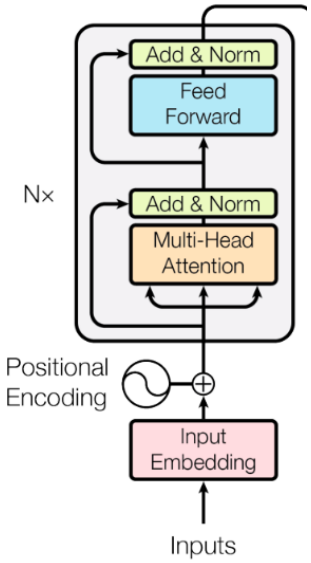
بخش امتیازی پروژه بازیابی اطلاعات

بخش اول، تعبیه برت

تا اینجا مشاهده کرده اید که هر سند تبدیل به یک بردار tf-idf می‌شود و همینطور برداری برای کوئری ساخته می‌شود و در نهایت شباهت کسینوسی بردار کوئری با سند ها مقایسه شده و مشابه‌ترین سند ها انتخاب می‌شوند.



حال می‌خواهیم روش دیگری برای تبدیل سند ها به بردار را بررسی کنیم.
جمله‌های زیر را در نظر بگیرید:

جمله اول:
هنگام طلوع خورشید، پرندگان با آواز های دلنشین خود روز را آغاز می‌کنند.

جمله دوم:
سپیده دم، مرغان با نغمه‌های دلپذیرشان شروع صبح را نوید می‌دهند.

مشاهده می‌کنید که معنی این دو جمله اول و دوم به شدت به هم نزدیک است است اما کلمات مشترک با معنی ای ندارند (صرفا یک "با" و یک "را" که معمولا به عنوان کلمات پر تکرار حذف می‌شوند).

جمله سوم:
شکارچیان کلاه هایی می‌پوشند که بالای سرشان پر پرندگان است و در هنگام شکار آواز می خوانند و خورشید را نگاه می کنند.

جمله سوم با جمله اول ارتباط معنایی کمی دارد، اما کلمات مشترک بیشتری با آن دارد و به نظر می‌رسد اگر به وسیله tf-idf از این سه جمله بردار بسازیم، بردارهای جمله‌های اول و سوم شباهت بیشتری خواهند داشت نسبت به بردارهای جمله‌های اول و دوم.

برای حل این مشکل می‌خواهیم به روشی دیگر متن سند هایمان را تبدیل به بردار کنیم که به معنای کلمات توجه می‌کنند.

به سراغ استفاده از مدل‌های bert-based فارسی می‌رویم، پیشنهاد ما ParsBert است، اما اگر مدلی پیدا کردید که عملکرد بهتری داشت، مانعی برای استفاده وجود ندارد.

با راهنمایی این لینک می‌توانید مدل انتخابی خود را به صورت local ذخیره کنید تا هر بار نیاز به دریافت آن از huggingface نباشید.

برای راهنمایی بیشتر برای نحوه استفاده از مدل انتخابی و محاسبه تعبیه ها توسط این مدل، می‌توانید به بخش Sentence Similarity در این لینک مراجعه کنید.

حال برای هر سند، بردار تعبیه جمله (sentence embedding) آن را به دست آورده و در فایل مربوط به متن آن را نیز ذخیره کنید.
برای این کار هر سند را به شکل یک فایل ذخیره کنید که داخل آن فایل به شکل زیر باشد:

```
{
        “doc_id”: DOCUMENT_ID,
        “content”: CONTENT,
        “embedding”: [//embedding entries]
}
```

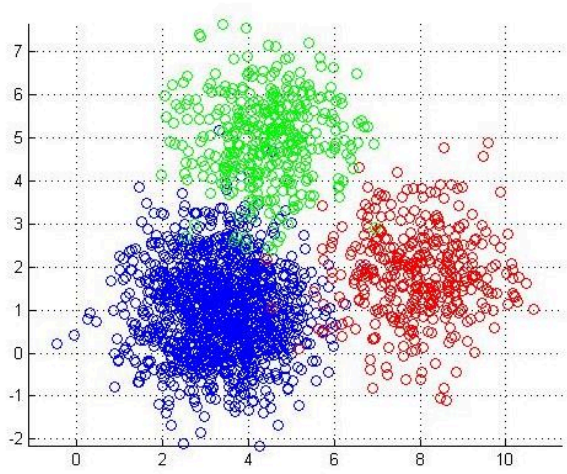
بخش دوم، جست و جوی همه اسناد

فرض کنید که نمی‌توانیم تمام سند ها و تعبیه آن‌ها را در RAM نگهداری کنیم.
برای این کار باید سند ها در disk ذخیره شده باشند.
حال هر کوئری که دریافت می‌شود را به وسیله مدل بردار تعبیه اش را به دست آورده و با بردار تعبیه تمام سند ها مقایسه کنید، نیاز است با دریافت هر کوئری، تمامی فایل سند ها (که هرکدام یک سند هستند و به شکل json ای بالا توصیف شده اند) را بخوانید، تعبیه آن‌ها را از فایل دریافت کنید (دیگر با مدل آن را به دست نیاورید) و با تعبیه ای که از کوئری به دست آورده اید similarity بگیرید و در هر لحظه، d تا شبیه ترین سند را در ram ذخیره داشته باشید.
در نهایت d تا شبیه ترین سند را به کوئری به عنوان خروجی برگردانید.

زمان اجرای جست و جو (تبدیل کوئری به بردار، خواندن تک تک سند ها از دیسک و محاسبه شباهت با کوئری، نگهداری d مرتبط ترین سند در ram) را برای یک کوئری مرتبط با یک سندی به انتخاب خودتان به دست آورده و گزارش کنید.

بخش سوم، جست و جو در خوشه‌ها

در بخش اول شباهت بردار کوئری با بردار هر سند محاسبه می‌شد و شبیه‌ترین سند برگردانده می‌شد.
در این بخش برای کاهش محاسبات، می‌خواهیم از خوشه بندی استفاده کنیم.



الگوریتم k-means را که در کلاس تدریس شده را می‌توانید از کتابخانه sklearn استفاده کنید.
فرض کنید ورودی یک ماتریس m * n است، که m تعداد بردارها (تعداد سند ها) و n تعداد درایه‌های هر بردار است (برای مثال، بردار هر سند که به وسیله ParsBert تبدیل به بردار می‌شود، 768 تا درایه دارد).
برای آشنایی با نحوه کار الگوریتم می‌توانید به اسلاید clustering درس و این لینک مراجعه کنید.

حال تمامی اسناد را به وسیله این الگوریتم و بردارهای به دست آمده در بخش قبل، خوشه بندی کنید.
سندهایی که در یک خوشه قرار می‌گیرند را در یک directory مشترک قرار دهید (برای مثال تمام سندهای خوشه 10 در directory ای به نام 10 ذخیره شوند).
مدل خوشه بندی خود را نیز ذخیره کنید.
حال برای بازیابی سند ها، ابتدا کوئری را تبدیل به بردار کرده، به وسیله مدل k-means خود تشخیص دهید که کوئری در کدام خوشه قرار دارد، سپس d شبیه‌ترین سند را در آن خوشه (که در directory آن خوشه هستند) جست و جو کرده و برگردانید.

بخش چهارم، جست و جوی بهینه در فضای برداری

به جای استفاده از خوشه بندی و جست و جوی دو مرحله ای، از یک کتابخانه vector space search (مانند faiss) استفاده کنید و هر بار d سند مرتبط با کوئری را برگردانید.
برای راهنمایی استفاده از faiss، می‌توانید به این لینک مراجعه کنید.

کارهای مورد انتظار برای هر فاز:

برای راحتی استفاده از کتابخانه های مورد نیاز و دسترسی به پردازنده گرافیکی می‌توانید از google clolab استفاده کنید.
در google colab می‌توانید google drive خود را mount کنید، پیشنهاد می‌شود فایل‌های مورد نیاز از جمله dataset اسناد و خروجی‌های هر مرحله را در drive خود نگهداری کنید.
در نهایت لینک directory ای که نتایج هر مرحله را در آن نگهداری کرده اید در گزارش خود قرار دهید. برای این کار لازم است دسترسی عمومی به عنوان viewer به آن directory باز کنید.

فاز ۱:

فایل json ای مخصوص این فاز که در اختیارتان قرار داده شده را بخوانید، به ازای هر سند یک content خواهید داشت.
برای هر سند، به وسیله مدل زبانی مبتنی بر مدل برت که انتخاب کرده اید، تعبیه آن را تولید کرده و به شکل زیر در فایلی با نامی به شکل زیر:

{doc_id}.json

که doc_id نام سند مورد نظر است، در directory ای به نام phase_1_result ذخیره کنید.
محتوای فایل هر سند باید به شکل زیر باشد:

```
{
	“doc_id”: DOCUMENT_ID,
	“content”: CONTENT,
	“embedding”: [//embedding entries]
}
```

در گزارش خود، آن بخش از کد که این فایل‌ها را به ازای هر سند می‌سازد را نشان دهید.
همینطور تصویری از تعدادی از سندهای ساخته شده در پوشه phase_1_result خود نشان دهید.

کد به دست آوردن تعبیه اسناد و ذخیره فایل آن‌ها به صورت گفته شده را در گزارش خود آورده و خلاصه توضیح دهید.

فاز ۲:

باید بتوانید با دریافت یک کوئری متنی، به وسیله مدل زبانی مبتنی بر برت گفته شده بردار تعبیه آن را تولید کنید، حال تمامی فایل‌های در پوشه phase_1_result را خوانده و شباهت آن را با تعبیه ای که از کوئری دارید به دست آورید.

ابتدا بررسی کنید که برای سندهای مختلف، اندازه بردار تعبیه شان یکسان خواهد بود یا خیر؟ اگر یکسان بود چه روشی برای محاسبه فاصله دو بردار را استفاده می‌کنید؟ اگر اندازه بردارها یکسان بود چطور؟ برای مدل زبانی انتخابی شما چه روشی بهتر است؟

بعد از محاسبه شباهت بردار هر سند با بردار کوئری و نگه داشتن شباهت (فاصله) آن d تا سند که بیشترین نزدیکی به کوئری را دارند برگردانید.

باید توانایی تغییر d را در کد خود داشته باشید، برای گزارش خود d را برابر 5 در نظر بگیرید.

با نگاهی به متن بعضی سندها، کوئری مناسبی انتخاب کنید، پیشنهاد می‌شود که تعداد کلمات کوئری کمتر از 4 کلمه نشود، توجه داشته باشید جایگاه کلمات برای مدل زبانی bert اهمیت دارد.

در گزارش خود کوئری انتخابی، متن و id سندهای بازیابی شده، امتیازی که به هر سند داده اید، و زمان بازیابی را گزارش کنید.

منظور از زمان بازیابی، مدت زمان محاسبه بردار تعبیه برای کوئری، خواندن دانه دانه سندها از فایل‌ها، محاسبه شباهت هر سند با کوئری، و در نهایت مرتب سازی اسناد بر اساس شباهت و برگرداندن شبیه ترین سندها به کوئری است.
کد بخش بازیابی را نیز در گزارش خود آورده و خلاصه توضیح دهید.

فاز ۳:

فایل‌های اسناد را که در phase_1_result ذخیره کرده اید بخوانید و تعبیه هایشان را در یک لیست درج کنید.
در نهایت باید به تعداد اسنادی که دارید، بردار تعبیه داشته باشید. یعنی یک ماتریس با تعداد سطرهای برابر با تعداد اسناد و تعداد ستون‌های برابر با درایه‌های تعبیه یک سند.

حال یک مدل الگوریتم خوشه بندی k-means روی این ماتریس آموزش دهید.
تعداد خوشه‌ها را چند در نظر می‌گیرید؟ علت آن را در گزارش خود بیاورید.

یک پوشه بسازید به نام phase_3_result و در آن به ازای هر خوشه، یک پوشه به نام آن خوشه بسازید.
حال فایل هر سند را که در phase_1_result ساخته بودید، در پوشه خوشه ای که در آن قرار می‌گیرد کپی کنید.

حال باید بتوانید با دریافت یک کوئری، ابتدا تشخیص دهید که این کوئری در کدام خوشه قرار دارد، و بازیابی را صرفا با اسنادی که در آن خوشه قرار دارند انجام دهید. برای این مرحله از همان کوئری فاز ۲ استفاده کنید. شما باید همچنان بتوانید d سند مرتبط به کوئری را بازیابی کنید.

در گزارش خود کوئری انتخابی (که همان کوئری فاز ۲ است)، متن و id سندهای بازیابی شده، امتیازی که به هر سند داده اید، و زمان بازیابی را گزارش کنید. دقت کنید که برای این مرحله، باید زمان خواندن مدل k-means هم در نظر بگیرید.

آیا نتایج بازیابی اسناد شما تغییر چشمگیری کرد؟ دلیل بیاورید.
مدل k-means خود را کنار پوشه phase_3_result ذخیره کنید.
مانند فاز ۲ مدت زمان بازیابی اسناد را گزارش کنید.
همینطور گزارش کنید که خوشه ای که کمترین تعداد اسناد را دارد، تعداد اسنادش چند است؟ همینطور خوشه ای که بیشترین تعداد اسناد را دارد؟

کد بخش بازیابی به وسیله k-means هم در گزارش خود آورده و خلاصه توضیح دهید.

فاز ۴:

ماتریسی که برای تعبیه اسناد به دست آورده بودید را به IndexFlatL2 از کتابخانه faiss داده و به وسیله این index، باید d سند مرتبط به کوئری را بازیابی کنید.
مدل IndexFlatL2 خود را در پوشه ای به نام phase_4_result ذخیره کنید.
به وسیله این index، باید d سند مرتبط با کوئری را بتوانید بازیابی کنید.
در گزارش خود کوئری انتخابی (که همان کوئری فاز ۲ است)، متن و id سندهای بازیابی شده، امتیازی که به هر سند داده اید، و زمان بازیابی را گزارش کنید. دقت کنید که برای این مرحله، باید زمان خواندن مدل IndexFlatL2 هم در نظر بگیرید.
کد بخش بازیابی به وسیله faiss هم در گزارش خود آورده و خلاصه توضیح دهید.