



دانشگاه صنعتی خواجه نصیرالدین طوسی

گزارشکار پروژه طراحی الگوریتم

فاز 1

زهرا آقایی 40214923

یگانه ظفرزاده 40219303

۱. تعریف مسئله

عنوان پروژه: طراحی یک موتور خلاصه‌سازی ترکیبی متن با استفاده از الگوریتم‌های کلاسیک و مدل زبانی (LLM)

هدف این پروژه طراحی و تحلیل یک «موتور خلاصه‌سازی ترکیبی» برای متن‌های متون است که با ترکیب یک روش کلاسیک خلاصه‌سازی الگوریتمی و یک مدل زبانی بزرگ (LLM)، خلاصه‌ای با کیفیت مناسب تولید کند. در این سیستم، بخش الگوریتمی وظیفه استخراج اطلاعات کلیدی را بر عهده دارد و بخش مبتنی بر LLM وظیفه تولید خلاصه‌ای روانتر و بازنویسی شده را انجام می‌دهد.

تعریف رسمی مسئله:

یک متن ورودی (T) داده شده است که از (n) جمله تشکیل شده است:

$$T = \{S_1, S_2, \dots, S_n\}$$

هدف، تولید یک خلاصه (S) با طول محدود است؛ به طوری که خلاصه بتواند مفاهیم مرکزی متن را پوشش دهد، از تکرار اطلاعات جلوگیری کند و تا حد امکان انسجام معنایی و خوانایی مناسبی داشته باشد. طول خلاصه می‌تواند به صورت یکی از حالت‌های زیر تعیین شود:

- انتخاب (k) جمله از متن (خلاصه‌ی جمله‌محور)، یا
- محدودیت درصدی از طول متن (مثلاً ۲۰٪ متن).

خروجی نهایی سیستم از سه جزء تشکیل می‌شود:

۱. **خلاصه الگوریتمی کلاسیک (Extractive):** انتخاب مجموعه‌ای از جمله‌های مهم از متن اصلی (بدون بازنویسی).

۲. **خلاصه مبتنی بر LLM (Abstractive):** تولید خلاصه بازنویسی شده و مفهومی توسط مدل زبانی.

۳. **خلاصه نهایی ادغام شده:** خروجی مرحله‌ای که در آن دو خلاصه فوق با یک سیاست ادغام/مقایسه ترکیب شده و خلاصه نهایی تولید می‌شود (طراحی دقیق ادغام در ادامه پروژه تکمیل خواهد شد).

ورودی سیستم یک متن خام است و خروجی آن یک خلاصه کوتاه‌شده از همان متن می‌باشد که قابلیت ارائه و ارزیابی دارد.

۲. تحلیل مسئله

مسئله‌ی خلاصه‌سازی متن از دیدگاه طراحی الگوریتم، قابل مدل‌سازی به صورت یک مسئله روی رشته‌ها و در بخش الگوریتم کلاسیک به صورت یک مسئله گراف محور است. برای پیاده‌سازی بخش کلاسیک، از الگوریتم **TextRank** استفاده می‌شود که در آن جمله‌ها به شکل گره‌های یک گراف در نظر گرفته می‌شوند و یال‌ها میزان شباهت بین جمله‌ها را نمایش می‌دهند.

۲.۱ دسته‌بندی مسئله از دید الگوریتمی

- پردازش متن / رشته‌ها (**String Processing**): چون ورودی، متن و جمله‌ها هستند و پردازش روی توکن‌ها و جملات انجام می‌شود.
- گراف (**Graph-based**): **TextRank** در یک گراف وزن‌دار بین جمله‌ها ساخته می‌شود. هر جمله یک رأس است و وزن یال‌ها برابر میزان شباهت دو جمله است. سپس یک الگوریتم رتبه‌بندی شبیه **PageRank** روی این گراف اجرا می‌شود تا اهمیت هر جمله تعیین گردد.
- الگوریتم‌های رتبه‌بندی و انتخاب: بسیاری از خلاصه‌سازهای کلاسیک، مسئله را به شکل «امتیازدهی به جملات و انتخاب Top-k» می‌بینند (Frequency-). (based / Greedy)
- بهینه‌سازی تحت محدودیت (**Constrained Selection**): خروجی خلاصه باید کوتاه و محدود باشد (مثلًا k جمله یا حداقل طول مشخص)، پس انتخاب جملات نوعی انتخاب بهینه تحت قیود طول است.
- سیستم **Hybrid** و ادغام نتایج: بخش کلیدی مسیر، طراحی الگوریتمی است که خلاصه کلاسیک و خلاصه LLM را ادغام/مقایسه/رتبه‌بندی کند تا خلاصه نهایی ساخته شود.

۲.۲ مدل گرافی در **TextRank**

فرض می‌کنیم (n) جمله داریم. یک گراف ($G = (V, E)$) تعریف می‌کنیم که:

- $V = \{s_1, s_2, \dots, s_n\}$
- برای هر دو جمله (s_i) و (s_j)، اگر شباهت آن‌ها مثبت باشد، یک یال وزن‌دار با وزن (w_{ij}) در نظر گرفته می‌شود.

- معیار شباخت می‌تواند مبتنی بر اشتراک واژه‌ها، TF-IDF و شباخت کسینوسی و روش‌های مشابه باشد (در فاز اول هدف اصلی طراحی و تحلیل است و جزئیات دقیق معیار شباخت می‌تواند در فاز دوم نهایی شود).

ایده اصلی TextRank این است که:

<جمله‌ای مهم‌تر است که به جمله‌های مهم بیشتری شباخت داشته باشد><بنابراین، با اجرای یک فرآیند تکرارشونده رتبه‌بندی، به هر جمله یک امتیاز داده می‌شود و جمله‌های با امتیاز بالاتر به عنوان خلاصه انتخاب می‌شوند.

2.3 نقش LLM در مسئله

در این پروژه، LLM به عنوان یک مؤلفه کمکی (Oracle-like Component) در نظر گرفته می‌شود که صرفاً یک خلاصه‌ی دوم از متن تولید می‌کند LLM. جایگزین الگوریتم کلاسیک نیست و در بخش تصمیم‌گیری الگوریتمی (مثل محاسبه شباخت، رتبه‌بندی و انتخاب جمله‌ها) دخالتی ندارد. هدف از استفاده LLM تولید خلاصه‌ای **Abstractive** است که از نظر خوانایی و روانی بهتر باشد و بتوان آن را با خلاصه‌ی استخراجی TextRank مقایسه و سپس ادغام کرد.

2.4 سختی محاسباتی و ویژگی‌های مسئله

- این مسئله در حالت کلاسیک (TextRank) عموماً **NP-Hard** محسوب نمی‌شود و یک فرآیند رتبه‌بندی روی گراف است.
- مهم‌ترین چالش الگوریتمی، هزینه محاسبه شباخت برای همه جفت جمله‌ها و اجرای تکرارشونده رتبه‌بندی است.
- خروجی LLM می‌تواند غیرقطعی باشد (ممکن است در اجراهای مختلف کمی تفاوت داشته باشد)، بنابراین وجود بخش الگوریتمی کلاسیک باعث ایجاد یک پایه‌ی پایدار و قابل تحلیل در سیستم می‌شود.

2.5 معیارهای ارزیابی

برای اینکه مسئله «رسمی و قابل سنجش» باشد، باید معیار تعریف کنیم. معیارها برای فاز اول

- طول خلاصه: دقیقاً k جمله یا حداقل L کلمه.
- پوشش محتوایی (Coverage): درصد پوشش کلمات کلیدی یا مفاهیم مهم متن.
- عدم افزونگی (Non-Redundancy): شباخت بین جملات انتخابی از یک آستانه بیشتر نشود.

- وفاداری به متن (Faithfulness) : خلاصه نهایی با متن ناسازگار نباشد (برای LLM حساس‌تر است).
- روانی/خوانایی (Readability) : (می‌تواند کیفی یا امتیازدهی ساده باشد).

3. فرضیات و محدودیتها

در این پروژه، فرضیات و محدودیت‌های زیر در نظر گرفته می‌شود:

- متن ورودی به صورت متن ساده (**Plain Text**) ارائه می‌شود.
- متن ورودی دارای جمله‌بندی صحیح بوده و امکان تفکیک آن به جمله‌ها وجود دارد.
- طول خلاصه از پیش مشخص است و می‌تواند به صورت:
 - تعداد ثابتی از جمله‌ها، یا
 - درصدی از طول متن ورودی تعیین شود.
- خلاصه‌سازی الگوریتمی به صورت **Extractive** انجام می‌شود؛ یعنی خروجی آن شامل انتخاب جمله‌ها از متن اصلی بدون بازنویسی است.
- خلاصه‌سازی مبتنی بر مدل زبانی (LLM) به صورت **Abstractive** انجام می‌شود؛ یعنی خلاصه‌ی تولیدشده می‌تواند بازنویسی مفهومی متن باشد.
- کیفیت خروجی LLM ممکن است دارای عدم قطعیت یا خطای معنایی باشد و کاملاً قطعی نیست.
- سیستم برای متون بسیار کوتاه (کمتر از چند جمله) بهینه نشده است.

4. طراحی الگوریتم

طبق داک پروژه مسیر *Algorithmic Summarization Engine*، باید حداقل یک الگوریتم کلاسیک داشته باشیم و بهتر است چند روش را مقایسه کنیم. سه روش کلاسیک که می‌توان برای خلاصه‌سازی الگوریتمی استفاده کرد عبارت‌اند از:

- **Frequency-based Summarization** مبتنی بر فراوانی کلمات
- **Sentence Ranking** رتبه‌بندی ساده جمله‌ها

• **TextRank (Graph-based) الگوریتم اصلی انتخاب شده**

در ادامه ابتدا دو الگوریتم فرعی را خیلی خلاصه معرفی می‌کنم و سپس TextRank را مفصل توضیح می‌دهم.

4.1 دو الگوریتم فرعی (برای مقایسه)

Frequency-based Summarization

ایده: کلماتی که در متن زیاد تکرار می‌شوند (غیر از کلمات خیلی عمومی مثل «است»، «و»، ...) معمولاً حامل موضوع اصلی‌اند. جمله‌هایی که تعداد بیشتری از این کلمات مهم را دارند امتیاز بالاتری می‌گیرند و برای خلاصه انتخاب می‌شوند.

- مزیت: ساده و سریع
- ضعف: ارتباط بین جمله‌ها و ساختار کلی متن را مدل نمی‌کند (ممکن است جمله‌های پراکنده انتخاب کند)

Sentence Ranking (Rule-based/Feature-based)

ایده: هر جمله بر اساس چند ویژگی ساده امتیاز می‌گیرد (مثلاً طول جمله، وجود کلمات کلیدی، موقعیت جمله در متن، تعداد کلمات مهم). سپس جمله‌های با امتیاز بیشتر انتخاب می‌شوند.

- مزیت: ساده و قابل کنترل
- ضعف: شدیداً به قواعد دستی وابسته است و «مرکزیت مفهومی» جمله‌ها را مثل روش گرافی نمی‌سنجد

4.2 (الگوریتم اصلی) : TextRank

4.2.1) ایده‌ی اصلی:

یک الگوریتم گراف محور برای خلاصه‌سازی Extractive است. Extractive یعنی خلاصه از خود جمله‌های متن اصلی انتخاب می‌شود (بازنویسی نمی‌شود).

ایده‌ی مرکزی TextRank این جمله است:

یک جمله زمانی مهم است که به جمله‌های مهم دیگر شبیه باشد.

این دقیقاً همان ایده‌ی PageRank در وب است:

- در وب: صفحه مهم است اگر صفحات مهم به آن لینک دهند
- در TextRank جمله مهم است اگر جمله‌های مهم به آن "شباهت/ارتباط" داشته باشند

4.2.2) گراف جمله‌ها یعنی چی؟

برای یک متن با (n) جمله:

- هر جمله (s_i) یک گره (Node) است.
- بین هر دو جمله (s_i) و (s_j) یک یال وزن‌دار می‌سازیم که وزن آن (w_{ij}) نشان‌دهنده شباهت بین جمله‌هاست.

پس گراف ما تقریباً یک گراف «کامل» می‌شود (چون هر جمله با همه جمله‌ها مقایسه می‌شود). اگر بخواهیم هزینه را کمتر کنیم، می‌توانیم فقط یال‌هایی را نگه داریم که شباهتشان از یک آستانه بیشتر باشد (در فاز دوم به عنوان بهینه‌سازی).

4.2.3) شباهت جمله‌ها چطور حساب می‌شود؟

TextRank خودش به ما نمی‌گوید similarity را دقیقاً چطور بسازیم، فقط می‌گوید باید وزن یال‌ها «شباهت» باشد.

در پروژه‌های کلاسیک معمولاً از یکی از این‌ها استفاده می‌شود:

- همپوشانی واژه‌ها (Overlap)
- TF-IDF + Cosine Similarity

4.2.4) چرا TextRank نیاز به اجرای تکرارشونده دارد؟

امتیاز جمله‌ها به هم وابسته‌اند:

- اگر جمله A به جمله B شبیه باشد، امتیاز A به امتیاز B ربط پیدا می‌کند

- و امتیاز B هم ممکن است به امتیاز A و بقیه ربط داشته باشد این وابستگی حلقه‌ای باعث می‌شود نتوانیم امتیازها را «یک بار برای همیشه» حساب کنیم. پس امتیازها را چند بار به روزرسانی می‌کنیم تا به یک حالت پایدار برسند (همگرایی).

4.2.5) امتیازدهی TextRank دقیقاً چه منطقی دارد؟

در هر مرحله، امتیاز یک جمله (s_i) از دو بخش تشکیل می‌شود:

- یک مقدار پایه (برای اینکه هیچ جمله‌ای صفر نشود)
 - مجموع سهم جمله‌های دیگر که به آن شبیه‌اند
- و سهم هر جمله‌ی دیگر (s_j) به (s_i) این‌طور محاسبه می‌شود:
- اگر (s_j) جمله مهمی باشد \Rightarrow سهمش بیشتر
 - اگر شباخت (s_j) به (s_i) زیاد باشد \Rightarrow سهمش بیشتر
 - اگر (s_j) به خیلی جمله‌ها شبیه باشد \Rightarrow سهمش بین همه تقسیم می‌شود (یعنی سهم هرکدام کمتر می‌شود)

این دقیقاً منصفانه است:

یک جمله‌ی مهم "اعتبارش" را بین همسایه‌هایش پخش می‌کند، نه اینکه همه را یکسان بالا ببرد.

4.2.6) خروجی TextRank چگونه ساخته می‌شود؟

بعد از اینکه امتیازها همگرا شدند:

- جمله‌ها را بر اساس امتیاز مرتب می‌کنیم
- جمله را انتخاب می‌کنیم Top-k

5) (سودوکد) TextRank

- Input: Text T, summary size k, damping d, max_iter, eps

2. sentences \leftarrow Split T into sentences
3. n \leftarrow number of sentences

4. Create empty weighted graph G
5. for i = 1 to n do
6. Add node i to G
7. end for

8. for i = 1 to n do
9. for j = 1 to n do
10. if i \neq j then
11. w \leftarrow Similarity(sentences[i], sentences[j])
12. if w > 0 then
13. Add edge (i \rightarrow j) with weight w to G
14. end if
15. end if
16. end for
17. end for

18. for i = 1 to n do
19. score[i] \leftarrow 1 / n
20. end for

```

21. for iter = 1 to max_iter do
22.   max_change ← 0
23.   for i = 1 to n do
24.     new_score[i] ← 1 - d
25.     for each j such that ( $j \rightarrow i$ ) exists in G do
26.       new_score[i] ← new_score[i] +
27.          $d \times (\text{weight}(j \rightarrow i) / \text{sum\_outgoing\_weights}(j)) \times \text{score}[j]$ 
28.     end for
29.     max_change ← max(max_change, |new_score[i] - score[i]|)
30.   end for
31.   score ← new_score
32.   if max_change < eps then break
33. end for

34. Sort sentences by score in descending order
35. Select top k sentences as summary
36. Output selected sentences

```

5.1) توضیح سودوکد

خطوط ۱ تا ۳ — ورودی‌ها و جمله‌بندی

خط ۱: ورودی‌های الگوریتم را می‌گیریم: متن، اندازه خلاصه، پارامترها

- خط 2: متن را به جمله‌ها تبدیل می‌کنیم
- خط 3: تعداد جمله‌ها را به دست می‌آوریم
- خطوط 4 تا 7 — ساخت گراف و اضافه کردن گره‌ها
 - خط 4: یک گراف وزن دار خالی می‌سازیم
 - خط 5-7: برای هر جمله یک گره در گراف اضافه می‌کنیم
- خطوط 8 تا 17 — محاسبه شباهت‌ها و ساخت یال‌ها
 - خط 8-9: همه جفت‌های جمله‌ها را بررسی می‌کنیم دلیل $O(n^2)$
 - خط 10: جمله با خودش مقایسه نمی‌شود
 - خط 11: شباهت دو جمله حساب می‌شود
 - خط 12-14: اگر شباهت مثبت بود، یال وزن دار اضافه می‌کنیم
- خطوط 18 تا 20 — مقداردهی اولیه امتیازها
 - ابتدا همه جمله‌ها امتیاز یکسان می‌گیرند (نقطه شروع)
- خطوط 21 تا 32 — بخش تکرارشونده (TextRank هسته)
 - خط 21: الگوریتم را تا حد اکثر max_iter تکرار می‌کنیم
 - خط 22: max_change را صفر می‌کنیم تا تغییرات را بسنجیم
 - خط 23-29: برای هر جمله، امتیاز جدید حساب می‌شود
 - خط 24: مقدار پایه $(1-d)$ را قرار می‌دهیم
 - خط 25: روی جمله‌هایی می‌رویم که به جمله i وصل‌اند (همسایه‌ها)
 - خط 26: سهم هر همسایه j را اضافه می‌کنیم:
 - اگر j امتیاز بالا دارد \Rightarrow اثر بیشتر
 - اگر وزن شباهت $i \rightarrow j$ بالا است \Rightarrow اثر بیشتر
 - اگر j به خیلی‌ها وصل است \Rightarrow سهم تقسیم می‌شود

- خط 28: تغییر امتیاز را ثبت می‌کنیم
 - خط 30: بعد از محاسبه همه، امتیازها را آپدیت می‌کنیم
 - خط 31: اگر تغییرات خیلی کم شد (همگرایی) حلقه را می‌شکنیم
- خطوط 33 تا 35 – انتخاب خلاصه**
- جمله‌ها را بر اساس امتیاز مرتب می‌کنیم و top-k را خروجی می‌دهیم

6. تحلیل زمان و فضای

در این بخش، پیچیدگی زمانی و فضایی الگوریتم خلاصه‌سازی الگوریتمی مبتنی بر **TextRank** تحلیل می‌شود.

فرض می‌کنیم:

- n = تعداد جمله‌های متن ورودی
- $iter$ = تعداد تکرارهای الگوریتم TextRank تا همگرایی

1. پیش‌پردازش و جمله‌بندی متن

عملیات:

- تقسیم متن به جمله‌ها
- حذف علائم نگارشی و توقف‌واژه‌ها (در صورت استفاده)

پیچیدگی زمانی:
 $O(n)$

پیچیدگی فضایی:
 $O(n)$

2. ساخت گراف شباهت جمله‌ها

در این مرحله، برای هر جفت جمله، میزان شباهت محاسبه می‌شود.

عملیات:

- مقایسه همه جفت‌های جمله‌ها

- محاسبه وزن یال‌ها

پیچیدگی زمانی:

$$O(n^2)$$

پیچیدگی فضایی:

$$O(n^2)$$

زیرا گراف به صورت کامل یا نیمه کامل ذخیره می‌شود.

3. مقداردهی اولیه امتیاز جمله‌ها

به هر جمله مقدار اولیه مساوی داده می‌شود.

پیچیدگی زمانی:

$$O(n)$$

پیچیدگی فضایی:

$$O(n)$$

4. اجرای الگوریتم (Iterative)

در هر تکرار:

- امتیاز هر جمله با توجه به جمله‌های مرتبط به آن محاسبه می‌شود

- این محاسبه برای همه جمله‌ها انجام می‌شود

پیچیدگی زمانی هر تکرار:

$$O(n^2)$$

تعداد تکرارها:

• معمولاً ثابت و کوچک (مثلاً 20 تا 30)

پیچیدگی زمانی کل:

$$O(\text{iter} * n^2)$$

پیچیدگی فضایی:

$$O(n)$$

فقط بردار امتیازها نگهداری می‌شود.

5. مرتب‌سازی نهایی جمله‌ها

پس از محاسبه امتیاز نهایی، جمله‌ها بر اساس امتیاز مرتب می‌شوند.

پیچیدگی زمانی:

$$O(n \log n)$$

پیچیدگی فضایی:

$$O(n)$$

پیچیدگی کلی الگوریتم

پیچیدگی زمانی نهایی:

$$O(\text{iter} * n^2)$$

زیرا جمله‌ی غالب مربوط به ساخت گراف و اجرای TextRank است.

پیچیدگی فضایی نهایی:

$$O(n^2)$$

به دلیل ذخیره گراف شباهت جمله‌ها.

تحلیل بهترین، بدترین و میانگین حالت

• بهترین حالت:

متن کوتاه با تعداد جمله کم \Rightarrow اجرای سریع

• بدترین حالت:

متن طولانی با تعداد جمله زیاد => هزینه‌ی (n^2) بالا

• حالت میانگین:

متون با اندازه متوسط => قابل اجرا در زمان مناسب

7. نقش مدل زبانی بزرگ (LLM) در سیستم

در این پژوهه، مدل زبانی بزرگ (LLM) به عنوان یک مولفه‌ی کمکی در کنار الگوریتم کلاسیک خلاصه‌سازی استفاده می‌شود و جایگزین الگوریتم اصلی نیست. هدف از به کارگیری LLM افزایش کیفیت زبانی و روانی خلاصه نهایی است، در حالی که ساختار تصمیم‌گیری الگوریتمی سیستم حفظ می‌شود.

در معماری سیستم، خلاصه‌سازی در دو مرحله مستقل انجام می‌گیرد:

1. خلاصه‌سازی الگوریتمی کلاسیک (TextRank)

این مرحله به صورت extractive عمل کرده و جمله‌های مهم متن را بر اساس تحلیل گرافی و رتبه‌بندی الگوریتمی انتخاب می‌کند.

2. خلاصه‌سازی مبتنی بر LLM

در این مرحله، LLM با دریافت متن ورودی، یک خلاصه‌ی abstractive و بازنویسی شده تولید می‌کند که از نظر زبانی روان‌تر و منسجم‌تر است.

مدل زبانی در این پژوهه نقش تصمیم‌گیرنده ندارد و در هیچ‌یک از مراحل زیر دخالت نمی‌کند:

- محاسبه شباهت بین جمله‌ها
- رتبه‌بندی یا امتیازدهی جمله‌ها
- انتخاب جمله‌ها برای خلاصه الگوریتمی

به عبارت دیگر، LLM به عنوان یک Oracle در نظر گرفته می‌شود که تنها یک خروجی زبانی تولید می‌کند و فرآیند داخلی آن در تحلیل الگوریتمی لحاظ نمی‌شود. خروجی LLM در فاز بعدی پژوهه، توسط یک الگوریتم ادغام با خلاصه الگوریتمی ترکیب خواهد شد تا خلاصه نهایی سیستم تولید شود.

به دلیل ماهیت غیرقطعی LLM، خروجی آن ممکن است در اجراهای مختلف کمی متفاوت باشد یا برخی جزئیات را حذف کند؛ ازین‌رو، وجود بخش الگوریتمی کلاسیک به عنوان پایه‌ای پایدار و قابل تحلیل برای سیستم ضروری است.

8. مثال‌های دستی

تعریف کوتاه TextRank برای مثال دستی

1. هر جمله = یک گره در گراف.
2. وزن یال بین دو جمله = شباهت (Similarity) بین آن‌ها.
3. با اجرای PageRank روی گراف، هر جمله امتیاز می‌گیرد.
4. خروجی خلاصه = انتخاب Top-k جمله با امتیاز بالاتر.

معیار شباهت برای مثال دستی

برای دو جمله S_i و S_j :

$$\frac{|jW_i \cap W|}{|jW_i \cup W|} = Sim(S_i, S_j)$$

که Wi مجموعه کلمات مهم جمله i است (بعد از حذف کلمات خیلی رایج مثل «از، به، و، را، که...»).

مثال ۱ — کیس ساده (۴ جمله، $k=2$)

متن ورودی:

S_1 : «.هوش مصنوعی در پزشکی برای تشخیص بیماری استفاده می‌شود»

S_2 : «.مدل‌های یادگیری ماشین می‌توانند الگوهای داده‌های پزشکی را پیدا کنند»

S_3 : «.تشخیص زودهنگام بیماری باعث افزایش شанс درمان می‌شود»

S_4 : «.امروز هوا بارانی است و ترافیک زیاد شده است»

هدف: تولید خلاصه با $k=2$ جمله.

مرحله ۱: استخراج کلمات کلیدی هر جمله (بعد از حذف کلمات رایج)

$W_1 = \{\text{هوش، مصنوعی، پزشکی، تشخیص، بیماری، استفاده}\}$

$W_2 = \{\text{مدل‌ها، یادگیری، ماشین، الگوها، داده‌های، پزشکی، پیدا}\}$

$W_3 = \{\text{تشخیص، زودهنگام، بیماری، افزایش، شانس، درمان}\}$

$W_4 = \{\text{هوا، بارانی، ترافیک، زیاد}\}$

مرحله ۲: محاسبه شباهت‌ها (Jaccard)

$\text{Sim}(1,2) = \{\text{پزشکی}\} \Rightarrow 1$

$$\text{اجتماع} = 0.083 \approx 12/1 \Rightarrow 12 = 1-7+6$$

$\text{Sim}(1,3) = \{\text{تشخیص، بیماری}\} \Rightarrow 2$

$$\text{اجتماع} = 0.2 = 10/2 \Rightarrow 10 = 2-6+6$$

$\text{Sim}(1,4) = \emptyset \Rightarrow 0$

$\text{Sim}(2,3) = \emptyset \text{ (در این ساده‌سازی)} \Rightarrow 0$

$\text{Sim}(2,4) = 0$

$\text{Sim}(3,4) = 0$

پس گراف عملای بین (S_1, S_2, S_3) ارتباط دارد و S_4 جداست.

مرحله ۳: نتیجه رتبه‌بندی (تحلیل شهرودی (TextRank))

در TextRank، جمله‌ای که به جملات بیشتری با وزن بالاتر وصل است، امتیاز بالاتری می‌گیرد.

S_1 به S_2 و S_3 وصل است (دو اتصال غیر صفر)

S_3 فقط به S_1 وصل است

S_2 فقط به S_1 وصل است

S_4 هیچ اتصالی ندارد

بنابراین ترتیب امتیاز (تقریبی/شهودی):

$$\text{Score}(S_1) > \text{Score}(S_3) > \text{Score}(S_2) > \text{Score}(S_4)$$

خروجی خلاصه ($k=2$)

S_1 : «هوش مصنوعی در پزشکی برای تشخیص بیماری استفاده می‌شود»

S_3 : «تشخیص زودهنگام بیماری باعث افزایش شанс درمان می‌شود»

این خلاصه هم مرتبط است هم جمله‌ی پر (S₄) حذف شده است.

مثال ۲ — کیس متوسط (۵ جمله، $k=2$) با رقابت بین جملات)

متن ورودی:

S_1 : «دانشگاهها برای ثبت‌نام، مدارک هویتی و ریزنمرات را بررسی می‌کنند»

S_2 : «بررسی مدارک باعث جلوگیری از ثبت‌نام غیرمجاز می‌شود»

S_3 : «فرآیند ثبت‌نام شامل پرداخت شهریه و انتخاب واحد است»

S_4 : «انتخاب واحد باید بر اساس پیش‌نیازها انجام شود»

S_5 : «هوای شهر امروز خیلی سرد است»

$k = 2$

کلمات کلیدی:

$W_1 = \{\text{دانشگاهها، ثبت‌نام، مدارک، هویتی، ریزنمرات، بررسی}\}$

$W_2 = \{\text{بررسی، مدارک، جلوگیری، ثبت‌نام، غیرمجاز}\}$

$W_3 = \{\text{فرآیند، ثبت‌نام، پرداخت، شهریه، انتخاب، واحد}\}$

$W_4 = \{\text{انتخاب، واحد، پیش‌نیازها}\}$

$W_5 = \{\text{هوای، شهر، امروز، سرد}\}$

شباهت‌های مهم (فقط غیر صفرها) :

$\text{Sim}(1,2)$: ثبت‌نام = ۳ اشتراک {بررسی، مدارک، ثبت‌نام}

$$\text{اجتما} = 8/3 \Rightarrow 8 = 3+5+6$$

$\text{Sim}(1,3)$: ثبت‌نام = ۱ اشتراک {ثبت‌نام}

$$\text{اجتما} = 11/1 \Rightarrow 11 = 1-6+6$$

$\text{Sim}(2,3)$: ثبت‌نام = ۱ اشتراک {ثبت‌نام}

$$\text{اجتما} = 10 = 1-6+5$$

$\text{Sim}(3,4)$: واحد = ۲ اشتراک {انتخاب، واحد}

$$\text{اجتما} = 7/2 \Rightarrow 7 = 2-3+6$$

ساختمان با S_5 صفر

تحلیل رتبه (شهودی) :

- S_2 به S_1 (وزن بالا) و S_3 وصل است \rightarrow مرکزی
- S_3 به S_1, S_2 و S_4 وصل است \rightarrow مرکزی
- S_1 به S_2 (خیلی قوی) و S_3 وصل است \rightarrow امتیاز بالا
- S_4 به S_3 وصل است \rightarrow متوسط
- S_5 جدا \rightarrow پایین

پس سه جمله اصلی $S_1/S_2/S_3$: چون $k=2$ ، معمولاً دو جمله مرکزی‌تر انتخاب می‌شوند:

$$(S_2 \& S_3) \text{ یا } (S_1 \& S_3)$$

اگر هدف خلاصه «کل فرآیند ثبت‌نام» باشد، بهترین دو جمله:

- S_1 مدارک و بررسی
- S_3 مراحل ثبت‌نام

($k=2$ خروجی خلاصه)

«دانشگاه‌ها برای ثبت‌نام، مدارک هویتی و ریزنمرات را بررسی می‌کنند»: S_1

«فرآیند ثبت‌نام شامل پرداخت شهریه و انتخاب واحد است»: S_3

جمله پرت S_5 حذف شد، و دو بخش اصلی فرآیند پوشش داده شد. ✓

مثال ۳ – Edge Case – (همه شباهت‌ها نزدیک صفر)

متن ورودی:

S_1 : «گربه روی دیوار پرید»

S_2 : «اقتصاد کشور رشد کرد»

S_3 : «برنامه‌نویسی تمرین می‌خواهد»

$k=1$

کلمات مشترک تقریباً وجود ندارد، پس گراف تقریباً بدون یال است.

نتیجه:

در این حالت TextRank نمی‌تواند "مرکزی‌ترین" را با اتصالات تشخیص دهد و معمولاً:

- یا همه امتیازها تقریباً برابر می‌شوند،
- یا الگوریتم با tie-break ساده (مثل ترتیب جمله یا طول جمله) تصمیم می‌گیرد.

خروجی نمونه: (tie-break) با "اولین جمله"

S_1 : «گربه روی دیوار پرید»

مثال ۴ – Edge Case – (جملات خیلی مشابه و افزونگی بالا)

متن ورودی:

S_1 : «دانشجو باید پروژه را در گیت‌هاب آپلود کند»

S_2 : «آپلود پروژه در گیت‌هاب الزامی است»

S_3 : «گیت‌هاب برای مدیریت نسخه‌ها کاربرد دارد»

$k=2$

S_1 و S_2 خیلی شبیه‌اند؛ TextRank ممکن است هر دو را انتخاب کند و خلاصه تکراری شود.

(چون اتصال قوی دارند) S_1 و S_2 خروجی احتمالی TextRank:

۹. طراحی تستهای ابتدایی

فرضهای تست

- الگوریتم خلاصه‌سازی کلاسیک TextRank (Extractive):
- خروجی خلاصه: انتخاب Top-k جمله بر اساس امتیاز TextRank.
- برای اینکه خروجی‌ها دقیق و قابل تصحیح باشند، در این تست‌ها tie-break را مشخص می‌کنیم:

قانون Tie-break (برای یکسان شدن امتیازها)

اگر امتیاز دو جمله برابر شد:

1. جمله‌ای انتخاب شود که زودتر در متن آمده (index) کمتر)
2. اگر باز هم برابر بود، جمله با طول بیشتر انتخاب شود.

— کیس ساده (وجود جمله پرت/نامرتبط) Test 1

هدف: بررسی حذف جمله‌ای که به بقیه وصل نیست.

Input (k=2):

«هوش مصنوعی در پزشکی برای تشخیص بیماری استفاده می‌شود»: S₁

«مدل‌های یادگیری ماشین می‌توانند الگوهای داده‌های پزشکی را پیدا کنند»: S₂

«تشخیص زودهنگام بیماری باعث افزایش شанс درمان می‌شود»: S₃

«امروز هوا بارانی است و ترافیک زیاد شده است»: S₄

Expected Output:

- «هوش مصنوعی در پزشکی برای تشخیص بیماری استفاده می‌شود» (S1)
- «تشخیص زودهنگام بیماری باعث افزایش شанс درمان می‌شود» (S3)

Reason:

S₄ ارتباط معنایی/وازنگانی با بقیه ندارد و در گراف وزن/اتصال نزدیک صفر دارد.

Test 2 — کیس متوسط (دو خوشه مرتبط در یک متن)

هدف: بررسی اینکه TextRank از «مرکز شبکه» انتخاب کند و جمله پر حذف شود.

Input (k=2):

S_1 : «دانشگاهها برای ثبت‌نام، مدارک هویتی و ریزنمرات را بررسی می‌کنند»

S_2 : «بررسی مدارک باعث جلوگیری از ثبت‌نام غیرمجاز می‌شود»

S_3 : «فرآیند ثبت‌نام شامل پرداخت شهریه و انتخاب واحد است»

S_4 : «انتخاب واحد باید بر اساس پیش‌نیازها انجام شود»

S_5 : «هوای شهر امروز خیلی سرد است»

Expected Output:

- «دانشگاهها برای ثبت‌نام، مدارک هویتی و ریزنمرات را بررسی می‌کنند» (S_1)

- «فرآیند ثبت‌نام شامل پرداخت شهریه و انتخاب واحد است» (S_3)

Reason:

S_1 و S_3 جملات مرکزی‌اند (به چند جمله دیگر وصل‌اند) S_5 . پر حذف است.

Test 3 — کیس سخت (افزونگی بالا / جملات بسیار مشابه)

هدف: سنجش مشکل تکرار در خلاصه (Redundancy) و ثبت آن به عنوان سناریوی شکست اولیه.

Input (k=2):

S_1 : «دانشجو باید پروژه را در گیت‌هاب آپلود کند»

S_2 : «آپلود پروژه در گیت‌هاب الزامی است»

S_3 : «گیت‌هاب برای مدیریت نسخه‌ها کاربرد دارد»

S_4 : «کامیت‌های مرحله‌ای روند پیشرفت پروژه را نشان می‌دهند»

TextRank)Expected Output خام:

- «.دانشجو باید پروژه را در گیت‌هاب آپلود کند» (S_1)
- «.آپلود پروژه در گیت‌هاب الزامی است» (S_2)

Reason:

S_1 و S_2 بیشترین شباهت را دارند و معمولاً امتیاز بالا می‌گیرند.

Observation: این خروجی ممکن است تکراری شود؛ در فاز دوم می‌توان یک مرحله کاهش افزونگی اضافه کرد.

Test 4 — Edge Case (شباهت تقریباً صفر بین همه جملات)

هدف: بررسی رفتار الگوریتم وقتی گراف تقریباً بدون یال است (tie-break) مهم می‌شود.

Input (k=1):

S_1 : «.گربه روی دیوار پرید»

S_2 : «.اقتصاد کشور رشد کرد»

S_3 : « برنامه‌نویسی تمرین می‌خواهد »

Expected Output:

- «.گربه روی دیوار پرید» (S_1)

Reason:

امتیازها تقریباً برابر می‌شوند؛ طبق قانون tie-break جمله اول انتخاب می‌شود.

Test 5 — Edge Case (تعداد جملات کمتر از k)

هدف: بررسی شرط مرزی و جلوگیری از خطأ.

Input (k=5):

S_1 : «.این متن فقط دو جمله دارد»

S_2 : «پس خلاصه باید همان دو جمله باشد»

Expected Output:

- «این متن فقط دو جمله دارد» (S_1)
- «پس خلاصه باید همان دو جمله باشد» (S_2)

Reason:

$n < k$ ، خروجی باید همه جملات را برگرداند و الگوریتم کرش نکند.

Test 6 – کیس متوسط (تمرکز روی جمله مرکزی با کلمات مشترک زیاد)

هدف: سنجش اینکه جمله‌ای که موضوع اصلی را جمع‌بندی می‌کند بالاتر می‌آید.

Input ($k=2$):

S_1 : «آلودگی هوا باعث افزایش بیماری‌های تنفسی می‌شود»

S_2 : «ذرات معلق یکی از عوامل اصلی آلودگی هوا هستند»

S_3 : «آلودگی هوا در زمستان بیشتر می‌شود»

S_4 : «ورزش منظم به سلامت بدن کمک می‌کند»

Expected Output:

- «آلودگی هوا باعث افزایش بیماری‌های تنفسی می‌شود» (S_1)
- «آلودگی هوا در زمستان بیشتر می‌شود» (S_3)

Reason:

S_1 و S_3 به موضوع مرکزی "آلودگی هوا" وصل‌اند؛ S_4 پرت است.