Information Retrieval and Question Answering

Kusal Bista
a1881044

The University of Adelaide

4333_COMP_SCI_7417 Applied Natural Language Processing
Lecturer: Dr. Alfred Krzywicki

Table of Contents

1. Abstract

Information retrieval system (IRS) plays a vital role in extracting information efficiently from a large volume of data. For this project we have utilized an Information retrieval system for matching the user query based on the news article. The project aims to retrieve matches of the text timely and accurately for the query asked by the user using TF-IDF, BERT, roBERTa, and spaCy. The major component of the system includes data preprocessing, coreference resolution for entity identification, text matching for relevant sentence retrieval and testing utilities for validation. Several processing techniques like text vectorization, Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec for word embeddings, cosine similarity for vector comparison are used to quantify the similarity of text. The system's effectiveness is evaluated using MRR which calculates the average reciprocal ranks of the relevant text retrieved for the query. In our investigation, BERT performed well as compared to other techniques whereas TF-IDF performed worst.

2. Introduction

The process of retrieving relevant information from a large collection of data is called Information retrieval. The primary goal of information retrieval is to provide users with the most relevant and useful information in the response to their queries. (Mishra & Vishwakarma, 2015)The ability to precisely match the exact answers to the queries can greatly enhance user satisfaction in many practical scenarios. Instead of having to sift through lists of retrieved documents, Information retrieval can provide direct answers to the queries reducing the mental effort required for traditional search tasks. This functionality is important for improving existing IR systems, such as search engines and emergent use cases. (Otten, 2023)QA systems have diverse applications across various domains as well as it can facilitate quick and accurate response to the inquiries reducing the human intervention. It can be used in various sectors such as customer services, healthcare, education, finance, e-commerce, voice assistants, chatbots, virtual assistants etc. (Otten, 2023)

QA systems can be implemented with different methodologies as per requirement. Five major types of QA systems are Information Retrieval-based QA, Knowledge-based QA, Generative QA, Hybrid QA, and Rule-based QA. Each approach offers distinct advantages and challenges, but for our project we have implemented Information Retrieval-based QA. This method automatically answers the question by searching the relevant passages containing the answer (Otten, 2023).It calculates the answer by finding the most similar answer text between the candidate answer texts. We can achieve this by using different language processing models like TI-IDF and spaCy. They are tools used for generating tokenization, pos tagging, name entity recognition etc. (Wu, 2023)

The current advancements in natural language processing (NLP) have revolutionized QA systems. Machine learning techniques of using pre-trained language models like BERT and RoBERTa. They have demonstrated remarkable performance in understanding context, and semantics and have enhanced the capabilities of QA systems. These models are trained on a large corpus of text, enabling them to capture precise patterns in the natural language and answer any query accurately. BERT is an impressive bidirectional language model that has a good capacity for transfer learning, helping to adapt to different tasks through pretraining and fine-tuning. (Jiajia , et al., 2024)

Over the past four decades, research in Question Answering Systems (QAS) has progressed where hundreds of questions answering systems have been developed. These systems could receive "Question Phase" in Natural language and provide a set of relevant responses as a "Answer Type".

Firstly, the term "Question Phrase" refers to the segment of the question containing the search keywords. Similarly, "Question Type" denotes the classification of the question based on its intended purpose. Meanwhile, "Answer Type" signified the categorization of the potential responses sought by the question. "Question Focus" relates to the specific aspect or subject matter targeted by the question. On the other hand, "Candidate passage" encompasses anything from a document that is retrieved by the search system. Lastly, a "candidate answer" represents the suitable response to the question. The demand for such systems has exponentially increased due to the escalating need for precise information retrieval in response to the exponential growth of data and information (Sarah, Nazar, & Mutsam, 2021). In 2014, the landscape of neural information retrieval (IR) began to take shape with the notable contribution made by Gupta et al. (2014) who introduced an auto-encoder approach focusing on the mixed-script query expansion addressing transliterated search queries. Concurrently, Zhang et al. (2014) tackled the task of local text reuse by employing three annotators to label passages as either representing reuse or not. Le and Mikolov (2014) introduced the paragraph vector (PV) method for composing word embeddings, aiming to induce semantic representations over longer textual units. The momentum continued into 2015, with a significant expansion in neural IR research where Word2vec gained wider adoption within the IR community. (Sarah, Nazar, & Mutsam, 2021).

Throughout this project, we have used several QA approaches, including TF-IDF, spaCy, RoBERTa, and BERT, to facilitate the text matching. This project tests all the technologies and compares them with each other. For testing, we use MRR as an evaluation metric.

2.1. Information Retrieval based Question Answering

Information retrieval-based Question Answering (IR-QA) is a type of question-answering system that depends on retrieving relevant passages from a large corpus of text in response to a user query. What the IR system does is, it identifies the relevant information from the existing source and extracts the response. IR-QA works using following steps:

a) **Query Processing:** The query is provided by the user which is parsed to identify the keywords. The pre-processing of the query is done to understand the meaning and intent of the user.
b) **Document Retrieval:** The system searches through a pre-indexed corpus of documents to find the relevant document that contains the relevant information. We have used techniques like keyword matching and vector space model to retrieve the document.
c) **Document Ranking:** Once a set of candidate documents has been retrieved, the IR-QA system ranks them based on their similarity to the query.
d) **Answer Extraction:** Finally, the system extracts relevant information from the top-ranked document to generate an answer to the user's query.

The level effectiveness of IR-QA systems varies depending on components such as the size and quality of the corpus, the complexity of searching and ranking algorithms, and the accuracy of the answer extraction methods.

3. Preprocessing

a) **Dataset:** For dataset we are provided with 1000 news_dataset having following columns:

- Id: It refers to the article id.
- Author: It refers to the writer of the document.
- Date: It refers to the date when the article was written.
- Year: It refers to the year when it was written.
- Month: It refers to the month when it was written.
- Article: The passage of the news.

We just use the article and the article id for this project.

b) **Removing duplicate:** The duplicate data are removed and only first data is kept, rest are dropped.

c) **Checking missing data:** The missing data is checked and replaced.

d) **Sampling data:** 100 out of 1000 data are sampled for testing.

e) **Stopword removal:** Stopwords refers to the common words in a natural language (e.g. 'is', 'the', 'and'). We remove these words as they don't carry significant meaning.

f) **Lemmatization:** Lemmatization is the process of reducing words to their base or root form. It helps in normalizing different forms of a word to its base form. For example: "running" becomes "run". We use NLTK's 'WordNetLemmatizer' to lemmatize the word.

g) **Text cleaning:** During this cleaning, we use regular expressions to match the desired pattern of the text. Firstly, non-ASCII characters are removed from the text, then, multiple spaces are replaced with a single space removed from the text. Question mark-related problems are addressed using regular expression substitutions. Finally, leading and trailing white spaces are stripped from the text.

4. System Architecture

4.1. Classical approach

Starting off with the most common question answering system we use keyword-based methodology called TF-IDF which works by combining the statistical elements to discover relationships between different elements in the test by recognizing the patterns. The articles are preprocessed into a representation of a vector using the TF-IDF method. It is a classical approach which works by weighing the frequently used information by calculating the relative frequency of words in a specific document comparing the inverse proportion of the word in the entire document. The architecture of TF-IDF can be divided into two:

a. TF- It means now often a term appears in an article.

$$\text{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

b. IDF- It means inverse document frequency that measures how important a term is.

$$\text{IDF}(t,D) = \log \frac{\text{Total number of documents in } D}{\text{Number of documents containing term } t}$$

c. TF-IDF Score: This score is obtained by calculating the weight of term t in the document d and is given by:

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D)$$

(Patil, 2020)

Secondly, we implement a different model named spaCy to compare two objects such as documents, spans, or tokens to find out their similarity. The 'similarity' function returns a floating-point number between -0 and 1 to indicate their similarity level. The word vectorization is a crucial function to determine the similarity. We train Word2Vec using trained word vectors from the raw text. For our project we use 'md' medium spaCy model. The similarity returned by the spaCy is the cosine similarity of two vectors. This vector can be adjusted if required. If any object contains multiple tokens, then it computes the average of their token vector.

For objects containing multiple tokens, such as `Doc` and `Span`, the default approach is to compute the average of their token vectors. Consequently, shorter phrases with fewer irrelevant words tend to provide more valuable similarity scores.

Cosine Similarity:
It is a technique of calculating similarity between the likeness of the extracted feature vector from the article. Doing so, we can select the most suitable solution for the feature vector. This solution would be a close answer to the query provided by the user. (Patil, 2020) Mathematically, it calculates the similarity of two non-zero vectors of an inner product space which measures the cosine of the angle between them. The cosine of 0 degrees is equal to 1, while for any angle within the range of 0 to π radians, the cosine is always less than 1.

$$\text{Cosine Similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

Here, a.b is a dot product and ||a|| ||b|| is a Euclidean norm of vectors a and b respectively. Both TF-IDF and spaCy use this technique to find the similarity between the query and the closest possible sentence in the article.

4.2. Pretrained model

It is a machine learning model that has been already tested on a large dataset and can be used for another dataset to perform a specific task. In our project we have utilized three pretrained models to perform question answering tasks. These models are very powerful tools and are trained on large scale question-answering pairs.

a) **Base BERT trained with squad2 data:**
   Base BERT refers to (Bidirectional Encoder Representations from Transformers) and has fewer trainable parameters as compared to larger variants. We utilize 'bert-base-cased-squad2' meaning it was trained on SQuAD2.0 dataset. It is well suited for general question answering tasks and is very efficient for deployment in resource-constrained environments.
b) **Roberta model trained in Squad 2 dataset:**
   We have implemented a base version of the RoBERTa(Robustly optimized BERT) model trained on SQuAD 2.0 question and answering dataset. It incorporates few modifications over the BERT model and can perform better. The major enhancement includes larger batch size processing, longer training sequence processing. It is robust and has improved performance and even deals with medical or legal question answering. (Hugging Face, n.d.)
c) **Large BERT fine-tuned squad model:**
   Instead of base this model is finetuned on larger model with higher parameter in a SQuaD1.0 dataset. Here, fine-tuning helps us achieve question answering more precisely compared to other models. It is able to answer the question accurately by understanding its requirements. Another reason for its accuracy is that it masks whole words during the

training, enhancing its ability to comprehend word-level semantics within the input text. It uses 24 layers, 1024 hidden dimensions, 16 attention heads and 336 million parameters making it a remarkable pretrained model for question answering. (Hugging Face, 2018)

These pretrained models are trained with large question-answer pairs, including unanswerable questions. (McCormick, 2020)

4.3 System Architecture Components

The section discusses the system architecture of fine-tuned large BERT model in SQuaD dataset as it had comparatively good performance overall. BERT architecture is composed of transformer encoder blocks. It uses a multi-headed self-attention mechanism to understand the contextual relations between words (or sub-words) in a sentence. As our fine-tuned BERT model uses MLM i.e. Masked Language Model, it can randomly mask input tokens and try predicting the masked word depending on the surrounding. By doing so, the model can learn bidirectionally and can capture a deeper relationship of words in a sentence. Not only this, but it also makes a model robust to noise and encourages the prediction during the noise. The selected model has another important feature named 'Next Sentence Prediction' where the model is fine tuned to make a prediction of the next sentence in a sequence enabling the model to handle relationships between multiple sentences.

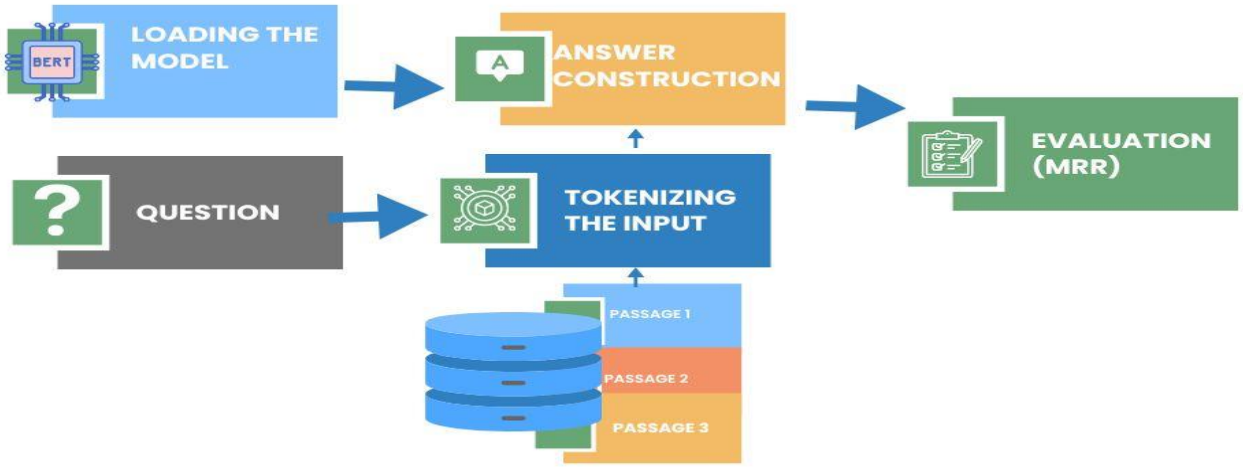

Figure 1: System architecture

a) **Loading the model:**
We load the BERT pretrained model, and select a fine-tuned variant named "bert-large-uncased-hole-word-masking-fine-tuned-squad" from Hugging Face transformers library.

b) **Tokenizing the inputs:**
Secondly, the BERT model takes in the input question and a passage as a single packed sequence. Following is the input breakdown before feeding to the model:

- **Token embedding:** A [CLS] token is added to the input word tokens at the question and a [SEP] token is concatenated at the end of both the question and the passage. This allows the segmentation of the passage and the question indicating separation. (McCormick, 2020)
- **Segment Embeddings:** It is utilized in BERT to differentiate sentences added to each token. After preprocessing the input, the BERT model incorporates start and end vectors. The probability of each word being the start-word is computed using a dot product between the final embedding of the word and the start vector. Then, we

perform softmax in all the words. The word with the highest probability value is the start-word. Similarly, this process is repeated to calculate the end-word also. Coreference resolution also takes place during this stage. It analyzes the text to identify its pronouns and referring expressions linking them to their corresponding entities. (McCormick, 2020)

c) Construction of answers:
This is the major component where answers are generated. It continuously iterates through the start score until it gets the highest match for the score. Finally, the answer with the highest score is presented to the user. In this QuestionAnswering class, we can extract the top k answer constructed by the model.

d) Evaluation of model:
To evaluate the model, we generate the answer prediction for ten questions from the news articles. Performance is measured using MRR which works by assessing the average rank of the first correct answer among the predicted answers. Greater the MRR better the performance of the model.

5. Model Selection and Training

5.1. Model evaluation

Firstly, the experiment started with the classical retrieval namely "TF-IDF" and "spaCy" calculating the similarities between the matrices. This classical retrieval had very little accuracy in terms of accuracy. In this project, we determine the accuracy by using TextUtility which calculates the MMR of each model. We can find two test question sets; one is from the same article whereas another is the questions from all randomly selected articles. While running the test on the question from the same article, TF-IDF and spaCy could not match any golden question answer, making them the worst model. However, spaCy had comparatively more accuracy than TF-IDF.

After the failure of the first two models, pre-trained BERT and RoBERTa models on SQuaD and SQuaD2 dataset were tested.

Table 1. *Performance metrics for models*

Model	**Question sets**	**MRR**
bert-base-cased-squad2	10 same articles	0.317
	10 different articles	0.21
deepset/roberta-base-squad2	10 same articles	0.60
	10 different articles	0.80
bert-large-uncased-whole-word-masking-finetuned-squad	10 same articles	0.60
	10 different articles	0.80

Two set of questions are prepared 10 questions from the same article and 10 questions from the different articles. The questions are very simple and most has one word answer.

5.2. Discussion

bert-base-cased-squad2: This model had the accuracy of 30%. Out of ten only 3 question was answered by this mode. Being a base model, it had the MRR score of 0.317 for questions from same article, the MRR dropped when it was exposed to 10 different questions from different articles. This is because it is a base model and has few parameters. This model has 71.1% exact match and F1 of 74.671 in SQuaD 2 dataset. It has the lowest result compared to other BERT model.

deepset/roberta-base-squad2: For the questions from same article, it could only match 6 out of 10 whereas for the question list from different article it could match 8 questions out of 10. For our test dataset of different article, Roberta base model tunned in squad2 outperformed other models. As it is fine-tuned using SQuAD 2.0 dataset it has achieved an "exact" match score of 79.87% and an "F1" score of 82.91%, indicating its robust performance in accurately answering questions with both answerable and unanswerable contexts.

bert-large-uncased-whole-word-masking-finetuned-squad: Out of 10 questions from the same article, it managed to answer 8 questions from the same article, whereas it could not perform well for questions from different articles. It is a finetuned model and trained in SQuAD dataset. It is large model with 24-layers, with 1024 hidden dimension, 16 attention head and 336M parameters. Hence, it performed very well.

6. User Interaction with the System

The interaction with the system is a straightforward iterative process run directly from the jupyter notebook shell.

a) **Input from user:** The user provides the article ID and a question. If the article id is valid then the system asks the user about the question.
b) **Answer retrieval:** If the question is irrelevant and cannot be answered then the system alerts users about answers not found. If both article and question is correct then the system will provide the user the answer.
c) **Exiting from system:** After providing the answer, the user interaction is looped back to the beginning allowing the user to input another article ID and question. This process of question and answering continues until the user enters 'quit' or 'exit' when the article ID is asked.

Following is the example of the interaction with the system:

Enter the article ID (type 'exit' to quit): 121513
Invalid article number.

Enter the article ID (type 'exit' to quit): 17574
Ask your question: *Who is the vice chairman of Samsung?*
=
Answer: jay y . lee

Enter the article ID (type 'exit' to quit): 17776
Ask your question: Who is the Roman Catholic Archbishop of New York?
Answer: cardinal dolan

Enter the article ID (type 'exit' to quit): 17776
Ask your question: How many religious leaders are scheduled to participate in Donald J. Trump's inauguration ceremony?
Answer: six

Enter the article ID (type 'exit' to quit): 17574
Ask your question: What amount did Samsung contribute to Ms. Choi's winter sports program? (Mishra & Vishwakarma, 2015)**Answer:** $ 1 . 3 million

Enter the article ID (type 'exit' to quit): What amount did Samsung contribute to Ms. Choi's winter sports program?
Article ID should be an integer. Please try again.
Enter the article ID (type 'exit' to quit): exit
Exiting the program...

7. Conclusion

This report discusses on the Information retrieval -based Question Answering (IR-QA) system using several modelling techniques like TF-IDF, spaCy, BERT, and RoBERTa. The QA model used query and article ID from user as an input and the system matched the relevant passages from the news articles.

During evaluation of models, we found that the classical methods like TF-IDF and spaCY was limited and was not effective even after coreference resolution. Later, pretrained transformer models like BERT and RoBERTa was implemented to improve the performance of our QA system. Specifically, the RoBERTa model fine-tuned on SQuAD 2 demonstrated good performance in handling answers from the different article, whereas BERT fine tunned on SQuAD data also performed well on the questions from same article. After selecting the model, we implement the system for the interaction of user.

This project could be improved more by additional pre-processing techniques along with different tokenization strategies can be used on fine-tuned parameters to enhance the performance. (Prakash, 2021) Training on larger and variety of question answer pair dataset may enhance the performance of the model. For the further refinement, the user interface can be enhanced in different application platforms along with the feedback mechanism for the user if it does any mistake.

8. References

Sarah, A., Nazar, M., & Mutsam, J. (2021, January 12). *Question Answering Systems: A Systematic Literature Review.* Retrieved from researchgate: https://www.researchgate.net/publication/350550855_Question_Answering_Systems_A_Systematic_Literature_Review

Hugging Face. (2018). *BERT large model (uncased) whole word masking finetuned on SQuAD*. Retrieved from huggingface: https://huggingface.co/google-bert/bert-large-uncased-whole-word-masking-finetuned-squad

Hugging Face. (n.d.). *roberta-base for QA.* Retrieved from huggingface: https://huggingface.co/deepset/roberta-base-squad2

Jiajia , W., Jimmy Xiangji, H., Xinhui , T., Wang, J., Huang, A., Laskar, M., & Bhuiyan, A. (2024, July). *Utilizing BERT for Information Retrieval: Survey, Applications, Resources, and Challenges.* Retrieved from dl.acm: https://dl.acm.org/doi/10.1145/3648471

McCormick, C. (2020, March 10). *Question Answering with a Fine-Tuned BERT.* Retrieved from mccormickml: https://mccormickml.com/2020/03/10/question-answering-with-a-fine-tuned-BERT/

Mishra, A., & Vishwakarma, S. (2015). *Analysis of TF-IDF Model and its Variant for Document Retrieval.* Retrieved from ieeexplore: https://ieeexplore.ieee.org/abstract/document/7546200

Otten, N. V. (2023, 01 20). *What is a question-answering System?* Retrieved from spotintelligence: https://spotintelligence.com/2023/01/20/question-answering-qa-system-nlp/

Patil, R. (2020, August 4). *Question Answering System using A TF-IDF* . Retrieved from ijset: https://www.ijset.in/wp-content/uploads/IJSET_V8_issue4_257.pdf

Prakash, P. (2021, September 10). *An Explanatory Guide to BERT Tokenizer.* Retrieved from analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/09/an-explanatory-guide-to-bert-tokenizer/

Wu, J. (2023, July 21). *Information Retrieval 1: TF-IDF based search engine with python code.* Retrieved from AI Mind: https://pub.aimind.so/information-retrieval-1-tf-idf-based-search-engine-with-python-code-54cd085786