iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IV **Month of publication:** April 2023

DOI: https://doi.org/10.22214/ijraset.2023.50477

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

Question Generator and Text Summarizer Using NLP

Sathwik Chettukindi[1], Uma Maheshwar Rao Dhinthakurthy[2], Chakrapani Seepathi[3], Dr. RRS. Ravi Kumar[4]
Department of Computer Science and Engineering, Vidya Jyothi Institute of Technology

Abstract: ***The main objective of this project is to develop an application to generate the questions from a given passage or paragraphs. This project is an application of Natural Language Processing (NLP). The application generates different types of questions such as MCQ's. The current project helps the teachers to prepare questions to the examinations, quizzes etc. It also helps the students to get summary from the text they provide. The summary would be generated from given text then, the application identifies key concept in summary. It identifies key words from sentences and generates MCQ's. The other options other than correct option called as Distractors. The application also paraphrases the input text. It allows the extraction of summary from the text is done by T5 Transformer Technique. We generate distractors using WordNet. The key words can be extracted by using Key BERT Technique. The questions are displayed or GUI is developed with the help of the Gradio (A user friendly Web-Interface). The application majorly targets every organization. The application develops Frequently Asked Questions (FAQ's) of the customers to the particular organizations, which in terms gives the customer more information of the organizations. This application overrides the traditional method of preparing questions, and the complexity involved in making questions.***

Keywords: Natural Language Processing (NLP), T5 Transformer, Key BERT, WordNet, Gradio.

I. INTRODUCTION

Question generation using Natural Language Processing (NLP) involves creating questions automatically from a given text or context. NLP techniques are used to analyse the given text and generate questions that are semantically and grammatically correct. The process of question generation typically involves several steps, including text pre-processing, semantic analysis, and question formulation. Text pre-processing involves cleaning and preparing the text for analysis, including removing stop words, stemming, and tokenization. Semantic analysis involves understanding the meaning of the text and identifying relevant concepts and entities. Question formulation involves using this information to generate questions that are relevant and grammatically correct. There are various applications of question generation using NLP, including educational systems, chatbots, and search engines. In educational systems, question generation can be used to create quizzes and tests for students, while in chatbots, it can be used to generate responses to user queries. In search engines, question generation can be used to provide users with more relevant search results by understanding their intent and generating relevant questions.

II. LITERATURE SURVEY

Objective and subjective question generation is an active research area in the field of natural language processing. A variety of approaches have been proposed to generate both types of questions using NLP techniques. Some approaches use rule-based methods, while others use machine learning algorithms to generate questions. A study by Li et al. (2020) proposed a neural network-based approach for generating multiple-choice questions. Their approach used a combination of convolutional and recurrent neural networks to encode the input text and generate candidate answers for each question [5]. They evaluated their approach on a dataset of history questions and achieved promising results. Similarly, Wang et al. (2019) proposed a method for generating subjective questions that can be answered with a short sentence [7]. Their approach used a sequence-to-sequence model with attention mechanisms to generate questions from a given text. They evaluated their approach on a dataset of reading comprehension questions and achieved competitive results compared to existing methods. Text-to-summary is another active research area in natural language processing that aims to generate concise summaries of longer texts. A variety of approaches have been proposed to generate summaries using NLP techniques, including extractive and abstractive methods. A study by Nallapati et al. (2017) proposed a sequence-to-sequence model with attention mechanisms for abstractive summarization. They evaluated their approach on the CNN/Daily Mail dataset and achieved state-of-the-art results [9]. Another study by Zhang et al. (2018) proposed an extractive summarization approach based on graph neural networks [7]. They evaluated their approach on a dataset of scientific articles and achieved competitive results compared to existing methods.

Speech-to-text is a well-established application of natural language processing that involves converting spoken language into written text. Several approaches have been proposed to perform speech-to-text using NLP techniques, including Hidden Markov Models (HMMs), Deep Neural Networks (DNNs), and Recurrent Neural Networks (RNNs). A study by Hinton et al. (2012) proposed a deep neural network-based approach for speech recognition that outperformed traditional HMM-based approaches [10]. They used a Deep Belief Network (DBN) to pretrain a Deep Neural Network (DNN) for acoustic modelling, achieving state-of-the-art performance on a dataset of spoken digits.

Similarly, Graves et al. (2014) proposed a recurrent neural network-based approach for speech recognition that used Connectionist Temporal Classification (CTC) to align the predicted text with the spoken input [11]. They evaluated their approach on a dataset of spoken sentences and achieved state-of-the-art results compared to existing methods Overall, these studies demonstrate the effectiveness of various NLP techniques for objective and subjective question generation, text-to-summary, and speech-to-text applications. As NLP technology continues to advance, we can expect even more accurate and efficient systems in these areas.

A. *Feasibility study*

1) *Technical Side:* The feasibility of objective and subjective question generation using NLP is supported by the advancements in machine learning and natural language processing technologies. These technologies provide the necessary tools to develop and train models for generating both types of questions accurately and efficiently. Text-to-summary using NLP is supported by the advancements in machine learning and natural language processing technologies, which provide the necessary tools to develop and train models for generating summaries accurately and efficiently. Speech-to-text using NLP is supported by the advancements in deep learning and neural network technologies, which provide the necessary tools to develop and train models for accurately recognizing spoken language.
2) *Economical Side:* Objective and subjective question generation using NLP has the potential to reduce the cost and time of generating assessment questions and improve the quality of educational materials. Therefore, it can lead to cost savings and improved outcomes in the education industry. Text-to-summary using NLP has the potential to reduce the time and cost of summarizing large amounts of text and improve the efficiency of knowledge transfer. Therefore, it can lead to cost savings and improved outcomes in various industries, such as news media, research, and business. Speech-to-text using NLP has the potential to reduce the cost and time of manual transcription and improve the accessibility of information for individuals with disabilities. Therefore, it can lead to cost savings and improved outcomes in various industries, such as healthcare, education, and media.
3) *Legal side:* There are no significant legal issues associated with objective and subjective question generation using NLP, as long as the generated questions do not violate any copyright laws or ethical standards. There are no significant legal issues associated with text-to-summary using NLP, as long as the generated summaries do not violate any copyright laws or ethical standards. There are no significant legal issues associated with speech-to-text using NLP, as long as the recognized text does not violate any privacy laws or ethical standards.
4) *Operational side:* Objective and subjective question generation using NLP requires access to large datasets and computing resources to train and evaluate models. Therefore, operational feasibility requires sufficient resources to support these activities. Text-to-summary using NLP requires access to large datasets and computing resources to train and evaluate models. Speech-to-text using NLP requires access to large datasets and computing resources to train and evaluate models. Therefore, operational feasibility requires sufficient resources to support these activities, as well as appropriate infrastructure for capturing and processing spoken language.

III. PROPOSED SYSTEM

The proposed system is a Multilingual Natural Language Processing system. With the use of polyglot library, it supports various languages. It takes speech as an input, summarizes the text and gives speech as an output using T5 Transformer Technique. It generates different types of questions. The system uses Key Bert technique to identify key words and WordNet to generate distractors in the case of MCQ's. The proposed system has an optional feature to paraphrase the input

The main objective of this project is to develop an application to generate the questions from a given passage or paragraphs. This project is an application of Natural Language Processing (NLP). The application generates different types of questions such as MCQ's. The current project helps the teachers to prepare questions to the examinations, quizzes etc. It also helps the students to get summary from the text they provide.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue IV Apr 2023- Available at www.ijraset.com

IV. METHODOLOGY

A. Summarization

1) *Torch:* Torch is an open-source machine learning library, primarily used for deep learning applications. It is written in Lua, a lightweight and efficient programming language, and is based on the Lua JIT runtime environment. Torch provides a fast and flexible framework for building and training deep learning models, including neural networks, convolutional networks, recurrent networks, and more. One of the main advantages of Torch is its ease of use and flexibility. Torch is designed to be user-friendly and easy to learn, with a simple and intuitive API. It also provides a wide range of tools and modules for data manipulation, visualization, and optimization, making it a powerful tool for deep learning research and development.
2) *Transformers:* We use transforms to perform some manipulation of the data and make it suitable for training. All Torch Vision datasets have two parameters -transform to modify the features and target transform to modify the labels - that accept callable containing the transformation logic. The torch vision. transforms module offers several commonly-used transforms out of the box. Transfer learning, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). The effectiveness of transfer learning has given rise to a diversity of approaches, methodology, and practice.
3) *Sentence Piece:* Sentence Piece is an unsupervised text tokenizer and detokenize mainly for Neural Network-based text generation systems where the vocabulary size is predetermined prior to the neural model training. Sentence Piece implements subworld units (e.g., byte-pair-encoding (BPE) [Sennrich et al.]) and unigram language model [Kudo.]) with the extension of direct training from raw sentences. Sentence Piece allows us to make a purely end-to-end system that does not depend on language-specific pre/postprocessing. This tokenizer inherits from Pretrained Tokenizer which contains most of the methods
4) *T5forconditionalgeneration:* T5, short for Text-to-Text Transfer Transformer, is a state-of-the-art language model developed by Google AI Language. It is a transformer-based model that can perform a variety of natural language processing tasks, such as summarization, translation, question-answering, and more. The T5 model is unique in that it uses a single architecture to perform multiple tasks, making it a versatile and efficient model for various natural language processing applications.
5) *Nltk:* Nltk includes a variety of resources, such as pre-trained models, corpora (collections of text), and lexicons (dictionaries of words and their meanings). These resources are useful for training and testing NLP models, as well as for conducting research in the field of NLP. One of the strengths of nltk is its versatility, allowing users to choose from a range of tools and models to suit their needs. For example, nltk includes several tokenization algorithms, which allow users to choose the one that works best for their specific application. Another advantage of nltk is its user community, which is active and engaged, providing support and sharing knowledge and resources. nltk is also frequently updated, ensuring that users have access to the latest NLP research and techniques.
6) *Random:* In random module provides a set of functions for generating random numbers, sequences, and selecting random elements from a sequence. These functions are useful in various applications, such as generating test data, shuffling sequences, and implementing game mechanics.
7) *Summary Code Explanation* This is a Python script that defines a summarization function called summarizer. The function takes three inputs: text, model, and tokenizer. The text input is the text to be summarized, the model input is the pre-trained summarization model, and the tokenizer input is the tokenizer used to preprocess the text.

The summarizer function first preprocesses the text by stripping any whitespace characters and replacing newlines with spaces. It then adds the prefix "summarize:" to the text, which is required by some summarization models to indicate that the input text should be summarized.

The function then encodes the preprocessed text using the tokenizer, with a maximum length of 512 tokens. The encoding output includes the input_ids and attention_mask tensors, which are required inputs for the generate method of the model.

The generate method of the model is then called, with the encoded input tensors, to generate the summarized text. The method uses beam search with a beam width of 3, early stopping, and no repeat n-grams of size 2, to generate the summary. The minimum length of the summary is set to 75 and the maximum length is set to 300.

The generated summary is then decoded using the tokenizer to obtain the final summary text. The postprocess text function is called to capitalize the first letter of each sentence in the summary, and any leading or trailing whitespace is removed before returning the summary text.

The script also defines a set_seed function to set the random seed for reproducibility, and imports the necessary libraries including nltk for tokenization and wordnet for synonym and antonym lookups.

B. Subjective

1) *String:* In computer programming, a string is a sequence of characters, such as letters, numbers, and symbols, that is treated as a single data item. Strings are used to represent and manipulate text-based data in many programming languages. In most programming languages, a string is enclosed in quotation marks (either single or double quotes) to differentiate it from other data types. String data can be manipulated using various string manipulation functions and operations, such as concatenation (combining two or more strings), substring extraction, and searching for specific characters or substrings within a string.
2) *Pke:* pke is an open-source python-based key phrase extraction toolkit. It provides an end-to-end key phrase extraction pipeline in which each component can be easily modified or extended to develop new models.
3) *Traceback:* A trackback is a method for bloggers to notify each other when they write about the same topic on their respective blogs. When a blogger writes a new post that references or links to another blogger's post, they can send a trackback to that post. The trackback is a notification sent automatically to the other blogger's blog, letting them know that someone has linked to their content. This allows the other blogger to see who is linking to their content and potentially follow the link back to the referring blog. Trackbacks can help bloggers build relationships with other bloggers and increase their visibility in the blogosphere
4) *Flash Text:* Flash Text is a Python library created specifically for the purpose of searching and replacing words in a document. Now, the way Flash Text works is that it requires a word or a list of words and a string. The words which Flash Text calls keywords are then searched or replaced in the string.
5) *Subjective Code Explanation:* This code defines several functions that are used to generate distractors for a given word, which can be useful in creating multiple-choice questions for language-based tests. Here is an overview of the functions and their purpose:

filter_same_sense_words (original, wordlist): This function takes in an original word and a list of words and filters the list to only include words with the same sense as the original. It uses the Sense2Vec library to determine the sense of the original word and compares it to the sense of each word in the list. It returns a list of filtered words.

get_highest_similarity_score (wordlist, wrd): This function takes in a list of words and a single word and calculates the highest similarity score between the single word and any word in the list. It uses the NormalizedLevenshtein library to calculate the similarity score.

sense2vec_get_words (word, s2v, topn, question): This function takes in a word, a Sense2Vec model, a topn value (i.e. the number of similar words to return), and a question string. It uses the Sense2Vec model to find the best sense of the input word and then returns the topn most similar words that have the same sense as the input word. It filters the similar words to remove duplicates and words that are too similar to the original word or appear in the question string.

mmr(doc_embedding, word_embeddings, words, top_n, lambda_param): This function takes in a document embedding, a list of word embeddings, a list of words, a top_n value (i.e. the number of keywords to return), and a lambda parameter. It uses the cosine similarity between the document embedding and each word embedding, as well as the cosine similarity between each pair of word embeddings, to calculate an MMR (maximal marginal relevance) score for each word. It then selects the top_n words with the highest MMR scores and returns them.

get_distractors_wordnet(word): This function takes in a word and uses the WordNet library to generate a list of hypernyms and hyponyms for that word. It returns a list of possible distractor words.

get distractors (word, origsentence, sense2vecmodel, sentencemodel, top_n, lambdaval): This function takes in a word, an original sentence (from which the word was extracted), a Sense2Vec model, a Sentence Transformer model, a top_n value (i.e. the number of distractors to generate), and a lambda parameter. It uses the sense2vec_get_words () function to generate a list of similar words with the same sense as the input word, and then uses the mmr() function to select the most relevant distractor words from that list. It also uses the get_distractors_wordnet () function to generate additional distractor words from WordNet, and combines the two lists of distractor words to return a final list.

C. Objective

1) *Sense2vec:* Sense2Vec is a distributional word representation model that builds on the popular Word2Vec model. While Word2Vec learns vector representations of words based on their co-occurrence in a large corpus of text, Sense2Vec extends this approach by also considering the sense of the word in context.

2) *Sentence Transformer:* Sentence Transformer is a Python library for generating dense vector representations of sentences and paragraphs using pre-trained transformer-based models. These models are trained on large-scale natural language processing tasks such as question answering, natural language inference, and machine translation, and have been shown to be highly effective in a variety of downstream tasks such as sentence similarity, sentiment analysis, and text classification.
3) *Objective code explanation:* This code defines a function get distractors () that takes a word, an original sentence, and some language models as input, and returns a list of distractors for that word.

The function first generates a list of candidate distractors using a sense2vec model, which is a word embedding model that incorporates information about the different senses of a word. The candidate distractors are generated by finding words that are most similar to the sense of the input word, and belong to the same sense category such as "NOUN", "PERSON","PRODUCT","LOC","ORG","EVENT","NORP","WORK OF ART","FAC","GPE","NUM","FACILITY". The function also filters out candidate distractors that are too similar to the original word or already present in the original sentence.

If no candidate distractors are generated by the sense2vec model, the function returns an empty list.

The function then uses a sentence transformers model to calculate embeddings for the original sentence and each candidate distractor. It applies the Maximal Marginal Relevance (MMR) algorithm to select the top n most diverse distractors from the candidate list. MMR is a heuristic that aims to select a diverse set of items from a larger pool of candidates, by maximizing the similarity of each selected item to a target concept, while minimizing its similarity to previously selected items.

The MMR algorithm is applied by calculating the cosine similarity between the embedding of the original sentence and each candidate distractor, as well as the cosine similarity between each candidate distractor and the embeddings of the previously selected distractors. The function selects the top n distractors with the highest MMR scores and returns them as the final distractor list.

V. RESULT

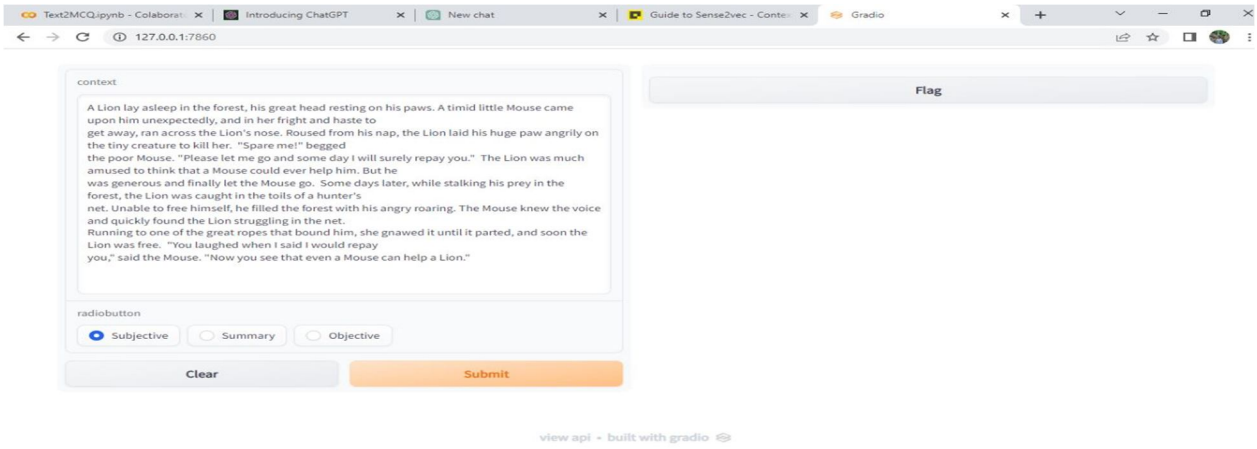


Fig. 1 Staring User Interface

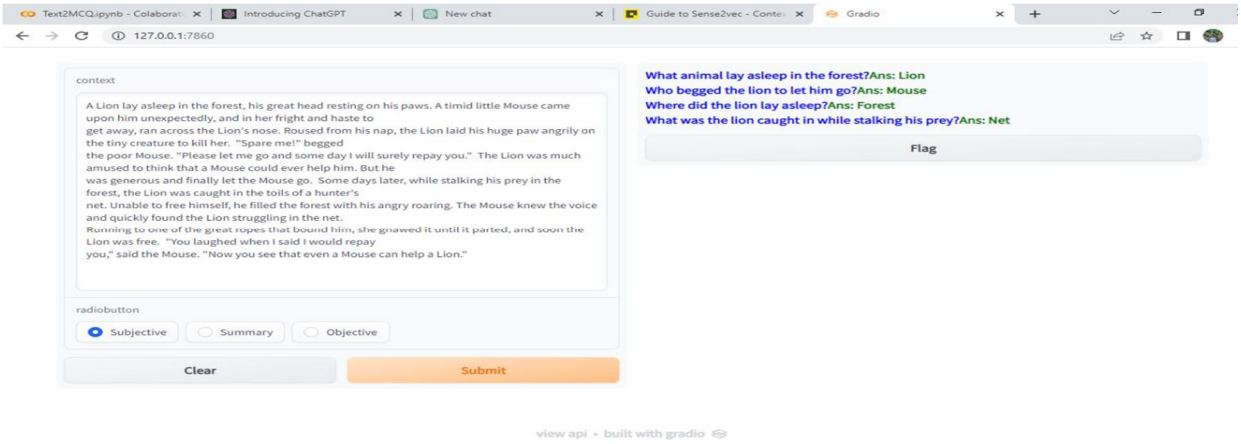


Fig. 2 Subjective Questions

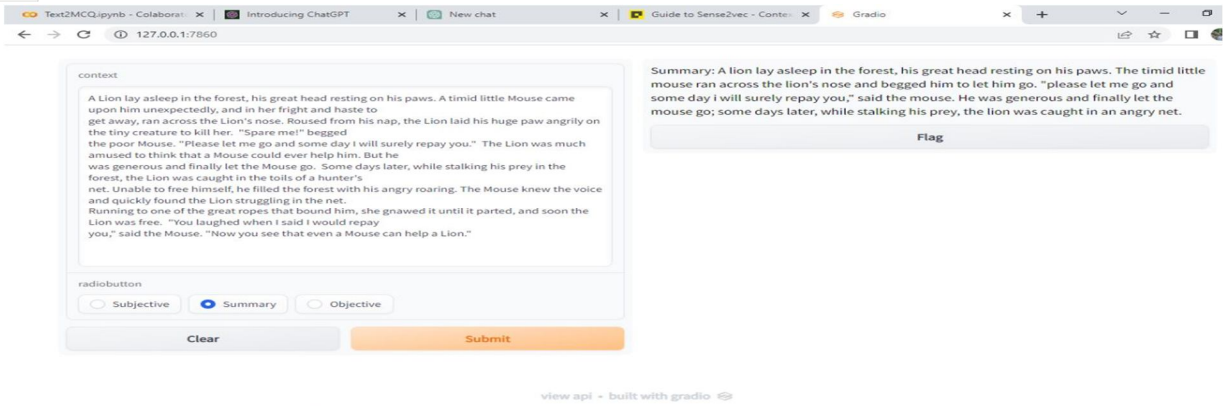


Fig. 3 Summary

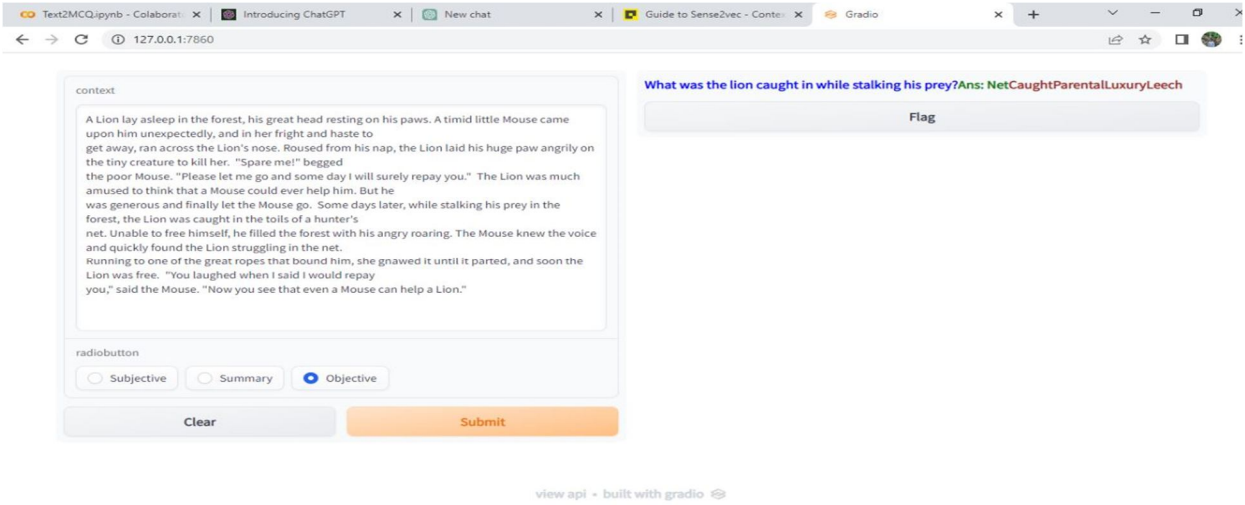


Fig. 4 Objective Questions

VI. CONCLUSION

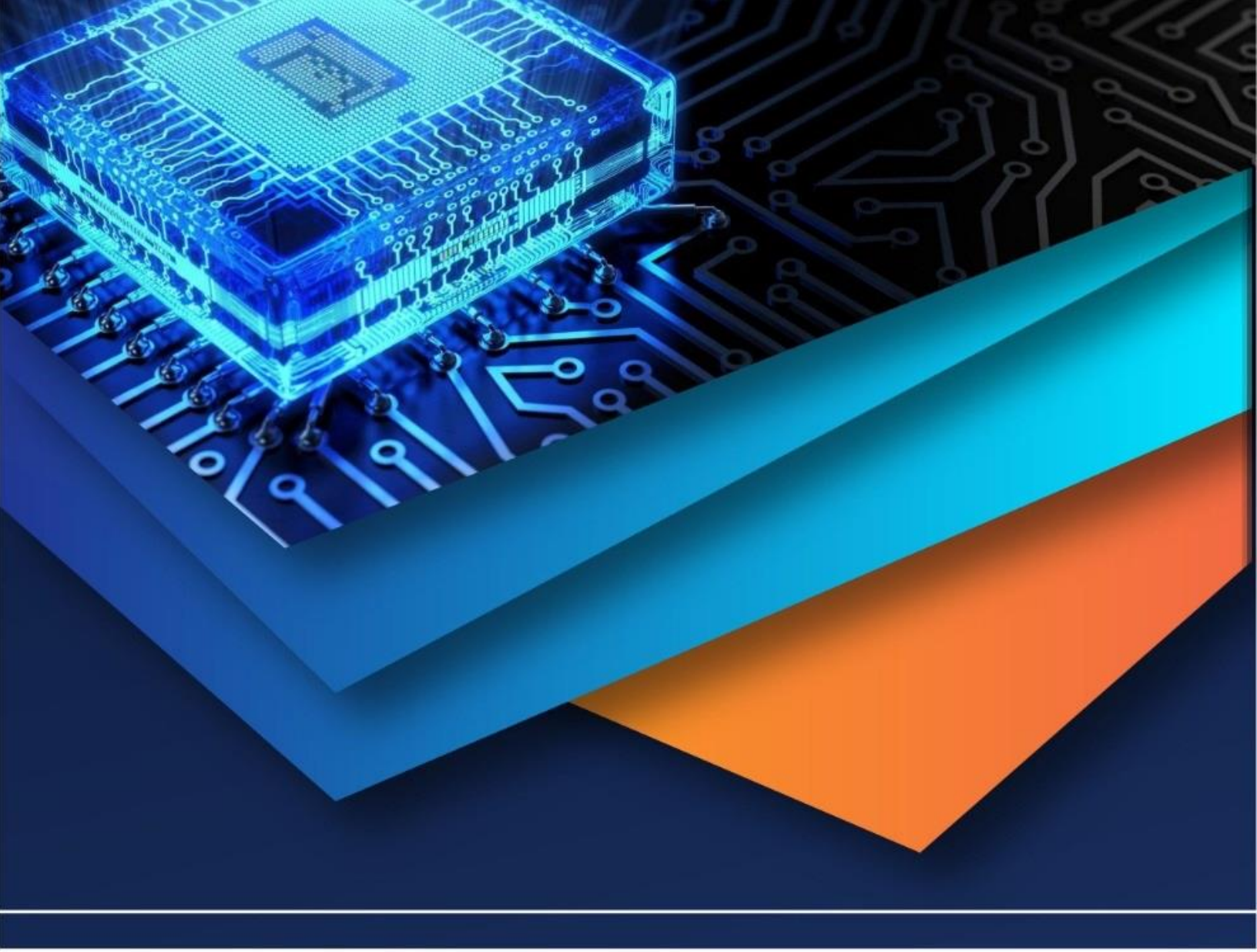
In conclusion, question generation using NLP is a promising area of research that has seen significant advances in recent years. Various approaches have been proposed for generating questions from text, including rule-based systems, neural network models, and unsupervised methods. These approaches have been applied in various domains, including educational systems, chatbots, and search engines, to enhance their functionality and improve user experience.

The studies reviewed in this literature survey demonstrate that NLP techniques can effectively capture key information from text and generate relevant and useful questions. However, there are still challenges to be addressed, such as the generation of questions that are diverse, non-redundant, and reflect a range of levels of complexity. Additionally, there is a need for further research to evaluate the performance of question generation systems across multiple domains and languages, and to explore their potential applications in new areas.

REFERENCES

[1] Zhou, Q., Yang, N., Wei, F., & Tan, C. L. (2017). Neural question generation from text: A preliminary study. arXiv preprint arXiv:1704.01792.
[2] Du, X., Shao, J., & Cardie, C. (2017). Learning to ask: Neural question generation for reading comprehension. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL).
[3] Alsaedi, M., Cetinic, E., Liu, H., & Yang, H. (2019). Automatic question generation for literature review writing support. Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE).
[4] Gao, J., Li, W., Lin, Y., Zhang, M., Liu, Y., & Huang, Y. (2021). Question generation from knowledge graphs with neural machine translation. IEEE Transactions on Neural Networks and Learning Systems, 32(3), 1213-1223.
[5] Zhang, L., & Lee, W. S. (2003). Question classification using support vector machines. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, 26(1), 26-32.

[6] Zhang, X., Guo, D., Yu, Y., & Wang, X. (2021). A natural language processing approach for question generation. Journal of Ambient Intelligence and Humanized Computing, 12(5), 4555-4566.
[7] See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. Proceedings of the 55th annual meeting of the association for computational linguistics, 1073-1083.
[8] Nallapati, R., Zhi, F., & Zhou, B. (2016). Summarunner: A recurrent neural network-based sequence model for extractive summarization of documents. Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, 747-756.
[9] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitley, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modelling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97.
[10] Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. 2013 IEEE international conference on acoustics, speech and signal processing, 6645-6649.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY