


```
def preprocess(text):

    text = text.lower()

    tokens = word_tokenize(text)

    stop_words = set(stopwords.words('english')) stemmer = PorterStemmer()

    words = [stemmer.stem(word) for word in tokens if word.isalnum() and word not in
    stop_words]

    return words documents = { }

for filename in os.listdir():

    if filename.endswith(".txt"):

        with open(filename, 'r', encoding='utf-8', errors='ignore') as f:

            text = f.read()

            documents[filename] = preprocess(text)

print(f"Total documents loaded: {len(documents)}")

inverted_index = defaultdict(set)

for doc_id, words in documents.items():

    for word in set(words): # avoid duplicates per document

        inverted_index[word].add(doc_id)

vocab_size = len(inverted_index)

print(f"\nVocabulary Size: {vocab_size} words")

print("\nSample inverted index terms:")

for term in list(inverted_index)[:10]:

    print(f'{term}: {sorted(inverted_index[term])}')
```

OUTPUT:

Total documents loaded: 11

```
defaultdict(<class 'set'>, {'today': {'HI.txt'}, 'work': {'HI.txt'}, 'warm': {'untitled4.txt', 'HI.txt'},
'hi': {'HI.txt'}, 'professor': {'HI.txt'}, 'aditya': {'HI.txt'}, 'sushuma': {'HI.txt'}, 'assist': {'HI.txt'},
```



```
'sunni': { 'untitled4.txt' } })
```

Vocabulary Size: 9 words

Sample inverted index terms:

hi: ['HI.txt']

sushuma: ['HI.txt']

today: ['HI.txt']

aditya: ['HI.txt']

work: ['HI.txt']

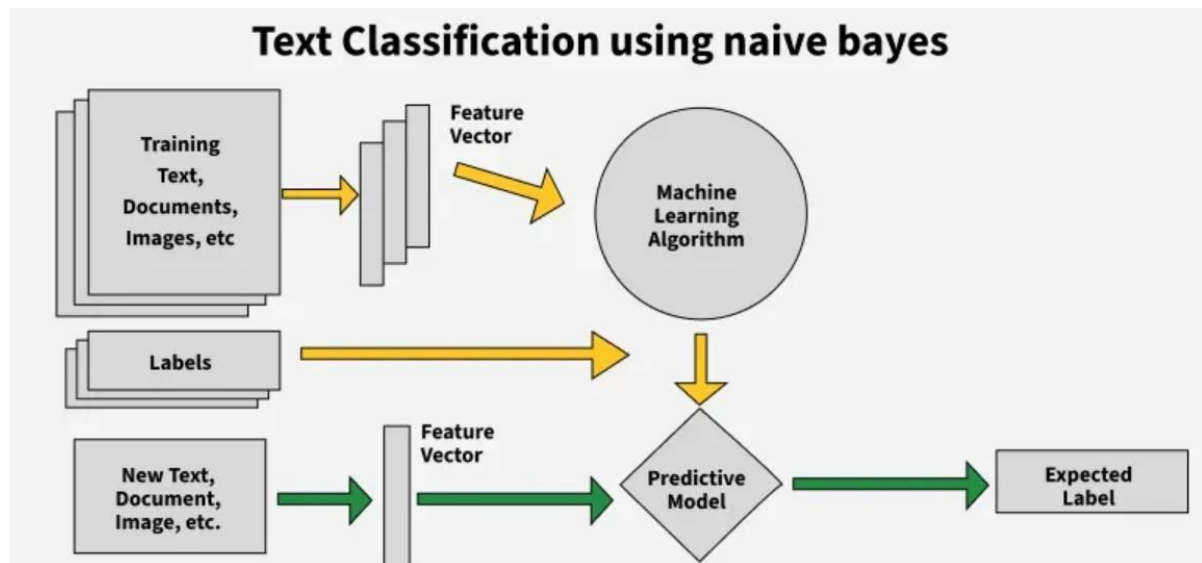
professor: ['HI.txt']

assist: ['HI.txt']

```
warm: ['HI.txt', 'untitled4.txt']
```

sunni: ['untitled4.txt']

[illegible]

**PROGRAM:**

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report

categories = ['sci.space', 'rec.sport.hockey', 'comp.graphics', 'alt.atheism']

newsgroups= fetch_20newsgroups(subset='all',categories=categories,shuffle=True, random_state=42)

print(f"Total documents: {len(newsgroups.data)}")

print(f"Target classes: {newsgroups.target_names}")

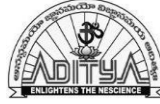
X_train, X_test, y_train, y_test = train_test_split (newsgroups.data, newsgroups.target, test_size=0.2,
random_state=42 )

vectorizer = TfidfVectorizer(stop_words='english')

X_train_tfidf = vectorizer.fit_transform(X_train)

X_test_tfidf = vectorizer.transform(X_test)

nb = MultinomialNB()
```

```
contingency_matrix[i][j] = count

return np.sum(np.max(contingency_matrix, axis=1)) / np.sum(contingency_matrix)

# Create a label mapping from cluster to majority class

def map_clusters_to_labels(y_true, y_pred):

    label_mapping = { }

    for cluster in np.unique(y_pred):

        indices = np.where(y_pred == cluster)[0]

        if len(indices) == 0:

            continue

        majority_label = mode(y_true[indices], keepdims=True).mode[0]
        label_mapping[cluster] = majority_label

    # Map each prediction to the true label

    mapped_preds = np.array([label_mapping[cluster] for cluster in y_pred])

    return mapped_preds

y_pred_mapped = map_clusters_to_labels(y_true, y_pred)

# Compute Metrics

purity = purity_score(y_true, y_pred)

precision = precision_score(y_true, y_pred_mapped, average='macro') recall =
recall_score(y_true, y_pred_mapped, average='macro')

f1 = f1_score(y_true, y_pred_mapped, average='macro')

# Print Results

print("Purity Score:", round(purity, 4))

print("Precision:", round(precision, 4))

print("Recall:", round(recall, 4))

print("F1-Score:", round(f1, 4))
```

[illegible]

Expt. No. :
Date :



Page No. :

OUTPUT:

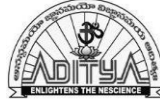
Purity Score: 1.0

Precision: 0.6767

Recall: 0.5225

F1-Score: 0.5253

[illegible]



```
websites = input("Enter comma-separated websites to limit crawling (e.g., bbc.com,cnn.com):")

SERP_API_KEY = '8d6bc2b3eef2e66c277a5a34be29b70d490834e929934539b15ae91c71dd569c'

search_url = 'https://serpapi.com/search.json'

def search_news(topic, websites):

    all_results = []

    for site in websites:

        params = {

            "engine": "google",

            "q": f"{topic} site:{site.strip()}",

            "api_key": SERP_API_KEY

        }

        response = requests.get(search_url, params=params)

        data = response.json()

        if "organic_results" in data:

            for result in data["organic_results"]:

                title = result.get("title")

                link = result.get("link")

                snippet = result.get("snippet", "")

                all_results.append((title, link, snippet))

    return all_results

def display_results(results):

    for idx, (title, link, snippet) in enumerate(results, start=1):

        print(f"\nNews {idx}:")

        print(f"Title : {title}")
```

[illegible]

```
print(f"URL : {link}")

print(f"Summary : {snippet}")

results = search_news(topic, websites)

if results:

    display_results(results)

else:

    print("No results found.")
```

OUTPUT:

Enter the news topic to search for: AI in healthcare

Enter comma-separated websites to limit crawling (e.g., `bbc.com,cnn.com`): `bbc.com,cnn.com`

News 1:

Title : AI in healthcare: what are the risks for the NHS?

URL : <https://www.bbc.com/news/articles/c6233x9k4dlo>

Summary : Generative AI will be transformative for NHS patient outcomes, a senior government advisor says.

News 2:

Title : How AI can spot diseases that doctors aren't looking for

URL : <https://www.bbc.com/news/articles/c9q7zqy1xlpo>

Summary : AI can take a second look at medical scans and flag up potential problems that doctors might not see.

News 3:

Title : How AI Has Transformed Healthcare's Future

URL : <https://www.bbc.com/storyworks/hpe-greenlake/how-ai-has-transformed-healthcares-future>

Summary : AI can link seemingly unrelated information to reveal new research pathways that yield better results. For example, AI models have identified potential ...

News 4:

Title : Hospitals will use AI to speed up patient care

URL : <https://www.bbc.com/news/articles/cye0yywdegdo>

Summary : Hospitals across the region are to use artificial intelligence (AI) technology to reduce unnecessary admissions and lengthy stays, ...

News 5:

Title : Can AI help modernise Ireland's healthcare system?

URL : <https://www.bbc.com/news/articles/cly7yxm3py5o>

Summary : Ireland is investing billions of euros to revamp its healthcare service - will AI help?

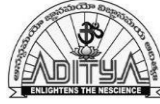
News 6:

Title : How artificial intelligence is matching drugs to patients

URL : <https://www.bbc.com/news/business-65260592>

Summary : Health-tech firms around the world are increasingly using AI to help tailor drugs for patients.

[illegible]



```

topics_map = {}

for page in tree.findall('page'):

    title = page.find('title').text.strip()

    links = [link.text.strip() for link in page.findall('link')]

    topics = page.find('topics').text.strip().split(",") if page.find('topics') is not
        None else []

    graph[title] = links

    topics_map[title] = [t.strip() for t in topics]

return graph, topics_map

```

Step 2: Build Adjacency Matrix

```
def build_adj_matrix(graph):

    pages = list(graph.keys())

    idx = {page: i for i, page in enumerate(pages)}

    n = len(pages)

    M = np.zeros((n, n))

    for page, links in graph.items():

        if links:

            for link in links:

                if link in idx:

                     $M[idx[link]][idx[page]] = 1 / \text{len}(\text{links})$ 

            else:

                 $M[:, idx[page]] = 1 / n$  # dangling node handling

    return M, pages
```

Step 3: Compute Topic-Specific PageRank

```
def topic_specific_pagerank(M, pages, topics_map, topic, d=0.85, tol=1e-6, max_iter=100):
```

[illegible]

```
n = len(pages)

teleport = np.array([1.0 if topic in topics_map[p] else 0.0 for p in pages])

if teleport.sum() == 0:

    teleport = np.ones(n)

teleport = teleport / teleport.sum() # normalize

r = np.ones(n) / n # initial rank

for i in range(max_iter):

    r_new = d * M @ r + (1 - d) * teleport

    if np.linalg.norm(r_new - r, 1) < tol:

        break

    r = r_new

return dict(zip(pages, r))
```

Step 4: Visualize the Web Graph with Topic Highlight

```
def draw_web_graph(graph, topics_map, topic):

    G = nx.DiGraph()

    for page, links in graph.items():

        for link in links:

            G.add_edge(page, link)

# Node colors: highlight pages having the topic

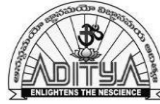
    node_colors = []

    for page in G.nodes():

        if topic in topics_map.get(page, []):

            node_colors.append("lightgreen") # highlight topic pages

        else:
```



```

node_colors.append("skyblue")        # normal pages

plt.figure(figsize=(6, 4))

pos = nx.spring_layout(G, seed=42)

nx.draw(G, pos, with_labels=True, node_color=node_colors, node_size=1500,
        font_size=10, arrowsize=15, edge_color="gray")

plt.title(f"Web Graph (Highlighted Topic: {topic})")

plt.show()

# Step 5: Input and Execute

xml_text = "<web>

    <page>

        <title>PageA</title>

        <link>PageB</link>

        <link>PageC</link>

        <topics>science,education</topics>

    </page>

    <page>

        <title>PageB</title>

        <link>PageC</link>

        <topics>science</topics>

    </page>

    <page>

        <title>PageC</title>

        <topics>sports</topics>

    </page>

</web>"

```

[illegible]

- SVD is a mathematical technique that breaks a large matrix into three smaller matrices.
- For a document-term matrix (like TF-IDF), SVD helps find patterns/relationships between terms and documents.
- It identifies the most important concepts (latent topics) in the data

PROGRAM:

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.decomposition import TruncatedSVD

from sklearn.metrics.pairwise import cosine_similarity

import numpy as np

# Step 1: Load dataset (subset for speed)

categories = ['sci.space', 'rec.sport.hockey', 'comp.graphics']

newsgroups = fetch_20newsgroups(subset='train', categories=categories, remove=('headers',
                                     'footers', 'quotes'))

# Step 2: TF-IDF Vectorization

vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)

X_tfidf = vectorizer.fit_transform(newsgroups.data)

print(f"Original TF-IDF shape: {X_tfidf.shape}") # (docs x terms)

# Step 3: SVD for LSI (Latent Semantic Indexing)

k = 100 # number of latent dimensions

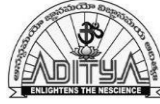
svd = TruncatedSVD(n_components=k)

X_lsi = svd.fit_transform(X_tfidf)

print(f"Reduced LSI shape: {X_lsi.shape}") # (docs x k topics)

# Step 4: Show similarity between some documents

def show_similar_docs(query_idx, top_n=5):
```

- Provides insights into emerging topics, popular hashtags, and influential entities.

PROGRAM:

```
import tweepy

# Replace with your own Bearer Token from Twitter Developer Portal

bearer_token =
"AAAAAAAAAAAAAAAAAAAAADhM4QEAAAAAEhGXBfqa4kNwgb3%2F3XEC8JceL
Ys%3D0AVX5bRfhoQTvuRjjokbg7zOQ6egn1VOGtL2xEXIW4N7IGsX9P"

# Initialize Tweepy client with bearer token

client = tweepy.Client(bearer_token=bearer_token)

# Define your search query

query = "AI OR Machine Learning"

# Fetch recent tweets matching the query

tweets = client.search_recent_tweets(

    query=query,

    max_results=100,          # maximum results per request (up to 100)

    tweet_fields=['created_at', 'text'] # request tweet creation time and text

)

# Check if tweets are returned

if tweets.data is not None:

    # Print tweet creation date and text

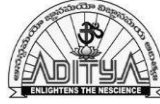
    for tweet in tweets.data:

        print(f"Created at: {tweet.created_at}")

        print(f"Tweet text: {tweet.text}\n")

else:

    print("No tweets found for this query.")
```

**OUTPUT:**

Created at: 2025-09-23 03:51:48+00:00

Tweet text: RT @leiane1: Good morning, family

How are you?


The @recallnet Arena is NOW open.

Trade proven, high-volume pairs with real liquidity....

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @icanvardar @stripe Stripe is becoming an ai labs

Created at: 2025-09-23 03:51:48+00:00

Tweet text: RT @GaiAIio:  GaiAI Discord is live!

Join our growing community of creators, developers, and Web3 AI explorers.

Discuss ideas, share gen...

Created at: 2025-09-23 03:51:48+00:00

Tweet text: RT @psicolut: a virginia sambando daquele jeito como rainha de bateria e voce aí se cobrando pra tirar um projeto do papel porque ainda não...

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @JnglJourney LOL....AI...UFOs.....the spooky ghouls of Halloween arriving early....



Or is it EU countries confabulating fake narratives to blame Russia for these mystery sightings.

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @OpenledgerFdn @kbwofficial @OpenledgerHQ Openledger is really building the fair layer of the OPEN internet AI

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @69on_ai 这是哪部作品的人物呀

Created at: 2025-09-23 03:51:48+00:00

Tweet text: RT @FractionAI_xyz: Here's a crazy thought:

Every Tuesday, we've been shipping something new and exciting, week in and week out.

This sh...

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @darwinmda_ @jaofranko As vezes é mais sobre o traços do artista, mas acho que se ele fizesse o cara chorando ai sim seria

Created at: 2025-09-23 03:51:48+00:00

Tweet text: RT @skywongraveeee: ผมเก็บสิ่งนี้มาได้มันคืออะไร ถาม ai ก็ไม่รู้ ใครทราบรบกวนแจ้งที
#MuTeLuvNotMyFatherEP1 <https://t.co/VzYjCk7Dv5>

Created at: 2025-09-23 03:51:48+00:00

Tweet text: @KAMADAN AI めっちゃ共感です！

PROGRAM:

OUTPUT: