-effective way of generating high-quality answers from historical text-based Questions?

Fine-tuning an LLM model is time-consuming and expensive. But, we can finetune T5, and FlanT5 transformer models with more epochs on free or almost free GPU machines. For historical question-answering work, we can get the best model that can answer all the questions of the given book.

RQ3: What different NLP/NLU/NLG technologies are available for Question Answering tasks in general?

We need to decide what kind of question-answer we want to generate. For MCQ or Extractive kind of QA system, almost all transformers can perform this task. For extractive QA, creating a suitable dataset is a laborious process because you need to mark the start and end of the answer in the chunk. For MCQ, you need to create a dataset with multiple options. In our case, we were interested in generative answers so we need not do that step. And ChatGPT is good at creating generative question answers.

RQ5: How to create a system that can answer any question from a historical book without any reference question answer dataset? Using RAAG system, we demonstrated that this is possible and we can generate good-quality answers.

RQ6: What are the linguistic nuances of Indian Historical text that impact the quality of the translation of questions and answers between the Indian Language? This question was not related to experiments but analysis of Indian Historical text. If we create a QA dataset and QA system for the Mahabharat book in the English language and we want to translate these questions into Hindi or Tamil then it will generate lots of spelling errors around the noun. Secondly, we cannot ask questions in Devanagari or Tamil script in this system. Thirdly, the answer generated will always be in English language Latin script.

To solve these problems we need to translate questions asked by the user into Hindi and Devanagari. But keep in mind the spelling of nouns. When we translate historical text into the Indian language then there will be errors in transliterating/transcribing the noun. This will lead to an overall wrong translation of the question and RAAGS or AGS will not work correctly. Therefore we need a system that can transcribe NER of historical text correctly.

4.4 Summary

From the results of QAS, we have seen that ChatGPT Front End is the most economical way to create question answers if we want to generate QA in the range of 1000. For this, we need to have a clean corpus. And, we need to break the entire text into smaller chunks. We suggest don't keep longer chunks size. In that case, we need more expensive hardware and powerful models that can handle longer contexts. If the chunk size is too small that will cause processing overhead and not able to learn the context. Ideally, a question chunk size can be between 150-300 words, although we have used longer chunk sizes. When we are using ChatGPT API to create QA dataset it is expensive. Sometimes, even if we don't like the questions and we have to rerun the program we need to pay for this. Although it generates QA very fast compared to using the front end.

If the chunk size is longer than the max token size an embedding model can handle then it will cause truncation. This will cause a dilution of learning. If the chunk size is smaller than the max token size then it will handled by adding the pads. Which is unnecessary overhead for no gain. If words (in the case of word embedding) or word sequence (in the case of sentence embedding) are not available in the embedding model then the word is split into multiple smaller words. In our case most of the words are from Sanskrit vocabulary therefore words will be split into multiple tokens. Because of this when non-English text is given a tokenization and embedding system, then it is not one token for one word. As per OpenAI, on average 4 tokens for 3 words, but this doesn't apply to our kind of work. Keeping all this mind in mind we need to decide on chunk size.

We have seen the results of QAGS and established yes it is possible to generate 1000+ questions within 30 min but we need to pay for that. On the other hand, we use use ChatGPT front end to generate these questions and it will be free, in fact quality of these QA is better than ChatGPT API-generated questions.

In the DRS experiment, we used 5 text embedding, refer Table:3.9, namely distilbert, minilm, distilroberta, mpnet, and roberta. The mean Reciprocal Ranking of mpnet is the highest 0.55. Recall@10 of mpnet is the highest 0.81. It means if the DRS system is returning 10 documents that can contain the possible answer for a given question then 81% chance that the document is correctly identified in the top 10. The all-mpnet-base-v2 is pretrained embedding model from SentenceTransformer is best for our work. The size of this model is 420 MB, it generates a 768-dimensional vector, and the max sequence length it can handle is 384 characters. It gives normalized embedding, which means the cos and dot products will return the same result. We know cos is a more costly operation compared to dot product therefore to find the distance between two documents we can use dot product. Thus for historical text mpnet embedding model is the best model.

In AGS experiments we learned T5-small finetuned over 30 epoch model generates the best answers., Metrics of this finetuned model are BLEU1 : 0.196, ROUGE1_P : 0.513, ROUGE1_R : 0.455, ROUGE1_F1 : 0.466, ROUGEL_P : 0.409, ROUGEL_R : 0.367, ROUGEL_F1 : 0.373, Precision : 0.393, Recall : 0.35, Cosine : 0.827

In RAAGS experiments we learned that even if we do not have context and we just know the question then with a finetuned model we can generate high-quality answers to any question related to a given historical book.

Chapter 5

DISCUSSION, CONCLUSION AND IMPLICATIONS

5.1 Introduction

In this chapter, we provide a detailed analysis of the findings and insights garnered through our research. Our research questions led us to develop and evaluate four distinct sub-systems. All these four systems can work independently and have their role in our day-to-day life. Collectively these sub-systems form a robust and innovative approach to addressing the challenges in historical book question answering. These sub-systems include:
Question Answer Generation System (QAGS): This system is designed to formulate precise and contextually relevant questions based on the content of historical books, a critical component in the quest for accurate and informative answers. This system is useful when

(a) a learner does not know anything about any historical book.

(b) a teacher wants to check a learner's understanding of a historical book but the teacher doesn't have time or he/she cannot test the understanding of thousands of people within a limited time.

(c) Learner want to quickly understand what she/he will learn from the work.

In these situations if we have ready-made high-quality questions and answers with us then using software apps we can engage with the learners at a large scale. The quality of engagement depends

(a) How good those questions are?

(b) In the learning process when we introduce those questions

(c) How correct, complete, concise, and creative those answers are?

From a large volume of historical text creating good-quality questions and corresponding answers is a challenging and laborious task. And evaluating how creating these questions and answers is another challenging task. Using QAGS systems we are solving the first part of the problem. To evaluate the quality of generated questions and answers we used the cosine metric. We saw cosine between the answer and the chunk is 1, which means answers are picked from the related chunk and not from other unrelated chunks. We also see cosine between question & chunk, and question & answer is the same, it is 0.91. It means questions are aligned to chunk and questions are aligned to answer.

Document Retrieval System (DRS): Retrieving relevant historical documents (text/ chunk) is another aspect of our research. From a large body of knowledge around historical text if we are interested in knowing which book, or section, or chapter, or page has the answer to a question we can use this system. We found the highest MRR of DRS is .55. We got this from the vector database created on mpnet sentence embedding SentenceTransformer.

Answer Generation System (AGS) Generating coherent and informative answers is a fundamental goal in historical book question answering. Answering a limited number of questions or based on predefined questions and answers is one aspect of learning systems. But people can ask the same question in different ways, in different styles, using different words, sometimes using unrelated words. We want our system should be able to answer the question no matter what the style of a question is. For this, we fined-tune some pretrained models and used that fine-tuned model to answer our unlimited questions related to a given historical book(s). We found we got the best results when we finetuned the pretrained t5-small model. T5-small finetuned over 30 epochs gives us the following performance: the mean of the cosine metric is 0.820, mean of the ROUGE1_P is 0.483, BLEU1 score is 0.204. BLEU1 score is the worst score and cosine score is the best score. But, when we compare BLEU1 score of T5-30ep model with other models the 0.204 BLEU score of T5-30ep is the best score among all the models.

Retrieval Augmented Answer Generation System (RAAGS): This was build upon the previous three systems, this component introduces a novel approach that leverages DRS, and AGS to enhance the quality and relevance of generated answers. When we want to get the answer to a question without knowing the section, page, chapter, or book name, but we know that this answer is available in the given text. In that situation, RAAGS helps us. The metrics of this system are as follows: BLEU1 : 0.14, ROUGE1_P : 0.411, ROUGE1_R : 0.393, ROUGE1_F1 : 0.384, ROUGEL_P : 0.323, ROUGEL_R : 0.313, ROUGEL_F1 : 0.303, Precision : 0.316, Recall : 0.304, Cosine : 0.763

5.2 Summary of the study and findings Conclusions

In this work we have worked on 4 components namely Question Answer Generation System (QAGS), Document Retrieval System (DRS), Answer Generation System (AGS), and Retrieval Augmented Answer Generation Systems (RAAGS).

5.2.1 Question Answer Generation System (QAGS)

In this work, we created Question Answer Datasets using 3 techniques namely ChatGPT Front End, ChatGPT API, and ChatPDF.

1. We created a corpus of English Translation work of Mahabharat Book's 18 Parvas, 2100 sections. This dataset is available Parvawise/bookwise and chapterwise. Bookwise dataset is available in text and pdf format. Chapterwise dataset is available in only in text format. 18 PDF of 18 Parva, is 19MB, 18 TXT file 13MB, 2100 section in TXT file 13MB

2. Using various Parvas of Mahabharat we created a Chunk dataset. The chunk dataset has 2746 chunks from all the Parvas. Chunks are generated using langchain CharacterTextSplitter at 8000 character long and 300 character overlap.

3. We selected 331 chunks from the chunk dataset and created 5 chunk vector datasets using 5 sentence embedding models. We didn't create vectors of those chunks which were not used in our work.

4. Using ChatGPT-FE and these 331 chunks we created 1102 question-answer dataset using prompt engineering. HBQA Dataset has question, ref-answer, chunk, and text-reference of answer. Most of these questions are from Book03-Vana Parva. Text reference was used to randomly verify the correctness of answer. Apart from this we used cosine between question-answer, question-chunk, and answer-chunk to ensure the alignment between text, generated questions, and answers.

5. Using ChatGPT API we created 552 QA pairs. Most of these questions are from Book01-Adiparva. Context of these questions is generated using langchain CharacterTextSplitter at 8000 charecter long and 300 characters

overlap. We generated 1084 questions using ChatGPT API but 532 QA pairs were duplicate QA pairs.

6. Using ChatPDF API we created 116 QA pairs. All of these questions are from Book01-Adiparva. The context of these questions is text on a specific page. Adi Parva has 334 pages, so we have 334 context. In this technique, we created 3 (average) questions from every page. But most of those questions were duplicated.

5.2.2 Document Retrieval System (DRS)

In this technique, using 5-sentence embedding models we created 5 question-answer Vector Dataset for 1102 questions, which were created using ChatGPT-FE. We also created 5-chunk-vector datasets. For each question, we searched, the corresponding 5 chunk-vector datasets for the chunk-id. An embedding technique that can retrieve chunk-id most correctly represents the text in the best way. For every question, we retrieved 10 best chunk-id in their order of relevance. The correct chunk-id of every question is available in the HBQA dataset. Using this information, we calculated MRR (mean reciprocal ranking), which shows how far is actual chunk-id from the first place. We found mpnet, a SentenceTransformer model, generated chunk vector produces the best MRR, 0.55. We selected this chunk embedding in the AGS and RAAGS system.

5.2.3 Answer Generation System (AGS)

In this system we used following SOTA models for answer generation, t5, bert, bigbird, bloom, distilbert, roberta, flan-t5, gpt2. Without finetuning the model we supplied questions and chunks. This models predicted different answer for the given questions. We logged all the answers and calculated the BLEU score, ROUGE score, Precision, Recall, and Cosine score for each model's predicted answer. Although none of these scores are satisfying, the t5 model performed best on all these scores. We took t5-small and flan-t5-small and finetuned models on our dataset. We took a smaller version of these pretrained model because of hardware resources availability for model training/finetuning. We got the best results when we trained these models for 30 epochs. Both, t5 and flan-t5 produced almost the same results, t5 a little better. So we selected the t5-small finetuned model for the RAAG sub-system. Metrics of this finetuned model are as follows:, BLEU1 : 0.196, ROUGE1_P : 0.513, ROUGE1_R : 0.455, ROUGE1_F1 : 0.466, ROUGEL_P : 0.409, ROUGEL_R : 0.367, ROUGEL_F1 : 0.373, Precision : 0.393, Recall : 0.35, Cosine : 0.827

A sample answer after the model is trained on T5

> question_chatgpt-FE = ”What predicament does Yudhishthira face, and how does he seek guidance to resolve it?”
>
> ref_ans_from_chatpgpt-F = ”Yudhishthira faces the predicament of being unable to support the Brahmanas who are following him as he departs for the forest.
>
> predicted_answer-T5: He seeks guidance to resolve this dilemma by approaching his priest, Dhaumya, and inquiring about the appropriate course of action.

5.2.4 Retrieval Augmented Answer Generation Systems (RAAGS)

We took another 50 chunks from the Parva-01 of the book. from this, we took those chunks which were not part of DRS and AGS. We created 150 question-answer pairs from this chunk. These chunks and QA pairs were not seen by AGS. We combined these chunks and QA pairs with chunks and QA pairs which we used to develop DRS, and AGS.

In RAAGS, we took the questions from this combined QA dataset. Using DRS we retrieved 5 chunks for each of these questions. With these five chunks, and using AGS we predicted 5 answers for each question. Following this, we concatenated these five answers and supplied them to AGS as context along with the corresponding questions and predicted the final answer. We compared this answer with the reference answer. On the different metrics used for RAAGS evaluation, we found these metrics are close to AGS metrics, where answers were predicted when questions and context were supplied by the user. Metrics of this sub-system are as follows: BLEU1 : 0.14, ROUGE1_P : 0.411, ROUGE1_R : 0.393, ROUGE1_F1 : 0.384, ROUGEL_P : 0.323, ROUGEL_R : 0.313, ROUGEL_F1 : 0.303, Precision : 0.316, Recall : 0.304, Cosine : 0.763

5.2.5 Why Metrics Does Not Look Attractive?

- In QAGS, ChatGPT Generated Questions Answers are open-ended answers, longer answers, sometimes synonym words are used, and sometimes different parts of the chunks are combined. Which is close to any creative, insightful answer generated by any expert human. Generating those kinds of answers from the light model is difficult. We can either experiment with LLM on expensive hardware or a normal model for many more iterations and many hours of training to get better results.

- We need other metrics than today's metrics to evaluate the performance of QAGS, and AGS. Reliance on cosine is not good enough. Neither BLEU nor ROUGE scores are great metrics for evaluating descriptive-generative-creative answers.

- In AGS, for model finetuning, we need longer hours of training on better hardware and with heavy LLMs using the entire corpus, while we used only selected chunks.

- Text cleaning may help further. Old-style English words can be replaced with modern words because today embedding models represent today's languages not ancient-style English or any other old-time language. The use of single or double quotation marks can be removed. Because of the nature of work in Mahabharat, there are many backstories that are written in double quotes. In the chunking process, these quotes are not respected. Our text splitter considered only paragraph marker for breaking the text.

5.3 Implications and applications Future research

5.3.1 Contribution to knowledge

Broadly our work has made following contributions.

1. QA dataset on Mahabharat. 1850+ questions in the English Language.

2. Explored approaches for Question Answer Generation

3. Explored approaches for Answer generation from a historical book.

4. Explored metrics that can help in evaluating this kind of work.

5. Explored sentence embedding models that can be used for Indian historical books translated into English language.

6. Evaluated different SOTA Question Answering Models for Historical text and finetued models for our work.

7. Explored whether we can answer a question without any context.

8. Explored how efficient is document retrieval in the case of Indian Historical text.

5.3.2 The implications and application of our work are as follows.

1. Although we have worked using The Mahabharat Book, evaluating AI technologies for Historical Book Question Answering in the field of AI and NLP is quite interesting and has the potential to contribute to both academia and practical applications. We can extend this work to any other language or any other historical book.

2. Relevance and Significance: The historical book Question Answering is a meaningful and valuable application of AI and NLP. It can help make historical texts more accessible and understandable to a wider audience, including researchers, students, and enthusiasts.

3. Interdisciplinary Nature: Our research involves elements of AI, NLP, and history. AI and NLP are evolving fast and history is becoming more relevant to solve the current struggle, to understand the background, and to understand the value system of societies. Advancing this work can potentially lead to innovative solutions.

4. Real-World Applications: Effective historical book question-answering systems can have practical applications in education, research, and even cultural heritage preservation. Our research could bridge the gap between historical texts and how modern people understand that today.

5. Evaluation and Benchmarking: While building and evaluating our system for historical book question answering, we contribute to the understanding of what metrics should be used for evaluating this kind of work and what value we can expect for those metrics.

6. Data Quality and ChatGPT Limitations: While using ChatGPT for generating reference answers is a creative approach, we need to keep in mind that ChatGPT might not capture all nuances and historical accuracy. Manual verification or expert validation of reference answers might be required. And this become more relevant when we are doing this work around a controversial history work.

7. Ethical and Cultural Sensitivity: Historical texts like the Mahabharat contain sensitive and sacred content. We need to understand to pick up the most authentic translation for this kind of work. Or generate multiple answers and mention as per which book what answer is generated. This helps people learn different viewpoints.

8. Baseline Comparison: Our work helps us understand high dimension embedding does not mean the best sentence vector. We need to understand what kind of sentence embedding model is suitable for our work. Secondly, in the race for LLM which is expensive to train and deploy, we can still work with models like T5.

9. During Academic Class: Based on the content that was taught last week, last day, or today, a teacher can walk through the lessons with the help online quiz. These quizzes can be generated from thousands of AI-generated QA. Sometimes students ask tough questions which is not easy to answer or that need to throw the light from a different perspective. We can use AI-generated answers for this.

10. Creating Question-Answer Books: Rather than having plain history books with tons of stories, we can print a book in the question-answer format. Or we may develop mobile apps based on these question answers.

5.3.3 Business Applications of HBQAS

1. Teaching history without the physical teacher. We can create expert systems that understand history from different perspectives and help students learn the facts and context.

2. We can take the system to any domain like Banking, Insurance, Legal, and Land Records where we have a huge number of text and PDF documents. Users can interact with the system and quickly know the facts that are spread across different documents.

3. In spiritual, philosophical, and political discussions to understand the history from different viewpoints we need SME. We can take the help of HBQAS system to understand the viewpoints.

4. Archaeology sites: They can develop content for their individual sites and engage with tourists in a more entertaining and exciting way. Tourists also feel that they have learned something new about the site. In the absence of this people still go there but most of the time it is like a picnic.

5. Publication: We can create Question-Answer booklets. Rather than having plain history books with tons of stories, we can print a book in the question-answer format. Or we may develop mobile apps based on these question answers.

6. Live TV Shows: Generally television hot debate to understand the history and perspective of people sucks lots of energy and makes people bitter. An anchor

can use QAGS, AGS, RAAGS to drive the debate in a more meaningful way.

7. Quiz in family, Group: People are turning towards their phones and TV, and human-to-human connection is reducing. We can discuss the topic with family members and create a quiz that they can use to play with each other in a family setup.

5.3.4 Future Recommendations

1. Perform AGS experiments on LLM with more parameters and bigger size models.
2. Perform AGS experiments with more epoch
3. In QAGS, generate multiple answers to generated questions. Compare the predicted answer against all the alternative answers.
4. In DRS, create multiple high-dimensional sentence embeddings for Indian Language historical text which is translated into English. And use these embedding in DRS.
5. Create QA in Indian languages. We can translate QA generated using English language and Latin script.
6. Create a sentence embedding model for Indian script (Devanagari, Kannada, Tamil) books.
7. Create a model to predict the correct Devanagari spelling of Roman script Indian Culture name entities like names of people, places, rituals, festivals, etc. Use these spellings during the translation process.
8. In AGS, to calculate BLEU and ROUGE scores, keep only NER in reference and predicted answers.

9. In DRS, take multiple books on multiple topics of the same domain say history. Create smaller chunks and chunk vectors to understand how DRS works

10. In RAAGS, accept questions from multiple books and multiple topics.

11. In RAAGS, instead of joining multiple answers, stuff the original chunks using langchain and pass that final chunk to AGS, to get the answer.

12. Human-in-the-Loop Evaluation: Given the unique nature of the historical text, involving domain experts or historians in the evaluation process could provide valuable insights and enhance the quality of the evaluation.

13. Types of Questions: In our work, we have explored open-ended answers but we can take this work forward for other formats like fill-in-the-blank, and MCQ.

14. The QA dataset created in this project can be translated into other Indian Languages. For that purpose, we need to do NER on the English text. Get the correct spelling of the word. Roman language has a limitation in writing Sanskrit names correctly. Due to this reason when Indian language text is translated into English language noun spelling becomes wrong. For example, Yagnya, Pruthuvi, Sravanan, Ramayan, Ram, Krushna these words have sounds that are not in European languages hence these words are wrongly written in Roman script.

5.4 Summary

In this chapter, we have undertaken a thorough examination of our research on AI-powered Historical Book Question Answering, which involved the development and evaluation of four critical sub-systems: the Question Answer Generation System, Document Retrieval System, Answer Generation System, and Retrieval Augmented Answer Generation System. Our research has provided valuable insights and it has the potential of revolutionizing the learning process around the historical text. The following are the key points that emerge from our investigation:

System Performance and Contributions: Our evaluation of the four sub-systems revealed their individual strengths and limitations. The Question Answer Generation System demonstrated its ability to formulate contextually relevant questions, while the Document Retrieval System efficiently accessed historical chunks/texts. The Answer Generation System (AGS) produced coherent responses, and the Retrieval Augmented Answer Generation System enhanced answer quality by combining DRS and AGS. These systems collectively contribute to the advancement of historical book question answering.

Overall System Integration: The successful integration of these sub-systems into a holistic AI-powered Historical Book question-answering framework has showcased the potential for an end-to-end solution that can facilitate historical knowledge extraction. The synergy among these systems represents a critical milestone in the field, offering a comprehensive approach to addressing the complexities of historical document analysis.

Performance Metrics and Evaluation: Our rigorous evaluation, including metrics such as BLUE, ROUGE, Cosine, Accuracy, Recall, and Precision, has provided quantitative evidence of the effectiveness of our systems. We have demonstrated improved performance in terms of question relevance, document retrieval efficiency, and answer quality, as compared to traditional methods. Those traditional systems are not scalable, slow, and prone to human biases and judgments.

Challenges and Future Directions: While our research has achieved considerable success, it is important to acknowledge the challenges that remain. Issues related to domain-specificity, handling diverse historical texts, solubility, finding new metrics to evaluate these systems, using LLMs for training, and inference on cheaper hardware represent avenues for further exploration. Future research should aim to enhance the adaptability, generalizability, and cost-effectiveness of these systems.

Broader Implications: Our work in AI-powered Historical Book Question Answering extends beyond the immediate applications. It offers substantial implications for the broader fields of artificial intelligence, information retrieval, and historical research. By improving access to historical knowledge, our systems contribute to preserving and disseminating the past, setting the historical context in the right way, and analyzing and answering questions from different translations in real-time at the least cost. Understanding history in a correct way, in the context opens doors for peace in the society. Otherwise, we keep fighting for the wrong reasons and keep justifying that by out-of-context past.

Cross-Disciplinary Relevance: The cross-disciplinary nature of this research is noteworthy. It holds significance for historians, archivists, educators, and researchers in diverse domains. The tools and methodologies developed can empower these professionals in their endeavors to explore and understand historical texts.

In conclusion, our research in ”AI Powered Historical Book Question Answering” has brought forth innovative solutions for the challenging task of extracting historical knowledge from vast archives of texts. Through the development and integration of the four sub-systems, we have demonstrated the feasibility of enhancing the accessibility of historical information. This work stands as a testament to the power of AI in augmenting historical research and has the potential to revolutionize the way we interact with and learn from historical texts. As we conclude this chapter, it is essential to acknowledge that the journey is far from over. The pursuit of refining and expanding the capabilities of our systems, addressing existing limitations, and exploring new applications in the realm of historical document analysis is a promising avenue for future research. The implications of our work extend into a future where AI and historical scholarship intersect to unlock the treasures buried within the pages of history.

Bibliography

Acharya, S., Sornalakshmi, K., Paul, B. & Singh, A. (2022), Question answering system using nlp and bert, IEEE, pp. 925–929.
URL: *https://ieeexplore.ieee.org/document/9952050/*

Agrawal, A., Shukla, P., Panicker, A. & Dhawe, T. (2019), 'Automatic fill in the blank question with distractor generation using nlp', *International Journal of Computer Sciences and Engineering* **7**, 892–897.
URL: *http://www.ijcseonline.org/full_paper_view.php ?paper_id=4650*

Anita, R. & Subalalitha, C. (2019), An approach to cluster tamil literatures using discourse connectives, IEEE, pp. 1–4.
URL: *https://ieeexplore.ieee.org/document/8938315/*

Arbaaeen, A. & Shah, A. (2021), 'Ontology-based approach to semantically enhanced question answering for closed domain: A review', *Information* **12**, 200.
URL: *https://www.mdpi.com/2078-2489/12/5/200*

Berger, G., Rischewski, T., Chiruzzo, L. & Rosa, A. (2022), Generation of english question answer exercises from texts using transformers based models, IEEE, pp. 1–5.
URL: *https://ieeexplore.ieee.org/document/9981171/*

Bevir, M. (2000), ‘The text as a historical object’.
URL: *https://escholarship.org/uc/item/6g97x5v2*

Brown, M. & Shackel, P. (2023), ‘Text mining oral histories in historical archaeology’, *International Journal of Historical Archaeology* .
URL: *https://link.springer.com/10.1007/s10761-022-00680-5*

Butterfield, S., Costello, E., Henderson, C. & Mourachov, S. (2013), ‘Slack’. [Online; accessed 07-September-2023].
URL: *https://en.wikipedia.org/wiki/Slack_(software)*

Calvo, R. A., O’Rourke, S. T., Jones, J., Yacef, K. & Reimann, P. (2011), ‘Collaborative writing support tools on the cloud’, *IEEE Transactions on Learning Technologies* **4**, 88–97.
URL: *http://ieeexplore.ieee.org/document/5669252/*

Chen, P., Wu, F., Wang, T. & Ding, W. (2018), ‘A semantic qa-based approach for text summarization evaluation’, *Proceedings of the AAAI Conference on Artificial Intelligence* **32**.
URL: *https://ojs.aaai.org/index.php/AAAI/article/view/11911*

Cho, J., Seo, M. & Hajishirzi, H. (2019), Mixture content selection for diverse sequence generation, Association for Computational Linguistics, pp. 3119–3129.
URL: *[https://www.aclweb.org/anthology/D19-1308, https://github.com/clovaai/FocusSeq2Seq]*

Clark, K., Luong, M.-T., Le, Q. V. & Manning, C. D. (2020), ‘Electra: Pre-training text encoders as discriminators rather than generators’.
URL: *https://github.com/google-research/ http://arxiv.org/abs/2003.10555*

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V. & Salakhutdinov, R. (2020), Transformer-xl: Attentive language models beyond a fixed-length context.

Debroy, B. & Chauhan, K. (2022), ‘Sanskrit manuscripts rotting, should be translated: Debroy’.
URL: *https://www.tribuneindia.com/news/archive/himachal/sanskrit-manuscripts-rotting-should-be-translated-debroy-704931*

Deena, G. & Raja, K. (2022), ‘Objective type question generation using natural language processing’, *International Journal of Advanced Computer Science and Applications* **13**.
URL: *http://thesai.org/Publications/ViewPaper ?Volume=13&Issue=2&Code=IJACSA&SerialNo=63*

Devlin, J., Chang, M. W., Lee, K. & Toutanova, K. (2019), ‘Bert: Pre-training of deep bidirectional transformers for language understanding’, *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* **1**, 4171–4186.

Donovan, A. M., Zhan, J. & Rapp, D. N. (2018), ‘Supporting historical understandings with refutation texts’, *Contemporary Educational Psychology* **54**, 1–11.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S0361476X17304009*

Durov, N. & Durov, P. (2013), ‘Telegram’. [Online; accessed 07-September-2023].
URL: *https://en.wikipedia.org/wiki/Telegram_(software)*

Ferrucci, D. (2013), ‘Ibm wastson’. [Online; accessed 07-September-2023].
URL: *https://en.wikipedia.org/wiki/IBM_Watson*

Furlan, R., Gatti, M., Menè, R., Shiffer, D., Marchiori, C., Levra, A. G., Saturnino, V., Brunetta, E. & Dipaola, F. (2021), ‘A natural language processing–based virtual patient simulator and intelligent tutoring system for the clinical diagnostic process: Simulator development and case study’, *JMIR Medical Informatics*

9, e24073.
URL: *https://medinform.jmir.org/2021/4/e24073*

Ganguli, K. M. (1896), ‘Mahabharata translation by ganguly accessed on 08-aug-23’.
URL: *https://www.mahabharataonline.com/*

GEMİRTER, C. B. & GOULARAS, D. (2021), ‘A turkish question answering system based on deep learning neural networks’, *Journal of Intelligent Systems: Theory and Applications* **4**, 65–75.
URL: *https://dergipark.org.tr/en/doi/10.38016/jista.815823*

Goar, V., Yadav, N. S. & Yadav, P. S. (2023), ‘Conversational ai for natural language processing: An review of chatgpt’, *International Journal on Recent and Innovation Trends in Computing and Communication* **11**, 109–117.
URL: *https://ijritcc.org/index.php/ijritcc/article/view/6161*

Gruener, G. & Warthen, D. (1996), ‘Ask.com’. [Online; accessed 07-September-2023].
URL: *https://en.wikipedia.org/wiki/Ask.com*

Gumaste, P., Joshi, S., Khadpekar, S. & Mali, S. (2020), ‘Automated question generator system using nlp’, *International Research Journal of Engineering and Technology (IRJET)* **7**.
URL: *https://www.irjet.net/archives/V7/i6/IRJET-V7I6848.pdf*

Guu, K., Lee, K., Tung, Z., Pasupat, P. & Chang, M. W. (2020), Realm: Retrieval-augmented language model pre-training, Vol. PartF168147-6.
URL: *https://arxiv.org/abs/2002.08909*

He, P., Liu, X., Gao, J. & Chen, W. (2021), Deberta: Decoding-enhanced bert with disentangled attention.

Hemanth*, C., Venkat, M., Babu, B. C., Rochasvi, U. & Daanesh, P. (2020), 'Random multiple choice questions generation using nlp', *International Journal of Innovative Technology and Exploring Engineering* **9**, 395–397.
URL: *https://www.ijitee.org/portfolio-item/I7104079920/*

Huang, Y.-F. & Li, Y.-H. (2021), 'Translating sentimental statements using deep learning techniques', *Electronics* **10**, 138.
URL: *https://www.mdpi.com/2079-9292/10/2/138*

Ignatow, G. & Mihalcea, R. (2018), *An Introduction to Text Mining: Research Design, Data Collection, and Analysis*, SAGE Publications, Inc.
URL: *https://methods.sagepub.com/book/an-introduction-to-text-mining*

Jabberwacky (1997), 'Jabberwacky ai'. [Online; accessed 07-September-2023].
URL: *http://www.jabberwacky.com/j2about*

Jacquemart, P. & Zweigenbaum, P. (2003), Towards a medical question-answering system: A feasibility study, Vol. 95.
URL:
https://www.researchgate.net/publication/5765534_Towards_a_medical_question-answering_system_A_feasibility_study

Jadhav, M. P. P. & Laddha, M. M. D. (2017), 'An automatic gap filling questions generation using nlp', *Ijcset.Com* **8**.
URL: *https://www.ijcset.com/docs/IJCSET17-08-08-039.pdf*

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C. & Socher, R. (2019), 'Ctrl: A conditional transformer language model for controllable generation'.
URL: *http://arxiv.org/abs/1909.05858*

Khanuja, S., Dandapat, S., Srinivasan, A., Sitaram, S. & Choudhury, M. (2020), Gluecos: An evaluation benchmark for code-switched nlp, Association for

Computational Linguistics, pp. 3575–3585.
URL: *https://www.aclweb.org/anthology/2020.acl-main.329*

Krishnamoorthy, V. (2021), ‘Evolution of reading comprehension and question answering systems’, *Procedia Computer Science* **185**, 231–238.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S1877050921011066*

Kulshreshtha, D., Shayan, M., Belfer, R., Reddy, S., Serban, I. V. & Kochmar, E. (2022), *Few-Shot Question Generation for Personalized Feedback in Intelligent Tutoring Systems*, Vol. 351.
URL: *https://ebooks.iospress.nl/doi/10.3233/FAIA220062*

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. & Soricut, R. (2019), ‘Albert: A lite bert for self-supervised learning of language representations’.
URL: *https://github.com/google-research/ALBERT. http://arxiv.org/abs/1909.11942*

Lekova, A., Tsvetkova, P., Tanev, T., Mitrouchev, P. & Kostova, S. (2022), ‘Making humanoid robots teaching assistants by using natural language processing (nlp) cloud-based services’, *Journal of Mechatronics and Artificial Intelligence in Engineering* **3**, 30–39.
URL: *https://www.extrica.com/article/22720*

Lende, S. P. & Raghuwanshi, M. (2016*a*), ‘Closed domain question answering system using nlp techniques’, *IJESRT* **2277**.
URL: *https://doi.org/10.5281/zenodo.49808*

Lende, S. P. & Raghuwanshi, M. M. (2016*b*), Question answering system on education acts using nlp techniques, IEEE, pp. 1–6.
URL: *http://ieeexplore.ieee.org/document/7583963/*

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V.

& Zettlemoyer, L. (2020), Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S. & Kiela, D. (2020), Retrieval-augmented generation for knowledge-intensive nlp tasks, Vol. 2020-December.

Liang, X., Cheddad, A. & Hall, J. (2021), 'Comparative study of layout analysis of tabulated historical documents', *Big Data Research* **24**, 100195.
URL: *https://linkinghub.elsevier.com/retrieve/pii/S2214579621000125*

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019), 'Roberta: A robustly optimized bert pretraining approach'.
URL: *https://github.com/pytorch/fairseq http://arxiv.org/abs/1907.11692*

Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J. & Roberts, A. (2023), 'The flan collection: Designing data and methods for effective instruction tuning', *arxiv* .
URL: *http://arxiv.org/abs/2301.13688*

Lopez, L. E., Cruz, D. K., Cruz, J. C. B. & Cheng, C. (2020*a*), 'Simplifying paragraph-level question generation via transformer language models', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **13032 LNAI**, 323–334.
URL: *http://arxiv.org/abs/2005.01107*

Lopez, L. E., Cruz, D. K., Cruz, J. C. B. & Cheng, C. (2020*b*), 'Transformer-based end-to-end question generation', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **13032 LNAI**.

URL: *https://www.researchgate.net/publication/341148367_Transformer-based_End-to-End_Question_Generation*

Luitse, D. & Denkena, W. (2021), 'The great transformer: Examining the role of large language models in the political economy of ai', *Big Data & Society* **8**, 205395172110477.
URL: *http://journals.sagepub.com/doi/10.1177/20539517211047734*

Mangrulkar, S. & Paul, S. (2023), 'Peft: Parameter-efficient fine-tuning of billion-scale models on low-resource hardware'. [Online; accessed 09-October-2023].
URL: *https://huggingface.co/blog/peft*

Marrese-Taylor, E., Nakajima, A., Matsuo, Y. & Yuichi, O. (2018), Learning to automatically generate fill-in-the-blank quizzes, Association for Computational Linguistics, pp. 152–156.
URL: *https://aclanthology.org/W18-3722.pdf*

Moruwawon, B. S. (2010), 'Translating african proper names in literary texts', *Journal of Universal Language* **11**, 207–226.
URL: *http://www.sejongjul.org/archive/view_article?pid=jul-11-2-207*

N, S. S. & B, S. (2022), A convolutional autoencoder based keyword spotting in historical handwritten devanagari documents, IEEE, pp. 356–362.
URL: *https://ieeexplore.ieee.org/document/9850900/*

Nakanishi, M., Kobayashi, T. & Hayashi, Y. (2019), Towards answer-unaware conversational question generation, Association for Computational Linguistics, pp. 63–71.
URL: *https://www.aclweb.org/anthology/D19-5809*

Nakatsuji, M. & Okui, S. (2020), Answer generation through unified memories over multiple passages, Vol. 2021-January, International Joint Conferences on Artificial Intelligence Organization, pp. 3823–3829.
URL: *https://www.ijcai.org/proceedings/2020/529*

Nimavat, K. (2010), ‘chatbots-an overview types, architecture, tools and future possibilities’. [Online; accessed 07-September-2023].
URL: *https://www.researchgate.net/publication/320307269_Chat-bots_An_overview_Types_Architecture_Tools_and_Future_Possibilities*

Palasundram, K., Sharef, N. M., Kasmiran, K. A. & Azman, A. (2021), ‘Seq2seq++: A multitasking-based seq2seq model to generate meaningful and relevant answers’, *IEEE Access* **9**.

Panchal, P., Thakkar, J., Pillai, V. & Patil, S. (2021), ‘Automatic question generation and evaluation’, *Journal of University of Shanghai for Science and Technology* **23**, 751–761.
URL: *https://jusst.org/automatic-question-generation-and-evaluation/*

Paranjape, B., Lamm, M. & Tenney, I. (2022), Retrieval-guided counterfactual generation for qa, Vol. 1, Association for Computational Linguistics, pp. 1670–1686.
URL: *https://aclanthology.org/2022.acl-long.117*

Patel, R. & Patel, S. (2021), *Deep Learning for Natural Language Processing*, Vol. 190, pp. 523–533.
URL: *https://link.springer.com/10.1007/978-981-16-0882-7_45*

Peng, X., Zheng, Y., Lin, C. & Siddharthan, A. (2021), Summarising historical text in modern languages, Association for Computational Linguistics, pp. 3123–3142.
URL: *https://aclanthology.org/2021.eacl-main.273*

Pereira, J., Fidalgo, R., Lotufo, R. & Nogueira, R. (2023), *Visconde: Multi-document QA with GPT-3 and Neural Reranking*, Vol. 13981 LNCS, pp. 534–543.
URL: *https://link.springer.com/10.1007/978-3-031-28238-6_44*

Pettersson, E., Megyesi, B. & Nivre, J. (2013), 'Normalisation of historical text using context-sensitive weighted levenshtein distance and compound splitting', *... of the 19th Nordic Conference of ...* **1**, 163–179.
URL: *http://www.ep.liu.se/ecp/085/017/ecp1385017.pdf*

Quarteroni, S., Kingdom, U. & Manandhar, S. (2007), 'A chatbot-based interactive question answering system', *Decalog* .

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J. (2019), 'Exploring the limits of transfer learning with a unified text-to-text transformer', *Journal of Machine Learning Research* **21**.
URL: *http://arxiv.org/abs/1910.10683*

Rahaman, M. S., Ahsan, M. M. T., Anjum, N., Terano, H. J. R. & Rahman, M. M. (2023), 'From chatgpt-3 to gpt-4: A significant advancement in ai-driven nlp tools', *Journal of Engineering and Emerging Technologies* **1**, 50–60.
URL: *https://journals.cspc.edu.ph/index.php/jeet/article/view/188*

Rami, P. & Pawar, K. (2017), 'A framework for restricted domain question answering system using advanced nlp tools and software', *International Journal of Research In Science and Reviews in Information Sciences* .
URL:
http://cdn.iiit.ac.in/cdn/ltrc.iiit.ac.in/icon/2013/proceedings/File50-paper54.PDF

Ramli, F. & Noah, S. A. M. (2016), 'Building an event ontology for historical domain to support semantic document retrieval', *International Journal on Advanced*

Science, Engineering and Information Technology **6**, 1154.
URL: *http://ijaseit.insightsociety.org/index.php ?option=com_content&view=article&id=9&Itemid=1&article_id=1634*

Ramli, F., Noah, S. A. M. & Kurniawan, T. B. (2020), 'Using ontology-based approach to improved information retrieval semantically for historical domain', *International Journal on Advanced Science, Engineering and Information Technology* **10**, 1130.
URL: *http://ijaseit.insightsociety.org/index.php ?option=com_content&view=article&id=9&Itemid=1&article_id=10180*

Rawat, A. & Samant, S. S. (2022), Comparative analysis of transformer based models for question answering, IEEE, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/10046525/*

Riza, L. S., Firdaus, Y., Sukamto, R. A., Wahyudin & Samah, K. A. F. A. (2023), 'Automatic generation of short-answer questions in reading comprehension using nlp and knn', *Multimedia Tools and Applications* .
URL: *https://link.springer.com/10.1007/s11042-023-15191-6*

Riza, L. S., Pertiwi, A. D., Rahman, E. F., Munir, M. & Abdullah, C. U. (2019), 'Question generator system of sentence completion in toefl using nlp and k-nearest neighbor', *Indonesian Journal of Science and Technology* **4**, 294–311.
URL: *https://ejournal.upi.edu/index.php/ijost/article/view/18202*

Sanh, V., Debut, L., Chaumond, J. & Wolf, T. (2019), 'Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter'.
URL: *https://github.com/huggingface/transformers http://arxiv.org/abs/1910.01108*

Sarkar, S., Singh, P., Kumari, N. & Kashtriya, P. (2023), *The Task of Question*

Answering in NLP: A Comprehensive Review, Vol. 1011 LNEE, pp. 603–611.
URL: *https://link.springer.com/10.1007/978-981-99-0601-7_46*

Sharma, V., Law, W., Balick, M. J. & Sarkar, I. N. (2017), 'Harnessing biomedical natural language processing tools to identify medicinal plant knowledge from historical texts', *AMIA ... Annual Symposium proceedings. AMIA Symposium* **2017**, 1537–1546.
URL: *https://pubmed.ncbi.nlm.nih.gov/29854223/*

Shoeybi, M., Patwary, M., Puri, R., Legresley, P., Casper, J. & Catanzaro, B. (n.d.), 'Megatron-lm: Training multi-billion parameter language models using model parallelism'.
URL: *https://github.com/*

Singh, D., Suraksha, K. & Nirmala, S. (2021), Question answering chatbot using deep learning with nlp, IEEE, pp. 1–6.
URL: *https://ieeexplore.ieee.org/document/9622709/*

Soni, S., Kumar, P. & Saha, A. (2019), 'Automatic fill-in-the-blank type questions and answer generation from text using nlp', *Journal of Advanced Research in Dynamical and Control Systems* **11**, 36–43.
URL: *http://jardcs.org/abstract.php*
?id=2871

Sprugnoli, R. & Tonelli, S. (2019), 'Novel event detection and classification for historical texts', *Computational Linguistics* **45**, 229–265.
URL: *https://direct.mit.edu/coli/article/45/2/229-265/1630*

Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H. & Wu, H. (n.d.), 'Ernie: Enhanced representation through knowledge integration'.
URL: *https://github.com/PaddlePaddle/*

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y. & Zhou, D. (2020), Mobilebert: a compact task-agnostic bert for resource-limited devices, Association for Computational Linguistics, pp. 2158–2170.
URL: *https://www.aclweb.org/anthology/2020.acl-main.195*

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. & Lample, G. (2023), 'Llama: Open and efficient foundation language models'.
URL: *http://arxiv.org/abs/2302.13971*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser & Polosukhin, I. (2017), 'Attention is all you need', *Advances in Neural Information Processing Systems* **2017-Decem**, 5999–6009.

Vincentio, K. & Suhartono, D. (2022), 'Automatic question generation monolingual multilingual pre-trained models using rnn and transformer in low resource indonesian language', *Informatica* **46**.
URL: *https://www.informatica.si/index.php/informatica/article/view/4236*

Wang, X., Fan, S., Houghton, J. & Wang, L. (2022), Towards process-oriented, modular, and versatile question generation that meets educational needs, Association for Computational Linguistics, pp. 291–302.
URL: *https://aclanthology.org/2022.naacl-main.22*

Wang, X., Kapanipathi, P., Musa, R., Yu, M., Talamadupula, K., Abdelaziz, I., Chang, M., Fokoue, A., Makni, B., Mattei, N. & Witbrock, M. (2019), 'Improving natural language inference using external knowledge in the science questions domain', *Proceedings of the AAAI Conference on Artificial Intelligence* **33**, 7208–7215.
URL: *https://ojs.aaai.org/index.php/AAAI/article/view/4705*

Weizenbaum, J. (1966), 'Eliza: a very basic rogerian psychotherapist chatbot'.

[Online; accessed 07-September-2023].
URL: *https://web.njit.edu/ ronkowit/eliza.html*

Winograd, T. (1971), 'Lunar (qa) system'. [Online; accessed 07-September-2023].
URL: *https://www.gabormelli.com/RKB/LUNAR_(QA)_System*

Witteveen, S. & Andrews, M. (2019), Paraphrasing with large language models, Association for Computational Linguistics, pp. 215–220.
URL: *https://www.aclweb.org/anthology/D19-5623*

Workshop, B., Scao, T. L. & Authors, . (2022), 'Bloom: A 176b-parameter open-access multilingual language model', *arXiv* .
URL: *http://arxiv.org/abs/2211.05100*

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L. & Ahmed, A. (2020), Big bird: Transformers for longer sequences, Vol. 2020-December.
URL: *https://arxiv.org/abs/2007.14062*

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T. & Zettlemoyer, L. (2022), 'Opt: Open pre-trained transformer language models'.
URL: *http://arxiv.org/abs/2205.01068*

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J. & Dolan, B. (2020), Dialogpt : Large-scale generative pre-training for conversational response generation, pp. 270–278.
URL: *https://github.com/mgalley/*

Zhou, S. & Zhang, Y. (2021), 'Datlmedqa: A data augmentation and transfer learning based solution for medical question answering', *Applied Sciences*

11, 11251.

URL: *https://www.mdpi.com/2076-3417/11/23/11251*

APPENDIX A: List and Definition of Commonly Used Metrics Used in NLP

Exact match (EM)

This metric measures the percentage of questions that the system answers perfectly, meaning that the system's answer exactly matches one of the gold answers. This metric is suitable for evaluating extractive answers or MCQ problems.

$EM = \frac{Number\ of\ exact\ matches}{Number\ of\ questions}$

Recall

Number of common words/tokens between reference answer and predicted answer divided by number of words/tokens in reference answer. This is used to evaluate each predicted answer. This is not for overall model performance. We can take an average of this for all the predictions and calculate the Mean Recall. The value of recall varies between 0 to 1. 1 being the best.

Precision

Number of common words/tokens between reference answer and predicted answer divided by number of words/tokens in predicted answer. This is used to evaluate each predicted answer. This is not for overall model performance. We can take the average of this for all the predictions and calculate the Mean Precision. The value of precision varies between 0 to 1. 1 being the best.

F1 Score

We can calculate this for each predicted answer. This is not for overall model performance. We can take the average of this for all the predicted answers and calculate the Mean F1 Score. The value of the F1 Score varies between 0 to 1. 1 being the best.

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

Mean average precision (MAP)

MAP is a metric that evaluates the ranking performance of an information retrieval system, such as a question-answering or text classification system. To calculate MAP, we need to first calculate the average precision (AP) for each query in the test set. AP is a measure of how well the system retrieves relevant documents for a given query. It is calculated by taking the precision of each relevant document and averaging over all relevant documents.

Precision is a measure of how many of the retrieved documents are relevant to the query. It is calculated by dividing the number of relevant documents by the number of retrieved documents. For example, if a system retrieves 10 documents for a query, and 4 of them are relevant, then the precision is 4/10 = 0.4.

Rank	Document	Relevance
1	Chunk 1	Yes
2	Chunk 2	No
3	Chunk 3	Yes
4	Chunk 4	No
5	Chunk 5	Yes

Rank	Document	Relevance	Precision
1	Chunk 1	Yes	1/1 = 1
2	Chunk 2	No	-
3	Chunk 3	Yes	2/3 = 0.67
4	Chunk 4	No	-
5	Chunk 5	Yes	3/5 = 0.6

AP = (1 + 0.67 + 0.6) / 3 = 0.756

Query	AP
Question 1	0.756
Question 2	0.8
Question 3	0.9

Cosine similarity

Cosine similarity can be used to evaluate the answers to open-ended questions. However, it is important to note that cosine similarity is a measure of similarity, not accuracy. This means that cosine similarity can be used to identify answers that are similar to each other, but it cannot be used to determine whether an answer is correct.

Cosine similarity can be used to evaluate the answers to open-ended questions. The cosine similarity between the model's answer and each of the reference answers can then be calculated. The model's answer is considered to be relevant if it has a high cosine similarity with one of the reference answers.

Found@n

It is like reciprocal rank. If a reference document is found in predicted document sets at first place then Found@1 is Yes, RR = 1/1= 1. If the reference document is found at the 2nd position then Found@1=False, Found@2=True. RR=1/2=0.5. If reference document is found at the 5th position then Found@1, @2, @3, @4 = False, Found@5=True, RR=1/5=0.2. To maintain readability we are keeping this metric in our performance metrics.

MRR

MRR stands for "Mean Reciprocal Rank," and it is a metric commonly used in information retrieval and evaluation tasks, including those in natural language processing (NLP). MRR is used to assess the effectiveness of ranking algorithms or systems in presenting relevant information to users. MRR is often applied to tasks such as question answering, search engines, and recommendation systems. A higher MRR indicates that relevant results tend to appear higher in the ranked lists, which suggests a better user experience.
Formula: $MRR = \frac{1}{N} * \sum_{i=1}^{N} \frac{1}{rank_i}$
N is the total number of queries or questions. rank_i is the position of the correct result for the i-th query.
$MRR@5 = \frac{1}{5} * \sum_{i=1}^{5} \frac{1}{rank_i}$
Calculation Steps:

1. Step 1: Ranking of Results: Imagine you have a system that retrieves a list of possible answers or documents in response to a user's query or question. These results are usually ranked based on their perceived relevance to the query.

2. Step 2: Reciprocal Rank: For each query or question, the reciprocal rank of

the first correct (relevant) result in the ranked list is calculated as 1 divided by the position of that correct result. If the correct result is the second item, the reciprocal rank would be 1/2; if it's the fifth, the reciprocal rank would be 1/5, and so on.

3. Step 3: Mean Reciprocal Rank (MRR): To calculate the MRR, you take the average of the reciprocal ranks across all queries or questions in your evaluation dataset. The formula for MRR is:

BLEU (Bilingual Evaluation Understudy)

BLEU is based on the n-gram overlap between the machine-generated and reference translations. It is a recall metric. An n-gram is a contiguous sequence of n items from a given sample of text. For example, in the sentence "The cat sat on the mat", the 2-grams are "The cat", "cat sat", "sat on", "on the", and "the mat". BLEU computes a score between 0 and 1, where 1 indicates a perfect match between the machine-generated and reference translations. The score is calculated as a weighted geometric mean of the n-gram precision, where the weights are determined by the length of the machine-generated translation. BLEU score is usually computed for unigrams (BLEU-1), bigrams (BLEU-2), trigrams (BLEU-3), and four-grams (BLEU-4). This metric is like the precision metric of classical machine learning. This is created for evaluating translation tasks but can be used for summarization and question-answering tasks.
The formula for calculating the BLEU score is as follows:
BLEU = $BP \cdot exp(\frac{1}{n}\sum_{i=1}^{n} w_i \cdot log(p_i))$
where:

BP is the $\boldsymbol{brevitypenalty}$, which is a correction factor that penalizes translations that are shorter than the reference translations.

n is the maximum n-gram length (usually 4).

w_i is the weight assigned to the i-th n-gram precision score.

p_i is the i-th n-gram precision score.

The precision score for an n-gram is calculated as the number of common tokens between the machine-generated and reference translations divided by the number of tokens in the machine-generated translation. The weights w_i are usually set to $\frac{1}{n}$.
The brevity penalty is calculated as follows:
BP = if c>r then 1 else $e^{(1-\frac{r}{c})}$
where:

c is the length of the machine-generated translation.

r is the length of the reference translation that has the closest length to c.

GLUE General Language Understanding Evaluation)

GLUE benchmark is developed by Google, it consists of a collection of nine natural language processing (NLP) tasks. We can say it is Google BLEU and much more. BLEU is designed to compare prediction with 1+ reference answers. While GLUE considers only 1 reference answer. It was designed to evaluate the general-purpose language understanding of NLP models. The nine tasks on nine datasets in GLUE are:

1. MNLI: Multi-Genre Natural Language Inference
2. QNLI: Question Answering over Natural Language Inferences
3. SST-2: Stanford Sentiment Treebank
4. MRPC: Microsoft Research Paraphrase Corpus

5. CoLA: Corpus of Linguistic Acceptability
6. STS-B: Semantic Text Similarity Benchmark
7. RTE: Recognizing Textual Entailment
8. WNLI: Winograd NLI
9. BoolQ: Boolean Questions

GLUE is designed to evaluate NLP models on a variety of tasks, while BLEU is specifically designed to evaluate machine translation models. GLUE uses a variety of metrics to evaluate model performance, while BLEU uses a single metric (n-gram precision). GLUE is more challenging than BLEU, as it includes tasks that require a deeper understanding of language.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

This metric measures how well a model's output summary matches a human-generated summary. It is commonly used in text summarization and question-answering tasks. This metric is like the recall metric of classical machine learning.

Formula for ROUGE is ROUGE-2 = $\frac{\sum_{i=1}^{m}(S_i \cap R_i)}{\sum_{i=1}^{m}(R_i)}$

To calculate ROUGE-2, we first need to identify the bigrams (pairs of words) in the generated text and the reference text. We then count the number of bigrams that are shared between the generated text and the reference text for each question. We then sum the number of shared bigrams for all questions and divide this sum by the total number of bigrams in the reference text.

Here are some examples:

Example 1:

Generated text: ”The capital of Kauravas was Hastinapur.” Reference text: ”Hastinapur was the capital of Kauravas.”

Bigrams in the generated text: ”The capital”, ”capital of”, ”of Kauravas”, ”Kauravas was”, ”was Hastinapur” Bigrams in the reference text: ”Hastinapur was”, ”was the”, ”the capital”, ”capital of”, ”of Kauravas” Shared bigrams: ”Hastinapur was”, ”was the”, ”capital of”, ”of Kauravas”

ROUGE-2: 4 / 5 = 0.80

Example 2: Generated text: ”The Shiva is blue.” Reference text: ”The Shiva is blue and the Kailasha is white.”

Bigrams in the generated text: ”The Shiva”, ”Shiva is”, ”is blue” Bigrams in the reference text: ”The Shiva”, ”Shiva is”, ”is blue”, ”and the”, ”the Kailasha”, ”Kailasha is”, ”is white” Shared bigrams: ”The Shiva”, ”Shiva is”, ”is blue”

ROUGE-2: 3 / 7 = 0.43

In general, a higher ROUGE-2 score indicates that the generated text is more similar to the reference text in terms of bigrams. However, it is important to note that ROUGE-2 is not a perfect metric for evaluating the quality of generated text. It is possible for a generated text to have a high ROUGE-2 score but still be semantically incorrect.

For example, suppose that the generated text is ”The capital of Nagas is Hastinapur.” and the reference text is ”Padmavati is the capital of Nagas.” The bigrams in the generated text and the reference text are identical, so the ROUGE-2 score would be 1.0. However, the generated text is semantically incorrect, because the capital of Nagas is not Hastinapur.

If our dataset has only above sentences then ROUGE-2 = (4+3)/(5+7) = 7/12 =.58

Despite its limitations, ROUGE-2 is a useful metric for evaluating the quality of generated text. It is a relatively simple metric to calculate, and it is well-correlated with human judgments of the quality of the generated text.

Precision at K (P@K)

Precision at K measures the precision of the system's answer among the top K-ranked answers. It assesses how well the system performs when considering the top K answers, where K is a predefined number.

P@K = $\frac{Number\ of\ correct\ answers\ in\ top\ K}{K}$

Let's say for a question we have 10 answers (prediction). 3 of those answers are correct, as per the reference answer. We want to calculate P@5. So, we will take the top 5 results (the model also returns cosine values, and ranking is done on this cosine). Now in these 5 answers, we find only 1 answer is correct and the remaining 4 are wrong. P@5 = 1/5

P@k needs to be calculated for each answer and finally, we can take the average of all values.

Recall at K (R@K)

Recall at K measures the recall of the system's answer among the top K-ranked answers. It assesses how well the system retrieves the correct answer when considering the top K answers.

R@K = $\frac{Number\ of\ correct\ answers\ in\ top\ K}{Total\ number\ of\ correct\ answers}$

Let's say for a question we have 10 answers (prediction). 3 of those answers are correct, as per the reference answer. We want to calculate R@5. So, we will take the top 5 results (the model also returns cosine values, and ranking is done on this cosine). Now in these 5 answers, we find only 1 answer is correct and the remaining 4 are wrong. R@5 = 1/3

R@k needs to be calculated for each answer and finally, we can take the average of all values.

Normalized Discounted Cumulative Gain (NDCG)

NDCG stands for Normalized Discounted Cumulative Gain. It is a measure of the quality of a ranking of items, such as the results of a search engine query or the recommendations or plausible answers. NDCG takes into account the relevance of the items in the ranking, as well as their position in the ranking. It assigns higher scores to correct answers that are ranked higher. The ranking of the predicted answer is done by a human agent. It calculates the cumulative gain (CG) by summing the relevance scores of the ranked items. However, it discounts the gain of items as they appear lower in the ranking. The formula for NDCG is as follows: NDCG@K = $\frac{DCG@K}{IDCG@K}$

> DCG@K (Discounted Cumulative Gain at K) calculates the cumulative gain for the top K items in the ranked list.
>
> IDCG@K (Ideal Discounted Cumulative Gain at K) calculates the ideal cumulative gain, assuming the most relevant items are at the top.

Imagine you're using a model to get some answers. The model generates the answer and ranks them based on some ranking metrics (like cosine, and TF-IDF). User provides the feedback about the generated answer and this is called the relevancy score, which varies between 0-3. You ask the below question and the model generates 10 responses.
Question: Who is the mother of Arjuna?
Answer:

1. Kunti is mother of Arjuna (Relevance: 3)
2. The mother of Arjuna is Kunti. (Relevance: 3)
3. Kunti and Gandhari both are mothers of Arjuna (Relevance: 1)

4. Arjuna's mother is Kunti. (Relevance: 3)

5. Madri is also Arjuna's mother (Relevance: 2)

6. Nakula and Arjuna are brothers therefore Nakula's mother is Kunti's mother. (Relevance: 1)

7. Pandu was married to Kunti and Arjuna was Pandu's son therefore he is Kunti's son also. (Relevance: 1)

8. Kunti is sister of Vasudeva. (Relevance: 2)

9. Draupadi is wife of Arjuna. Kunti is Draupadi's mother-in-law. Therefore Kunti is Arjuna's mother. (Relevance: 1)

10. Kunti blessed Arjuna and 4 brothers when they came back with Draupadi. (Relevance: 1)

Let's calculate NDCG@5 for our example:

> DCG@5: (3 + 3 + 1 + 3 + 2 = 12) (Sum of the relevance scores of the top 5 items)
>
> Ideal ranking: If we assume the most relevant items are at the top, then the ideal DCG@5 would be (3 + 3 + 3 + 2 + 2 = 13).

Now, calculate NDCG@5:

$$NDCG@5 = \frac{DCG@5}{IDCG@5} = \frac{12}{13} = .92$$

The NDCG@5 score is .92, indicating a very good ranking where all the relevant items are at the top.

$\boldsymbol{R}@5(\boldsymbol{Recallat}5)$:

R@5 measures how many of the relevant items are included in the top 5 positions of the ranked list. It's a binary metric that tells us how many of the most relevant items were retrieved in the top 5.
In our example, the top 5 ranked items are:

1. Kunti is mother of Arjuna (Relevance: 3)
2. The mother of Arjuna is Kunti. (Relevance: 3)
3. Kunti and Gandhari both are mothers of Arjuna (Relevance: 1)
4. Arjuna's mother is Kunti. (Relevance: 3)
5. Madri is also Arjuna's mother (Relevance: 2)

Among these, the relevant items are answer 1, answer 2 and answer 4, they all are relevant answers (Relevance: 3). So, in this case, R@5 is 3, indicating that 3 out of the top 5 items are relevant.
We need to keep in mind, NDCG assesses the quality of ranking by considering both relevance and position, while R@5 measures how many relevant items are present in the top 5 positions of a ranked list.

Semantic answer similarity (SAS)

This metric measures the semantic similarity between the system's answer and the gold answers. It is calculated using a variety of methods, such as word embeddings and contextual embeddings. Common metrics used for SAS include Precision at K (P@K), Recall at K (R@K), or F1 Score at K (F1@K), where K represents the number of reference answers considered.
Word Overlap Metrics: Metrics like BLEU, METEOR, ROUGE, and CIDER are often used in machine translation and text generation tasks but can also be adapted for

question answering. They measure the overlap of words or n-grams between the system's answer and the reference answer.

METEOR (Metric for Evaluation of Translation with Explicit ORdering)

This metric measures how well a model's output matches human-generated text, taking into account synonyms and paraphrases. It is commonly used in machine translation tasks, and it can be used for question-answering.
Components of METEOR:

- Exact Matching (Precision and Recall): METEOR starts by counting the number of exact word matches between the generated translation and the reference translations. It calculates precision (P) and recall (R) based on these exact matches.
- Word Overlap: METEOR then calculates a score based on the harmonic mean of precision and recall (F-mean), giving equal weight to both. This accounts for partial matches and the overall word overlap.
- Alignment Score: METEOR uses an alignment score to measure how well words or phrases in the generated translation align with words or phrases in the reference translations. It considers stemming and synonymy during this process.
- Penalty for Unaligned Words: METEOR also penalizes the inclusion of unaligned words in the generated translation.

APPENDIX B: Mahabharata Book Examples of QA Generated by QAGS

A typical QA Example

Just to give you an idea of the chunk size, the content inside a chunk, and the question, and answer created with reference from the chunk from the Mahabharata Book you can look at this example. From this, you will also get an idea of how heavily this kind of text is influenced by Sanskrit text even after translation into English. It is not easy for any non-Indian to read this text Historical text even in English. Keep in mind, when we transliterated these nouns back to any Indian script then it will have lots of spelling mistakes. In that situation if we use translation of English question answers into multiple Indian languages it will be a problem. We need to fix this noun spelling problem first.

This question and text is from Aranyaka Parva (The Book of Forest/ Vana Parva)

Question: How did Yudhishthira adore the sun, and what is the significance of this adoration?

Note: We need to keep in mind here sun is a common noun and is written in small letters after the translation. However, in the original work, it is a proper noun, Surya. Karna is a son of Surya.

Answer: Yudhishthira adored the sun by performing ablutions in the Ganges, standing in the stream with a concentrated mind, and singing a hymn of praise to the sun. This adoration is significant because it is believed to bestow various benefits, including the acquisition of prosperity, a wife, offspring, riches, memory of past lives, patience, and protection from grief and calamities.

Reference: Yudhishthira's adoration of the sun and the significance of this adoration are detailed in the text, where it describes Yudhishthira's actions and the hymn of praise he recites.

Chunk:

"Section III ""Vaisampayana said, 'Yudhishthira the son of Kunti, thus addressed by Saunaka, approached his priest and in the midst of his brothers said, 'The Brahmanas versed in the Vedas are following me who am departing for the forest. Afflicted with many calamities I am unable to support them. I cannot abandon them, nor have I the power to offer them sustenance: Tell me, O holy one, what should be done by me in such a pass.' ""Vaisampayana said, 'After reflecting for a moment seeking to find out the (proper) course by his yoga powers, Dhaumya, that foremost of all virtuous men, addressed Yudhishthira, in these words, 'In days of old, all living beings that had been created were sorely afflicted with hunger. And like a father (unto all of them), Savita (the sun) took compassion upon them. And going first into the northern declension, the sun drew up water by his rays, and coming back to the southern declension, stayed over the earth, with his heat centered in himself. And while the sun so stayed over the earth, the lord of the vegetable world (the moon), converting the effects of the solar heat (vapours) into clouds and pouring them down in the shape of water, caused plants to spring up. Thus it is the sun himself, who, drenched by the lunar influence, is transformed, upon the sprouting of seeds, into holy vegetable furnished with the six tastes. And it is these which constitute the food of all creatures upon the earth. Thus the food that supporteth the lives of creatures is instinct with solar energy, and the sun is, therefore, the father of all creatures. Do thou, hence, O Yudhishthira, take refuge even in him. All illustrious monarchs of pure descent and deeds are known to have delivered their people by practising high asceticism. The great Karttavirya, and Vainya and Nahusha, had all, by virtue of ascetic meditation preceded by vows, delivered their people from heavy afflictions. Therefore, O virtuous one, as thou art purified by the acts do thou likewise, entering upon a file of austerities. O Bharata, virtuously support the regenerate ones.' ""Janamejaya said, 'How did that bull among the Kurus, king Yudhishthira, for the sake of the Brahmanas adore the sun of wonderful appearance?"" ""Vaisampayana said, 'Listen attentively, O king, purifying thyself and withdrawing thy mind from every other thing. And, O king of kings, appoint thou a time. I will tell thee everything in detail, And, O illustrious one, listen to the one hundred and eight names (of the sun) as they were disclosed of old by Dhaumya to the high-souled son of Pritha. Dhaumya said, 'Surya, Aryaman, Bhaga, Twastri, Pusha, Arka, Savitri. Ravi, Gabhastimat, Aja, Kala, Mrityu, Dhatri, Prabhakara, Prithibi, Apa, Teja, Kha, Vayu, the sole stay, Soma, Vrihaspati, Sukra, Budha, Angaraka, Indra, Vivaswat, Diptanshu, Suchi, Sauri, Sanaichara, Brahma, Vishnu, Rudra, Skanda, Vaisravana, Yama, Vaidyutagni, Jatharagni, Aindhna, Tejasampati, Dharmadhwaja, Veda-karttri, Vedanga, Vedavahana, Krita, Treta, Dwapara, Kali, full of every impurity, Kala, Kastha, Muhurtta, Kshapa, Yama, and Kshana; Samvatsara-kara, Aswattha, Kalachakra, Bibhavasu, Purusha, Saswata, Yogin, Vyaktavyakta, Sanatana, Kaladhyaksha, Prajadhyaksha, Viswakarma, Tamounda, Varuna, Sagara, Ansu, Jimuta, Jivana, Arihan, Bhutasraya, Bhutapati, Srastri, Samvartaka, Vanhi, Sarvadi, Alolupa, Ananta, Kapila, Bhanu, Kamada, Sarvatomukha, Jaya, Visala, Varada, Manas, Suparna, Bhutadi, Sighraga, Prandharana, Dhanwantari, Dhumaketu, Adideva, Aditisuta, Dwadasatman, Aravindaksha, Pitri, Matri, Pitamaha, Swarga-dwara, Prajadwara, Mokshadwara, Tripistapa, Dehakarti, Prasantatman, Viswatman, Viswatomukha, Characharatman, Sukhsmatman, the merciful Maitreya. These are the hundred and eight names of Surya of immeasurable energy, as told by the self-create (Brahma). For the acquisition of prosperity, I bow down to thee, O Bhaskara, blazing like unto gold or fire, who is worshipped of the gods and the Pitris and the Yakshas, and who is adored by Asuras, Nisacharas, and Siddhas. He that with fixed attention reciteth this hymn at sunrise, obtaineth wife and offspring and riches and the memory of his former existence, and by reciting this hymn a person attaineth patience and memory. Let a man concentrating his mind, recite this hymn. By doing so, he shall be proof against grief and forest-fire and ocean and every object of desire shall be his.' ””Vaisampayana continued, ’Having heard from Dhaumya these words suitable to the occasion, Yudhishthira the just, with heart concentrated within itself and purifying it duly, became engaged in austere meditation, moved by the desire of supporting the Brahmanas. And worshipping the maker of day with offerings of flowers and other articles, the king performed his ablutions. And standing in the stream, he turned his face towards the god of day. And touching the water of the Ganges the virtuous Yudhishthira with senses under complete control and depending upon air alone for his sustenance, stood there with rapt soul engaged in pranayama. And having purified himself and restrained his speech, he began to sing the hymn of praise (to the sun).' ’Yudhishthira said, ””Thou art, O sun, the eye of the universe. Thou art the soul of all corporeal existences. Thou art the origin of all things. Thou art the embodiment of the acts of all religious men. Thou art the refuge of those versed in the Sankhya philosophy (the mysteries of the 1.”

Descriptive Question

A question without W5H format.

This question is from section Book03_017 (Parva 3, Section 17)
Question: Describe the confrontation between Pradyumna and Salwa in the battle. What were the initial actions of Salwa, and how did Pradyumna respond?
Answer: Pradyumna challenged Salwa and rushed towards him with a desire for an encounter. Salwa, maddened by anger, descended from his chariot to face Pradyumna. Salwa began by discharging a shower of arrows at Pradyumna, but Pradyumna parried this arrowy assault. In response, Pradyumna overwhelmed Salwa with a thick shower of arrows.

Same Question Different worded Answer

Example 1

Question: How did Garuda become the vehicle of Lord Vishnu, and what was his role in the churning of the ocean?
Answer1: Garuda became the vehicle of Lord Vishnu by performing severe austerities and pleasing him. His role in the churning of the ocean was to carry the pot of nectar on his back and protect it from the demons who tried to steal it.
Answer2: Garuda became the vehicle of Lord Vishnu by performing severe austerities and pleasing him. During the churning of the ocean, Garuda played a crucial role by carrying the pot of nectar on his back and protecting it from the demons.

Example 2

How many sons did Vinata have, and what were they promised by Kasyapa?
Vinata had two sons, and they were promised by Kasyapa that they would become the lords of all winged creatures.

Vinata had two sons, and Kasyapa promised that they would be respected in all the worlds and capable of assuming any form at will.

Different Ways of Asking the Same Questions

Who were the two sons of Vinata and Kasyapa, and what was their destiny?

Who were the two sons of Vinata and Kasyapa, and what were their names?

Who were the two sons of Kasyapa and Vinata?

APPENDIX C: Mahabharata Book Examples of Answers Generated by Zeroshot and Finetuned Models

We experimented with different models in zeroshot learning mode and finetuning mode. We got a very interesting answer from the Mahabharat book. Some randomly selected answers are presented below. It gives you an idea of the prediction power of these various models over Indian Historical Books.
While reading these answers make note of the following.

- ZS: Zero Shot Model
- Pred_Answer : Answer predicted by the model.
- t5: Text to Text Transfer Transformer Model.
- bert: Bidirectional Encoder Representations from Transformers
- bert512: BERT model trained with max_length 512
- bigbird: BirBird Transformer Model
- flant5: Finetuned (by google) Language T5 (FlanT5) Model
- flant530ep: Finetuned Language T5 (FlanT5) Model finetuned by us over 30 epoch

- bloom: Bloom 560 million Parameter Model
- bloom17: Bloom 1.7 billion Parameter Model.
- distilbert: DistilBERT Model
- t51ep: T5 Finetuned Model with 1 epoch
- t530ep: T5 Finetuned Model with 30 epoch
- xxx : The answer could not be predicted by the model

We find t530ep_Pred_Answer and flant30ep_Pred_Answer are predicting very close and plausible answers.

Ques_Id: 836

Question: What special water did Vibhishana offer to Rama and Lakshmana, and what was its significance?
Ref_Answer: Vibhishana offered a special water to Rama and Lakshmana, which was a gift from Kuvera, the king of the Guhyakas. This water, when applied to the eyes, made invisible creatures visible. It allowed them to see beings that could not be seen by the unassisted eye.
ZSt5_Pred_Answer: the king of the monkeys. the king of the monkeys. And he slain the king of the monkeys. And he a woman a man! ? a he dragged that car into the city. he slain his son. he said,
ZSbert_Pred_Answer: so that all creatures that are invisible may, o chastiser of foes, become visible to thee! this water laved over the eyes will make every invisible creature visible to thee, as also to any other person to whom thou mayst give it!’- - saying - - so be it, - - rama took that sacred water, and
ZSbigbird_Pred_Answer: xxx

ZSbloom_Pred_Answer: es, at the command of the king of the Guhyakas, a Guhyaka hath come from the White mountains, bringing with him his water! O great king, this water is a present to thee from Kuvera, so that all creatures that are invisible may, O chastiser of foes, become visible to thee! This water laved over the eyes will make every invisible creature visible to thee, as also to any other person to whom thou mayst give it!’–Saying–So be it,–Rama took that sacred water, and sanctified his own eyes therewith. And the high-minded Lakshmana also did the same. And Sugriva and Jambuvan, and Hanuman and Angada, and Mainda and Dwivida, and Nila and many other foremost of the monkeys, laved their eyes with that water. And thereupon it exactly happened as Vibhishana had said, for, O Yudhishthira, soon did the eyes of all these became capable of beholding things that could not be seen by the unassisted eye! ”Meanwhile, Indrajit, after the success he had won, went to his father. And having informed him of the feats he had achieved, he speedily returned to the field of battle and placed himself at the van of his army. The son of Sumitra then, under Vibhishana’s guidance, rushed towards that wrathful son of Ravana coming back, from desire of battle, to lead the attack. And Lakshmana, excited to fury and receiving a hint from Vibhishana, and desiring to slay Indrajit who had not completed his daily sacrifice, smote with his arrows that warrior burning to achieve success. And desirous of vanquishing each other, the encounter that took place between them was exceedingly wonderful like that (in days of yore) between the Lord of celestials and Prahrada. And Indrajit pierced the son of Sumitra with arrows penetrating into his very vitals. And the son of Sumitra also pierced Ravana’s son with arrows of fiery energy. And pierced with Lakshmana’s arrows, the son of Ravana became senseless with wrath. And he shot at Lakshmana eight shafts fierce as venomous

ZSbloom17_Pred_Answer: to the field of battle and placed himself at the van of his army. The son of Sumitra then, under Vibhishana’s guidance, rushed towards that wrathful son of Ravana coming back, from desire of battle, to lead the attack. And Lakshmana, excited to fury and receiving a hint from Vibhishana, and desiring to slay Indrajit who had not completed his daily sacrifice, smote with his arrows that warrior burning to achieve success. And desirous of vanquishing each other, the encounter that took place between them was exceedingly wonderful like that (in days of yore) between the Lord of celestials and Prahrada. And Indrajit pierced the son of Sumitra with arrows penetrating into his very vitals. And the son of Sumitra also pierced Ravana's son with arrows of fiery energy. And pierced with Lakshmana's arrows, the son of Ravana became senseless with wrath. And he shot at Lakshmana eight shafts fierce as venomous snakes. Listen now, O Yudhishthira, as I tell thee how the heroic son of Sumitra then took his adversary's life by means of three winged arrows possessed of the energy and effulgence of fire! With one of these, he severed from Indrajit's body that arm of his enemy which had grasped the bow. With the second he caused that other arm which had held the arrows, to drop down on the ground. With the third that was bright and possessed of the keenest edge, he cut off his head decked with a beautiful nose and bright with ear-rings. And shorn of arms and head, the trunk became fearful to behold. And having slain the foe thus, that foremost of

ZSdistilbert512_Pred_Answer: xxx

ZSroberta_Pred_Answer: sacred water, and sanctified his own eyes therewith. And the high-minded Lakshmana also did

t5_1ep_Pred_Answer: Vibhishana offered Rama and Lakshmana special water, and what was its significance? Indrajit pierced Ravana's son with arrows of fiery energy, and pierced Ravana's son with arrows of fiery energy. He also pierced Ravana's son with arrows of fiery energy.

t5_30ep_Pred_Answer: Vibhishana offered a special water to Rama and Lakshmana, and it was a present from Kuvera, so that invisible creatures can be seen to all. It was a present to Rama and Lakshmana, and it was a present from Kuvera.

flant5_30ep_Pred_Answer: Vibhishana offered a Guhyaka water to Rama and Lakshmana, a gift from Kuvera. This water laved over the eyes will make every invisible creature visible to the king, as well as any other person to whom he might give it.

Ques_Id: 107

Question: Why did Arjuna undertake his journey to meet Indra?

Ref_Answer: Arjuna embarked on his journey to meet Indra because he desired to obtain all the celestial weapons from him. He sought these divine weapons to enhance his prowess and fulfill his mission.

t5_Pred_Answer: ceased. And he departed. And Arjuna, a mighty ascetic, walked round his brothers and round his daughter of Pandu. And Arjuna, a strong-armed one, departed. And Arjuna, a strong-armed one, departed. And Arjuna, a strong-armed one, departed. And Arjuna, a strong-armed one, departed. And

ZSbert_Pred_Answer: desire of beholding indra, took. and that slayer of foes passed over many mountains inhabited by ascetics, and then reached the sacred himavat, the resort of the celestials. and the high - souled one reached the sacred mountain in one day,

ZSbigbird_Pred_Answer: Himavat, the resort of the celestials. And the high-souled one reached the sacred mountain in one day, for like the winds he was gifted with the speed of the mind, in consequence of his ascetic austerities. And having

ZSbloom_Pred_Answer : xxx

ZSbloom17_Pred_Answer : xxx

ZSdistilbert512_Pred_Answer: desire of beholding Indra, took. And that slayer of foes passed over many mountains inhabited by ascetics, and then reached the sacred Himavat, the resort of the celestials. And the high - souled one reached the sacred mountain in one day

ZSroberta_Pred_Answer: the desire of beholding Indra, took. And that slayer of foes passed over many mountains inhabited by ascetics, and then reached the sacred Himavat, the resort of the celestials. And the high-souled one reached the sacred mountain in one day

t5_1ep_Pred_Answer: Arjuna, a strong-armed son of Pandu, walked round his brothers and round Dhaumya, set out. He took up his handsome bow, set out. He walked round his brothers and round Dhaumya, and took up his handsome bow, set out.

t5_30ep_Pred_Answer: Arjuna embarked on his journey to meet Indra because he wanted to obtain all the celestial weapons from Indra. He wanted to obtain the sight of the highest of the gods and obtain the sight of the highest of the gods.

flant5_30ep_Pred_Answer: Arjuna embarked on his journey to meet Indra because he wanted to learn from Indra’s divine abilities and desires. He wanted to learn from Arjuna’s divine abilities and desires, and he wanted to obtain all the celestial weapons.

Ques_Id: 275

Question: What is the significance of the river Narmada, and why is it considered sacred?

Ref_Answer: The river Narmada is considered sacred because it flows westward through the country of the Anarttas. It is graced by Priyangu and mango trees and adorned with thick canes. All the tirthas, sacred spots, rivers, woods, and mountains in the three worlds are believed to come to bathe in the sacred waters of the Narmada. Additionally, the sacred asylum of the Muni Visravas and the birthplace of the lord of treasures, Kuvera, are associated with the river Narmada.

t5_Pred_Answer: said, ” O mighty king, and O mighty king, and foremost of the Kurus! Here ” ” the Narmada, and ? the sacred and auspicious Vaidurya peak, the, a sacred lake called Punya, and the sacred O O king of earth, O king of earth,

ZSbert_Pred_Answer: graced by priyangu and mango trees, and engarlanded with thickest of canes. all the tirthas and sacred spots, and rivers and woods and foremost of mountains that are in the three worlds, all the gods with the grandsire, along with the siddhas, the rishis and the charanas

ZSbigbird_Pred_Answer: xxx

ZSbloom_Pred_Answer: graced by Priyangu and mango trees, and engarlanded with thickest of canes. All the tirthas and sacred spots, and rivers and woods and foremost of mountains that are in the three worlds, all the gods with the Grandsire, along with the Siddhas, the Rishis and the Charanas, O best of the Kurus, always come, O Bharata, to bathe in the sacred waters of the Narmada. And it hath been heard by us that the sacred asylum of the Muni Visravas, had stood there, and that there was born the lord of treasures, Kuvera, having men for his vehicles. There also is that foremost of hills, the sacred and auspicious Vaidurya peak abounding with trees that are green and which are always graced with fruit and flowers. O lord of the earth, on the top of that mountain is a sacred tank decked with full-blown lotus and resorted to by the gods and the Gandharvas. Many are the wonders, O mighty monarch, that may be seen on that sacred mountain which is like unto heaven itself and which is visited by celestial Rishis

ZSbloom17_Pred_Answer: of the Anarttas, O Bharata, there, flows in a westward course the sacred river Narmada, graced by Priyangu and mango trees, and engarlanded with thickest of canes. All the tirthas and sacred spots, and rivers and woods and foremost of mountains that are in the three worlds, all the gods with the Grandsire, along with the Siddhas, the Rishis and the Ch

ZSdistilbert512_Pred_Answer: xxx

ZSroberta_Pred_Answer: capable of producing merit that lie on the west, in the country of the Anarttas, O Bharata, there, flows in a westward course the sacred river Narmada, graced by Priyangu and mango trees, and engarlanded

t5_1ep_Pred_Answer: The sacred river Narmada, and why is it considered sacred? The sacred river Narmada is a sacred area, inhabited by birds and deer, and inhabited by the regenerate ones.

t5_30ep_Pred_Answer: The river Narmada flows westward course through the country of the Anarttas. It is graced by Priyangu and mango trees, and engarlanded with thick canes. It is considered sacred because it is composed of tirthas, sacred spots, rivers, woods, and mountains. It is considered sacred because it is a subjugator of hostile cities, and it is considered sacred.

flant5_30ep_Pred_Answer: The river Narmada is considered sacred because it flows westward in the country of the Anarttas. It is graced by Priyangu and mango trees and engarlanded with thick canes. All the tirthas, sacred spots, rivers, woods, and mountains are in the three worlds, all the gods with the Grandsire, Siddhas, Rishis, and Charanas.

Ques_Id: 634

Question: What boon did Lord Hari (Vishnu) seek from Madhu and Kaitabha?

Ref_Answer: Lord Hari sought the boon that Madhu and Kaitabha would submit to be slain by him for the greater good of the world.

t5_Pred_Answer: said, ’O Brahmana, I desire to know all this!’ ” he said, ’O Brahmana, I am gratified with you!’ ” ”And as the Lord Hari lay asleep on that lotus, he trembled and trembled, and as the Grandsire Brahma, possessed of great prowess, sprang from his navel and

ZSbert_Pred_Answer : xxx

ZSbigbird_Pred_Answer : xxx

ZSbloom_Pred_Answer: unfading Vishnu, He who is called by Munis crowned with ascetic success as the Supreme Lord of the Universe, that Being of great holiness, then lay in Yoga sleep on the wide hood of the Snake Sesha of immeasurable energy, and the Creator of the Universe, that highly-blessed and holy Hari, knowing no deterioration, lay on the hood of that Snake encircling the whole Earth and as the Deity lay asleep on that bed, a lotus, endued with great beauty and effulgence equal unto that of the Sun, sprang from his navel. And from that lotus possessed of effulgence like unto the Sun's, sprang the Grandsire Brahma, that lord of the worlds who is the four Vedas, who hath four forms and four faces, who is invinc

ZSbloom17_Pred_Answer : xxx

ZSdistilbert512_Pred_Answer : xxx

ZSroberta_Pred_Answer : xxx

t51ep_Pred_Answer: Lord Hari (Vishnu) sought from Madhu and Kaitabha, who were seated on that lotus, and sprang the Grandsire Brahma, who was invincible in consequence of his own energy. He was surrounded by a lotus-leaves and a crown and the Kaustubha gem and attired in purple silk.

t5_30ep_Pred_Answer: Lord Hari sought a boon from Madhu and Kaitabha, a sage of great energy, possessed of great beauty and effulgence. He wanted to know all this, aided by his brave sons, and aided by his brave sons who had given up their weapons.

flant5_30ep_Pred_Answer: Lord Hari (Vishnu) sought a boon from Madhu and Kaitabha, who had never spoken an untruth or truth before. He sought to slay them at a spot that was absolutely uncovered.

Ques_Id: 501

Question: Why did Hanuman obstruct Bhima's path and refuse him passage in the direction of that path?

Ref_Answer: Hanuman obstructed Bhima's path and refused him passage because the path Bhima was attempting to take was one of the paths to heaven meant for the celestials, not mortals. Hanuman wanted to ensure that no mortal would pass through this path, both to preserve its sanctity and to prevent Bhima from facing any adverse consequences by attempting it.

t5_Pred_Answer: he slain the king of the Rakshasas. he returned to his own city, Ayodhya, inaccessible to mortals. he said, 'O Lord, O Lord, this path is impassable to mortals.' Section CXVII Hanuman asked, 'O Lord, this path is impassable to mortals.' Section CXVII Han

ZSbert_Pred_Answer : xxx

ZSbigbird_Pred_Answer : xxx

ZSbloom_Pred_Answer: Hearing everything from me the lotus-eyed Rama at once ascertained his course of action, and having for the passage of his army constructed a bridge across the deep, crossed it followed by myriads of monkeys. Then by prowess Rama slew those Rakshasas in battle, and also Ravana, the oppressor of the worlds together with his Rakshasa followers. And having slain the king of the Rakshasas, with his brother, and sons and kindred, he installed in the kingdom in Lanka the Rakshasa chief, Vibhishana, pious, and reverent, and kind to devoted dependants. Then Rama recovered his wife even like the lost Vaidic revelation. Then Raghu's son, Rama, with his devoted wife, returned to his own city, Ayodhya, inaccessible to enemies; and that lord of men began to dwell there. Then that foremost of kings, Rama was established in the kingdom. Thereafter, I asked a boon of the lotus-eyed Rama, saying, 'O slayer of foes, Rama, may I live as long as the history of thy deeds remaineth extant on earth!" Thereupon he said, 'So be it. O represser of foes, O Bhima, through the grace of Sita also, here all excellent objects of entertainment are supplied to me, whoever abide at this place. Rama reigned for the thousand and ten hundred years. Then he ascended to his own abode. Ever since, here Apsaras and Gandharvas delight me, singing for aye the deeds of that hero, O sinless one. O son of the Kurus, this path is impassable to mortals. For this, O Bharata, as also with the view that none might defeat or curse thee, have I obstructed thy passage to this path trod by the immortals. This is one of the paths to heaven, for the celestials; mortals cannot pass this way. But the lake in search of which thou hast come, lieth even in that direction." 1.

ZSbloom17_Pred_Answer : xxx

ZSdistilbert512_Pred_Answer : xxx

ZSroberta_Pred_Answer : xxx

t5_1ep_Pred_Answer: Hanuman obstructed Bhima's path and refused him passage in the direction of that path. He asked a boon of the lotus-eyed Rama, saying, 'O slayer of foes, Rama, may I live as long as the history of thy deeds remaineth extant on earth'.

t5_30ep_Pred_Answer: Hanuman obstructed Bhima's path and refused him passage in the direction of that path because he believed that none might defeat or curse Bhima or the mortals would pass the path. Hanuman believed that this path would be impossible for mortals to pass through the path of the immortals.

flant5_30ep_Pred_Answer: Hanuman obstructed Bhima's path and refused him passage because he believed that none might defeat or curse him. He believed that mortals could pass this path because they could not pass through it. Hanuman believed that Bhima's path was one of the paths to heaven, and he believed that mortals could not pass through it.

APPENDIX D: Mahabharata Book Examples of Answers Generated by AGS & RAAGS

Example 1

Ques: What is the significance of performing the Agnihotra, and what consequences are mentioned for neglecting it?

Reference Ans: Performing the Agnihotra is considered important for eternal morality. Neglecting it, as mentioned in the text, leads to the consumption of sin. Those who do not perform the Agnihotra are said to suffer the consequences, including not waiting upon bulls and neglecting their kinsmen, guests, friends, sons, wives, and servants.

RAAGS Ans: The Agnihotra is performed by a celestial being, and it is important because it is performed by a celestial being. The consequences of neglecting it are mentioned in the article. The Agnihotra is performed by Rishis, Siddhas, and Vishnu.

AGS Ans: The Agnihotra is performed without sacrificing the gods, pitris, and guests. It is considered a significant sacrifice and it is associated with sin. It is believed that none should cook his food alone and slay an animal without dedicating it to the gods, pitris, and guests. It is believed that none should eat it alone, and it is considered an act of emancipation.

Example 2

Ques: What predicament does Yudhishthira face, and how does he seek guidance to resolve it?
Reference Ans: Yudhishthira faces the predicament of being unable to support the Brahmanas who are following him as he departs for the forest. He seeks guidance to resolve this dilemma by approaching his priest, Dhaumya, and inquiring about the appropriate course of action.
RAAGS Ans: Yudhishthira faces a predicament, and seeks guidance to resolve it by bathing in tirthas, avoiding temptations of Daityas and Danavas, vows, forgiveness, prosperity, morality. He seeks guidance to resolve this predicament, seek guidance to guide him through ascetic penances, sacrifices, gifts, and ascetic penances.
AGS Ans: Yudhishthira faces a predicament, including the need for guidance from his priest and the Brahmanas. He seeks guidance to resolve this predicament by reciting the hymn at sunrise, obtaining wife, offspring, riches, and memory of his former existence.

Example 3

Ques: What advice does Dhaumya offer to Yudhishthira regarding his predicament?
Reference Ans: Dhaumya advises Yudhishthira to take refuge in the sun, who is considered the father of all creatures. He explains that the sun, by his influence,

transforms solar energy into food, which sustains all living beings. Dhaumya suggests that Yudhishthira should purify himself, engage in ascetic meditation, and support the Brahmanas virtuously.

RAAGS Ans: Dhaumya advises Yudhishthira to visit the tirthas and give away kine to avoid the curse caused by Agastya's presence. He advises Yudhishthira to go to the Kurus to make a peace that all the sons of Bharata may live together with cheerful hearts and contentedly.

AGS Ans: Dhaumya advises Yudhishthira to purify himself and withdraw his mind from every other thing. He advises Yudhishthira to listen to the hundred and eight names of the sun as they were disclosed of old by Dhaumya to the high-souled son of Pritha.

Example 4

Ques: How did Yudhishthira adore the sun, and what is the significance of this adoration?

Reference Ans: Yudhishthira adored the sun by performing ablutions in the Ganges, standing in the stream with a concentrated mind, and singing a hymn of praise to the sun. This adoration is significant because it is believed to bestow various benefits, including the acquisition of prosperity, a wife, offspring, riches, memory of past lives, patience, and protection from grief and calamities.

RAAGS Ans: Yudhishthira adored the sun because he believed that Vindhya would not honour the sun by circumambulations, and that it would lead to the great happiness of men. This adoration is significant because it is believed to be the result of the great dissolution of the universe.

AGS Ans: Yudhishthira adored the sun by purifying himself and withdrawing his mind from every other thing. This adoration is significant because it is believed to be the result of the creation of the sun.

Example 5

Ques: What are the hundred and eight names of the sun as revealed by Dhaumya to Yudhishthira?
Reference Ans: The hundred and eight names of the sun, as revealed by Dhaumya to Yudhishthira, include names like Surya, Aryaman, Bhaga, Ravi, Vivaswat, Indra, Soma, Brahma, Vishnu, Rudra, and many others. These names signify various attributes and aspects of the sun's divine nature.
RAAGS Ans: Dhaumya revealed the hundred and eight names of the sun to Yudhishthira. These names are derived from the abode of Indra and King Vaisravana. They are a symbol of the sun's presence in the firmament and appear in the nether regions, causing great terror in the hearts of gods, Asuras, Gandharvas, Yakshas, Snakes, Raks
AGS Ans: The hundred and eight names of the sun are immeasurable energy, as revealed by Dhaumya to Yudhishthira. These names are derived from the self-create (Brahma) and are a symbol of prosperity.

Example 6

Ques: What benefits are associated with reciting the hymn of praise to the sun, as mentioned in the text?
Reference Ans: Reciting the hymn of praise to the sun is believed to bestow several benefits, including the acquisition of prosperity, a wife, offspring, riches, memory of past lives, patience, and protection from grief and calamities. It is a practice that is said to bring about positive outcomes for the devotee.

RAAGS Ans: Reciting the hymn of praise to the sun is associated with a variety of benefits. It is believed that reciting the hymn of praise to the sun is beneficial for those who are discerning and discerning. It is associated with benefits such as obtaining wife, offspring, riches, memory of his former existence, patience, and memory.

AGS Ans: Reciting the hymn of praise to the sun is associated with benefits such as obtaining wife, offspring, riches, memory of his former existence, patience, and memory. It is believed that a person can attain patience and memory by concentrating his mind on the hymn.

Example 7

Ques: What were the specific actions and preparations undertaken by Yudhishthira before his adoration of the sun?

Reference Ans: Before adoring the sun, Yudhishthira purified himself, withdrew his mind from distractions, and performed ablutions in the Ganges. He stood in the stream with a concentrated mind and practiced pranayama (breath control). These preparations were made to ensure a focused and spiritually charged adoration.

RAAGS Ans: Before his adoration of the sun, Yudhishthira engaged in various actions and preparations. He entertained orders in proper measure with meat, beverages to eat, drink, fragrant garland, and various kinds of attire. He entertained the princes and Brahmanas by thousands, bestowing upon them wealth of various kinds, bade them farewell.

AGS Ans: Before adoring the sun, Yudhishthira purified himself and withdrawn his mind from all other things. He emphasized that he was virtuously assisting the Brahmanas in obtaining prosperity and reciting the hymn at sunrise.

Example 8

Ques: What was the primary concern expressed by Dhritarashtra to Vidura regarding the situation after the Pandavas had gone to the forest?
Reference Ans: Dhritarashtra expressed deep sorrow over the situation and was concerned about how to secure the goodwill of the citizens and maintain their support. He sought Vidura's advice on what should be done for the welfare of both his sons and the Pandavas.
RAAGS Ans: Dhritarashtra expressed a primary concern to Vidura about the situation after the Pandavas had gone to the forest. He expressed concern about the situation and expressed his desire to see Vidura.
AGS Ans: Dhritarashtra expressed his concern about the situation after the Pandavas had gone to the forest. He emphasized that the situation was not friendly to Vidura and that Vidura was not friendly to Vidura.