

Project: Intelligent Document Retrieval



Meet our dedicated team behind the scenes:

Sukhesh Chittanuri -RA2211026010171

Jami Jaswanth -RA2211026010173

Reddy Lakshmi Manoj -RA2211026010179

Thottamsetti Chatush Raj
-RA221102601053

Developing a Chat-Based Document Retrieval System

This project outlines the key steps involved in creating an intelligent chat-based system that can retrieve relevant information from a collection of documents. By leveraging natural language processing (NLP) and neural network models, we aim to build a user-friendly interface that allows users to seamlessly find the answers they need.



Requirement Analysis

1 Understand User Needs

Conduct interviews and surveys to gather insights into the specific information requirements of your target users. This will help you design a system that truly meets their needs.

2 Define Use Cases

Outline the various scenarios in which users might interact with the system, such as searching for relevant documents, getting answers to specific questions, or exploring related content.

3 Specify Performance Metrics

Determine the key performance indicators (KPIs) that will be used to measure the system's effectiveness, such as response accuracy, user satisfaction, and query response time.

4 Ensure Scalability

Design the system to handle growing volumes of documents and user requests without compromising performance or reliability.



Project Dataset Overview

Our project dataset includes several well-known collections for building robust question-answering systems:

- **SQuAD (Stanford Question Answering Dataset):** This dataset from Stanford University focuses on reading comprehension, making it great for training QA models and benchmarking different systems.
- **Natural Questions (NQ):** Provided by Google, this dataset contains real-world user queries and answers, helping us handle diverse, real-world questions.
- **MS MARCO:** Sourced from Microsoft, this dataset includes real-world search queries and answers, which can improve search engine optimization and contextual understanding.

TriviaQA: This challenging dataset of trivia questions and answers will help us build general knowledge-based QA capabilities and handle complex queries.

By leveraging these high-quality datasets, we can develop a robust, intelligent document retrieval system that meets the diverse needs of our users.



Data Collection and Preprocessing

Data Collection

Gather a diverse set of documents, including PDFs, web pages, and other textual sources, that cover the relevant topics and information your users might need. Ensure that the dataset is representative and comprehensive.

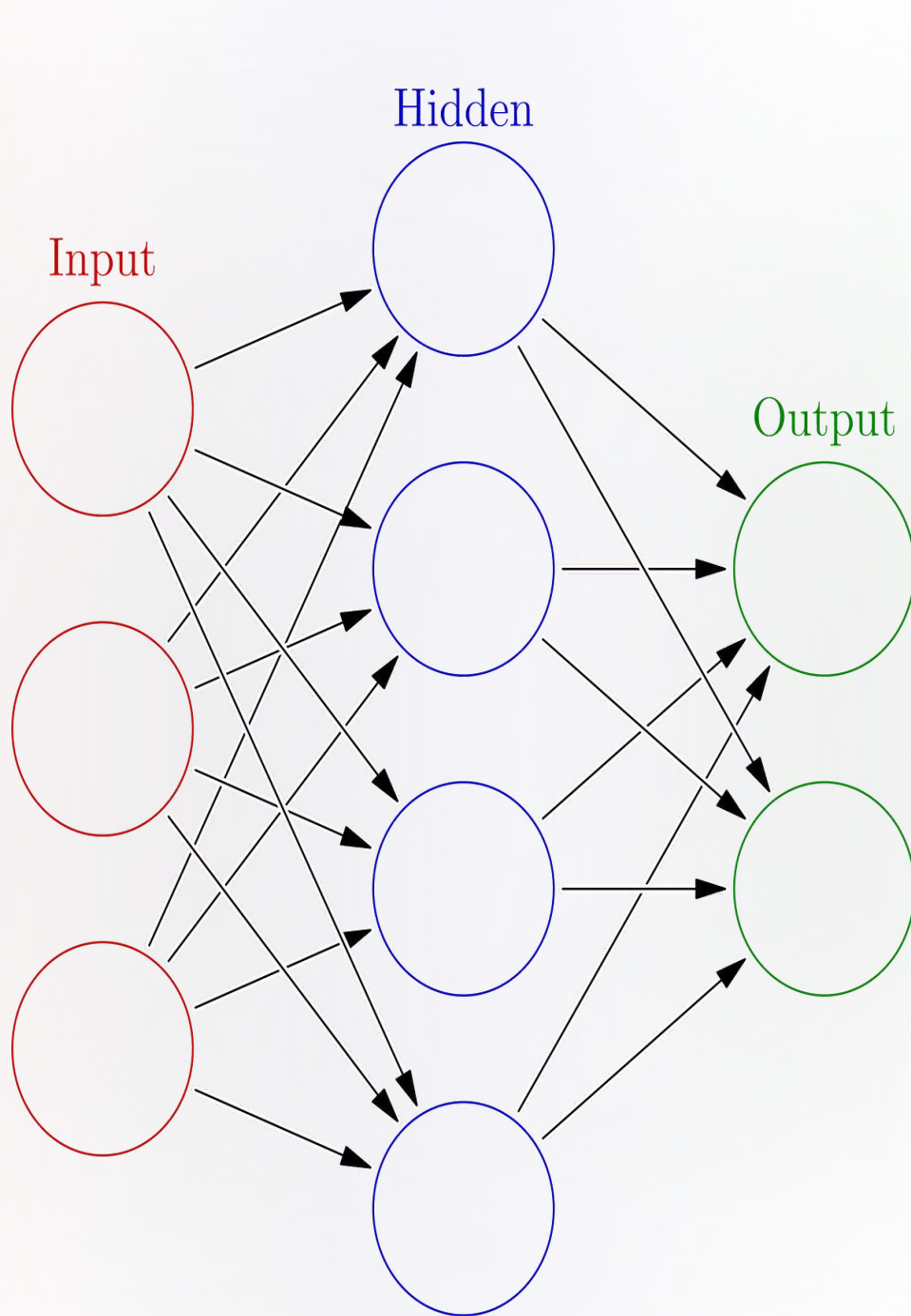
Text Extraction

Use tools like PyMuPDF and BeautifulSoup to extract the text content from the various document formats, ensuring that the data is in a clean, structured format suitable for further processing.

Data Cleaning

Apply text normalization techniques, such as removing stop words, handling spelling errors, and converting to a consistent format, to prepare the data for the machine learning models.

Model Selection and Training



1

Model Selection

Evaluate and choose an appropriate neural network model, such as BERT or GPT, that is well-suited for natural language understanding and question-answering tasks.

2

Dataset Preparation

Curate a high-quality training dataset that includes a diverse set of questions and their corresponding answers extracted from the collected documents.

3

Model Training

Fine-tune the selected model on the prepared dataset, optimizing hyperparameters and monitoring the model's performance on validation sets to ensure accurate retrieval of relevant information.

Integration and User Interface

Chat Interface

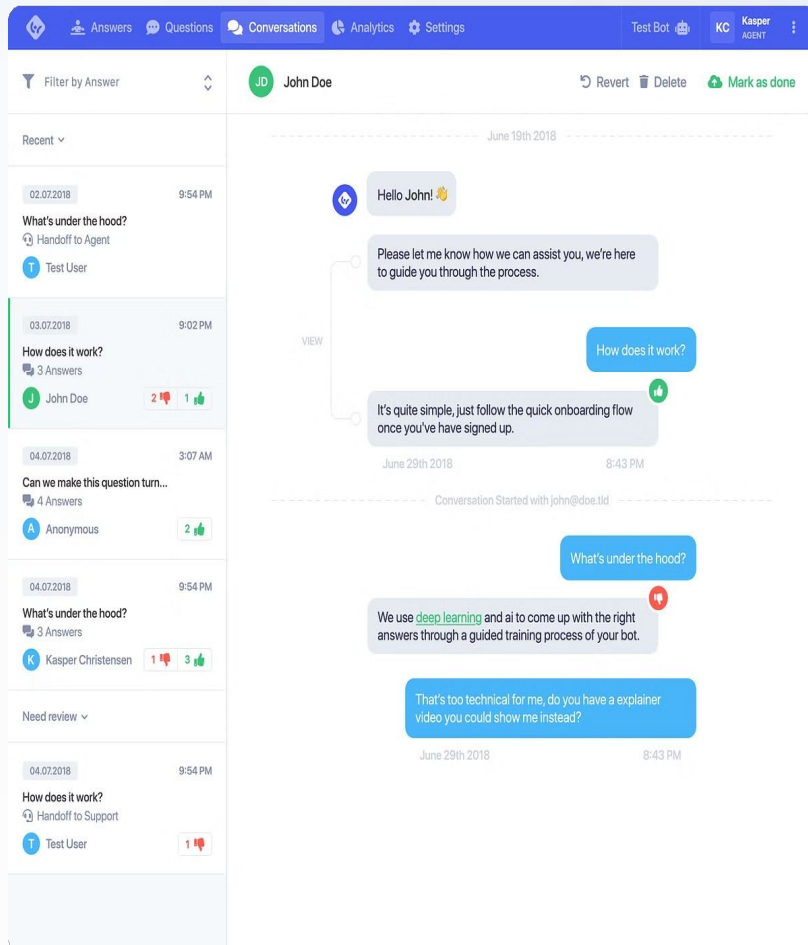
Develop a user-friendly chat interface that allows users to easily input their questions and receive the relevant information from the document collection. This can be implemented using web technologies like HTML, CSS, and JavaScript.

API Integration

Seamlessly integrate the trained NLP model with the chat interface, enabling the system to process user queries, retrieve the most relevant information, and present it to the user in a clear and concise manner.

Iterative Improvement

Continuously gather user feedback and monitor the system's performance to identify areas for improvement. Implement updates and refinements to enhance the user experience and the accuracy of the information retrieval.



Testing and Validation



Accuracy

Measure the system's ability to retrieve relevant information and provide accurate answers to user queries.



Response Time

Evaluate the system's performance in terms of the time it takes to process a user's question and deliver the response.



User Satisfaction

Collect user feedback and ratings to assess the overall satisfaction with the system's functionality and usability.



Scalability

Ensure the system can handle increasing volumes of documents and user requests without compromising performance.



Security and Privacy



1

Document Protection

Implement robust security measures to safeguard the documents and their content from unauthorized access or misuse.

2

User Data Privacy

Ensure that user queries and personal information are handled in compliance with data privacy regulations and best practices.

3

Auditing and Monitoring

Establish processes to regularly audit the system's security and privacy controls, and promptly address any identified vulnerabilities or compliance issues.

Methodology for Intelligent Document Retrieval Using ANN

1. Problem Definition and Requirements

The primary objective of this project is to develop an intelligent system capable of retrieving relevant documents based on a user's query. The documents could include text files, reports, web pages, etc., and the system will employ neural networks to learn document-query relationships for effective retrieval.

2. Data Collection and Preprocessing

Document Dataset: Collect or create a large dataset of documents relevant to a specific domain. These documents could come from sources like academic papers, news articles, or web pages.

3. Feature Extraction

Query Representation: Convert the user query into a vector form that captures its semantic meaning. This can be done using:

Word Embeddings: Represent each word in the query using pre-trained models like Word2Vec, GloVe, or BERT. These models capture the semantic meaning of the words.

Sentence Embeddings: Alternatively, embeddings can be created for the entire query, ensuring contextual information is preserved.

4. ANN Model Design

A Feedforward Neural Network (FNN) or a Siamese Network is commonly used for document retrieval tasks. A Siamese network consists of two neural networks sharing weights and is designed to rank the similarity between documents and queries. Alternatively, transformer-based architectures like BERT or GPT fine-tuned for retrieval tasks can be used for document-query similarity matching.



ACTIVATION FUNCTION

In an Intelligent Document Retrieval system using Artificial Neural Networks (ANN), the activation functions used depend on the architecture and the specific requirements of the retrieval task. However, based on the nature of document retrieval tasks, a typical architecture might use the following activation functions:

1. ReLU (Rectified Linear Unit)

Characteristics:

Outputs the input directly if it is positive; otherwise, it outputs zero.

Advantages:

Non-linearity: Helps the model learn complex patterns by introducing non-linearity.
Sparsity: Many neurons may output zero, leading to sparse representations, which can improve computational efficiency.
Mitigates Vanishing Gradient: Allows gradients to flow through the network more effectively, addressing the vanishing gradient problem often encountered with traditional activation functions.

Disadvantages:

Dying ReLU Problem: Some neurons can become inactive (output zero) for all inputs, leading to dead neurons that do not learn during training.

Impact on Performance:

ReLU often leads to faster training and better performance in deep networks, making it a popular choice in modern architectures. In the context of intelligent document retrieval, it helps capture complex relationships between documents and queries.

2. Tanh (Hyperbolic Tangent)

Characteristics:

Outputs values between -1 and 1.

Advantages:Zero-centered:

Outputs are zero-centered, which can help in faster convergence during training as it reduces bias in the gradients.Strong Gradient: It has a stronger gradient than Sigmoid for values near zero, which can improve learning in the earlier stages.

Disadvantages:

Vanishing Gradient Problem: Like Sigmoid, Tanh can still suffer from the vanishing gradient problem, especially for values far from zero.

Impact on Performance:

Tanh can perform well in hidden layers, especially for data that is centered around zero. In intelligent document retrieval, it can help with capturing relationships when the data has a more balanced distribution.

3. Sigmoid

Characteristics:

Outputs values between 0 and 1.

Advantages:

Probabilistic Interpretation: The output can be interpreted as probabilities, which is useful for binary classification tasks. Smooth Gradient: It has a smooth gradient, which can be beneficial in certain situations.

Disadvantages:

Vanishing Gradient Problem: The gradient is very small for extreme values (close to 0 or 1), leading to slow convergence or no learning in deep networks. Non-zero-centered: This can lead to bias in gradients and may slow down the training.

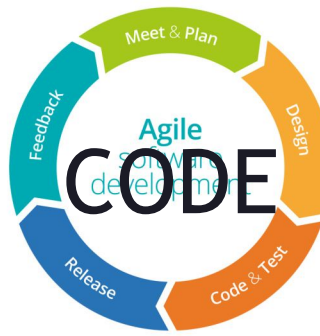
Impact on Performance:

Sigmoid is generally less effective in hidden layers of deep networks due to the vanishing gradient problem but can be useful in the output layer for binary classification. In multi-class classification scenarios like intelligent document retrieval, it is less preferred in hidden layers compared to ReLU and Tanh.



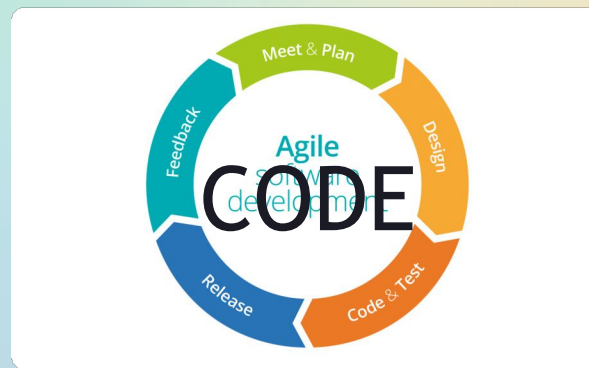
Continuous Improvement

Feedback Collection	Gather user feedback through surveys, usage analytics, and direct user interactions to identify areas for improvement.
Model Refinement	Continuously fine-tune the NLP model, incorporating new data and feedback to enhance the system's accuracy and responsiveness.
User Experience Optimization	Analyze user interactions and pain points to iteratively enhance the chat interface and overall user experience.
Deployment and Monitoring	Deploy updates and monitor the system's performance to ensure it continues to meet the evolving needs of your users.

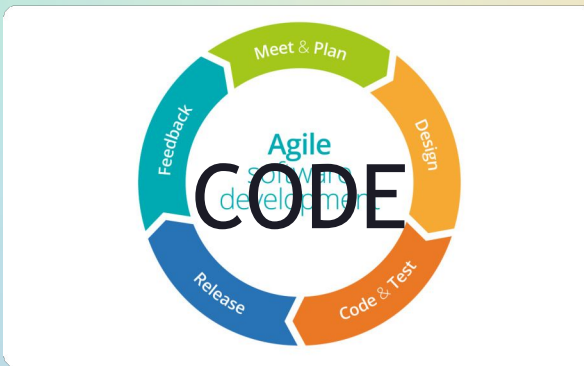


```
!pip install transformers torch pdfplumber gradio
from transformers import pipeline
import pdfplumberimport gradio as gr
```

```
# Load the question-answering model from Hugging Face
qa_pipeline = pipeline("question-answering", model="distilbert-base-uncased-distilled-squad")
def read_pdf(file_path):
    """Extracts text from a PDF file."""
    text = ""
    with pdfplumber.open(file_path) as pdf:
        for page in pdf.pages:
            text += page.extract_text()
    return text
def extract_questions(text):
    """Assumes each line in the document is a question."""
    questions = text.split("\n")
    return [q.strip() for q in questions if q.strip()]
def
generate_answers_for_questions(questions):
    """Generate answers for the given
    questions using the pre-trained model."""
    answers = []
    for question in questions:
        response = qa_pipeline({'question': question, 'context': ""}) # No context, only the question
    answers.append((question,
```



```
response['answer']))
return answers
def answer_questions_from_pdf(pdf_path):
    """Reads questions from a PDF and generates answers for them."""
    # Step 1: Extract text from PDF    text = read_pdf(pdf_path)
    # Step 2: Extract questions from the text    questions = extract_questions(text)
    # Step 3: Generate answers using the pre-trained model    answers = def
generate_answers_for_questions(questions)
return answersdef chatbot_interface(pdf_file):
answers = answer_questions_from_pdf(pdf_file.name)
formatted_answers = "\n\n".join([f"Q: {q}\nA: {a}" for q, a in answers])
return formatted_answers
# Customized Gradio Interfacegr.Interface(
    fn=chatbot_interface,
    inputs=gr.File(label="Upload PDF with Questions"),
    outputs=gr.Textbox(label="Generated Answers"),
    title="📄 Intelligent Document QA System",
```



description="Upload a PDF containing only questions.

The chatbot will generate relevant answers.

Perfect for educational and research purposes!", theme="compact", # Using compact theme for a cleaner look

css="""

body {background-color: #f5f5f5; font-family: 'Arial', sans-serif;}

.output_text {font-size: 16px; color: #333; line-height: 1.5;}

.input_file {background-color: #e0f7fa; color: #00796b; border: 1px solid #00796b;

border-radius: 5px;}

.title {color: #1565c0; font-weight: bold; text-align: center; font-size: 24px;}

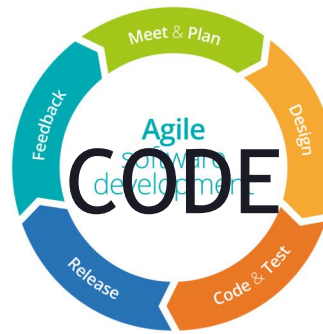
.description {color: #5e35b1; font-size: 18px; text-align: center;}

.button-primary {background-color: #42a5f5; border-radius: 5px; color: white;}

.button-primary:hover {background-color: #1e88e5;}

.gr-output .wrap {color: #424242; background-color: #f1f8e9; border-radius: 5px; padding: 10px;}


""").launch(share=True) # share=True for external access



Intelligent Document QA System

Upload a PDF containing only questions. The chatbot will generate relevant answers. Perfect for educational and research purposes!

Upload PDF with Questions

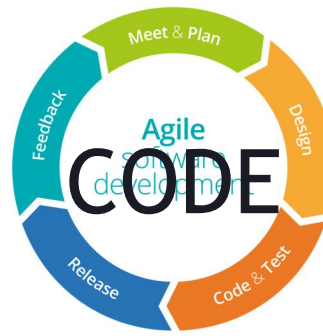

Drop File Here
- or -
Click to Upload

Generated Answers

Flag

Clear

Submit



Intelligent Document QA System

Upload a PDF containing only questions. The chatbot will generate relevant answers. Perfect for educational and research purposes!

 Upload PDF with Questions



ANN.pdf

72.2 KB ↓

Clear

Submit

Generated Answers

Flag