پروژه درس بازیابی پیشرفته اطلاعات

نیم‌سال دوم ۱۴۰۰-۱۴۰۱

مدرس: دکتر عسگری

دانشکده مهندسی کامپیوتر

مستندات پروژه

اسم و شماره گروه: فردوسی - ۱۶

در این قسمت قصد داریم تمام آنچه تا اینجا انجام شده است را بطور خلاصه مرور کنیم. این روش ها همگی بطور مستقیم و یا غیر مستقیم در این پروژه بکار رفته‌اند و مرور آنها مفید خواهد بود.

۱ یافتن ابیات مشابه با یک بیت

یافتن روابط میان ابیات مهمترین گام در پیاده‌سازی سامانه بازیابی ماست.حال برای اینکار باید نمایش مناسبی برای هر بیت پیدا کنیم که در آن ابیات مشابه نمایش مشابهی نیز داشته باشند. روش‌ها زیادی برای نمایش ابیات وجود دارد که در این قسمت می‌خواهیم ۴ مورد آنها را بررسی کنیم. این روش‌ها این امکان را برای ما فراهم میکنند تا بتوانیم با داشتن یک بیت ورودی (کوئری) ابیات مشابه آن را برای اساس کلمات بکار رفته آن و یا معنا بیت پیدا کنیم

۱.۱ روش **boolean**

در این روش به هر بیت یک بردار نسبت داده می‌شود که درایه ام آن وجود کلمه ام دیکشنری در این بیت نشان می‌دهد. از آنجایی که در این روش تعداد کلمات بکار رفته و معنا آن لحاظ نشده و صرفا وجود هر کلمه بررسی می‌شود، ابیاتی که بردارهای مشابی دارند کلمات مشترک بیشتری با هم دارند.

۲.۱ روش **tfidf**

از این روش به عنوان نسخه بهبود یافته روش boolean می‌توان یاد کرد چرا در این شیوه نمایش ابیات فرض می‌شوند کلماتی که در ابیات بیشتر تکرار می‌شوند اهمیت بیشتری دارند درحالی که آن‌هایی که در تعداد بیشتری بیت تکرار می‌شوند اهمیت کمتری دارند. دومین مورد مطرح شده همان idf است که فرض میکند میزان اهمیت هر کلمه با لگاریتم معکوس نسبت ابیات دارای کلمه مورد نظر به همه ابیات رابطه مستقیم دارد. در این روش ابیاتی که بردارهای مشابهی دارند معمولا شامل نسبت یکسانی از کلمات کلیدی هستند.

۳.۱ روش **word-embedding**

در این شیوه نمایش ابیات علاوه بر خود کلمات معنای آنها هم درنظر گرفته می‌شود. به این صورت که برای هر کلمه نمایشی برداری درنظر می‌گیریم که در آن کلمات هم معنا بردارهای مشابهی با هم دارند. به این نمایش کلمات word-embedding می‌گویند و قصد داریم با استفاده از آن نمایش مناسبی برای ابیات ارائه دهیم. حال میدانیم که می‌توانیم معنای هر بیت را از کلمات بکار رفته در آن مشخص کنیم ولی مسئله این است که هر کلمه در نمایش نهایی ما به چه میزان اهمیت دارد. اگر به روش قبلی یعنی tfidf دقت کنیم میبینیم که چنین معیاری در آن معرفی شده است. اینجا نیز مشابه قسمت قبل میتوانیم از idf کلمات برای مشخص کردن میزان اطلاعاتی که هر کلمه به ما می‌دهد و بطور کلی میزان اهمیت آن استفاده کنیم. پس در نهایت نمایش برداری هر بیت را به صورت مجموع نمایش برداری کلمات ضرب در idf هر کلمه تعریف میکنیم. ابیاتی که در این روش نمایش مشابهی دارند معولا کلمات مترادفی با هم دارند که باعث می‌شد معنای ابیات نیز مشابه یکدیگر باشند. حال اگر نمایش کلمات بی‌نقص باشند ترکیب خطی آنها فارغ از خود کلمات بکار رفته معنای بیت را می‌توان نشان دهد ولی معمولا چنین نیست و وابستگی به کلمات باقی می‌ماند. برای کاهش این وابستگی و نزدیکتر شدن به معنای ابیات از روش دیگری استفاده میکنیم که در ادامه توضیح داده می‌شود.

۴.۱ روش sent-embedding

استفاده از مدل‌های پیچیده زبانی و بطور خاص ترنسفورمر ها این امکان را به ما می‌دهد تا بتوانیم نمایشی برای ابیات پیدا کنیم که کمترین وابستگی به کلمات خود داشته باشند. در واقع به دنبال نمایشی هستیم که معنای خالص ابیات را نشان دهد. در چنین نمایشی بردارهای مشابه نشان دهنده مفهوم و معنای مشابه هستند. این بالاترین سطح نمایش انتزاعی است که در آن همگی ابیاتی که درباره "مدح و ستایش خدا" هستند مشابه و با بردارهای نشان دهنده "دلاوری و پهلوانی" تفاوت زیادی دارند.

۲ طبقه‌بندی ابیات شاهنامه

شاهنامه شامل داستان های زیادی است که بعضی از آنها شامل هزاران بیت می‌شوند و روایت کننده داستان های اسطوره ای آن هستند. حال قصد داریم یک مدل طبقه‌بندی ارائه کنیم که بتواند تشخیص دهد که هر بیت به کدام یک از داستان های شاهنامه تعلق دارد. البته اینکار فقط و فقط برای داستان هایی امکان پذیر است که به تعداد کافی بیت در اختیار داشته باشند. پس پیش از هر چیزی ۱۰ داستان بلند شاهنامه را به عنوان ورودی های مدل در نظر میگیرم. حال برای اینکه کارمان ساده‌تر باشد یکی از مدل‌های از پیش آموزش داده شده در بستر huggingface را انتخاب کرده و با استفاده از ابیات شاهنامه آن را بهبود می‌دهیم. حال میتوانیم مدل را آموزش دهیم ولی پیش آن باید توجه داشته باشیم که تعداد ابیات داستان‌های شاهنامه یکسان نیستند و این میتواند روی یادگیری مدل تاثیر منفی بگذارد. برای جلوگیری از چنین تاثیری ۲ راه حل وجود دارد. می‌توانیم داده‌های ورودی را resample کنیم تا تعداد ابیات تمامی داستان‌ها یکسان شود یا در loss حساب شده این تفاوت در تعداد را درنظر بگیریم و به خطای هر داستان وزنی نسبت دهیم تا میزان اهمیت آنها نسبت به هم مشخص شود که مورد دوم با توجه به اینکه از ورود داده تکراری به مدل جلوگیری میکند مناسب‌تر است و زمان کمی برای آموزش نیاز دارد.

۳ خوشه‌بندی ابیات

در این قسمت قصد داریم ابیات شاهنامه را در خوشه‌هایی قرار دهیم بطوریکه ابیات کمترین فاصله را از هم داشته باشند. اما مقصود از کمترین فاصله چیست ؟ اگر از نمایش برداری مبتنی بر ترنسفورمرها استفاده کنیم ابیاتی که از نظر معنایی به هم نزدیکتر هستند بردارهای نزدیکتری نیز خواهند داشت پس قصد داریم ابیات شاهنامه را براساس معنای ابیات به خوشه‌هایی تقسیم کنیم. برای اینکار مراحل زیر را طی میکنیم.

الف نمایش برداری هر بیت را با استفاده از یک مدل ترنسفورمر از پیش آماده شده در بستر Hugginface بدست میاوریم.

ب برای کاهش حجم بردار ها و عملکرد بهتر الگوریتم های کلاسترینگ بعد آن را به زیر ۱۰ کاهش میدهیم. برای اینکار از مدل PCA استفاده میکنیم

ج با استفاده از الگوریتم KMeans خوشه ها را بدست میاریم. این الگوریتم سعی در پیدا کردن مرکز هر خوشه دارد و برای اینکار از یک حدس اولیه شروع کرده و به مرور آنرا بهتر و بهتر میکند.

```
1
2  # To transformer embedding
3  texts = self.dataset['text'].tolist()
4  self.embeddings = self.get_transformer_embeddings(texts)
5
6  # To low dimention
7  self.pca = PCA(n_components=pca_dim)
8  self.embeddings = self.pca.fit_transform(self.embeddings)
9
10 # clustring by KMeans algorithm!
11 self.kmeans = KMeans(n_clusters=k, max_iter=max_iter)
12 self.labels = self.kmeans.fit_predict(self.embeddings)
```