```
!pip install PyPDF2 transformers scikit-learn
```

```
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.44.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.2
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (202
Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 896.9 kB/s eta 0:00:00
Installing collected packages: PyPDF2
Successfully installed PyPDF2-3.0.1
```

Start coding or generate with AI.

intent classification

```
import spacy
from transformers import pipeline

# Load language model
nlp = spacy.load('en_core_web_sm')

# Simple question intent classification using Hugging Face's Transformers
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

# Possible categories (HR, IT Support, Events)
categories = ["HR Policies", "IT Support", "Company Events"]

def classify_query(query):
    result = classifier(query, categories)
    return result["labels"][0]

user_query = "How can I fix my server?"
intent = classify_query(user_query)
print(f"Detected intent: {intent}")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                   1.15k/1.15k [00:00<00:00, 31.9kB/s]
model.safetensors: 100%                                  1.63G/1.63G [00:19<00:00, 78.0MB/s]
tokenizer_config.json: 100%                                26.0/26.0 [00:00<00:00, 877B/s]
vocab.json: 100%                                    899k/899k [00:00<00:00, 6.97MB/s]
merges.txt: 100%                                    456k/456k [00:00<00:00, 5.03MB/s]
tokenizer.json: 100%                                  1.36M/1.36M [00:00<00:00, 9.63MB/s]
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenization_spa
  warnings.warn(
Detected intent: IT Support
```

summarization

```
from transformers import pipeline

# Load summarization model
summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# Example document text
document_text = """
The leave policy allows employees to apply for paid leave up to 20 days per year...
"""
# Summarizing document
summary = summarizer(document_text, max_length=100, min_length=30, do_sample=False)
print(f"Document Summary: {summary[0]['summary_text']}")

# Keyword extraction using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

def extract_keywords(text, top_n=5):
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform([text])
    indices = X[0].toarray()[0].argsort()[-top_n:]
    features = vectorizer.get_feature_names_out()
    keywords = [features[i] for i in indices]
    return keywords

keywords = extract_keywords(document_text)
print(f"Extracted Keywords: {keywords}")
```

```
config.json: 100%                                   1.58k/1.58k [00:00<00:00, 73.6kB/s]
model.safetensors: 100%                                  1.63G/1.63G [00:21<00:00, 36.2MB/s]
generation_config.json: 100%                                363/363 [00:00<00:00, 7.63kB/s]
vocab.json: 100%                                    899k/899k [00:00<00:00, 18.6MB/s]
merges.txt: 100%                                    456k/456k [00:00<00:00, 8.43MB/s]
tokenizer.json: 100%                                  1.36M/1.36M [00:00<00:00, 15.7MB/s]
Your max_length is set to 100, but your input_length is only 22. Since this is a summarization task, where outputs shorter than
Document Summary: The leave policy allows employees to apply for paid leave up to 20 days per year... but only for a maximum of
Extracted Keywords: ['employees', 'paid', 'policy', 'year', 'leave']
```

filtering bad words

```
bad_words = ["shit", "trash", "bitch"]

def filter_bad_language(user_input):
    words = user_input.split()
    filtered_words = ["****" if word.lower() in bad_words else word for word in words]
    return " ".join(filtered_words)

user_input = "This is a trash example."
filtered_input = filter_bad_language(user_input)
print(f"Filtered Text: {filtered_input}")
```

```
Filtered Text: This is a **** example.
```

a full application with PDF file

```
from transformers import pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
import PyPDF2

# Function to extract text from a multi-page PDF file
def extract_text_from_pdf(pdf_path):
    with open(pdf_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        text = ""
        for page_num in range(len(reader.pages)):
            page = reader.pages[page_num]
            page_text = page.extract_text()
            if page_text:  # Ensure page is not empty
                text += page_text + " "
    return text

# Function to select important sentences based on TF-IDF importance
def select_important_sentences(text, token_limit=1024):
    vectorizer = TfidfVectorizer(stop_words='english')
    sentences = text.split(". ")  # Split text into sentences
    X = vectorizer.fit_transform(sentences)

    # Sum up TF-IDF scores for each sentence
    sentence_scores = X.sum(axis=1).A1
    # Sort by importance and select top sentences until we reach token limit
    important_sentences_idx = sentence_scores.argsort()[::-1]  # Sort in descending order
    selected_sentences = []
    total_tokens = 0

    # Select sentences until we reach the token limit
    for idx in important_sentences_idx:
        sentence = sentences[idx]
        sentence_tokens = len(sentence.split())
        if total_tokens + sentence_tokens > token_limit:
            break
        selected_sentences.append(sentence)
        total_tokens += sentence_tokens

    return ". ".join(selected_sentences)

# Summarization function using Hugging Face's transformer model
def summarize_text(text, max_len=100, min_len=30, token_limit=1024):
    summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

    # Break the text into chunks that fit the token limit
    chunk_size = token_limit
    text_chunks = [text[i:i+chunk_size] for i in range(0, len(text), chunk_size)]

    # Summarize each chunk separately and combine the summaries
    summaries = []
    for chunk in text_chunks:
        summary = summarizer(chunk, max_length=max_len, min_length=min_len, do_sample=False)
        summaries.append(summary[0]['summary_text'])

    # Combine the chunk summaries into one overall summary
    return " ".join(summaries)

# Keyword extraction function using TF-IDF
def extract_keywords(text, top_n=5):
    vectorizer = TfidfVectorizer(stop_words='english')
    X = vectorizer.fit_transform([text])
    indices = X[0].toarray()[0].argsort()[-top_n:]
    features = vectorizer.get_feature_names_out()
    keywords = [features[i] for i in indices]
    return keywords

# Main function to handle PDF summary and keyword extraction
def process_pdf(pdf_path):
    # Extract text from the PDF
    document_text = extract_text_from_pdf(pdf_path)

    # Check the length of the extracted text
    print(f"Length of extracted text: {len(document_text)}")

    #if len(document_text) > :  # Check if there's enough content for summarization
        # Summarize the extracted text by compressing and focusing on important parts
    summary = summarize_text(document_text)
    print(f"Document Summary: {summary}")

        # Extract keywords from the extracted text
    keywords = extract_keywords(document_text)
    print(f"Extracted Keywords: {keywords}")
    #else:
        #print("Extracted text is too short for summarization.")

# Example usage: PDF file path for a large document (3 pages)
pdf_path = "/content/gailpolicy.pdf"
process_pdf(pdf_path)
```

Document Summary: HR policies are guidelines for managing employees in an organization, providing guidance on HR issues. Policies define relationships, address issues like diversity, discipline, age, and employment, and ensure equal opportunities. Policies on health, safety, grievances, and redundancy ensure fair treatment, while procedures enforce compliance. The role of HR is to communicate and implement these policies consistently with the organization's values.

Document Summary: HR policies are continuing guidelines on how people should be managed in the organization. A policy provides generalized guidance on how HR issues should be dealt with. A procedure spells out precisely what steps should be taken to deal with major employment issues. Formalized HR policies can be used in induction, team leader and management training to help participants understand the philosophies and values of the organization. HR policies can be expressed formally as statements of the values of the organization. They are a means for defi ning the employment rela- tionship and the psychological contract. Selznick (1957) emphasized the key role of values in organizations. The values expressed in an overall statement of HR policies may explicitly or implicitly refer to the following concepts. These include treating employees fairly and justly by adopting an 'even-handed' approach. HR aims to improve the quality of working life. This involves increasing the sense of satisfaction people obtain fr om their work. It involves reducing monotony, increasing autonomy and avoiding placing people under too much stress. Employers are increasingly having to recognize that they are subject to external as well as internal pres- sures. The most common areas in which HR policies exist are age and employment, discipline, e-mails and the internet. Policy on age and employment should take into account the UK legislation on age dis- crimination. Age is a poor predictor of job performance. It is misleading to equate physical and mental ability with age. An anti-bullying policy will state that bullying will not be tolerated by the organization. Those who persist in bullying their staff will be subject to disciplinary action, which could be severe in particularly bad cases. Managing diversity is a concept that recognizes the benefi ts to be gained from differences. It does not focus exclusively on issues of discrimination. It should be supported by a disciplinary procedure. Diversity management is being hailed as a proactive, results-focused approach. It is a welcome departure from the equal opportunities approach, which has been seen as reactive. Employees' e-mails should be monitored to check on excessive or unaccepted use. A policy statement could be included to the effect that the company reserves the right to access all e-mail messages. Employee relations policy will set out the org anization's approach to the rights of employees. It will also cover the basis upon which the organiza-tion works with trade unions. Employment policies should be concerned with fundamental aspects of the employment rela- tionship. They should take account of the requirements of relevant UK and European legisla-ipienttion and case law, which is beyond the scope of this handbook to cover in detail. Recent Acts and Regulations that are important include those concerning the minimum wage, working Carbuncletime and part-time workers. Equal opportunity policy should spell out the or ganization's determination to give equal opportunities to all, irrespective of sex, ra ce, disability, age or marital status. The policy should also deal with the extent to which the organization wants to take 'affi rmative action' to redress imbalances between the numbers employed according to sex or race. discrimination against any employee on the grounds of race, nationality, sex, sexual orientation, disability, religion, marital status or age. The company will ensure that equal opportunity principles are applied in all its HR policies and procedures. The policy on grievances could state that employees have the right to raise their grievances with their manager. The policy should be supported by a grievance procedure (see Chapter 61). Health and safety policies cover how the organization intends to provide healthy and safe places and systems of work. The policy could recognize that there will be occasions when the organization's present and future needs can only be met by recruitment from outside. The point could be made that a vigorous organization needs infusions of fresh blood from time to time if it is not to stagnate. Redundancy through redeployment and retraining programmes. Those affected will be given fair and equitable treatment. The policy should be supported by a redundancy procedure (see Chapter 61) Employees subjected to sexual harassment will be given advice, support and counsel. Sexual harassment is regarded as gross misconduct and, if proved, makes the individual liable for instant dismissal. Employees identified as having substance abuse problems will be offered advice and help. Any reasonable absence from work necessary to receive treatment will be granted under the organization's sickness scheme. Work and Families Act 2006 gives carers the right to request exible working to care for an elderly or sick relative. It will emphasize that the numbers of hours worked must not be treated as a criterion for assessing performance. HR policies need to address key HR issues that have been identifi ed in the organization. They must also take account of external infl uences such as legislation. The maximum amount of consultation should take place with managers, employees and their representatives. HR policies are subject to UK employ-ment legislation, EC employment regulations, and Codes of Practice. Check with managers, preferably starting at the top, on their views about HR poli-phthalcies. The aim will be to implement policies fairly and consistently. Line managers have an impor- tant role in doing this. The role of HR is to com-��municate and interpret the policies, convince line managers that they are right. HR policies provide guidelines on how key aspects of people management should be handled. The aim is to ensure that any HR issues are dealt with consistently in accord with the values of the organization. HR policies cover age and employ-mentation, AIDS, bullying, discipline, e-mails and the internet. HR policies need to address the key HR issues that have been identifi ed in the organization. They must also take account of external infl uences such as legislation. The aim will be to implement policies fairly and consistently. Peters, T and Waterman, R (1982) In Search of Excellence , Harper & Row, New Y ork.Purcell, J, Kinnie, K, Hutchinson, S, Rayton, B and Swart, J (2003) People and Performance: How people management impacts on organisational performance, CIPD, London. Selznick, P (1957) Leadership and Administration , Row, Evanston, Ill

```
from transformers import pipeline

# Load a question-answering pipeline (using DistilBERT model fine-tuned on SQuAD)
qa_model = pipeline("question-answering", model="distilbert-base-uncased-distilled-squad")

# Function to answer questions based on the extracted text from the PDF
def ask_question(context, question):
    result = qa_model(question=question, context=context)
    return result['answer']

# Example usage: Pass the extracted text or summary to the Q&A system
def process_question_from_extracted_text(document_text):
    print("You can now ask questions based on the content of the PDF document.")
    question = input("Please enter your question: ")

    # You can choose to use either the full document text or the summary as the context
    context = document_text  # or use a summarized version

    # Get the answer from the model
    answer = ask_question(context, question)
    print(f"Answer: {answer}")

# Example function that calls the process function and then lets the user ask questions
def main_with_qa(pdf_path):
    # Process the PDF first to extract text
    document_text = extract_text_from_pdf(pdf_path)

    # After extracting the text, allow the user to ask questions
    process_question_from_extracted_text(document_text)

# Example usage: After processing the PDF, ask questions
pdf_path = "/content/gailpolicy.pdf"
main_with_qa(pdf_path)



# Function to find relevant paragraphs containing HR-related keywords
def find_relevant_section(text, keywords):
    sentences = text.split(". ")
    relevant_sentences = [sentence for sentence in sentences if any(keyword.lower() in sentence.lower() for keyword in keywords)]
    return ". ".join(relevant_sentences)


# Example usage: Pass the extracted text or summary to the Q&A system
def process_question_from_extracted_text(document_text):
    print("You can now ask questions based on the content of the PDF document.")
    question = input("Please enter your question: ")

    # You can choose to use either the full document text or the summary as the context
    # Example usage: searching for HR-related sections in the document
    hr_related_keywords = ['HR', 'human resources', 'employee', 'issues', 'grievances', 'disciplinary']
    relevant_context = find_relevant_section(document_text, hr_related_keywords)

    # Get the answer from the model
    answer = ask_question(relevant_context, question)
    print(f"Answer: {answer}")

# Example function that calls the process function and then lets the user ask questions
def main_with_qa(pdf_path):
    # Process the PDF first to extract text
    document_text = extract_text_from_pdf(pdf_path)

    # After extracting the text, allow the user to ask questions
    process_question_from_extracted_text(document_text)

# Example usage: After processing the PDF, ask questions
pdf_path = "/content/gailpolicy.pdf"
main_with_qa(pdf_path)
```

```
You can now ask questions based on the content of the PDF document.
Please enter your question: policy
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-11-3e043c74d302> in <cell line: 32>()
     30 # Example usage: After processing the PDF, ask questions
     31 pdf_path = "/content/gailpolicy.pdf"
---> 32 main_with_qa(pdf_path)
     33

                                   5 frames
/usr/local/lib/python3.10/dist-packages/transformers/pipelines/question_answering.py in normalize(self, item)
    167                     raise ValueError(f"`{k}` cannot be None")
    168                 elif isinstance(item[k], str) and len(item[k]) == 0:
--> 169                     raise ValueError(f"`{k}` cannot be empty")
    170
    171             return QuestionAnsweringPipeline.create_sample(**item)

ValueError: `context` cannot be empty
```

```
pip install better-profanity
```

```
Collecting better-profanity
  Downloading better_profanity-0.7.0-py3-none-any.whl.metadata (7.1 kB)
Downloading better_profanity-0.7.0-py3-none-any.whl (46 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.1/46.1 kB 722.6 kB/s eta 0:00:00
Installing collected packages: better-profanity
Successfully installed better-profanity-0.7.0
```

```
from better_profanity import profanity

# Initialize the profanity filter with the default bad words dictionary
profanity.load_censor_words()

def filter_and_detect_bad_language(user_input):
    # Check if the input contains any bad language
    if profanity.contains_profanity(user_input):
        print("Warning: Your input contains inappropriate language.")
        print("Warning: Your input contains inappropriate language.")

    # Censor the bad words in the input
    filtered_input = profanity.censor(user_input)
    return filtered_input

# Example usage
user_input = "This is a trash example. This policy is stupid. "
filtered_input = filter_and_detect_bad_language(user_input)
print(f"Filtered Text: {filtered_input}")
```

```
Warning: Your input contains inappropriate language.
Filtered Text: This is a trash example. This policy is ****.
```

Start coding or generate with AI.