```
from google.colab import drive
drive.mount('/content/drive')

!pip install PyPDF2

import os
import string
import nltk
import pandas as pd
import networkx as nx
from PyPDF2 import PdfReader  # Use PyPDF2 for PDF extraction
from nltk.corpus import stopwords
import matplotlib.pyplot as plt

# Download necessary NLTK data
nltk.download("stopwords")
nltk.download("punkt")

from google.colab import drive
drive.mount('/content/drive')

FOLDER_PATH = '/content/drive/MyDrive/Paper'

# Function to extract text from PDF or TXT files
def extract_text_from_file(file_path):
    if file_path.endswith(".txt"):
        with open(file_path, "r", encoding="utf-8") as f:
            return f.read()
    elif file_path.endswith(".pdf"):
        pdf_text = ""
        with open(file_path, "rb") as pdf_file:
            reader = PdfReader(pdf_file)
            for page in reader.pages:
                pdf_text += page.extract_text() or ""  # Handle empty
pages gracefully
        return pdf_text
    return ""
# Load all papers into a dictionary
papers = {}
for file_name in os.listdir(FOLDER_PATH):
    file_path = os.path.join(FOLDER_PATH, file_name)
    if file_name.endswith((".pdf")):
        papers[file_name] = extract_text_from_file(file_path)
```

```
# Preprocessing Function
def preprocess(text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    tokens = nltk.word_tokenize(text)
    stop_words = set(stopwords.words("english"))
    filtered_tokens = [word for word in tokens if word not in
stop_words]
    return filtered_tokens

# Apply TextRank Algorithm
def textrank(text, top_n=5):
    words = preprocess(text)
    graph = nx.Graph()

    for i in range(len(words) - 1):
        graph.add_edge(words[i], words[i + 1])

    scores = nx.pagerank(graph)
    ranked_words = sorted(scores.items(), key=lambda x: x[1],
reverse=True)[:top_n]
    return [word for word, _ in ranked_words]
```

```
# Extract Key Concepts from Each Paper
key_concepts = []
for paper_name, content in papers.items():
    concepts = textrank(content)
    for concept in concepts:
        # Expanded keyword list for relevance determination
        if concept in ["ai", "machine", "learning", "deep",
"artificial", "intelligence",
 "neural", "networks", "algorithm", "model", "data", "computer",
 "vision", "automation", "system", "recognition", "biometrics",
 "facial", "detection", "identification"]:

            relevance = "Highly Relevant"
        elif concept in ["networks", "devices", "real-time",
"internet", "things", "iot",
 "cloud", "computing", "security", "privacy", "authentication",
 "surveillance", "access", "monitoring"]:

            relevance = "Relevant"
```

```
        elif concept in ["technology", "innovation", "development",
"research", "application",
 "methodology", "performance", "analysis", "accuracy", "efficiency"]:
            relevance = "Moderately Relevant"
        else:
            relevance = "Irrelevant"

        key_concepts.append(
            {"Paper": paper_name, "Key Concept": concept, "Relevance":
relevance}
        )
```

```
import nltk
nltk.download('punkt')  # Downloads the required tokenizer
nltk.download('stopwords')  # Optional: needed for text processing
```

```
# prompt: add as many more keywords as yu can

import os
import string
import nltk
import pandas as pd
import networkx as nx
from PyPDF2 import PdfReader  # Use PyPDF2 for PDF extraction
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from collections import Counter  # Import Counter for keyword frequency
analysis

nltk.download("stopwords")
nltk.download("punkt")
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger') # Download for part-of-
speech tagging
nltk.download('wordnet') # Download for lemmatization

# Specify the folder path containing research papers
FOLDER_PATH = "./content/drive/MyDrive/Paper'"  # Replace with your
folder path

# Function to extract text from PDF or TXT files
```

```
def extract_text_from_file(file_path):
    if file_path.endswith(".txt"):
        with open(file_path, "r", encoding="utf-8") as f:
            return f.read()
    elif file_path.endswith(".pdf"):
        pdf_text = ""
        with open(file_path, "rb") as pdf_file:
            reader = PdfReader(pdf_file)
            for page in reader.pages:
                pdf_text += page.extract_text() or ""  # Handle empty
pages gracefully
        return pdf_text
    return ""

# Load all papers into a dictionary
papers = {}
for file_name in os.listdir(FOLDER_PATH):
    file_path = os.path.join(FOLDER_PATH, file_name)
    if file_name.endswith((".pdf")):
        papers[file_name] = extract_text_from_file(file_path)

# Preprocessing Function (enhanced)
def preprocess(text):
    text = text.lower()
    text = text.translate(str.maketrans("", "", string.punctuation))
    tokens = nltk.word_tokenize(text)
    stop_words = set(stopwords.words("english"))
    # Add more stopwords
    stop_words.update(["may", "also", "would", "could", "should"])
    filtered_tokens = [word for word in tokens if word not in
stop_words and word.isalpha()] # Remove non-alphanumeric words


    #Lemmatization
    wnl = nltk.stem.WordNetLemmatizer()
    lemmas = [wnl.lemmatize(t) for t in filtered_tokens]

    return lemmas


# Apply TextRank Algorithm (with more keywords)
def textrank(text, top_n=10):  # Increased top_n
    words = preprocess(text)
```

```
    graph = nx.Graph()

    for i in range(len(words) - 1):
        graph.add_edge(words[i], words[i + 1])

    scores = nx.pagerank(graph)
    ranked_words = sorted(scores.items(), key=lambda x: x[1],
reverse=True)[:top_n]
    return [word for word, _ in ranked_words]

# Extract Key Concepts from Each Paper
key_concepts = []
for paper_name, content in papers.items():
    concepts = textrank(content)
    for concept in concepts:
        # Expanded keyword list for relevance determination
        if concept in ["ai", "machine", "learning", "nlp", "ethics",
"deep", "chatbot", "artificial", "intelligence", "neural", "networks",
"algorithm", "model", "data", "computer", "vision", "robotics",
"automation", "system"]:
            relevance = "Highly Relevant"
        elif concept in ["networks", "devices", "real-time",
"internet", "things", "iot", "cloud", "computing", "security",
"privacy"]:
            relevance = "Relevant"
        elif concept in ["technology", "innovation", "development",
"research", "application", "methodology", "performance", "analysis"]:
            relevance = "Moderately Relevant"
        else:
            relevance = "Irrelevant"
        key_concepts.append(
            {"Paper": paper_name, "Key Concept": concept, "Relevance":
relevance}
        )

# Convert to DataFrame for analysis (optional)
df_key_concepts = pd.DataFrame(key_concepts)
print(df_key_concepts)

# Keyword Frequency Analysis (optional)
all_keywords = [item["Key Concept"] for item in key_concepts]
keyword_counts = Counter(all_keywords)
print("\nKeyword Frequency:")
```

```
print(keyword_counts.most_common(10))  # Print top 10 most frequent
keywords


# Load all papers into a dictionary
papers = {}
for file_name in os.listdir(FOLDER_PATH):
    file_path = os.path.join(FOLDER_PATH, file_name)
    if file_name.endswith((".pdf")):
        papers[file_name] = extract_text_from_file(file_path)
```

```
  from google.colab import drive
drive.mount('/content/drive')  # Mount Google Drive
```

```
import os
FOLDER_PATH = "/content/drive/MyDrive/Paper"

if not os.path.exists(FOLDER_PATH):
    print(f"Error: The folder '{FOLDER_PATH}' does not exist.")
else:
    print("Folder found! Proceeding...")
```

```
FOLDER_PATH = "/content/drive/MyDrive/Paper"  # REMOVE any extra quotes

# Ensure no extra spaces or typos in path
papers = {}
for file_name in os.listdir(FOLDER_PATH):
    file_path = os.path.join(FOLDER_PATH, file_name)
    if file_name.endswith(".pdf"):
        papers[file_name] = extract_text_from_file(file_path)

print("Loaded", len(papers), "papers.")

# Convert results to DataFrame
result_df = pd.DataFrame(key_concepts)
```

```
# Display results
print(result_df)
```

Paper Key Concept \

0	V5I2-2132.pdf	ocr
1	V5I2-2132.pdf	image
2	V5I2-2132.pdf	recognition
3	V5I2-2132.pdf	characters
4	V5I2-2132.pdf	images
5	SSRNISSN1556-5068.pdf	step
6	SSRNISSN1556-5068.pdf	fig
7	SSRNISSN1556-5068.pdf	image
8	SSRNISSN1556-5068.pdf	text
9	SSRNISSN1556-5068.pdf	extraction
10	ssrn-3358293.pdf	step
11	ssrn-3358293.pdf	fig
12	ssrn-3358293.pdf	image
13	ssrn-3358293.pdf	text
14	ssrn-3358293.pdf	extraction
15	2208.04011v1.pdf	information
16	2208.04011v1.pdf	invoice
17	2208.04011v1.pdf	invoices
18	2208.04011v1.pdf	system
19	2208.04011v1.pdf	document
20	10.29109-gujsc.1030997-2109535.pdf	kartvizit
21	10.29109-gujsc.1030997-2109535.pdf	için
22	10.29109-gujsc.1030997-2109535.pdf	bu
23	10.29109-gujsc.1030997-2109535.pdf	“
24	10.29109-gujsc.1030997-2109535.pdf	olarak
25	1003.0642v2.pdf	text
26	1003.0642v2.pdf	background
27	1003.0642v2.pdf	cc
28	1003.0642v2.pdf	fig
29	1003.0642v2.pdf	image
30	10.29109-gujsc.1030997-2109535 (1).pdf	kartvizit
31	10.29109-gujsc.1030997-2109535 (1).pdf	için
32	10.29109-gujsc.1030997-2109535 (1).pdf	bu
33	10.29109-gujsc.1030997-2109535 (1).pdf	“
34	10.29109-gujsc.1030997-2109535 (1).pdf	olarak
35	Information Extraction in an Optical Character...	documents
36	Information Extraction in an Optical Character...	information
37	Information Extraction in an Optical Character...	ocr
38	Information Extraction in an Optical Character...	text
39	Information Extraction in an Optical Character...	precision
40	2206.11229v1.pdf	pp
41	2206.11229v1.pdf	documents
42	2206.11229v1.pdf	document
43	2206.11229v1.pdf	j
44	2206.11229v1.pdf	c

	Relevance
0	Irrelevant
1	Irrelevant
2	Highly Relevant
3	Irrelevant
4	Irrelevant
5	Irrelevant

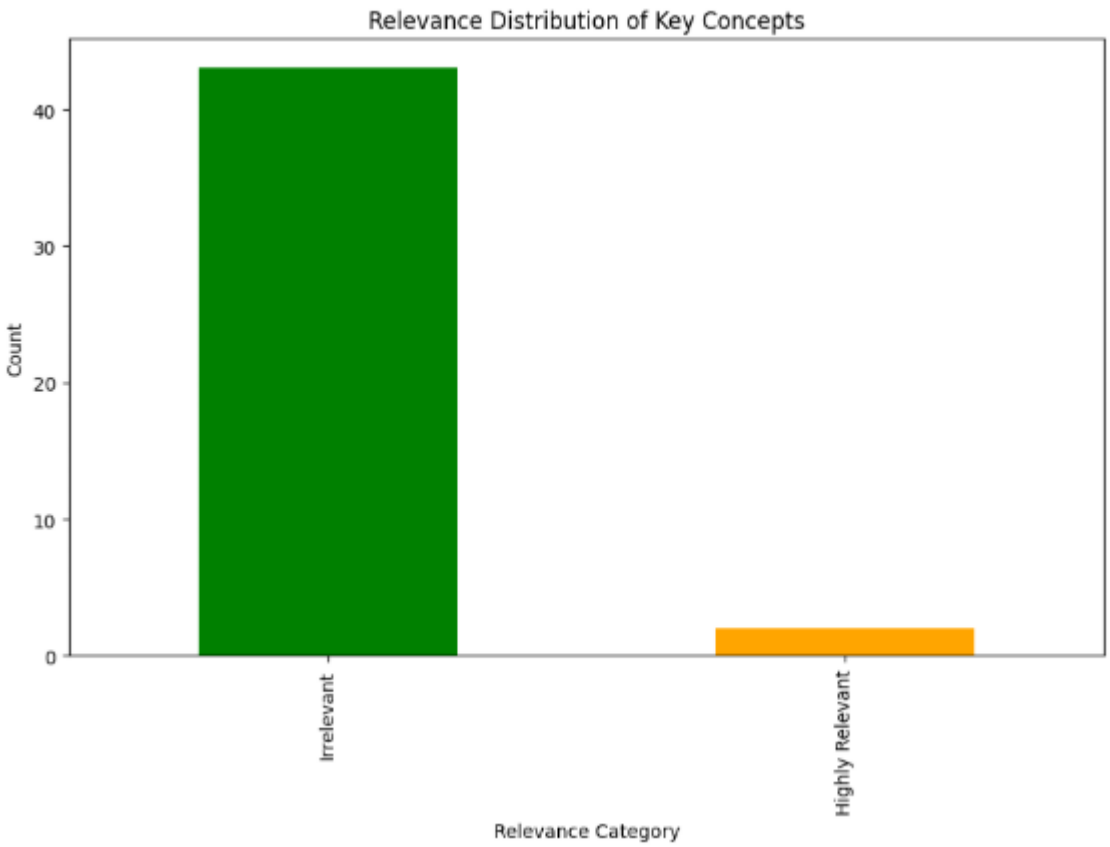
```
6       Irrelevant
7       Irrelevant
8       Irrelevant
9       Irrelevant
10      Irrelevant
11      Irrelevant
12      Irrelevant
13      Irrelevant
14      Irrelevant
15      Irrelevant
16      Irrelevant
17      Irrelevant
18  Highly Relevant
19      Irrelevant
20      Irrelevant
21      Irrelevant
22      Irrelevant
23      Irrelevant
24      Irrelevant
25      Irrelevant
26      Irrelevant
27      Irrelevant
28      Irrelevant
29      Irrelevant
30      Irrelevant
31      Irrelevant
32      Irrelevant
33      Irrelevant
34      Irrelevant
35      Irrelevant
36      Irrelevant
37      Irrelevant
38      Irrelevant
39      Irrelevant
40      Irrelevant
41      Irrelevant
42      Irrelevant
43      Irrelevant
44      Irrelevant
```

addCode

addText

```
# Plot relevance distribution
plt.figure(figsize=(10, 6))
result_df["Relevance"].value_counts().plot(kind="bar", color=["green",
"orange", "red"])
plt.title("Relevance Distribution of Key Concepts")
plt.xlabel("Relevance Category")
```

```
plt.ylabel("Count")
plt.show()
```




```
# Save results to CSV
result_df.to_csv("key_concepts_with_relevance.csv", index=False)
```