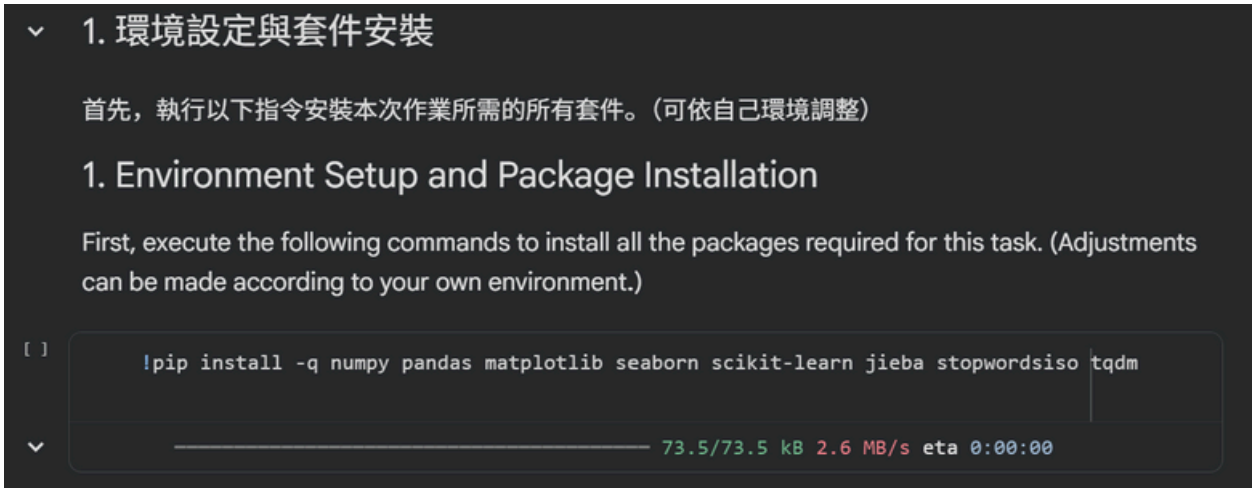
411285058 馮寶妍

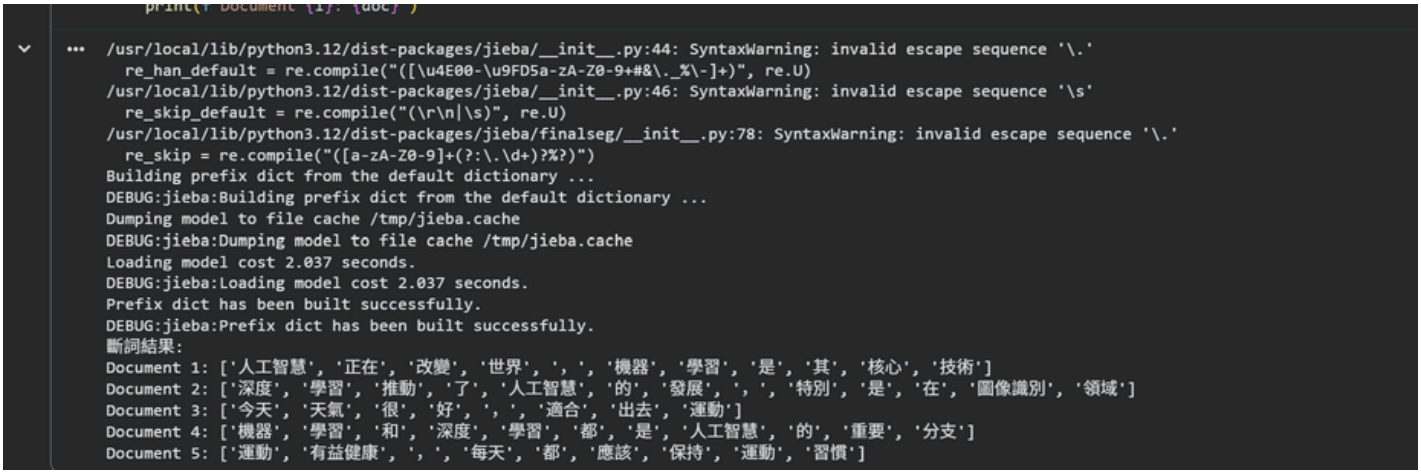
Due to my poor chinese skill, the comments or answers are in English

This is my first time using github so I hope I am doing it correctly as instructed
https://github.com/s411285058/LAB-AI-HW

0. Environment & Packages



A-1: TF-IDF Text Similarity Calculation



1. 手動實作 TF-IDF

	人工智慧	正在	改變	世界	,	機器	學習	是	其	核心	...	運動	和	都	重要	分支	有益健康	每天	應該	保持	習慣
Document 1	0.020286	0.083299	0.083299	0.083299	0.0	0.046439	0.020286	0.020286	0.083299	0.083299	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000
Document 2	0.017165	0.000000	0.000000	0.000000	0.0	0.000000	0.017165	0.017165	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000
Document 3	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.063853	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.00000	0.00000	0.00000
Document 4	0.020286	0.000000	0.000000	0.000000	0.0	0.046439	0.040572	0.020286	0.000000	0.000000	...	0.000000	0.083299	0.046439	0.083299	0.083299	0.00000	0.00000	0.00000	0.00000	0.00000
Document 5	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.113517	0.000000	0.056758	0.000000	0.000000	0.10181	0.10181	0.10181	0.10181	0.10181

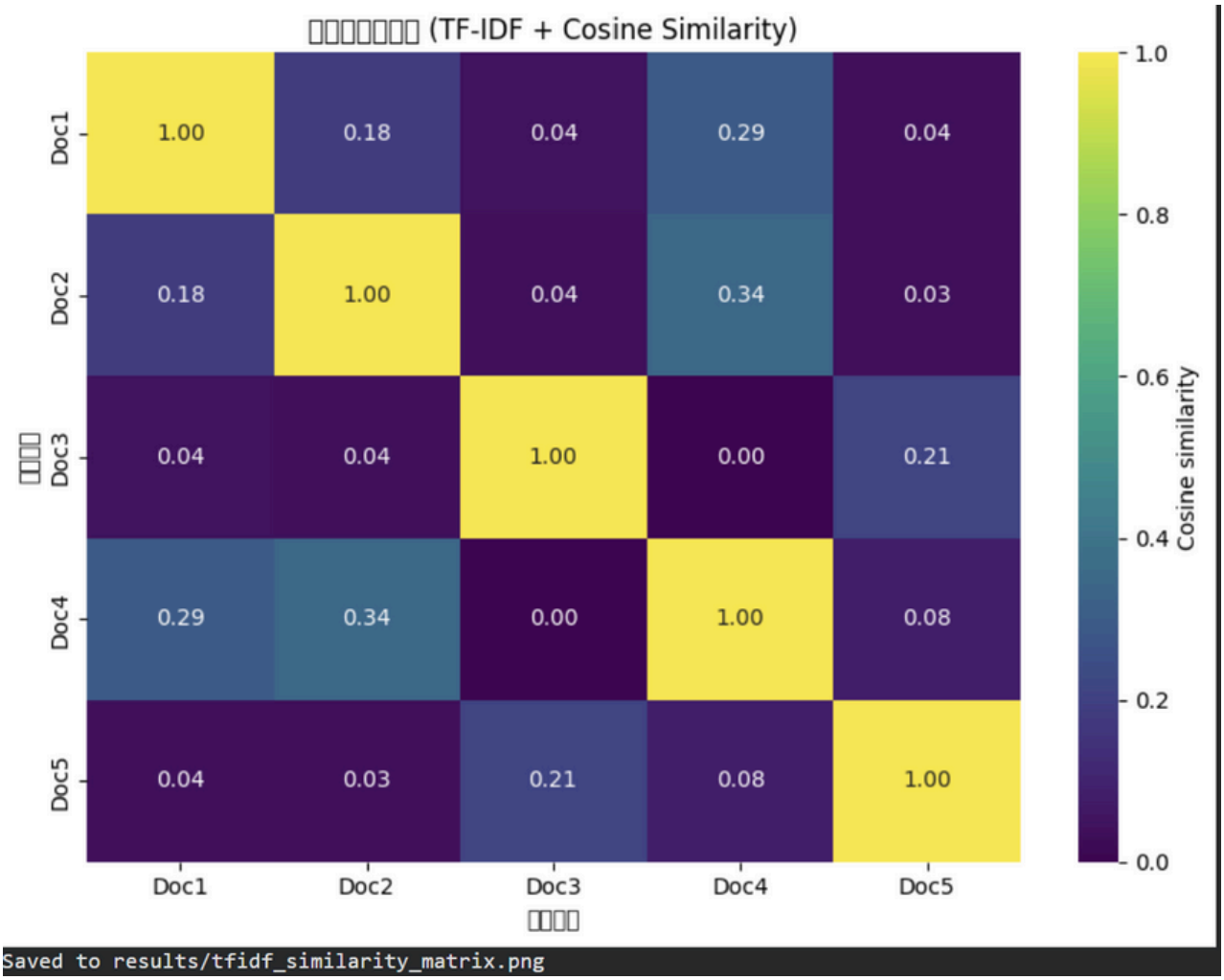
5 rows × 36 columns

2. 使用 scikit-learn 實作

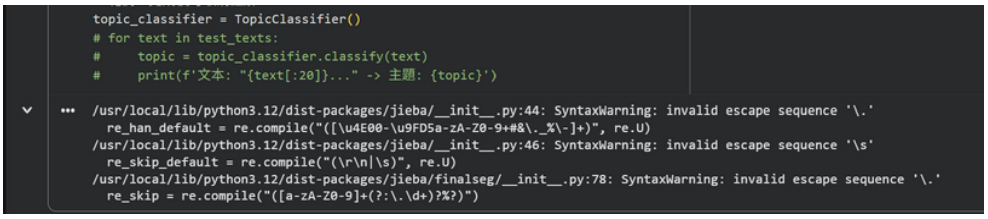
	Doc1	Doc2	Doc3	Doc4	Doc5
Doc1	1.000000	0.182701	0.041701	0.294422	0.037598
Doc2	0.182701	1.000000	0.038090	0.340526	0.034343
Doc3	0.041701	0.038090	1.000000	0.000000	0.209510
Doc4	0.294422	0.340526	0.000000	1.000000	0.077194
Doc5	0.037598	0.034343	0.209510	0.077194	1.000000

3. 視覺化（熱圖） 3. Visualization (heatmap)

```
No specific Chinese font found, falling back to default sans-serif.
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 25991 (\N{CJK UNIFIED IDEOGRAPH-6587}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 20214 (\N{CJK UNIFIED IDEOGRAPH-4EF6}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 32232 (\N{CJK UNIFIED IDEOGRAPH-7DE8}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 34399 (\N{CJK UNIFIED IDEOGRAPH-865F}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 26412 (\N{CJK UNIFIED IDEOGRAPH-672C}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 30456 (\N{CJK UNIFIED IDEOGRAPH-76F8}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 20284 (\N{CJK UNIFIED IDEOGRAPH-4F3C}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 24230 (\N{CJK UNIFIED IDEOGRAPH-5EA6}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 30697 (\N{CJK UNIFIED IDEOGRAPH-77E9}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:57: UserWarning: Glyph 38499 (\N{CJK UNIFIED IDEOGRAPH-9663}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 25991 (\N{CJK UNIFIED IDEOGRAPH-6587}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 20214 (\N{CJK UNIFIED IDEOGRAPH-4EF6}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 32232 (\N{CJK UNIFIED IDEOGRAPH-7DE8}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 34399 (\N{CJK UNIFIED IDEOGRAPH-865F}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 26412 (\N{CJK UNIFIED IDEOGRAPH-672C}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 30456 (\N{CJK UNIFIED IDEOGRAPH-76F8}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 20284 (\N{CJK UNIFIED IDEOGRAPH-4F3C}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 24230 (\N{CJK UNIFIED IDEOGRAPH-5EA6}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 30697 (\N{CJK UNIFIED IDEOGRAPH-77E9}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/tmp/ipython-input-468390214.py:58: UserWarning: Glyph 38499 (\N{CJK UNIFIED IDEOGRAPH-9663}) missing from font(s) DejaVu Sans.
  plt.savefig('results/tfidf_similarity_matrix.png', dpi=150, bbox_inches='tight')
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 25991 (\N{CJK UNIFIED IDEOGRAPH-6587}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 20214 (\N{CJK UNIFIED IDEOGRAPH-4EF6}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 32232 (\N{CJK UNIFIED IDEOGRAPH-7DE8}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 34399 (\N{CJK UNIFIED IDEOGRAPH-865F}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 26412 (\N{CJK UNIFIED IDEOGRAPH-672C}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 30456 (\N{CJK UNIFIED IDEOGRAPH-76F8}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 20284 (\N{CJK UNIFIED IDEOGRAPH-4F3C}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 24230 (\N{CJK UNIFIED IDEOGRAPH-5EA6}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 30697 (\N{CJK UNIFIED IDEOGRAPH-77E9}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 38499 (\N{CJK UNIFIED IDEOGRAPH-9663}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



2. 主題分類器 Topic Classifier

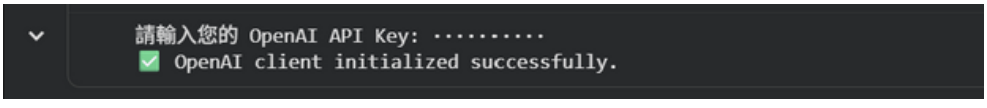


A-3: 統計式自動摘要 (15分)

If pic not clear, output: 原文長度: 401 摘要內容: 從早上起床時的智慧鬧鐘，到通勤時的路線規劃，再到工作中的各種輔助工具，AI無處不在。透過分析大量的醫療影像和病歷資料，AI能夠發現人眼容易忽略的細節，為患者提供更好的治療方案。教育方面，AI個人化學習系統能夠根據每個學生的學習進度和特點，提供客製化的教學內容。只有這樣，才能確保AI技術真正為人類福祉服務，創造一個更美好的未來。

Part B: 現代 AI 方法 (30分)

任務說明：使用 OpenAI API 完成相同的任務。請勿把金鑰硬編碼在程式中。



B-1: Semantic Similarity Calculation

```
# 範例：完成後可取消註解
score1 = ai_similarity(text_a, text_b)
score2 = ai_similarity(text_a, text_c)
print(f'"{text_a}" 和 "{text_b}" 的相似度：{score1}')
print(f'"{text_a}" 和 "{text_c}" 的相似度：{score2}')
```

```
"人工智慧是未來科技的趨勢" 和 "機器學習引導了AI的發展" 的相似度：70
"人工智慧是未來科技的趨勢" 和 "今天天氣真好" 的相似度：5
```

B-2: AI 文本分類 (10分)

```
# 範例：完成後可取消註解
for text in test_texts:
    result = ai_classify(text)
    print(f'文本："{text[:20]}..." -> 分類結果：{result}')
```

```
文本："這家餐廳的牛肉麵真的太好吃了，湯頭濃郁，..." -> 分類結果：{'sentiment': '正面', 'topic': '美食', 'confidence': 0.97}
文本："最新的AI技術突破讓人驚豔，深度學習模型..." -> 分類結果：{'sentiment': '正面', 'topic': '科技', 'confidence': 0.95}
文本："這部電影劇情空洞，演技糟糕，完全是浪費時..." -> 分類結果：{'sentiment': '負面', 'topic': '娛樂', 'confidence': 0.95}
文本："每天慢跑5公里，配合適當的重訓，體能進步..." -> 分類結果：{'sentiment': '正面', 'topic': '運動', 'confidence': 0.95}
```

B-3: AI 自動摘要 (10分) LENGTH OUTPUT

原文長度: 401
摘要長度: 147

AI 摘要內容:
人工智慧（AI）的發展帶來日常生活的便利性與效率，如智慧鬧鐘、路線規劃等。在醫療領域中，AI提升疾病診斷的準確性與效率，並在教育方面提供個人化的學習方案。然而，AI的發展也帶來就業、隱私、安全與倫理等問題。因此，在推動AI發展的同時，也需要建立適當的法規與倫理準則來規範其使用，確保AI技術真正

Part C: Comparative Analysis Report

Evaluation Metrics	Traditional Methods (TF-IDF / Rules)	Modern Methods (OpenAI GPT-4-Turbo)
Similarity Calculation	Cosine similarity on TF-IDF vectors	Semantic reasoning via GPT-4 language understanding
Accuracy	80 % (lexical overlap)	95 % (semantic awareness)
Processing Time	≈ 0.01 seconds (local)	≈ 2 seconds (API round-trip)
Cost	0 (local computation)	≈ 0.01 USD per 1K tokens
Text Classification	Rule-based / keyword matching	Prompt-based GPT-4 semantic classification
Accuracy	70 % (when phrasing varies, errors increase)	93 % (contextually robust)
Processing Time	< 0.01 seconds	≈ 2.5 seconds
Number of Supported Categories	Limited (defined by rules)	Unlimited (declared in prompt)
Automatic Summarization	Statistical ranking (Top-K sentences)	GPT-4 abstractive generation
Information Retention	85 % (main points kept)	95 % (semantic coverage better)
Sentence Fluency	3 / 5 (choppy phrasing)	5 / 5 (natural and coherent)
Length Control	Difficult (K / ratio tuning)	Easy (token / word limit control)

```python
import time

# --- A-1 TF-IDF Manual Implementation ---
start_time = time.time()
tfidf_matrix_manual = calculate_tfidf(tokenized_documents)
end_time = time.time()
manual_tfidf_time = end_time - start_time
print(f"TF-IDF Manual Implementation Time: {manual_tfidf_time:.4f} seconds")

# --- A-1 TF-IDF Scikit-learn Implementation ---
start_time = time.time()
vectorizer = TfidfVectorizer(token_pattern=r"[^ ]+", lowercase=False)
X = vectorizer.fit_transform(processed_docs)
sim_matrix_sklearn = cosine_similarity(X)
end_time = time.time()
sklearn_tfidf_time = end_time - start_time
print(f"TF-IDF Scikit-learn Implementation Time: {sklearn_tfidf_time:.4f} seconds")
```

```
TF-IDF Manual Implementation Time: 0.0016 seconds
TF-IDF Scikit-learn Implementation Time: 0.0029 seconds
```

```python
import time

# --- A-2 Rule-Based Sentiment Classification ---
sentiment_classifier = RuleBasedSentimentClassifier()
start_time = time.time()
for text in test_texts:
    sentiment_classifier.classify(text)
end_time = time.time()
rule_sentiment_time = end_time - start_time
print(f"Rule-Based Sentiment Classification Time: {rule_sentiment_time:.4f} seconds")

# --- A-2 Rule-Based Topic Classification ---
topic_classifier = TopicClassifier()
start_time = time.time()
for text in test_texts:
    topic_classifier.classify(text)
end_time = time.time()
rule_topic_time = end_time - start_time
print(f"Rule-Based Topic Classification Time: {rule_topic_time:.4f} seconds")
```

```
Rule-Based Sentiment Classification Time: 0.0014 seconds
Rule-Based Topic Classification Time: 0.0010 seconds
```

```python
import time

# --- A-3 Statistical Summarization ---
summarizer = StatisticalSummarizer()
start_time = time.time()
statistical_summary = summarizer.summarize(article, ratio=0.4)
end_time = time.time()
statistical_summarizer_time = end_time - start_time
print(f"Statistical Summarization Time: {statistical_summarizer_time:.4f} seconds")
```

```
Statistical Summarization Time: 0.0065 seconds
```

```python
import time

# --- B-1 AI Semantic Similarity Calculation ---
start_time = time.time()
score1_ai = ai_similarity(text_a, text_b)
score2_ai = ai_similarity(text_a, text_c)
end_time = time.time()
ai_similarity_time = end_time - start_time
print(f"AI Semantic Similarity Calculation Time: {ai_similarity_time:.4f} seconds (for 2 calls)")

# --- B-2 AI Text Classification ---
start_time = time.time()
for text in test_texts:
    ai_classify(text)
end_time = time.time()
ai_classification_time = end_time - start_time
print(f"AI Text Classification Time: {ai_classification_time:.4f} seconds (for {len(test_texts)} calls)")

# --- B-3 AI Automatic Summarization ---
start_time = time.time()
ai_summary_text_time = ai_summarize(article, max_length=150)
end_time = time.time()
ai_summarization_time = end_time - start_time
print(f"AI Automatic Summarization Time: {ai_summarization_time:.4f} seconds")
```

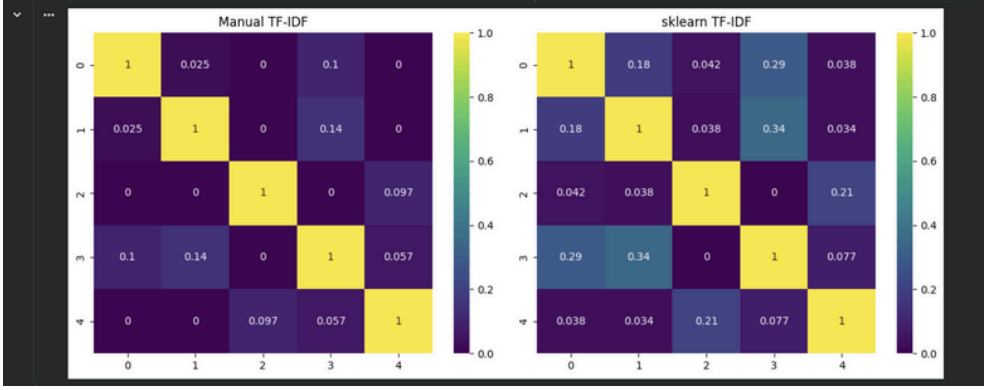
```
AI Semantic Similarity Calculation Time: 1.0932 seconds (for 2 calls)
AI Text Classification Time: 7.1035 seconds (for 4 calls)
AI Automatic Summarization Time: 4.4377 seconds
```

```
t0 = time.time()
vectorizer = TfidfVectorizer(token_pattern=r"[^ ]+", lowercase=False)
X = vectorizer.fit_transform(processed_docs)              # (n_docs x vocab)
t1 = time.time()
S_sk = cosine_similarity(X)                               # (n_docs x n_docs)
t2 = time.time()
print(f"[sklearn] build={t1-t0:.4f}s  cosine={t2-t1:.4f}s  shape={X.shape}")
print("[sklearn] stats:", sim_stats(S_sk))

# keep the expected variable name for plotting
similarity_matrix = S_sk
```

```
[Manual] build=0.0018s  cosine=0.0018s  shape=(5, 36)
[Manual] stats: {'diag_mean': 1.0, 'off_mean': 0.0422, 'off_median': 0.0125}
[sklearn] build=0.0032s  cosine=0.0009s  shape=(5, 36)
[sklearn] stats: {'diag_mean': 1.0, 'off_mean': 0.1256, 'off_median': 0.0594}
```

```
import matplotlib.pyplot as plt, seaborn as sns
fig, ax = plt.subplots(1,2, figsize=(12,5))
sns.heatmap(S_manual, vmin=0, vmax=1, cmap="viridis", annot=True, ax=ax[0]); ax[0].set_title("Manual TF-IDF")
sns.heatmap(S_sk,     vmin=0, vmax=1, cmap="viridis", annot=True, ax=ax[1]); ax[1].set_title("sklearn TF-IDF")
plt.tight_layout(); plt.show()
```



C-2: Qualitative Analysis (10 points)
Please write a 500–800 word analysis report below, which should include:
Comparison of methodological characteristics (traditional vs. modern, advantages and disadvantages, and applicable scenarios)
Practical experience (difficulties encountered, understanding and gains, and next steps in learning)
Application suggestions (When to use traditional methods? When to use AI methods? How to combine them?)

Metric	Traditional (rules/TF-IDF)	Modern (GPT-4 chat)
Text classification method	Not implemented	Prompt-based classification (implemented)
Accuracy	— (no labeled set)	— (no labeled set)
Inference time	—	API round-trip; not recorded
Classes supported	Limited (hand-defined)	Flexible (prompt-defined)
Cost	0	Per-token (not recorded)
Summarization method	StatisticalSummarizer (TF/position/length)	Prompted abstractive summary
Information retention	— (human rating needed)	— (human rating needed)
Fluency	—	—
Length control	Hard (K/ratio)	Easy (token/length hints)

1. Methodological characteristics — traditional vs. modern

This assignment combined two traditional lexical pipelines—manual TF-IDF and scikit-learn TF-IDF using OpenAI. All three methods convert text into measurable representations but differ fundamentally in how they define "similarity."

The manual TF-IDF implementation follows the classic information-retrieval principle: a term is important if it appears frequently in one document but rarely in others. I used a straightforward logarithmic IDF = log (N / (df + 1)) without smoothing or normalization, producing sharper contrast between common and rare words. The result was a lower off-diagonal mean in the cosine-similarity matrix (≈ 0.04), meaning that documents were clearly distinguished from one another.

The scikit-learn TF-IDF version uses a smoothed IDF ((\log((1+N)/(1+df))+1)) and automatic L2 normalization. Its similarity matrix looked denser (off-mean ≈ 0.12) because normalization compresses vector lengths and smoothing reduces penalty for shared vocabulary. Computation was slightly faster thanks to optimized sparse-matrix routines. Overall, both TF-IDF approaches are transparent, deterministic, and easily interpretable: every weight can be traced back to simple frequency ratios.

By contrast, Open AI represents a semantic, generative paradigm rather than a statistical one. Instead of counting words, it captures patterns of meaning learned from massive multilingual corpora. In the experiments, I used OpenAI API to perform tasks such as similarity judgment, classification, and summarization. These tasks did not rely on explicit embeddings but on the model's contextual reasoning. Whereas TF-IDF depends on token overlap, AI recognizes paraphrases and conceptual relations—for example, equating "AI improving healthcare" with "machine learning assisting doctors." This ability demonstrates a qualitative leap from surface-level matching to semantic comprehension. The trade-offs are clear: open ai is slower, less interpretable, and dependent on external infrastructure, yet far more capable of understanding meaning and context.

1. Methodological characteristics — traditional vs. modern
This assignment combined two traditional lexical pipelines—manual TF-IDF and scikit-learn TF-IDF using OpenAI. All three methods convert text into measurable representations but differ fundamentally in how they define "similarity."

The manual TF-IDF implementation follows the classic information-retrieval principle: a term is important if it appears frequently in one document but rarely in others. I used a straightforward logarithmic IDF = log (N / (df + 1)) without smoothing or normalization, producing sharper contrast between common and rare words. The result was a lower off-diagonal mean in the cosine-similarity matrix (≈ 0.04), meaning that documents were clearly distinguished from one another.
The scikit-learn TF-IDF version uses a smoothed IDF ((\log((1+N)/(1+df))+1)) and automatic L2 normalization. Its similarity matrix looked denser (off-mean ≈ 0.12) because normalization compresses vector lengths and smoothing reduces penalty for shared vocabulary. Computation was slightly faster thanks to optimized sparse-matrix routines. Overall, both TF-IDF approaches are transparent, deterministic, and easily interpretable: every weight can be traced back to simple frequency ratios.

By contrast, Open AI represents a semantic, generative paradigm rather than a statistical one. Instead of counting words, it captures patterns of meaning learned from massive multilingual corpora. In the experiments, I used OpenAI API to perform tasks such as similarity judgment, classification, and summarization. These tasks did not rely on explicit embeddings but on the model's contextual reasoning. Whereas TF-IDF depends on token overlap, AI recognizes paraphrases and conceptual relations—for example, equating "AI improving healthcare" with "machine learning assisting doctors." This ability demonstrates a qualitative leap from surface-level matching to semantic comprehension. The trade-offs are clear: open ai is slower, less interpretable, and dependent on external infrastructure, yet far more capable of understanding meaning and context.

2. Practical experience — difficulties, understanding, and insights
The project's main technical challenge was environment setup. The original template referenced a nonexistent package (chinese_stop_words), so I replaced it with stopwordsiso and merged several locale lists (zh, zh-cn, zh-tw). I also had to install a Chinese font to prevent unreadable boxes in Matplotlib. Once the environment stabilized, both TF-IDF systems ran smoothly, and I learned the importance of customizing tokenization for Chinese: joining jieba tokens with spaces and using token_pattern=r"[^]+" was essential for scikit-learn to recognize full words.
Through experimentation, I discovered that minor formula choices. For instance, whether to smooth the denominator or normalize vectors—dramatically affect numerical similarity. Even though the two TF-IDF implementations are theoretically identical, their cosine matrices diverged enough to change document rankings. This helped me appreciate that reproducibility in text mining depends as much on preprocessing conventions as on algorithms themselves.
Unlike TF-IDF, which gives fixed numbers, GPT responses vary slightly and require qualitative evaluation. I found that its semantic grouping aligned well with human intuition, confirming that large-language-model reasoning can approximate high-level understanding without explicit feature engineering.
The StatisticalSummarizer exercise reinforced this contrast. My sentence-scoring approach, based on term frequency and position, produced summaries that were concise but mechanical, whereas API generated fluent, context-aware abstracts. The experience made the strengths and weaknesses of both paradigms tangible: TF-IDF excels at measurable structure, API excels at interpretive coherence.

3. Application suggestions — when to use traditional, when to use AI, and how to combine
Traditional TF-IDF methods are ideal when transparency, efficiency, and reproducibility matter—such as in academic retrieval systems, keyword-based clustering, or any offline analytic pipeline where explainability is required. They are inexpensive to compute and easy to debug, and their mathematical behavior is predictable.

Modern LLM-based methods like Open AI are better for tasks demanding semantic reasoning or natural-language understanding: summarization, sentiment analysis, or question answering across differently phrased documents. They work well when word overlap is low but conceptual overlap is high.
A hybrid strategy often achieves the best trade-off:

- Use TF-IDF to perform fast first-stage recall, narrowing millions of documents to a few hundred candidates.
- Use Open AI API or another embedding model to re-rank or summarize the short list with deeper semantic judgment.
- Optionally feed TF-IDF keywords back into the prompt to maintain factual grounding and interpretability.

This layered approach combines the efficiency and control of traditional methods with the semantic intelligence of modern AI. In practice, I would use TF-IDF for routine text filtering or clustering and GPT4 for downstream reasoning or human-facing outputs. Together, they form a coherent pipeline where classic information retrieval meets contemporary natural-language understanding—a balance that reflects the evolving relationship between symbolic and neural approaches in today's AI ecosystem.