Engineer to Excel

SIMATS ENGINEERING

SIMATS

Saveetha Institute of Medical And Technical Sciences

(Declared as Deemed to be University under Section 3 of UGC Act 1956)

QUESTION TAGGING SYSTEM USING NATURAL LANGUAGE PROCESSING

AIM:

The aim of this project is to develop a robust and efficient question tagging system using natural language processing techniques. This system will accurately categorize user-inputted questions into predefined tags or categories, enhancing information retrieval, organization, and user experience in knowledge-based applications.

ABSTRACT:

- ❖ **Objective:** Develop an advanced question tagging system using NLP techniques.
- ❖ **Data Preparation:**Collect and clean a substantial dataset of questions from various domains.
- ❖ **Feature Extraction:**Extract key features from the questions, such as part-of-speech tags, named entities, and syntactic dependencies.
- ❖ **Model Training:**Train machine learning models like SVM, Random Forests, or deep learning models such as RNNs or BERT using the extracted features.
- ❖ **Performance Evaluation:**Assess the models' performance using standard metrics like accuracy, precision, recall, and F1-score.

  Integration and Deployment:Integrate the trained model into a user-friendly interface for inputting questions.

  Deploy the system in a production environment for real-world use.

- ❖ **Continuous Improvement:**Monitor the system's performance and gather user feedback for iterative enhancements.Utilize feedback and monitoring data to improve the system's accuracy and effectiveness over time.

OBJECTIVES:

- ❖ **Automate Question Categorization:** Develop algorithms for automatic categorization of questions.
- ❖ **Improve Information Retrieval:** Enhance the process of finding relevant answers to questions.
- ❖ **Enhance User Experience:** Create a user-friendly interface for inputting questions and receiving tags.
- ❖ **Ensure Scalability and Efficiency:** Design a system capable of handling large volumes of questions quickly.

- **Optimize Model Performance:**Train machine learning models to achieve high accuracy in tagging questions.
- **Integrate with Existing Systems:**Incorporate the tagging system into existing platforms or applications.
- **Continuous Improvement:**Implement processes for ongoing monitoring and refinement of the system.
- **Cross-Domain Adaptability:**Develop flexibility for use in various domains and contexts.

REQUIREMENTS DESIGN:

- **Data Collection and Preprocessing:**
  - Gather diverse question datasets and clean them for analysis.
- **Feature Extraction:**
  - Extract relevant linguistic features like part-of-speech tags and named entities.
- **Machine Learning Models:**
  - Implement models like SVM or deep learning for question classification.
- **Model Evaluation:**
  - Assess model performance using accuracy and other metrics.
- **User Interface:**
  - Design an intuitive interface for users to input questions and view tags.
- **Scalability:**
  - Ensure the system can handle large question volumes efficiently.
- **Integration:**
  - Integrate the system seamlessly with existing platforms.
- **Monitoring:**
  - Implement monitoring tools for performance tracking.
- **Documentation:**
  - Provide comprehensive guides and documentation for users.
- **Security:**
  - Implement security measures to protect user data and privacy.

INTRODUCTION:

An Automatic Question Generator using Natural language processing (NLP) generates relevant, syntactically, and semantically accurate questions based on various input formats such as text, a structured database, or knowledge bases. NLP is a subfield of Artificial Intelligence (AI) that aims to facilitate human-computer interaction using natural language that recognizes and understands human spoken and written language. It is the ultimate objective of NLP to read, analyze, understand, and make sense of written or textual information in natural languages in a meaningful manner. With the help of NLP, this project aims to set questions for the computer-based examination, educators, and students who are preparing for competitive exams

[1]. The purpose is to improve the method of setting MCQs and modifying them, along with creating a viable question bank that academicians and learners can use. As a result, you will ensure that the MCQ includes appropriate questions and options. These questions will align with the learning objectives and significance of the topics discussed in the tutorial material[2]. Automatic question generators can be applied to a range of domains including Massiveopen online courses, setting objective questions, search engines, automated help systems, chatbots (e.g. for customer interaction), and health care for analyzing mental health. However, it requires time and effort to manually create meaningful and relevant questions[3]. Setting questions is a challenging undertaking for both teachers and students who are preparing for competitive exams. The current approach entails manually setting the questions, which takes a lot of time and human effort. Consequently, there is an increasing need for a system that can easily, quickly, and with minimal human effort develop questions. Manually creating test papers and quizzes takes a lot of time from teachers, professors, and tutors. Similarly, students spend considerable time self-analyzing their knowledge. The remaining part of the paper comprises four sections. Section 2 gives necessary background knowledge of works carried out by various researchers on NLP. Section 3 describes the methodology adopted. Section 4 represents the implementation details, and the conclusion of the work is given in section 5.

LITERATURE REVIEW:

To realize the state-of-the-art work in NLP and its application in education sector analysis using machine learning models, a literature survey is conducted, and the following are thegist of the articles related to the proposed work. A rule-based methodology for automating the generation of questions is proposed by Onur et al. The paper proposes a method that considers both the syntactic and semantic structure of a sentence. The purpose of this paper is mainly to generate more comprehensive questions based on word semantic roles. This paper proposes to generate questions using an existing sentence by following a rule-based model. Specifically, reliance-based, named entity recognition (NER)-based, and semantic role (SRL) marking-based layouts/rules are utilized. To decide between who and what questions, the system has proposed using Chunking. Aleena et al's paper describes an implementation for a question generation system. The main idea is to ensure the system understands natural language so it can process and manipulate the data. The proposed system focuses on pre-processing of data, key phase extraction, and natural language processing. This method can be used to develop a fast, secure, and randomized system that is advantageous in many aspects, including education.As input, this paper proposed an automatic question paper generation system that accepts text, documents, or PDF files. The proposed system removes stop words and uses Natural Language Processing. A TF-IDF algorithm is used for key phrase extraction, and the existence of terms is checked on the wiki. Using the WordNet tool, create triplets for question paper generation as well as conduct input clarity checks. Priti et al., their paper discusses the generation of questions based on Bloom's taxonomy.The Pre-processing of data is mainly concerned with feature extraction and Stanford pos tagging. An analysis of syntactic structures and semantic structures is part of pre- processing. The POS tagging and Chunking are included in the syntactic analysis. It is carried out in the semantic analysis process to recognize named entities. Afterward, the text is pre-processed, appropriate ‘WH’ question words are mapped to it, and questions are generated. D. R. CH and S. K. Saha has proposed a survey on automatic multiple-choice question generation from the text. Questions are generated by reading articles from the database. For The text summarization and frequency count of words, an NLP-based summarizer is used, while pattern matching is used for selecting the keywords. They have used wordnets, pattern matching, domain ontologies, and semantic analysis to generate distractors. The creation of a general workflow for automatic MCQ generation is the main topic of this study .Ankita, K. A. et al. describe the intricacy of POS tagging as the number of computational levels required for determining POS tags. The focus of this paper is on how to reduce the complexity of POS labeling by using Hidden Markov Models. The authors have likewise recommended Named Entity Recognition (NER). Even though there are numerous POS taggers out there, individuals are still searching for a way that requires less effort to accomplish and is less likely to create complications, as well. HMM, bases taggers find tags in sentences, rather than in individual words.To summarize, the researchers have worked on NLP applications on question paper generation and multichoice questions. There is a scope to develop a model keeping in consideration the object-based learning and bloom's taxonomy and draw the questions. A web-based application is develop to generate the multi choice questions for the text/ paragraph given as input. Assess the score of the test and give the results and display the right answers to participants.

PROPOSED METHODOLOGY:

Many teachers, professors, and tutors (academicians) spend a significant amount of time preparing test papers and quizzes manually. Similarly, students spend a substantial amount of time on self-analysis (self-calibration). In addition, students rely on their mentors to help them with their self-analysis. This has led to working on the NLP area, which currently has large scope for improvement. Security is also a big concern for them, as well as the lack of teaching staff in any institute makes creating the papers difficult. In conclusion, the system proposes an Automatic Question Generator that stores the data, provides fast operations, and provides high security. The proposed system aims to boost the effectiveness of the existing one. Every constraint of the current system can be bypassed with this system. This reduces manual work and provides proper security and includes user interactions. The Educator provides inputs in text format, and it is processed by the NLP system so the system is identified as a layered architecture. The NLP system summarizes, extracts, and analyzes the user input text and generates different types of questions like multiple choice, question, and answers, landfill in the blanks. The Educator can upload these questions to students as an assessment. The main purpose of the proposed system is to reduce the time consumed by the setting of assessment questions like MCQ and take self-assessment to know how much the subject is understood. The system will generate questions with meaningful distractors and correct answers. Students can view these questions generated by educators along with answers, and answer the questions like an assessment for practice.

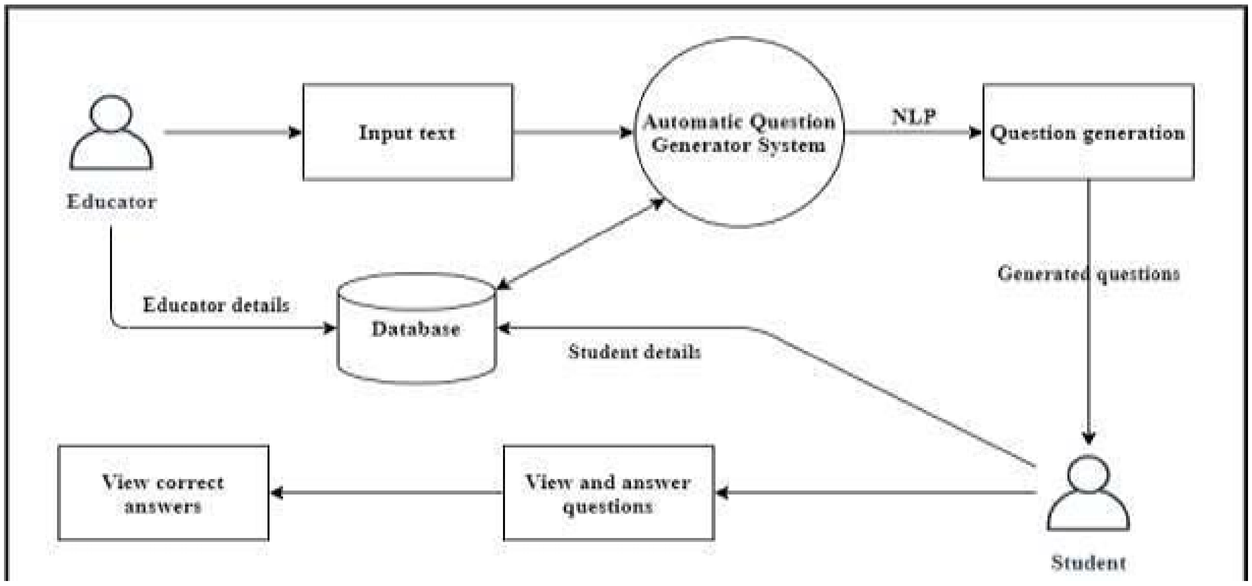


Fig.1 Proposed Methodology

The System Architecture of the Automatic Question Generator in Figure 1 represents the relationship between the principal elements of the system by using blocks and arrows as shown in the above figure. The above diagram represents a block diagram of an Automatic question generator. In this process Educator and Student need to provide valid credentials to log in to the system. The user details are then stored in the database. Educators need to provide the text as the input to generate questions. Questions are generated based on text using Natural Language Processing technology. The generated questions are stored in the database and later uploaded to students. Students can practice those questions and answer them. Students can also ask queries and view the results of answered and unanswered questions.

MODEL DESIGN:

SpaCy:

SpaCy provides Python APIs for Natural Language Processing (NLP) that is free and open source. Data processing and analysis using it are becoming increasingly popular in NLP. There are large volumes of unstructured textual data, so processing and interpreting this data are crucial. Data must be represented in a way that can be understood by computers to achieve this. A natural language processing system can help you achieve this. The NLP models included in spaCy can be used to perform the majority of commonly occurring NLPtasks. A number of tasks can be performed by SpaCy, it includes text tokenization, POS tagging, named entity recognition (NER), lemmatization, and word vectorization.

Fig2.Spacy Architecture

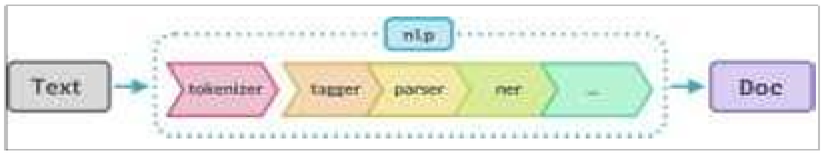


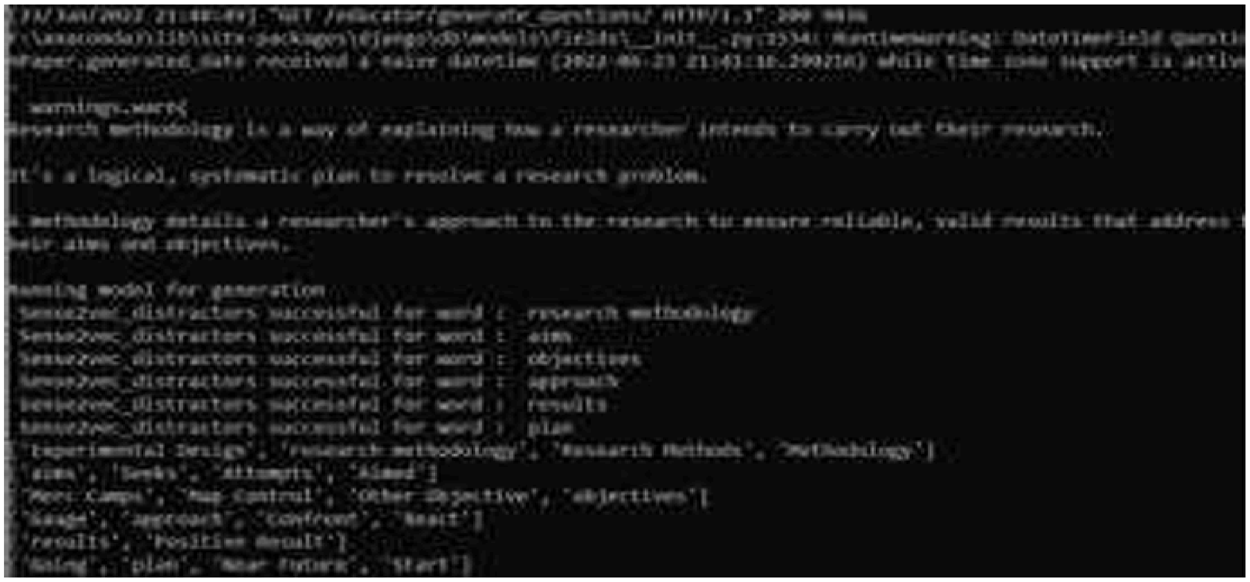
Figure 2 represents the architecture of spaCy library; it involves Tokenizer, Tagger, Parser, and NER, etc. The tokenization process involves breaking up a text into tiny segments, known as tokens. Depending on the target, tokenization can produce tokens of sentences at the document level, tokens of words at the sentence level, or tokens of characters at the word level. The POS tags provided by Spacy include NOUN, PUNCT, ADJ, ADV, etc. With the help of statistical models and a trained pipeline, spaCy can categorize tokens based on labels or tags. A syntactic dependency parser is included in spaCy, as well as a powerful API for navigating the tree. This parser is also used to detect sentence boundaries, and to iterate over phrases containing nouns, or chunks. Identifying and classifying named entities is a task of Named Entity Recognition, a subset of NLP. Taking the raw text and categorizing the named entities into organizations, persons, places,time, money, etc. The entity types are basically identified and segmented into various classes according to their characteristics.

Natural Language Processing Toolkit (NLTK):

With NLTK, one may create Python programs that operate with data from natural human language. It is one of the most popular Python platforms. Programming language models are introduced in a useful way by NLTK. Tokenization, parsing, lemmatization, chunking,POS tagging, and stemming are all included in the NLTK text processing libraries. NLTKis a massive toolkit for natural language processing (NLP), designed to assist with all aspects of the methodology. NLTK enables them to split sentences into paragraphs, split up words, recognize the words' parts of speech, highlight the main points, and even help the machine understand the content.

Sense2vec:

A neural network model called Sense2vec uses a large corpora of words to generate vector space representations of them. The Sense2vec algorithm is an extension of the infamous Word2VEC algorithm. Instead of tokenizing words, it embeds "senses' '. The concept of a sense is the combination of a word with a label, i.e. a statement representing the circumstances in which a word is used. POS Tags, Polarity, Entity Names,Dependency Tags, etc. can be used for this type of label. Neural word representations obtained with Word2vec are unable to encode context, despite being able to capture complex semantic and syntactic relationships among words. A contextually keyed word vector (i.e., one vector for each word sense) is the best way to solve this problem while disambiguating word senses. Contextually keyed vectors are the solution to this problem with Sense2vec. As a simple but powerful variation of word2vec, sense2vec was developed. Syntactic dependency parsing is improved while calculating representations of word senses is significantly reduced in terms of computational overhead. This package contains a sense2vec model that integrates seamlessly with spacy.



```
[23/Jun/2022 21:40:49] "GET /educator/generate_questions/ HTTP/1.1" 200 9836
F:\anaconda3\lib\site-packages\django\db\models\fields\__init__.py:1534: RuntimeWarning: DateTimeField Questi
nPaper.generated_date received a naive datetime (2022-06-23 21:41:16.299210) while time zone support is activ

  warnings.warn(
Research methodology is a way of explaining how a researcher intends to carry out their research.

It's a logical, systematic plan to resolve a research problem.

A methodology details a researcher's approach to the research to ensure reliable, valid results that address t
heir aims and objectives.

Running model for generation
 Sense2vec_distractors successful for word :  research methodology
 Sense2vec_distractors successful for word :  aims
 Sense2vec_distractors successful for word :  objectives
 Sense2vec_distractors successful for word :  approach
 Sense2vec_distractors successful for word :  results
 Sense2vec_distractors successful for word :  plan
['Experimental Design', 'research methodology', 'Research Methods', 'Methodology']
['aims', 'Seeks', 'Attempts', 'Aimed']
['Merc Camps', 'Map Control', 'Other Objective', 'objectives']
['Gauge', 'approach', 'Confront', 'React']
['results', 'Positive Result']
['Going', 'plan', 'Near Future', 'Start']
```

Fig3

CODE:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import pandas as pd

# Download NLTK resources (uncomment if not already downloaded)
# nltk.download('punkt')
# nltk.download('wordnet')
# nltk.download('stopwords')

# Sample dataset of questions and their corresponding tags
data = [
    ("What is the capital of France?", "Location"),
    ("Who wrote 'Harry Potter'?", "Author"),
    ("What is the boiling point of water?", "Science"),
    ("What is 2 + 2?", "Math"),
    ("Who painted the Mona Lisa?", "Artist"),
]

# Preprocess the data
```

```
def preprocess(text):
    # Tokenization
    tokens = word_tokenize(text)

    # Lowercase and remove stopwords
    stop_words = set(stopwords.words('english'))
        tokens = [token.lower() for token in tokens if token.isalnum() and token.lower() not in
stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    return ' '.join(tokens)

X = [preprocess(question) for question, _ in data]
y = [tag for _, tag in data]

# Create TF-IDF vectorizer and SVM classifier
tfidf_vectorizer = TfidfVectorizer()
svm_classifier = SVC(kernel='linear')

# Hyperparameter tuning using G

ridSearchCV
parameters = {'tfidf__ngram_range': [(1, 1), (1, 2)],
              'svm__C': [0.1, 1, 10]}
grid_search = GridSearchCV(pipeline, parameters, cv=5)
grid_search.fit(X, y)

# Example batch of new questions
new_questions = [
    "Who discovered gravity?",
    "What is the largest planet in our solar system?",
    "Who is the president of the United States?"
]

# Preprocess and tag new questions
new_questions_preprocessed = [preprocess(question) for question in new_questions]
predicted_tags = grid_search.predict(new_questions_preprocessed)

# Display results
results_df = pd.DataFrame({'Question': new_questions, 'Predicted Tag': predicted_tags})
print(results_df)
```

CODE EXPLANATION:

- Import Libraries:
  - Import necessary libraries such as NLTK for natural language processing, scikit-learn for machine learning tasks, and pandas for data manipulation.
  - Ensure NLTK resources are downloaded (uncomment the download lines if needed).
- Sample Dataset:
  - Define a sample dataset containing questions and their corresponding tags. Each tuple in the data list represents a question-tag pair.
- Preprocessing Function:
  - Define a preprocessing function preprocess(text) that tokenizer, removes stopwords, converts to lowercase, and lemmatizer the text.
- Preprocess Data:
  - Apply the preprocessing function to each question in the dataset to get preprocessed text (X) and corresponding tags (y).
- Create TF-IDF Vectorizer and SVM Classifier:
  - Initialize a TF-IDF vectorizer and an SVM classifier.
- Hyperparameter Tuning:
  - Define hyperparameters for tuning (ngram range and SVM's C value).
  - Use GridSearchCV to perform hyperparameter tuning on the pipeline (TF-IDF + SVM).
- New Questions:
  - Define a batch of new questions to be tagged.
- Preprocess and Tag New Questions:
  - Preprocess the new questions and predict their tags using the trained model.
- Display Results:
  - Create a Pandas DataFrame to display the new questions and their predicted tags.

This explanation outlines the steps involved in preprocessing the data, training a Support Vector Machine (SVM) model with TF-IDF features, hyperparameter tuning, tagging new questions, and displaying the results using Pandas DataFrame.

EVALUATION AND PERFORMANCE:

analysis After entering the text paragraph, click on Generate question the code will run on the command prompt as shown in Fig 3. Here for example entering a small paragraph, the paragraph will be split into sentences to generate questions for each sentence. There are three types of questions generated : multiple-choice, fill- in-the-blank, and Boolean-type questions. The sentence split is done using the NLTK package. The Sense Vector package is used to generate valid distractors with the correct answer for questions generated. It extracts keywords from each sentence and finds related words to generate distractors from that paragraph only. The questions are formed based on the keywords extracted from the paragraph. Boolean-type questions are also generated with the values Yes/ No.

RESULTS AND DISCUSSION:

The educator should select a subject from the dropdown, which wants to generate questions. Paragraph text should be provided, multiple paragraphs can be provided. By clicking on Generate questions, the question is generated and stored in the database, and it redirects to the question paper list page to view generated questions. The list of question papers with the subject and generated time and date is displayed. Educators can view the question by clicking on the button for different question types. The questions paper list consists of question types like MCQs, Fill in the blanks, and Boolean-type questions. The question column consists of questions and options consisting of one correct answer with other three distractors. The correct answer column displays the correct answer. show the process of question generation.

CONCLUSION AND FUTURE SCOPE:

The generation of questions from text is useful in numerous application domains, but manual generation is time-consuming and costly. In addition to being a web-based and desktop-based application system, it offers several features mainly for producing MCQs, Boolean type, and fill-in-the-blank questions. The goal of this application is to describe an automatic question generator built on NLP. This system helps students to self- analyze their knowledge about the subject. The application can be used in the educational field, and it is helpful in generating test papers for assessment. The generated questions are displayed on the student side with questions and answers and like an assessment for a practice test for students. It is helpful in competitive exams by generating questions and practicing them. This work can be further extended to generate paraphrasing questions, descriptive questions, and question paper generation for different types of questions. It is also possible to add additional automated answer copy-checking systems to verify the answers and provide the results to the students.