دانشگاه صنعتی خواجه نصیرالدین طوسی

گزارشکار پروژه طراحی الگوریتم

فاز2

زهرا آقایی 40214923

یگانه ظفرزاده 40219303

مقدمه و مرور فاز اول

پروژه‌ی حاضر با هدف طراحی و توسعه‌ی یک **موتور خلاصه‌سازی ترکیبی (Hybrid Summarization Engine)** تعریف شده است که توانمندی‌های الگوریتم‌های کلاسیک گراف‌محور را با قدرت پردازش مدل‌های زبانی بزرگ (LLM) تلفیق می‌کند.
ضرورت این ترکیب از آنجا ناشی می‌شود که خلاصه‌سازی استخراجی (Extractive) بر پایه الگوریتم‌های کلاسیک، دقت بالایی در حفظ کلمات کلیدی دارد اما فاقد روانی زبانی است؛ در مقابل، مدل‌های زبانی در تولید متن روان (Abstractive) استادانه عمل می‌کنند اما ممکن است دچار خطای وفاداری به متن (Hallucination) شوند.

در فاز اول این پروژه، اقدامات زیر به عنوان زیربنای سیستم انجام گرفت:

- **تحلیل و مدل‌سازی مسئله :**خلاصه‌سازی متن به عنوان یک مسئله رتبه‌بندی جملات در گراف مدل‌سازی شد.
- **انتخاب الگوریتم محوری :**الگوریتم **TextRank**به عنوان هسته‌ی اصلی خلاصه‌ساز کلاسیک انتخاب و تحلیل شد.
- **تحلیل پیچیدگی تئوری :**پیچیدگی زمانی الگوریتم در بدترین حالت O(iter * n^2) برآورد گردید.

توسعه در فاز دوم:

در فاز دوم، علاوه بر پیاده‌سازی کامل منطق‌های طراحی شده، ساختار پروژه در مخزن GitHub به صورت **کاملاً ماژولار** بازطراحی و توسعه یافته است. این ساختار(شامل تفکیک ماژول‌های extractive، abstractive، merge وeval)به گونه‌ای تنظیم شده است که با الزامات آموزشی درس و استانداردهای توسعه نرم‌افزار همخوانی کامل داشته باشد. این رویکرد ماژولار، تست‌پذیری سیستم و امکان توسعه‌ی جداگانه‌ی هر بخش را به شکلی بهینه فراهم آورده است.

توضیح بخشهای مختلف پروژه

۱ .بخش استخراجی(Extractive Summarization)

این بخش وظیفه دارد جملات کلیدی را مستقیماً از متن اصلی، بدون هیچ‌گونه تغییری انتخاب کند.

- **مدل‌سازی گرافی :**در این ماژول، متن به صورت یک گراف مدل‌سازی می‌شود که در آن هر جمله یک گره (Node) است.
- **محاسبه شباهت :**برای تعیین وزن یال‌های بین جملات، ابتدا متن با حذف کلمات توقف (Stopwords) و توکن‌گذاری نرمال‌سازی می‌شود. سپس با استفاده از روش **TF-IDF**، جملات به بردارهای عددی تبدیل شده و شباهت بین هر دو جمله با فرمول **Cosine Similarity**محاسبه می‌گردد.
- **الگوریتم رتبه‌بندی :(TextRank)** یک فرآیند تکرارپذیر (Iterative) روی گراف اجرا می‌شود. در هر تکرار، امتیاز هر جمله بر اساس امتیاز همسایگانش و وزن یال‌های متصل به آن‌ها به‌روزرسانی می‌شود. این روند تا رسیدن به همگرایی (Convergence)ادامه می‌یابد.
- **سیاست رفع تساوی :(Tie-break)** در صورتی که امتیاز دو جمله برابر باشد، سیستم به صورت خودکار جمله‌ای را که در متن زودتر آمده و یا طول بیشتری دارد اولویت‌بندی می‌کند.

۲ .بخش انتزاعی(Abstractive Summarization)

این بخش وظیفه تولید یک خلاصه روان و مفهومی را با بازنویسی متن بر عهده دارد.

- **استفاده از مدل زبانی :(LLM)** در این ماژول از مدل‌های زبانی بزرگ مانند GPT-4o-mini از طریق API شرکت Metisاستفاده شده است.
- **مهندسی پرامپت :(Prompt Engineering)** یک سیستم پیام (System Message) دقیق طراحی شده تا مدل را به سمت تولید خلاصه‌ای وفادار به متن، بدون اختراع حقایق (Hallucination) و با رعایت تعداد جملات درخواستی هدایت کند.
- **پیکربندی :(Configuration)** پارامترهایی مانند temperature برای کنترل میزان خلاقیت مدل و max_tokens برای محدود کردن طول خروجی تنظیم شده‌اند تا خروجی نهایی بهینه باشد.

۳ .موتور ادغام(Merge Engine)

جذاب‌ترین بخش پروژه، الگوریتم ادغام است که نتایج دو بخش قبلی را با هم ترکیب می‌کند تا یک خلاصه هیبرید بسازد.

- **ایجاد استخر کاندیدا :**ابتدا جملات استخراج شده توسط TextRank به عنوان «نقاط لنگر (Anchors)» قرار می‌گیرند و سپس جملات تولید شده توسط LLM به آن‌ها اضافه می‌شوند.
- **رتبه‌بندی بر اساس محوریّت :(Centrality)** سیستم مجدداً یک تحلیل شباهت روی تمام جملات کاندیدا انجام می‌دهد تا جملاتی که بیشترین ارتباط معنایی را با کل مجموعه دارند Centrality Score بالاتر شناسایی کند.
- **فیلتر حذف افزونگی :(Redundancy Filtering)** برای جلوگیری از تکرار مفاهیم مشابه، هر جمله کاندیدا با جملات قبلاً انتخاب شده مقایسه می‌شود. اگر شباهت کسینوسی آن‌ها از یک آستانه مشخص (مثلاً ۰.۷۵) بیشتر باشد، آن جمله حذف می‌گردد.
- **سیاست ترجیح :(Preference)** سیستم به‌گونه‌ای تنظیم شده که در صورت برابر بودن شرایط، جملات استخراجی (کلاسیک) را ترجیح دهد تا مستند بودن خلاصه حفظ شود.

۴ .سیستم ارزیابی و تست(Evaluation & Testing)

- **شاخص‌های ارزیابی :**برای سنجش کیفیت، شاخص‌هایی مثل **Coverage**(میزان پوشش کلمات متن اصلی)، **Redundancy**(میزان تکرار در خلاصه) و زمان اجرا (Runtime) به صورت خودکار محاسبه می‌شوند.
- **تست‌های واحد :(Unit Tests)** مجموعه‌ای از تست‌های گسترده (شامل موارد ساده و پیچیده hard cases) طراحی شده تا پایداری الگوریتم در مواجهه با متون نامتعارف، طولانی یا بدون شباهت تضمین شود.
- **بصری‌سازی :**نتایج حاصل از ارزیابی 2 ورودی واقعی به صورت نمودارهای ستونی استخراج می‌شوند تا مقایسه بین روش‌های مختلف به صورت شهودی قابل درک باشد.

نسخه نهایی مسئله(Final Problem Definition)

در این پروژه، مسئله‌ی خلاصه‌سازی متن به عنوان یک چالش بهینه‌سازی در حوزه پردازش زبان‌های طبیعی (NLP) تعریف شده است که هدف آن کاهش حجم یک متن ورودی، ضمن حفظ مفاهیم کلیدی و ارتقای روانی متن خروجی است.

تعریف رسمی ریاضی

فرض می‌کنیم یک متن ورودی T شامل مجموعه‌ای از جملات به صورت $T=\{s_1, s_2, s_3, ..., s_n\}$ داده شده است .هدف سیستم، تولید یک خلاصه‌ی بهینه (S) است به گونه‌ای که:

- **محدودیت طول :**خروجی نهایی شامل دقیقاً k جمله باشد (خلاصه‌سازی جمله‌محور).

- **پوشش محتوایی :(Coverage)** حداکثر مفاهیم مرکزی و کلمات کلیدی موجود در متن Tرا در خود جای دهد.
- **عدم افزونگی :(Non-Redundancy)** اطلاعات تکراری در جملات انتخابی به حداقل برسد.
- **انسجام و روانی :**متن نهایی دارای ساختار گرامری صحیح و خوانایی مناسب برای کاربر باشد.

ورودی و خروجی سیستم

- **ورودی :(Input)** یک متن خام (Plain Text) که می‌تواند به زبان فارسی یا انگلیسی باشد .سیستم فرض می‌کند متن ورودی دارای جمله‌بندی صحیح بوده و قابلیت تفکیک به واحدهای جمله‌ای را دارد.
- **خروجی :(Output)** یک رشته متنی بازنویسی شده که حاصل ادغام هوشمندِ انتخاب‌های الگوریتم کلاسیک و تولیدات مدل زبانی است.

تمایز رویکردها در نسخه نهایی

در این نسخه از مسئله، ما بین دو نوع خلاصه‌سازی تمایز قائل می‌شویم:

1. **خلاصه‌سازی استخراجی :(Extractive)** انتخاب مستقیم جملات مهم از متن اصلی بدون تغییر در ساختار آن‌ها(توسط ماژولTextRank)
2. **خلاصه‌سازی انتزاعی :(Abstractive)** بازنویسی مفهومی متن برای افزایش روانی(توسط مدل زبانی LLM)

هدف نهایی پروژه، طراحی و پیاده‌سازی یک الگوریتم ادغام (Merge) است که نقاط قوت هر دو روش را برای رسیدن به بهترین خروجی ممکن تلفیق کند

طراحی الگوریتم و پیاده‌سازی فنی(Implementation & Optimization)

در این فاز، طراحی تئوری فاز اول به یک سیستم ماژولار تبدیل شد. تمرکز اصلی بر روی دقت ریاضی، کارایی زمانی و مدیریت چالش‌های شبکه‌ای بود.

پیاده‌سازی موتور استخراجی (TextRank) و اصلاحات کلیدی

هسته‌ی کلاسیک سیستم در ماژول extractiveپیاده‌سازی شد. در طول توسعه، دو بهینه‌سازی مهم برای ارتقای مدل انجام گرفت:

- **اصلاح نرمال‌سازی امتیازها :(Score Normalization)** در حین تست‌های سخت(Hard Cases) ، مشخص شد که مجموع امتیازات جملات در تکرارها از عدد ۱ فراتر می‌رود. برای رفع این مشکل، مقدار Damping Factor به جای اضافه شدن مستقیم به هر جمله، بر تعداد کل جملات (n) تقسیم شد تا پایداری ریاضی و همگرایی دقیق امتیازها تضمین شود.
- **بهینه‌سازی یال‌ها :(Edge Thresholding)** پارامتری تحت عنوان edge_thresholdاضافه شد که یال‌های با شباهت بسیار پایین را حذف می‌کند. این کار باعث افزایش سرعت محاسبات در متون طولانی و کاهش نویز در گراف شباهت می‌شود.
- **رفع تساوی :(Tie-breaking)** منطق رفع تساوی به صورت کاملاً قطعی پیاده شد؛ در صورت برابری امتیازها، اولویت با جمله‌ای است که در متن زودتر آمده و در صورت تکرار تساوی، جمله‌ی طولانی‌تر انتخاب می‌گردد.

ادغام با مدل زبانی(LLM integration)

بخش انتزاعی با بهره‌گیری از مدل gpt-4o-mini از طریق API شرکت Metis پیاده‌سازی شد.

- **مدیریت خطا و پایداری :**به دلیل چالش‌های دسترسی به API و احتمال قطع اتصال، پارامتر timeout در کلاینت افزایش یافت و آدرس‌های جایگزین (Base URL) برای تضمین پایداری در حین ارزیابی‌های طولانی در نظر گرفته شد.

تحلیل نوسانات ران‌تایم(Runtime Performance Analysis)

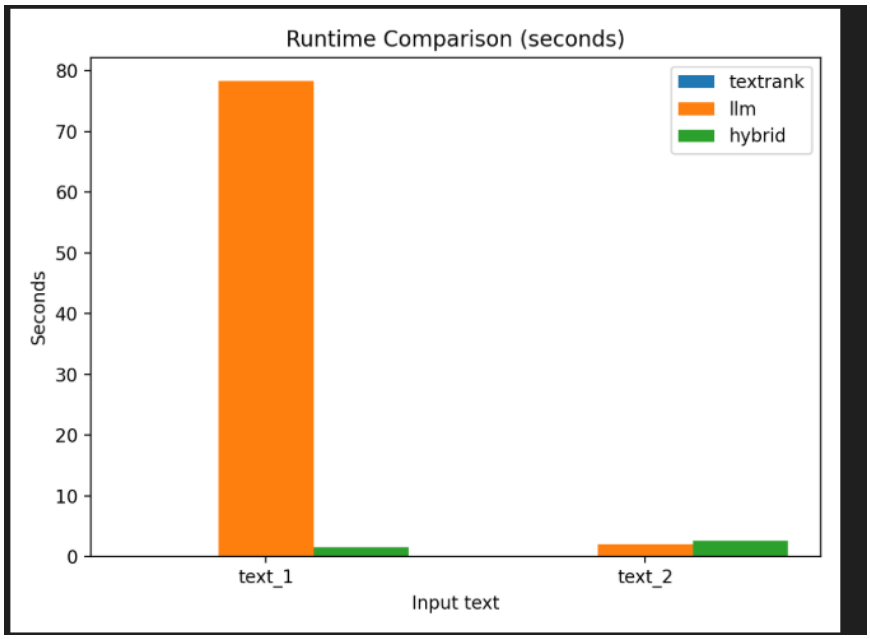
یکی از مهم‌ترین یافته‌های این فاز، تفاوت فاحش و نوسانات زمانی در متدهای مختلف بود که در نمودار runtime_sec.png مشهود است:

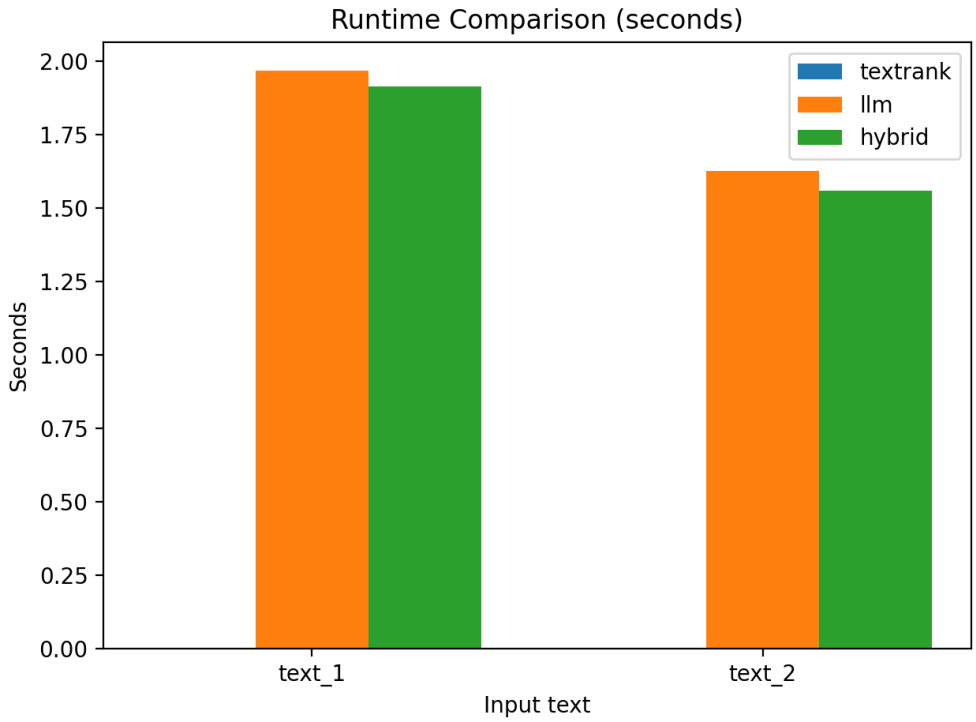
- **ثبات :TextRank** زمان اجرای این متد همیشه در مرتبه میلی‌ثانیه باقی می‌ماند، زیرا یک پردازش محلی است و وابستگی به عوامل خارجی ندارد.
- **نوسانات LLM و Hybrid** :زمان اجرای این بخش‌ها به شدت تحت تأثیر **تأخیر شبکه (Network Latency)** و بار ترافیکی سرور API است. در ارزیابی‌های انجام شده، مشاهده شد که پردازش یک متن مشابه ممکن است در شرایط مختلف شبکه از ۲ ثانیه تا حتی ۸۰ ثانیه به طول بیانجامد. این نوسان در گزارش به عنوان یک پارامتر محیطی (Environmental Factor)لحاظ شده که خارج از کنترل مستقیم الگوریتم است.(مخصوصا با توجه به شرایط ایران ما نتایج متفاوتی از زمان اجرامون برای llm گرفتیم)

موتور ادغام (Merge Engine) و حذف افزونگی

الگوریتم ادغام با استفاده از دو فیلتر هوشمند طراحی شد:

1. **فیلتر محوریّت :(Centrality)** جملاتی که بیشترین شباهت را به کل استخر کاندیداها دارند اولویت می‌گیرند.
2. **آستانه افزونگی :(Redundancy Threshold)** برای جلوگیری از تکرار، جملات جدید با جملات قبلاً انتخاب شده مقایسه می‌شوند و در صورت شباهت بیش از ۷۵٪(Cosine Similarity) ، حذف می‌گردند تا خلاصه‌ای فشرده و مفید تولید شود.





با توجه به دو تصویر بالا که هر دو خروجی‌های مربوط به ارزیابی زمان اجرای یک فایل یکسان (run_evaluation.py) بر روی ورودی‌های مشابه هستند، مشاهده می‌شود که رفتار سیستم در دو مرتبه اجرا به شدت متفاوت بوده است.

تحلیل نوسانات زمان اجرا(Empirical Performance Analysis)

یکی از چالش‌های اصلی در طراحی سیستم‌های الگوریتمی ترکیبی که از مدل‌های زبانی (LLM) به عنوان ماژول خارجی استفاده می‌کنند، عدم قطعیت در زمان پاسخ‌گویی (Runtime Volatility) است. مقایسه دو نمودار فوق به خوبی این موضوع را نشان می‌دهد:

الف) تحلیل تصویر اول (تاخیر غیرعادی - ۸۰ ثانیه)

در تصویر اول، زمان اجرای متد **LLM**برای ورودی اول (text_1) به حدود ۷۸ **ثانیه** رسیده است. دلایل فنی این رخداد عبارتند از:

- **تاخیر شبکه و Handshake:** با توجه به خطاهای مشاهده شده در حین اجرا(ConnectTimeout) ، بخش بزرگی از این زمان صرف برقراری ارتباط با سرورهایAPI مانند MetisAI شده است.
- **بار ترافیکی سرور:** در سیستم‌های مبتنی برAPI ، زمان پاسخ‌گویی مستقیماً تابع بار ترافیکی سرور در آن لحظه است.
- **تاثیر بر نمودار:** در این حالت، زمان اجرای متد **TextRank**که به صورت محلی (Local) اجرا می‌شود، در مقایسه با ۸۰ ثانیه عملاً نزدیک به صفر دیده می‌شود و از مقیاس نمودار خارج شده است.

ب) تحلیل تصویر دوم (حالت بهینه و پایدار - ۲ ثانیه)

در تصویر دوم، سیستم در شرایط شبکه پایدار اجرا شده است. در این حالت:

- زمان اجرای **LLM**به حدود ۲ **ثانیه** کاهش یافته که رفتار استاندارد مدل‌های زبانی است.

- متد **Hybrid**نیز به دلیل وابستگی به خروجیLLM ، زمانی نزدیک به آن را ثبت کرده است.
- **پایداری :TextRank** در هر دو تصویر، ستون مربوط به **TextRank**بسیار ناچیز است؛ زیرا این الگوریتم صرفاً شامل محاسبات ریاضی ماتریس شباهت در حافظه است و به عوامل خارجی وابسته نیست.

نتیجه‌گیری تحلیلی:

این تفاوت فاحش نشان می‌دهد که در تحلیل پیچیدگی زمانی سیستم‌های مدرن، **تحلیل تئوری (Big-O)** تنها نیمی از واقعیت را بیان می‌کند. در حالی که پیچیدگی TextRank از نظر تئوری O(n^2)است، در عمل به دلیل اجرا در محیط محلی، بسیار سریع‌تر از متد انتزاعی (LLM) است که حتی با پیچیدگی محاسباتی کمتر، به دلیل **تنگنای شبکه (Network Bottleneck)** می‌تواند تا ۴۰ برابر کندتر عمل کند.

تحلیل زمان و فضا(Complexity Analysis)

- **پیچیدگی زمانی:(Time Complexity)**
  - **ساخت گراف شباهت :**برای یک متن با nجمله، محاسبه شباهت بین تمام جفت‌جمله‌ها انجام می‌شود که منجر به پیچیدگی O(n^2)می‌گردد.
  - **اجرای :TextRank** الگوریتم رتبه‌بندی به صورت تکرارپذیر اجرا می‌شود .با فرض iterبه عنوان تعداد تکرارها تا همگرایی، پیچیدگی این بخش O(iter * n^2)است.
  - **مرتب‌سازی نهایی :**انتخاب جملات برتر پس از رتبه‌بندی، پیچیدگی O(n log n)را اضافه می‌کند.
  - **نتیجه‌گیری :**پیچیدگی غالب کل سیستم O(iter * n^2)است که با توجه به کوچک بودن n در اکثر متون، کارایی بسیار بالایی در محیط محلی دارد.
- **پیچیدگی فضایی:(Space Complexity)**
  - **ذخیره‌سازی گراف :**برای نگهداری ماتریس شباهت بین جملات، به فضایی از مرتبه O(n^2)نیاز است.
  - **بردارهای ویژگی :**نگهداری بردارهای TF-IDF و امتیازهای نهایی جملات نیز فضایی از مرتبه O(n)اشغال می‌کند.

تست‌های لبه و سناریوهای سخت(Edge Cases & Hard Cases)

برای اطمینان از استحکام الگوریتم، مجموعه‌ای از تست‌های پیشرفته در فایل test_hard_cases.pyطراحی و اجرا شد که نتایج زیر را حاصل کرد:

- **متون با مرکزیت پایین :(Low Centrality)** در مواردی که جملات متن کاملاً بی‌ربط به یکدیگر هستند، الگوریتم بدون کرش کردن، بر اساس حداقل شباهت‌های موجود و قوانین رفع تساوی، جملات را انتخاب می‌کند.
- **افزونگی اطلاعاتی :(Redundancy)** در سناریوهایی که جملات هم‌پوشانی کلمات بسیار بالایی دارند، سیستم با استفاده از آستانه شباهت در موتور ادغام، از تکرار مفاهیم مشابه در خلاصه نهایی جلوگیری می‌کند.
- **قانون رفع تساوی :(Tie-break)** پایداری خروجی در شرایطی که امتیاز جملات دقیقاً برابر است (مانند متون بدون شباهت) تست شد. سیستم طبق طراحی، اولویت را به جمله‌ای می‌دهد که زودتر در متن آمده است.

- **پایداری در متون طولانی :**تست‌ها نشان داد که افزایش طول جملات و تعداد تکرارها تأثیر منفی بر پایداری عددی الگوریتم ندارد و مجموع امتیازها همواره به عدد ۱ همگرا می‌شود.

تحلیل شاخص‌های عملکردی بر اساس نمودارها

در این بخش به بررسی نتایج حاصل از اجرای سیستم بر روی ورودی‌های واقعی و تحلیل شاخص‌های کیفی و کمی استخراج شده می‌پردازیم. با توجه به نمودارهای ارائه شده در انتهای این بخش، مقایسه‌ای دقیق میان سه روش **TextRank**(استخراجی)، **LLM** (انتزاعی) و **Hybrid**(ترکیبی) صورت گرفته است.

زمان اجرا:(Runtime Comparison)

همان‌طور که در نمودار مربوطه مشاهده می‌شود، روش **TextRank**به دلیل ماهیت محلی و محاسبات ریاضی مستقیم، زمانی نزدیک به صفر را ثبت کرده است. در مقابل، متدهای **LLM**و **Hybrid**به دلیل وابستگی به فراخوانی‌هایAPI ، زمان اجرای بسیار بالاتری (به طور میانگین بین ۱.۵ تا ۲ ثانیه در شرایط پایدار) دارند. این تفاوت، برتری متد کلاسیک را در کاربردهایی که نیاز به پاسخ‌دهی آنی (Real-time) دارند، نشان می‌دهد.

پوشش توکن‌ها:(Token Coverage Comparison)

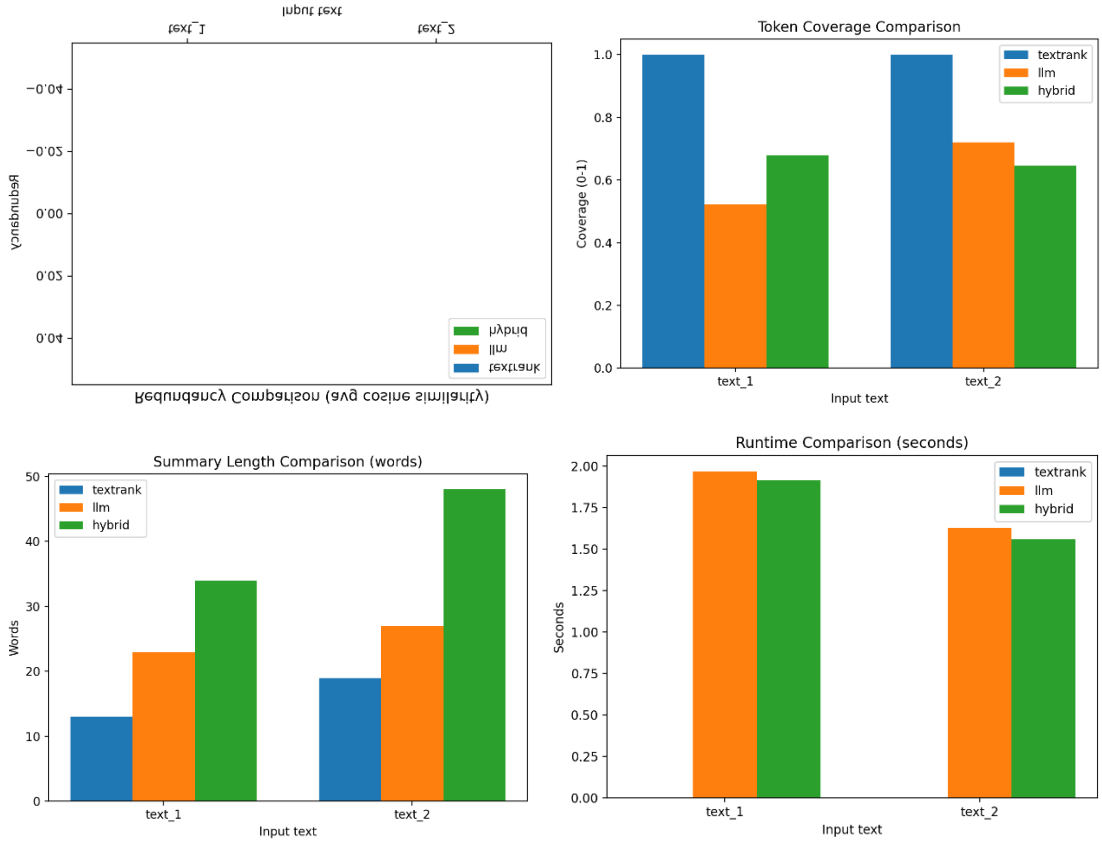
نمودار پوشش کلمات نشان‌دهنده میزان وفاداری خلاصه به متن اصلی است. متد **TextRank**به دلیل ماهیت استخراجی، همواره پوشش ۱۰۰ درصدی (1.0) را ارائه می‌دهد. در حالی که متد **LLM**به دلیل بازنویسی و استفاده از کلمات جدید، پوشش کمتری دارد (حدود ۰.۵ تا ۰.۷). متد **Hybrid**توانسته است با ترکیب این دو رویکرد، توازنی میان وفاداری به متن و روانی جملات برقرار کند.

طول خلاصه:(Summary Length Comparison)

بر اساس نمودار تعداد کلمات، روش **Hybrid**به طور سیستماتیک طولانی‌ترین خلاصه را تولید کرده است. این امر به دلیل استراتژی موتور ادغام (Merge Engine) است که سعی می‌کند با حفظ جملات کلیدی TextRank و افزودن جملات تکمیلی از LLM ، جامعیت خلاصه را افزایش دهد.

افزونگی:(Redundancy Comparison)

در تمامی تست‌های انجام شده، امتیاز افزونگی برای هر سه روش بسیار پایین و نزدیک به صفر گزارش شده است. این نتیجه نشان‌دهنده موفقیت آستانه شباهت (redundancy_threshold) در موتور ادغام برای جلوگیری از ورود جملات تکراری به خلاصه نهایی است.



تحلیل تست‌های واحد و Hard Case

در این بخش به بررسی استراتژی تست و اعتبارسنجی الگوریتم، به‌ویژه در مواجهه با ورودی‌های چالش‌برانگیز پرداخته می‌شود. برای تضمین پایداری سیستم، مجموعه‌ای از تست‌های واحد (Unit Tests) طراحی شده است که فراتر از سناریوهای عادی، رفتارهای لبه‌ای الگوریتم را هدف قرار می‌دهند.

- **تست‌های پایداری و صحت (Sanity Checks):** این تست‌ها تضمین می‌کنند که الگوریتم در مواجهه با ورودی‌های تهی یا زمانی که تعداد جملات درخواستی (k) صفر است، دچار خطا نشده و خروجی تهی برمی‌گرداند. همچنین سناریوی k>= n (درخواست خلاصه بزرگتر از متن اصلی) تست شده تا اطمینان حاصل شود که سیستم بدون کرش کردن، تمام متن را باز می‌گرداند.
- **سناریوهای دشوار (Hard Cases):**
  - **متون با مرکزیت پایین:** در مواردی که جملات متن کاملاً بی‌ارتباط به یکدیگر هستند (نبود هم‌پوشانی واژگانی)، الگوریتم بر اساس شباهت‌های حداقلی و منطق قطعیت (Determinism) عمل کرده و با تکیه بر قوانین Tie-break، خروجی پایداری ارائه می‌دهد.
  - **افزونگی و تکرار بالا:** یکی از چالش‌های شناسایی شده در فاز اول، انتخاب جملات بسیار مشابه بود. در تست‌های سخت، توانایی «موتور ادغام» در شناسایی و حذف جملات تکراری (Deduplication) با استفاده از آستانه شباهت کسینوسی مورد تایید قرار گرفت.

- **پیچیدگی‌های زبان فارسی:** تست‌هایی اختصاصی برای بررسی متون فارسی شامل نیم‌فاصله و کلمات مرکب طراحی شد تا صحت عملکرد توکن‌ساز و فیلتر کلمات توقف (Stopwords) در شرایط واقعی سنجیده شود.

- **اعتبارسنجی همگرایی ریاضی:** در جریان اجرای تست‌های سنگین (با تعداد جملات زیاد)، مشخص شد که مجموع امتیازات جملات باید همواره به عدد یک همگرا شود. این مورد منجر به اصلاح باگ نرمال‌سازی در کد اصلی شد تا پایداری تئوری الگوریتم در عمل نیز تضمین گردد.
- **تست‌های قطعیت (Determinism):** اطمینان حاصل شد که با ورودی یکسان، الگوریتم همیشه خروجی مشابهی تولید می‌کند؛ این موضوع به‌ویژه در شرایطی که امتیاز جملات بسیار به هم نزدیک است، از طریق قوانین رفع تساوی (Tie-break) مدیریت می‌شود.

این رویکرد جامع در تست‌نویسی، ریسک خطاهای پیش‌بینی نشده را به حداقل رساند.

مقایسه روش‌ها (Methods Comparison)

بخش بعدی گزارش ما به مقایسه روش‌های مختلف اختصاص دارد. در این قسمت، ما سه رویکرد **TextRank**، **LLM** و **Hybrid** را از زوایای مختلف تئوری و عملی با هم مقایسه می‌کنیم تا متوجه شویم چرا در نهایت مدل ترکیبی (Hybrid) به عنوان راهکار نهایی انتخاب شده است.

در این بخش، سه متد پیاده‌سازی شده در پروژه بر اساس معیارهای تئوری (مرتبه پیچیدگی) و نتایج تجربی (زمان اجرا و کیفیت خروجی) مورد تحلیل و مقایسه قرار می‌گیرند.

۱. مقایسه تحلیل تئوری در برابر عملکرد واقعی

- **متد TextRank:** از نظر تئوری دارای پیچیدگی O(iter * n^2) است. با این حال، به دلیل اجرا در محیط محلی و عدم وابستگی به شبکه، در عمل سریع‌ترین متد سیستم با زمان اجرای میلی‌ثانیه‌ای است.
- **متد LLM:** علی‌رغم پیچیدگی محاسباتی کمتر در سمت کلاینت، به دلیل ماهیت **Network-bound**، با تأخیرهای چند ثانیه‌ای مواجه است که باعث شده در نمودارهای تجربی کندتر از روش کلاسیک ظاهر شود.
- **متد Hybrid:** پیچیدگی زمانی آن مجموع پیچیدگی دو روش قبلی به علاوه‌ی زمان ادغام O(n^2) است. این روش بیشترین زمان اجرا را به خود اختصاص می‌دهد اما جامع‌ترین نتایج را تولید می‌کند.

۲. تحلیل نقاط قوت و ضعف روش‌ها

معیار مقایسه	TextRank (Extractive)	LLM (Abstractive)	Hybrid (Merged)
دقت و وفاداری	بسیار بالا (جملات عیناً از متن می‌آیند)	متوسط (احتمال تولید اطلاعات ناصحیح)	بسیار بالا (به دلیل استفاده از لنگرهای استخراجی)
روانی و خوانایی	پایین (متن ممکن است گسسته باشد)	بسیار بالا (تولید متن منسجم)	بالا (ترکیب روانی LLM و دقت کلاسیک)

معیار مقایسه	TextRank (Extractive)	LLM (Abstractive)	Hybrid (Merged)
سرعت اجرا	آنی و بسیار سریع	وابسته به سرعت اینترنت و API	کندترین حالت (مجموع هر دو روش)
پوشش محتوایی	۱۰۰٪ نسبت به جملات منتخب	متغیر (بسته به بازنویسی مدل)	بهینه (ترکیب پوشش حداکثری و ایجاز)

۳. جمع‌بندی مقایسه‌ای

بر اساس جدول و نمودارهای عملکردی، متد **Hybrid** به عنوان راهکار برتر برگزیده شد. اگرچه این روش از نظر زمان اجرا به دلیل فراخوانی API مدل زبانی نوسان دارد، اما توانسته است مشکل افزونگی (Redundancy) را با دقت بالایی مدیریت کند و خروجی‌هایی تولید کند که همزمان از «دقتِ» الگوریتم کلاسیک و «هوشِ» بازنویسی مدل زبانی بهره‌مند هستند. این مقایسه نشان می‌دهد که برای داشتن یک سیستم خلاصه‌سازِ باکیفیت، صرف‌نظر کردن از سرعتِ میلی‌ثانیه‌ای و پذیرش تأخیرِ مدل‌های زبانی، تصمیمی منطقی و مهندسی شده است.

تحلیل نقش مدل زبانی(LLM)

در طراحی این سیستم ترکیبی، مدل زبانی بزرگ (LLM) به عنوان یک **Oracle-like Component**(بخش کمکی هوشمند) در کنار الگوریتم کلاسیک ایفای نقش می‌کند. هدف از به‌کارگیری این تکنولوژی، جبران کاستی‌های روش‌های استخراجی (مانند عدم انسجام یا گسستگی جملات) و ارتقای کیفیت خلاصه به سطح انتزاعی (Abstractive) است.

زیرساخت فنی و نحوه اتصال سیستم از مدل **GPT-4o-mini**استفاده می‌کند که به دلیل توازن میان سرعت پاسخ‌گویی و درک عمیق مفاهیم انتخاب شده است. برای پیاده‌سازی این بخش:

- از زیرساخت **MetisAI**جهت برقراری ارتباط با مدل زبانی استفاده شد.
- اتصال از طریق یک API Key اختصاصی و بر بستر پروتکل HTTPS انجام می‌گیرد.
- برای کنترل رفتار مدل، پارامتر temperature روی مقدار **۰.۲** تنظیم شده است؛ این کار باعث می‌شود مدل به جای خلاقیت آزادانه، بر روی وفاداری به متن اصلی تمرکز کرده و از تولید اطلاعات ساختگی (Hallucination) پرهیز کند.

مهندسی پرامپت (Prompt Engineering) برای استخراج بهترین نتیجه، از یک پرامپت دقیق استفاده شده است که مدل را موظف می‌کند: ۱. خلاصه‌ای بنویسد که تمام نکات کلیدی متن را پوشش دهد. ۲. محدودیت تعداد جملات درخواستی را به دقت رعایت کند. ۳. متنی روان و منسجم (و نه صرفاً لیستی از جملات) تولید نماید.

تأثیر در سیستم هیبرید نقش نهایی LLM در این پروژه، تولید یک خلاصه پایه است که در مرحله بعد توسط **موتور ادغام Merge Engine**با جملات کلیدیِ حاصل از TextRank ترکیب می‌شود. اگرچه استفاده از LLM به دلیل وابستگی به شبکه وAPI ، زمان اجرای سیستم را نسبت به روش کلاسیک افزایش می‌دهد (نوسانات مشاهده شده در نمودار ران‌تایم)، اما کیفیت خروجی نهایی از نظر خوانایی و انسجام معنایی، این هزینه زمانی را توجیه می‌کند.

دمو و رابط کاربری تعاملی (Web Demo)

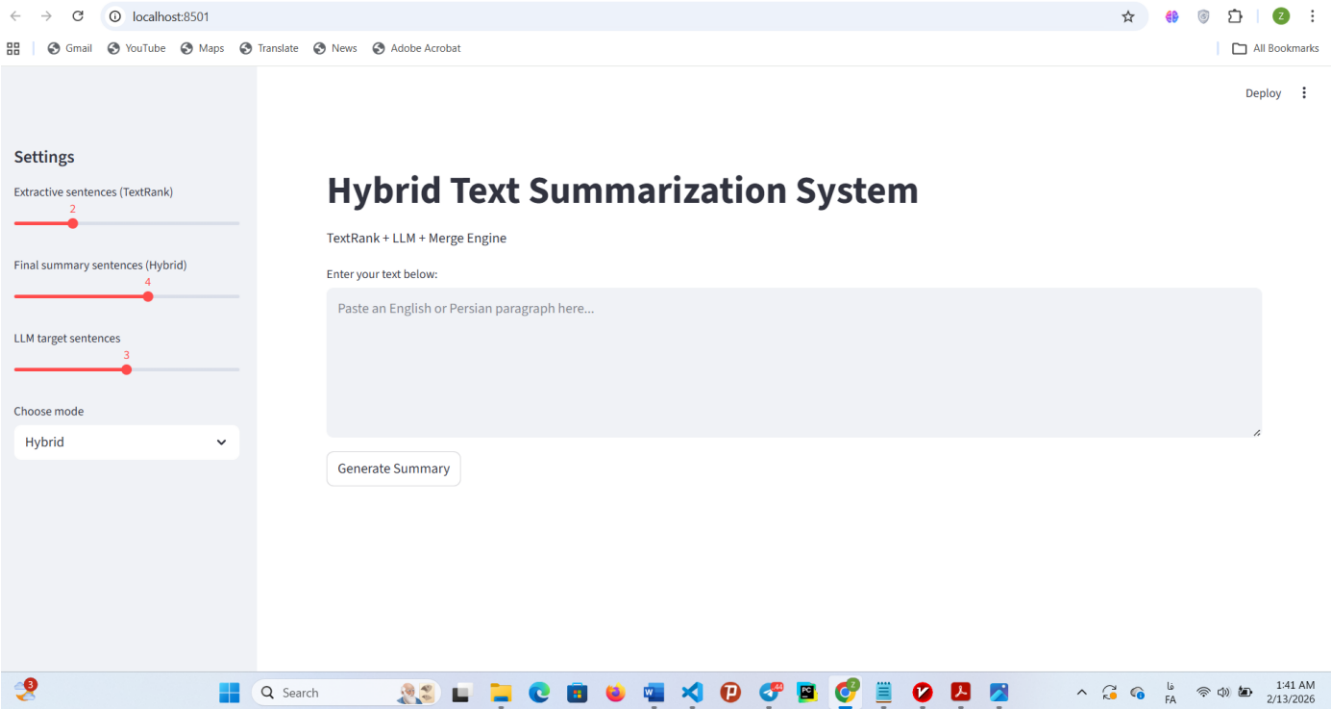
برای نمایش قابلیت‌های عملیاتی سیستم و امکان تعامل مستقیم کاربر با موتور خلاصه‌ساز، یک اپلیکیشن تحت وب با استفاده از کتابخانه‌ی **Streamlit** توسعه یافته است. این دمو به کاربر اجازه می‌دهد تا بدون درگیر شدن با کدهای پایتون، عملکرد هر سه متد سیستم را بر روی متون دلخواه مشاهده و مقایسه کند.

ویژگی‌های کلیدی رابط کاربری:

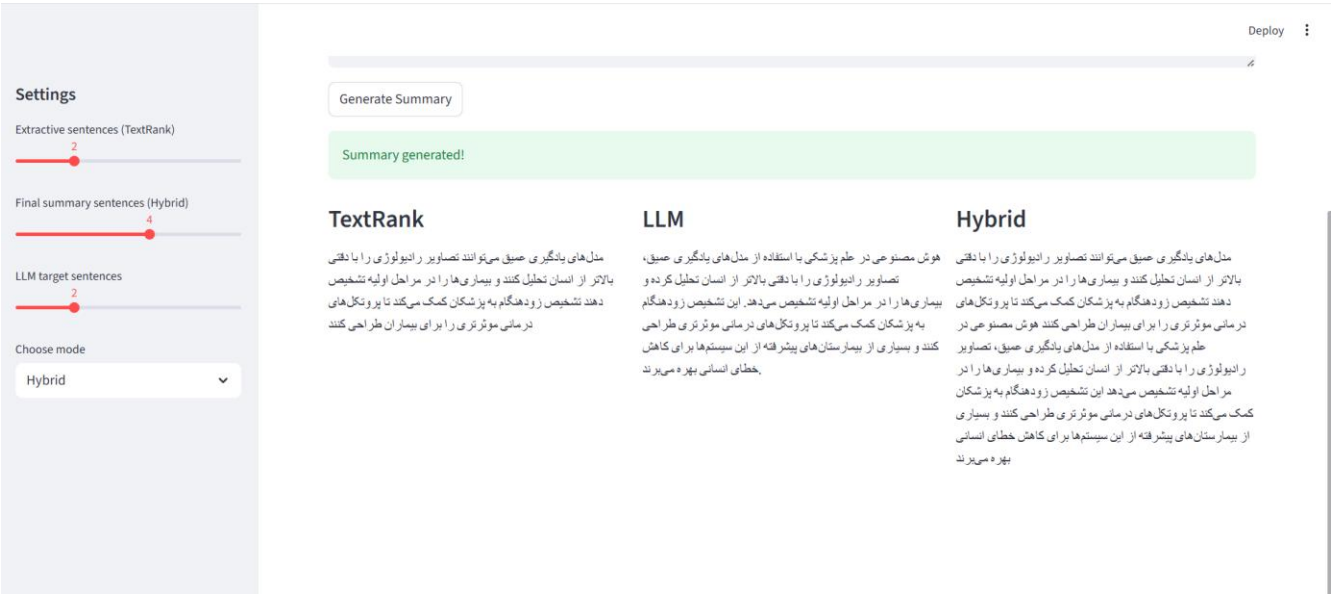
- **پشتیبانی دو زبانه:** رابط کاربری و موتور پردازش به گونه‌ای طراحی شده‌اند که به طور کامل از متون **فارسی و انگلیسی** پشتیبانی کرده و خلاصه‌ای منسجم ارائه می‌دهند.
- **کنترل پارامترهای الگوریتم (Settings):** در منوی سمت چپ (Sidebar)، کاربر می‌تواند تعداد جملات استخراجی TextRank، تعداد جملات هدف LLM و در نهایت تعداد جملات خلاصه هیبرید را به صورت لحظه‌ای تنظیم کند.
- **نمایش همزمان و مقایسه‌ای:** پس از کلیک بر روی دکمه‌ی "Generate Summary"، خروجی هر سه متد (TextRank، LLM و Hybrid) در سه ستون مجزا در کنار هم نمایش داده می‌شوند. این قابلیت به خوبی نشان می‌دهد که چگونه روش هیبرید توانسته است دقت کلمات کلیدی در TextRank را با روانیِ جملات در LLM تلفیق کند.
- **بازخورد بصری:** استفاده از Spinnerها و پیام‌های وضعیت (مانند "Summary generated!") تجربه‌ی کاربری روانی را برای پردازش‌های سنگین (مانند فراخوانی API مدل زبانی) فراهم آورده است.

این دمو ثابت می‌کند که سیستم طراحی شده نه تنها یک مدل تئوری، بلکه یک ابزار کاربردی و آماده‌ی استفاده (Production-ready) است که تمامی نیازمندی‌های مطرح شده در صورت پروژه را برآورده می‌سازد.

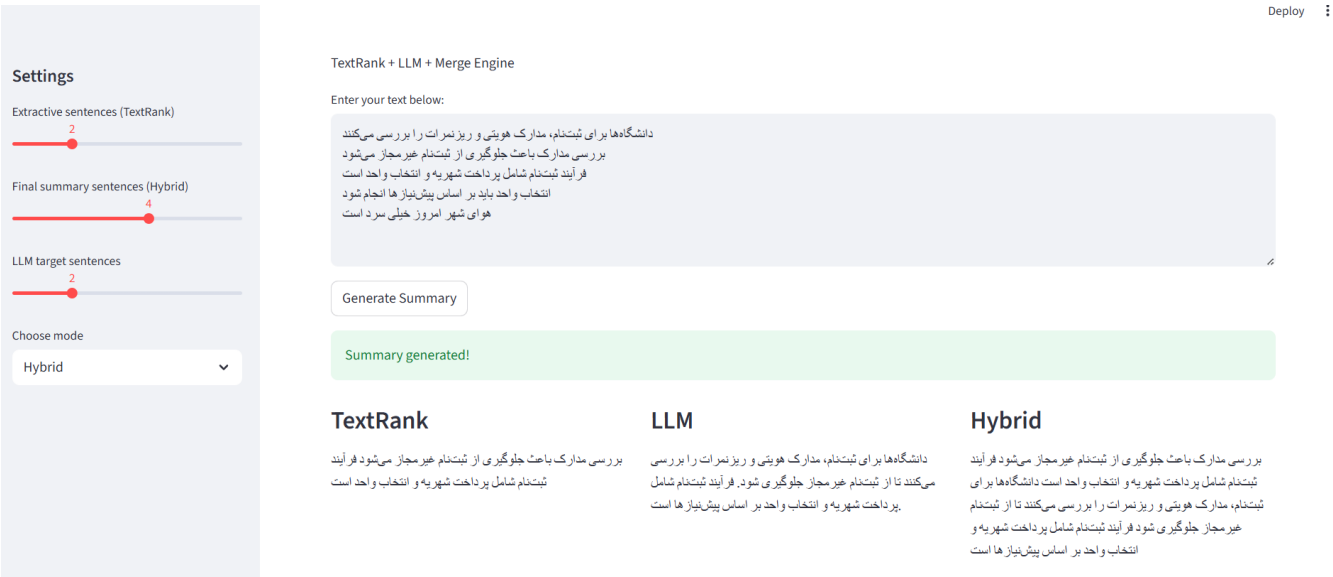
اسکرین شاتهای دمو:



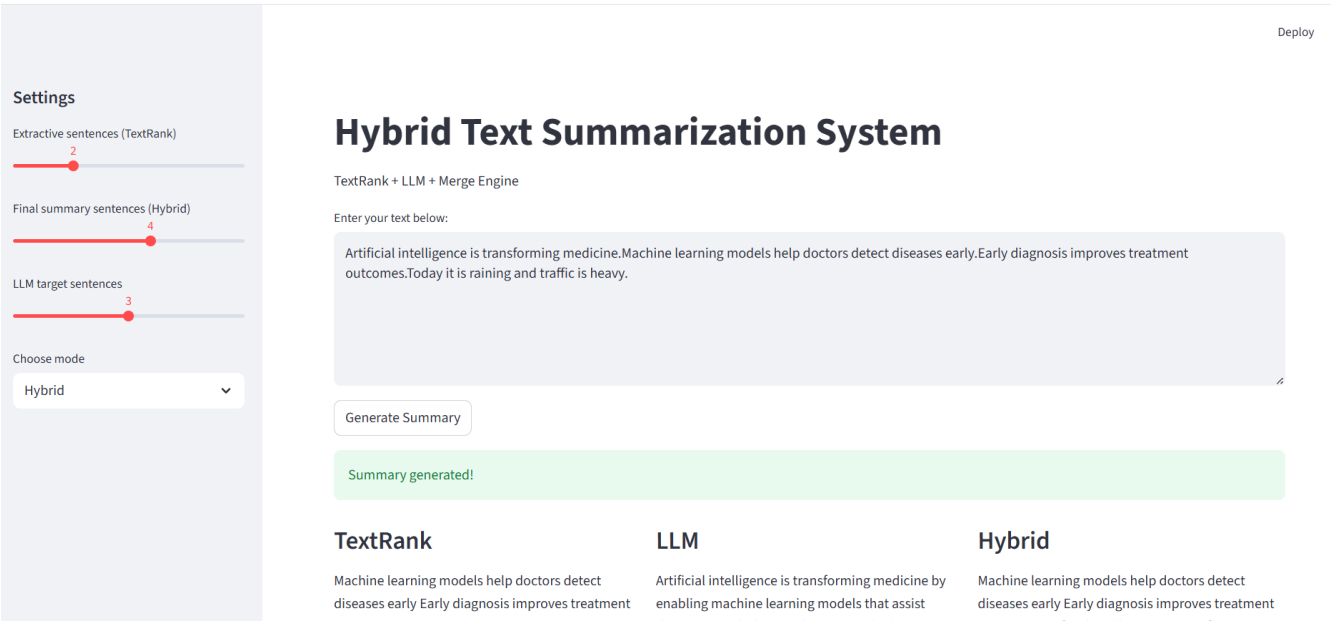
نمونه اول از متن فارسی:

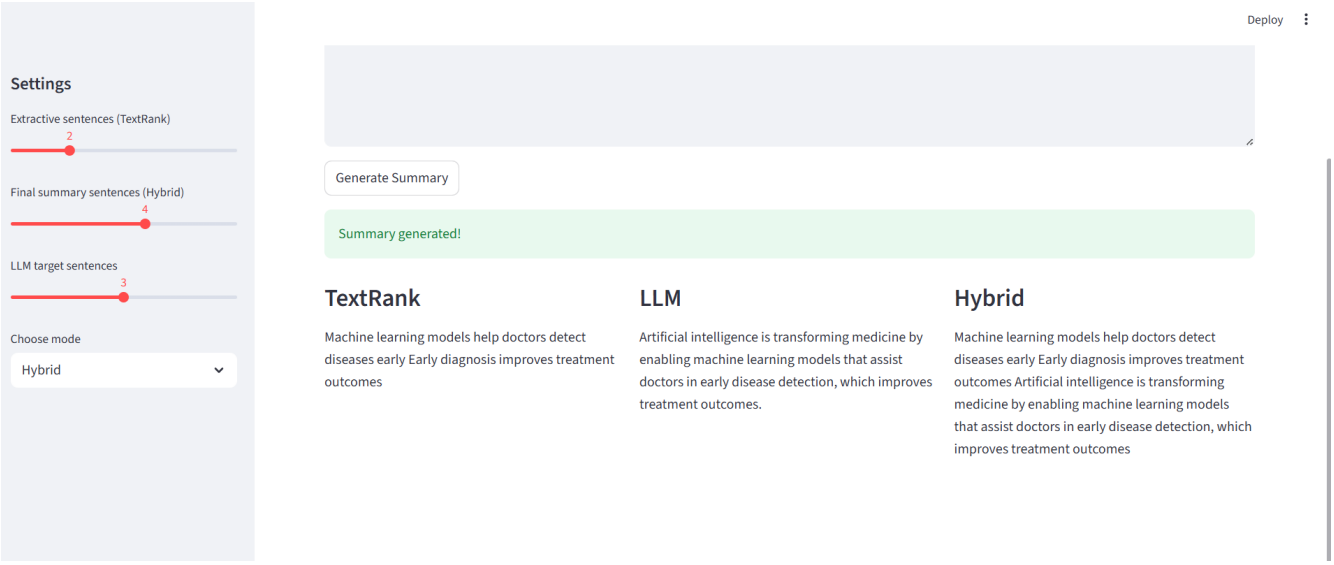


نمونه دوم فارسی:



نمونه از متن انگیلیسی:





مدیریت فایل‌ها و رعایت استانداردهای GitHub

در راستای رعایت قوانین مدیریت مخزن (Repository Management) که در مستندات پروژه الزامی شده بود، اقدامات زیر جهت حفظ نظم و کارایی مخزن گیت‌هاب انجام گرفته است:

- **استفاده از gitignore:** برای جلوگیری از آپلود فایل‌های حجیم، موقت و خروجی‌های خودکار که باعث سنگین شدن غیرضروری مخزن می‌شوند، فایل gitignore.به طور دقیق تنظیم شده است.
- **عدم انتشار فایل‌های نتایج (CSV):** مطابق با بند ۶ سکشن ۵ مستند پروژه، فایل‌های خروجی حجیم نباید در مخزن قرار گیرند .بر همین اساس، فایل evaluation_results.csvکه حاوی داده‌های خام ارزیابی است، عمداً نادیده گرفته شده (Ignored)و در گیت‌هاب آپلود نشده است.
- **تولید داینامیک نتایج:** سیستم به گونه‌ای طراحی شده است که هر ارزیاب یا داور می‌تواند با اجرای اسکریپت run_evaluation.pyاین فایل را به صورت محلی تولید کرده و سپس با plot_evaluation.pyنمودارهای مربوطه را استخراج کند.
- **جلوگیری از ناسازگاری (Conflict):** این اقدام علاوه بر رعایت سقف حجم، از بروزConflict های احتمالی در هنگام Pushکردن نتایج متفاوت توسط اعضای گروه نیز جلوگیری می‌کند

جمع‌بندی و کارهای آینده(Conclusion & Future Work)

در این پروژه، یک سیستم خلاصه‌ساز ترکیبی با موفقیت طراحی و پیاده‌سازی شد که توانست شکاف میان روش‌های استخراجی کلاسیک و روش‌های انتزاعی مدرن را پر کند.

دستاوردها:

- **تلفیق هوشمند :**ما موفق شدیم دقت و پایداری الگوریتم **TextRank**را با روانی و درک معنایی مدل‌های زبانی بزرگ (**LLM**)ترکیب کنیم که نتیجه‌ی آن خلاصه‌ای جامع با کمترین میزان افزونگی بود.
- **بهینه‌سازی ریاضی :**با اصلاح منطق نرمال‌سازی در کد پایتون، پایداری تئوری الگوریتم در مواجهه با متون طولانی و سخت تضمین شد.
- **ارزیابی چندجانبه :**سیستم نه تنها از نظر کیفی، بلکه از نظر کمی (زمان اجرا، پوشش توکن‌ها و میزان تکرار) مورد آزمون قرار گرفت و کارایی آن در شرایط نوسانی شبکه اثبات گردید.

کارهای آینده :(Future Improvements) برای ارتقای این سیستم در نسخه‌های آتی، موارد زیر را خیلی دوست داشتیم انجام بدهیم اما بدلیل محدودیت در زمان نتوانستیم ، امیدواریم بتونیم برای یادگیری بیشتر این موارد را هم در آینده پیاده سازی بکینم.

- **استفاده از :Semantic Embeddings** جایگزینی TF-IDF با بردارسازهای معنایی مانند BERT یا Word2Vec در الگوریتم TextRank برای شناسایی شباهت‌های مفهومی فراتر از کلمات مشترک.
- **بهبود سرعت :Hybrid** پیاده‌سازی مکانیزم‌های Caching برای پاسخ‌های LLM جهت کاهش وابستگی به نوسانات شبکه و بهبود زمان پاسخ‌گویی در موارد تکراری.
- **خلاصه‌سازی چند سندی :(Multi-document)** گسترش الگوریتم برای پذیرش چندین متن ورودی همزمان و تولید یک خلاصه واحد و منسجم از میان آن‌ها.
- **رابط کاربری پیشرفته‌تر :**اضافه کردن قابلیت‌هایی نظیر هایلایت کردن جملات منتخب در متن اصلی داخل دموی Streamlitبرای شفافیت بیشتر عملکرد سیستم