2

September 15, 2025

```
[ ]: # 6. Extract textual content from multiple web pages (e.g., news sites, blogs)␣ ↪using web
     scraping techniques. Analyze the extracted data to detect emerging␣ ↪trends and topics.
     Implement this using Python, R, or any other relevant␣ ↪tool.

[ ]: import os, re
     from bs4 import BeautifulSoup
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.decomposition import NMF
     import numpy as np
     import warnings

[ ]: # Suppress RuntimeWarnings from NMF
     warnings.filterwarnings("ignore", category=RuntimeWarning)

[ ]: # List of downloaded HTML file paths
     FILES = [
         r"Z:\BIDA\exp6\Artificial Intelligence.html",
         r"Z:\BIDA\exp6\Startups_TechCrunch.html"
     ]

[ ]: def extract_text_from_file(filepath):
         if not os.path.exists(filepath):
             print(f"File not found: {filepath}")
             return ""
         try:
             with open(filepath, "r", encoding="utf-8") as f:
                 html = f.read()
         except Exception as e:
             print(f"Error reading {filepath}: {e}")
             return ""

         soup = BeautifulSoup(html, "html.parser")
         for tag in soup(["script", "style", "noscript", "iframe", "form"]): tag.decompose()

         container = soup.find("article") or soup.find("main") or soup
```



```
     text = " ".join(p.get_text(" ", strip=True) for p in container.↪find_all("p"))
     if len(text) < 200:
     text = soup.get_text(" ", strip=True)[:5000]
```

```
return re.sub(r"\s+", " ", text).strip()

[ ]: # Extract and clean text
docs = [extract_text_from_file(f) for f in FILES]
docs = [d for d in docs if len(d) > 200]
if not docs:
raise SystemExit("No usable content found in files.")

[ ]: # TF-IDF Vectorization
vec = TfidfVectorizer(stop_words="english", ngram_range=(1,2), max_df=0.9, ␣ ↪min_df=1)
X = vec.fit_transform(docs)
terms = np.array(vec.get_feature_names_out())
scores = np.asarray(X.mean(axis=0)).ravel()
top_idx = scores.argsort()[::-1][:25]

[ ]: print("Top TF-IDF terms (trends):")
for t, s in zip(terms[top_idx], scores[top_idx]):
print(f"{t:30s} {s:.4f}")

[ ]: # Topic Modeling with NMF
k = min(6, max(1, X.shape[0]))
nmf = NMF(n_components=k, random_state=42, init="nndsvd", max_iter=1000) W =
nmf.fit_transform(X)
H = nmf.components_

def label_topic(terms):
return ", ".join(terms[:3])

print("\nTopics:")
for i, row in enumerate(H):
idx = row.argsort()[::-1][:10]
topic_terms = terms[idx]
print(f"Topic {i} ({label_topic(topic_terms)}): " + ", ".join(topic_terms)) 2
```