

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240070031 |
| **NAME** | ATCHAYAKANMANI K |
| **PROJECT TITLE** | Intelligent Quiz Generator from Text Documents |
| **DATE OF SUBMISSION** | 29/10/2025 |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

Artificial Intelligence (AI) is a rapidly evolving field that enables machines to perform cognitive tasks such as learning, reasoning, and problem-solving. In the field of education, AI plays a crucial role in automating the generation and evaluation of learning materials. One such application is the creation of quiz questions from textual data. Manually preparing quizzes from large documents is a time-consuming process for teachers and trainers. Hence, this project focuses on developing an intelligent quiz generator that extracts relevant information from a text document and automatically generates multiple-choice or short-answer questions. The system applies Natural Language Processing (NLP) techniques to understand the context and meaning of the input text and formulates meaningful questions along with correct answers. This project aims to reduce human effort and make learning assessments more efficient and interactive.

**PROBLEM STATEMENT**

Manually creating quizzes from large text materials such as notes, articles, and research papers is tedious and time-consuming. The challenge lies in automatically identifying important sentences, forming grammatically correct questions, and extracting accurate answers from the given content.

**GOAL**

The goal of this project is to design and implement an intelligent system that can automatically generate quiz questions from text documents using Natural Language Processing (NLP) techniques. The system should be able to:

* Extract key information from the text.
* Generate meaningful and grammatically correct questions.
* Provide corresponding answers for validation.

**THEORETICAL BACKGROUND**

The project is based on **Natural Language Processing (NLP)**, a branch of Artificial Intelligence that deals with the interaction between computers and human languages. NLP techniques like tokenization, part-of-speech (POS) tagging, and named entity recognition (NER) are used to understand text and extract relevant information.  
The **question generation** process involves identifying important sentences and transforming them into interrogative form. Popular algorithms and libraries such as **NLTK**, **spaCy**, and **Transformers (BERT, T5)** can be applied.  
After exploring various approaches, a rule-based and keyword extraction technique was chosen for simplicity and efficiency. This approach ensures that questions are meaningful and contextually correct for educational use.

**ALGORITHM EXPLANATION WITH EXAMPLE**

**Algorithm Steps:**

1. Load the input text document.
2. Preprocess the text by removing stop words and punctuation.
3. Use NLP techniques to identify important keywords and sentences.
4. Convert declarative sentences into questions by replacing keywords with interrogative words.
5. Store generated questions and answers for quiz generation.

**Example:**  
Input sentence: *“Artificial Intelligence enables machines to learn from data.”*  
Generated Question: *“What enables machines to learn from data?”*  
Answer: *“Artificial Intelligence”*

**IMPLEMENTATION AND CODE**

import random

import re

import spacy

# Load spaCy NLP model

nlp = spacy.load("en\_core\_web\_sm")

# ------------------------------

# 1. LOAD TEXT FILE

# ------------------------------

def load\_text(filename="sample.txt"):

    with open(filename, "r", encoding="utf8") as f:

        return f.read()

# ------------------------------

# 2. SPLIT TEXT INTO SENTENCES

# ------------------------------

def get\_sentences(text):

    doc = nlp(text)

    return [sent.text.strip() for sent in doc.sents if len(sent.text.strip()) > 30]

# ------------------------------

# 3. EXTRACT ENTITIES FOR QUESTIONS

# ------------------------------

def extract\_entities(text):

    doc = nlp(text)

    entities = [(ent.text.strip(), ent.label\_) for ent in doc.ents if len(ent.text.strip()) > 2]

    return entities

# ------------------------------

# 4. CREATE CLOZE (FILL-IN-THE-BLANK) QUESTIONS

# ------------------------------

def create\_cloze\_questions(sentences, entities, limit=10):

    questions = []

    used = set()

    for sent in sentences:

        for ent, label in entities:

            if ent in sent and ent not in used:

                question\_text = re.sub(re.escape(ent), "\_\_\_\_\_\_", sent, count=1)

                questions.append({

                    "type": "cloze",

                    "question": question\_text,

                    "answer": ent

                })

                used.add(ent)

                if len(questions) >= limit:

                    return questions

    return questions

# ------------------------------

# 5. CREATE TRUE/FALSE QUESTIONS

# ------------------------------

def create\_true\_false(sentences, limit=8):

    questions = []

    for s in sentences:

        year\_match = re.search(r"\b(19|20)\d{2}\b", s)

        if year\_match:

            year = int(year\_match.group(0))

            fake\_year = year + random.choice([-5, -10, +5, +10])

            false\_stmt = s.replace(str(year), str(fake\_year))

            if false\_stmt != s:

                questions.append({

                    "type": "truefalse",

                    "question": random.choice([s, false\_stmt]),

                    "answer": "True" if s == random.choice([s, false\_stmt]) else "False"

                })

        if len(questions) >= limit:

            break

    return questions

# ------------------------------

# 6. CREATE MULTIPLE CHOICE QUESTIONS

# ------------------------------

def create\_mcqs(sentences, entities, limit=7):

    questions = []

    random.shuffle(entities)

    for ent, label in entities:

        context = next((s for s in sentences if ent in s), None)

        if not context:

            continue

        # Generate distractors (same label if possible)

        same\_label\_ents = [e for e, l in entities if l == label and e != ent]

        distractors = random.sample(same\_label\_ents, min(3, len(same\_label\_ents))) if same\_label\_ents else []

        options = distractors + [ent]

        random.shuffle(options)

        q = {

            "type": "mcq",

            "question": f"{context}\n\nChoose the correct answer:",

            "options": options,

            "answer": ent

        }

        questions.append(q)

        if len(questions) >= limit:

            break

    return questions

# ------------------------------

# 7. RUN INTERACTIVE QUIZ

# ------------------------------

def run\_quiz(questions):

    print("\n🤖 Intelligent Quiz Generator — Interactive Mode")

    print("--------------------------------------------------\n")

    score = 0

    for i, q in enumerate(questions, start=1):

        print(f"\nQuestion {i}/{len(questions)}:")

        if q["type"] == "cloze":

            print(f"Fill in the blank:\n{q['question']}")

            user = input("Your answer: ").strip()

            if user.lower() == q["answer"].lower():

                print("✅ Correct!")

                score += 1

            else:

                print(f"❌ Incorrect. Correct answer: {q['answer']}")

        elif q["type"] == "mcq":

            print(q["question"])

            for idx, opt in enumerate(q["options"], start=1):

                print(f"{idx}. {opt}")

            user\_choice = input("Your choice (1-4): ").strip()

            try:

                user\_answer = q["options"][int(user\_choice) - 1]

                if user\_answer.lower() == q["answer"].lower():

                    print("✅ Correct!")

                    score += 1

                else:

                    print(f"❌ Wrong. Correct answer: {q['answer']}")

            except:

                print(f"⚠️ Invalid input. Correct answer: {q['answer']}")

        elif q["type"] == "truefalse":

            print(f"True or False:\n{q['question']}")

            user = input("Your answer (True/False): ").strip().capitalize()

            if user == q["answer"]:

                print("✅ Correct!")

                score += 1

            else:

                print(f"❌ Incorrect. Correct answer: {q['answer']}")

    print("\n----------------------------------------")

    print(f"🎯 Quiz Complete! Your Score: {score}/{len(questions)}")

    print("----------------------------------------\n")

# ------------------------------

# 8. MAIN FUNCTION

# ------------------------------

def main():

    text = load\_text("sample.txt")

    sentences = get\_sentences(text)

    entities = extract\_entities(text)

    cloze\_q = create\_cloze\_questions(sentences, entities, limit=10)

    mcq\_q = create\_mcqs(sentences, entities, limit=7)

    tf\_q = create\_true\_false(sentences, limit=8)

    all\_q = cloze\_q + mcq\_q + tf\_q

    random.shuffle(all\_q)

    run\_quiz(all\_q[:25])

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**OUTPUT**

**A computer screen with white text

Description automatically generated**

**RESULTS AND FUTURE ENHANCEMENT**

* The system successfully extracts key information from text and converts it into quiz questions with corresponding answers. This significantly reduces the manual effort of educators in creating assessments.
* In the future, this system can be enhanced using **deep learning models like T5 or BERT** for generating more complex and natural-sounding questions. A **web-based interface or mobile app** could also be developed for user-friendly access and automatic evaluation using Firebase or a local database.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/Atchayakanmani/AI\_Quiz\_Generator.git** |

**REFERENCES**

* Jurafsky, D. & Martin, J.H., *Speech and Language Processing*, Pearson, 2022.
* NLTK Documentation: https://www.nltk.org
* spaCy NLP Library: https://spacy.io
* T5 Model Overview – Google AI Research
* Towards Data Science Blog – *Automated Question Generation using NLP*