بخش‌های اصلی نسخه ی نمایشی پروژه ی متن کاوی

این پروژه براساس کتاب Text Analytics with Python و صرفا به عنوان demo ایجاد شده است.

نرمال سازی متن:

در این قسمت توابعی برای پاک سازی متن، حذف کاراکترها و کلمات اضافه و زائد، حذف کاراکترهای تکراری و تبدیل لغات به ریشه ی آنها آورده شده است.

- **تابع tokenize_text:** متن را به جملات و کلمات تشکیل دهنده تقسیم می کند.
- تابع remove_characters_after_tokenization: کاراکترهای اضافه مانند *%^@+) را بعد از انجام tokenization از متن حذف می کند.
- **تابع remove_characters_before_tokenization:** کاراکترهای اضافه مانند *%^@+) را قبل از انجام tokenization از متن حذف می کند.
- **تابع remove_stopwords:** کلماتی که دارای ارزش مفهومی نیستند و در معمولا در اکثر متن‌ها تکرار می شوند را از متن حذف می کند. a, any, anything, about, both, all, can نمونه هایی از stopWord ها هستند.
- **تابع remove_repeated_characters:** کاراکترهای تکراری را که احتمالا از تایپ کردن اشتباه ایجاد شده اند، از متن حذف می کند.

استخراج ویژگی‌ها[1]:

در این قسمت با استفاده از روش TFIDF[2]، لغاتی که دارای اهمیت هستند (از بین تمام مجموعه‌ی متن)، به عنوای ویژگی انتخاب می شوند تا هر متن به وسیله‌ی آن‌ها نمایش داده شود.
در روش TFIDF، لغات با توجه به تعداد تکرارشان در یک متن اهمیت بیشتری پیدا می کنند، اما به نسبت تعداد متن‌هایی که در آن‌ها تکرار شده‌اند اهمیتشان کاهش میابد.

- **تابع bow_extractor:** این تابع یک مجموعه متن[3] دریافت کرده و ماتریس ویژگی‌ها[4] و یک شی vectorizer را برمی گرداند. ماتریس ویژگی‌ها همان تعداد تکرار کلمات در متن است (Term Frequency) و vectorizer یک شی است که برای نمایش متن‌های جدید بر اساس ویژگی‌های استخراج شده از Corpus استفاده می شود
- **تابع tfidf_transformer:** این تابع همانند تابع bow_extractor عمل می کند با این تفاوت که ماتریس ویژگی‌ها براساس TFIDF محاسبه می‌شود و نه تنها TF.

جست و جوی متن:

در این قسمت، یک عبارت، در مجموعه‌ای ۴۰۰ متنی جست و جو شده و ۵ مورد از متن‌هایی که با عبارت مورد جست و جو[5] ارتباط بیشتری دارند، نمایش داده می شوند. جست و جو براساس شباهت کسینوسی محاسبه می شود.
مدل فضای برداری[6] استفاده شده در این قسمت بر اساس TFIDF است. ابتدا Corpus نرمال شده است و سپس مدل و ماتریس ویژگی‌ها ایجاد شده است.
از مزیت‌های این پیاده سازی می توان به جست و جو برای شکل‌های مختلف کلمات اشاره کرد. به این ترتیب که عبارت مورد جست و جو ابتدا نرمال می شود و طی نرمال شدن، با استفاده از عمل Lemmatization، هر کلمه ی آن به شکل لغت پایه‌ی خود تبدیل می شود.

- **تابع get_corpus:** این تابع نام فایل Corpus را به صورت رشته دریافت کرده، فایل را پردازش می‌کند و corpus را به صورت لیستی از متن‌ها برمی‌گرداند

---

[1] Feature extraction
[2] Term Frequency Inverse Document Frequency
[3] Corpus
[4] Feature Matrix
[5] Query
[6] Vector Space Model

- **تابع search:** این تابع یک Corpus را دریافت کرده و با دریافت کوئری از کاربر، به جست و جوی متن‌های مرتبط می پردازد. تا وقتی که کوئری جدید وارد شود جست و جوی جدید، ادامه می یابد، در صورت فشردن دکمه ی Enter (بدون نوشت کوئری) جست و جو خاتمه می یابد.

مدل سازی موضوع[7]:

در این بخش به استخراج موضوعات[8] یا مفاهیم[9] مختلفِ موجود در اسناد پرداخته شده است. وقتی یک مجموعه متن شامل اسنادی با مفاهیم یا موضوعات مختلف باشد، می توان با استفاده از روش‌های ریاضی مانند تجزیه‌ی ماتریس ها، گروه‌های یا خوشه‌هایی از کلمه‌ها[10] ایجاد کرد که از هم مجزا و قابل تشخیص باشند. این گروه‌ها یا خوشه‌ها موضوعات یا مفاهیم را تشکیل می‌دهند. از این مفاهیم می توان برای تفسیر زمینه‌ی اصلی Corpus استفاده کرد و همینطور بین کلماتی که به صورت همزمان در اسناد مختلف تکرار می شوند، ارتباط مفهومی ایجاد کرد.
پیاده سازی صورت گرفته در این بخش بر اساس سه روش SVD[11] و LSI[12] و NMF[13] صورت گرفته است.

مثال پیاده سازی شده در این قسمت بر روی یک Corpus با 8 متن، اجرا شده است:

```
toy_corpus = [
        "The fox jumps over the dog",
          "The fox is very clever and quick",
          "The dog is slow and lazy",
          "The cat is smarter than the fox and the dog",
          "Python is an excellent programming language",
          "Java and Ruby are other programming languages",
          "Python and Java are very popular programming languages",
          "Python programs are smaller than Java programs"]
```

همان طور که مشخص است 4 متن اول راجع به حیوانا و 4 متن آخر راجع به زبان برنامه نویسی می باشد.
با استفاده از هر 3 روش، کلمات مربوط به حیوانات در یک گروه و کلمات مربوط به زبان‌های برنامه نویسی در گروه دیگر قرار گرفته اند.
به این ترتیب دو مفهوم جدا از هم به خوبی مجزا شده اند.

LSI Result

Topic #1 with weights
[('dog', 0.72), ('fox', 0.72), ('jump', 0.43), ('smarter', 0.34), ('cat', 0.34), ('quick', 0.23), ('clever', 0.23), ('slow', 0.23), ('lazy', 0.23)]

Topic #2 with weights
[('programming', -0.73), ('language', -0.73), ('python', -0.56), ('java', -0.56), ('popular', -0.34), ('excellent', -0.33), ('ruby', -0.33), ('program', -0.21)]

[7] Topic Modeling
[8] Topics
[9] Concepts
[10] Term
[11] Singular Value Decomposition
[12] Latent Semantic Indexing
[13] Non Negative Matrix Factorization