به نام خدا

موضوع پروژه

پیشنهاد فیلم

نام و نام خانوادگش اعضای گروه

فرزانه ولایی شکن ـ محبوبه کریمی

زمستان 1404

گزارش پروژه

انتخاب مسیر

مسیر انتخاب‌شده برای این پروژه، حوزه پردازش زبان طبیعی بخش **Similarity search** در زمینه ی سیستم‌های توصیه‌گر (Recommender Systems) است. با توجه به رشد روزافزون پلتفرم‌های محتوایی و نیاز کاربران به دریافت پیشنهادهای شخصی‌سازی‌شده، طراحی یک سیستم پیشنهاددهنده فیلم به‌عنوان مسئله‌ای کاربردی و مرتبط با مباحث هوش مصنوعی انتخاب گردید.

در این پروژه از رویکرد Content-Based Filtering استفاده شده است؛ بدین معنا که پیشنهادها بر اساس ویژگی‌های محتوایی فیلم‌ها (ژانر، کلیدواژه‌ها، عوامل تولید و ...) تولید می‌شوند.

تعریف مسئله

مسئله اصلی پروژه طراحی سیستمی است که بتواند با دریافت نام یک فیلم، چند فیلم مشابه را بر اساس ویژگی‌های محتوایی آن پیشنهاد دهد.

چالش اصلی در این مسئله شامل موارد زیر است:

- استخراج و ترکیب ویژگی‌های متنی مرتبط با هر فیلم
- تبدیل داده‌های متنی به نمایش عددی قابل پردازش
- محاسبه میزان شباهت بین فیلم‌ها
- ارائه نتایج به کاربر از طریق یک رابط کاربری ساده

هدف نهایی، ارائه سیستمی است که بتواند شباهت معنایی بین فیلم‌ها را شناسایی کرده و پیشنهادهای مرتبط ارائه دهد.

معرفی دیتاست

در این پروژه از دیتاست فیلم‌ها شامل دو فایل اصلی استفاده شده است:

1. **movies.csv**

شامل اطلاعاتی نظیر: شناسه فیلم، عنوان، ژانرها، کلیدواژه‌ها، خلاصه داستان می باشد.

2. **credits.csv**

شامل اطلاعات بازیگران، کارگردان و سایر عوامل تولید است.

این دو فایل بر اساس شناسه فیلم (movie_id) با یکدیگر ادغام شده‌اند تا اطلاعات کامل هر فیلم در یک ساختار واحد قابل استفاده باشد.

ساختار ریپازیتوری

```
├── data/
│   ├── raw/              # movies.csv, credits.csv
│   └── processed/         # movies_merge.csv (output of preprocessing)
├── notebooks/
│   └── EDA.ipynb          # Exploratory data analysis
├── src/
│   ├── preprocessing/
│   │   └── preprocessing.ipy  nb
│   ├── training/
│   │   ├── training.ipynb
│   │   └── recommender.py # recommend(movie) used by UI
│   └── ui/
│       └── ui.ipynb       # Tkinter GUI
├── requirements.txt
└── README.md
```

پیش‌پردازش داده‌ها

در فایل نوت بوک پیش پردازش (`preprocessing.ipynb`) مراحل زیر انجام می‌شود.

بارگذاری داده

```
import pandas as pd

movies_df = pd.read_csv("../../data/raw/movies.csv")
credits_df = pd.read_csv("../../data/raw/credits.csv")
```

ادغام داده‌ها

ادغام فایل‌های فیلم و عوامل براساس شناسه

پاکسازی و مرتب سازی متن

1. حذف سطرهای خالی و بررسی تکراری داشتن

```
movies_merge_df=movies_df[['movie_id','title','overview','genres','keywords','cast','crew']]
movies_merge_df.dropna(inplace=True)
movies_merge_df.duplicated().sum()
```

۲. استخراج اطلاعات از ساختار JSON مانند (برای ژانر، کلیدواژه و بازیگران و ..)

```
import ast
def convert(obj):
        l=[]
        for i in ast.literal_eval(obj):
                l.append(i['name'])
        return l
movies_merge_df['genres']=movies_merge_df['genres'].apply(convert)
movies_merge_df['keywords']=movies_merge_df['keywords'].apply(convert)
```

۳. انتخاب تعداد محدودی از بازیگران اصلی برای کاهش ابعاد داده

۴. استخراج نام کارگردان

```
def convert3(obj):
        l=[]
        counter=0
        for i in ast.literal_eval(obj):
                if counter !=3:
                        l.append(i['name'])
                        counter+=1
                else:
                        break
        return l
movies_merge_df['cast']=movies_merge_df['cast'].apply(convert3)
```

```
def fetch_director(obj):
```

۵. اتصال کلمات چندبخشی در بخش (ژانر، کارگردان، بازیگران و کلمات کلیدی)

```
movies_merge_df['genres']=movies_merge_df['genres'].apply(lambda x:[i.replace(" ","")for i in x])
movies_merge_df['keywords']=movies_merge_df['keywords'].apply(lambda x:[i.replace(" ","")for i in x])
movies_merge_df['cast']=movies_merge_df['cast'].apply(lambda x:[i.replace(" ","")for i in x])
movies_merge_df['crew']=movies_merge_df['crew'].apply(lambda x:[i.replace(" ","")for i in x])
```

۶. ترکیب ویژگی‌های متنی در یک ستون جدید به نام tags

۷. تبدیل تمام متن‌ها به حروف کوچک

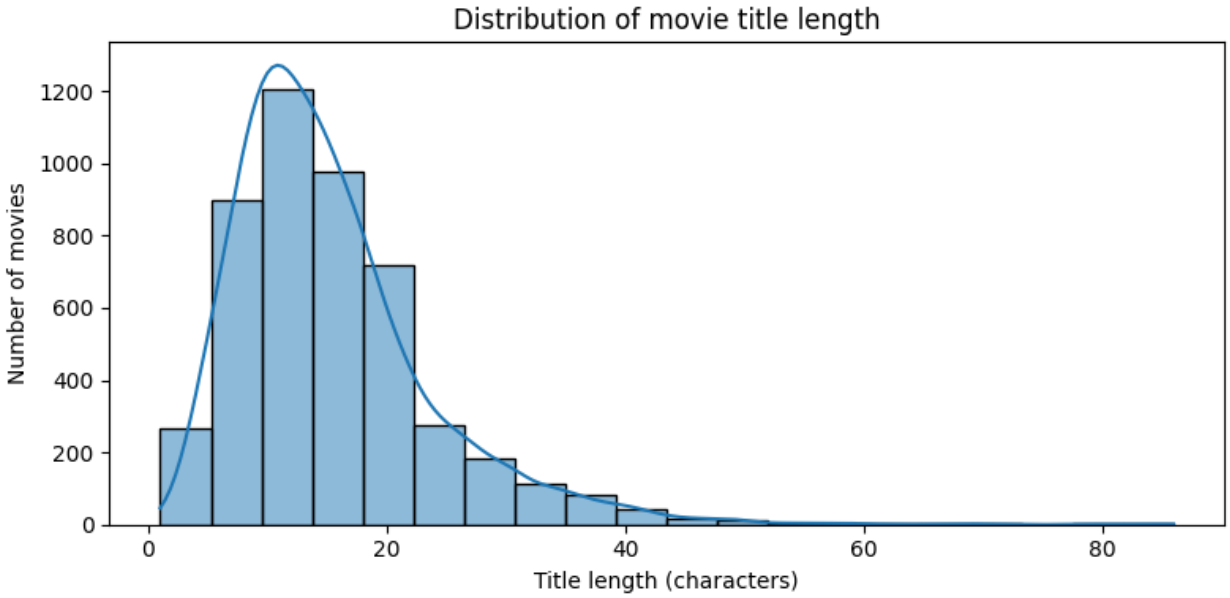
۸. حذف فاصله‌های اضافی و نرمال‌سازی متن

```
movies_merge_df['tags']=movies_merge_df['overview']+movies_merge_df['genres']+movies_merge_df['keywords']+movies_merge_df['cast']+movies_merge_df['crew']
new_df=movies_merge_df[['movie_id','title','tags']]
new_df['tags']=new_df['tags'].apply(lambda x:' '.join(x))
new_df['tags']=new_df['tags'].apply(lambda x:x.lower())
```

تحلیل داده و نمودارها

توزیع طول عنوان فیلم

این نمودار یک هیستوگرام است و توزیع طول عنوان فیلم‌ها (تعداد کاراکترها در هر عنوان) را نشان می‌دهد. این نمودار به ما کمک می‌کند تا ببینیم آیا عنوان‌ها کوتاه هستند یا بلند، چقدر توصیفی هستند و آیا عنوان‌های غیرمعمول (خیلی کوتاه یا خیلی بلند) وجود دارند که ممکن است موارد استثنا باشند یا خیر.

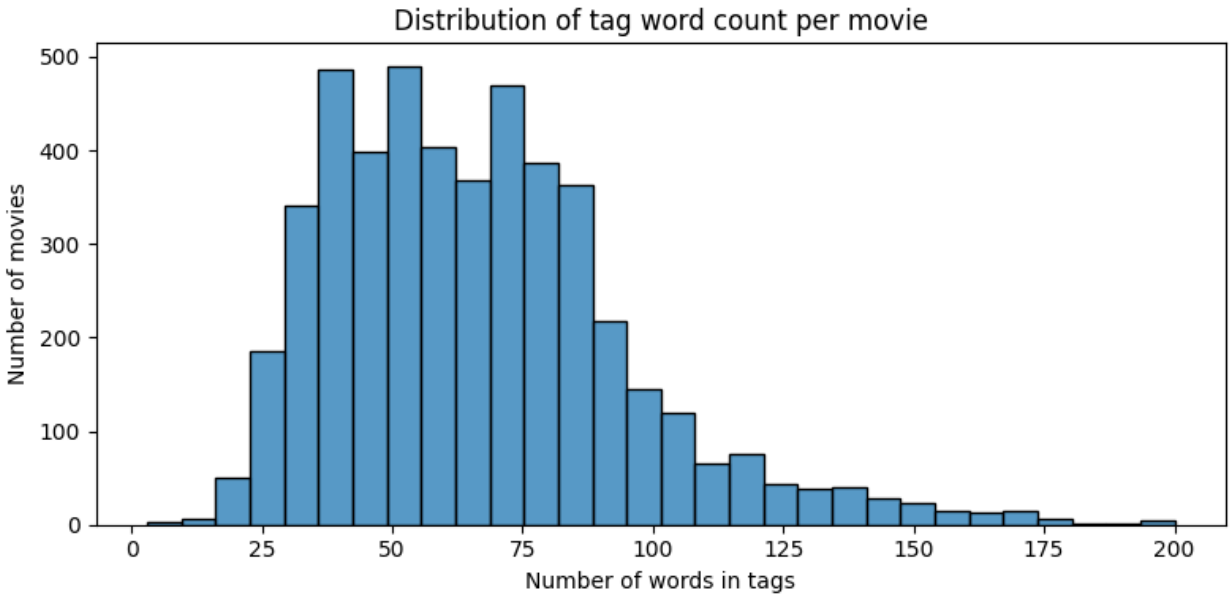


از روی این نمودار می‌توانیم ببینیم که:

1. بیشتر فیلم‌ها عنوان‌هایی با طول متوسط دارند (حدود 10 تا 25 کاراکتر).
2. تعداد کمی فیلم عنوان‌های بسیار کوتاه (زیر 5 کاراکتر) یا بسیار بلند (بیش از 30 کاراکتر) دارند.
3. این توزیع نشان می‌دهد که اکثر سازندگان فیلم تمایل دارند عنوان‌هایی با طول معقول و قابل‌حفظ انتخاب کنند.

توزیع تعداد کلمات برچسب در هر فیلم

این نمودار نشان می‌دهد که هر فیلم چند کلمه در فیلد «برچسب‌ها»ی خود دارد. مقادیر بالاتر به معنای متادیتاهای متنی بیشتر (ژانر، بازیگران، کلمات کلیدی و غیره) است. این توزیع به ما می‌گوید که چه سهمی از فیلم‌ها توضیحات برچسب کوتاه در مقابل طولانی دارند و به تشخیص ورودی‌های پراکنده یا بسیار متراکم کمک می‌کند.

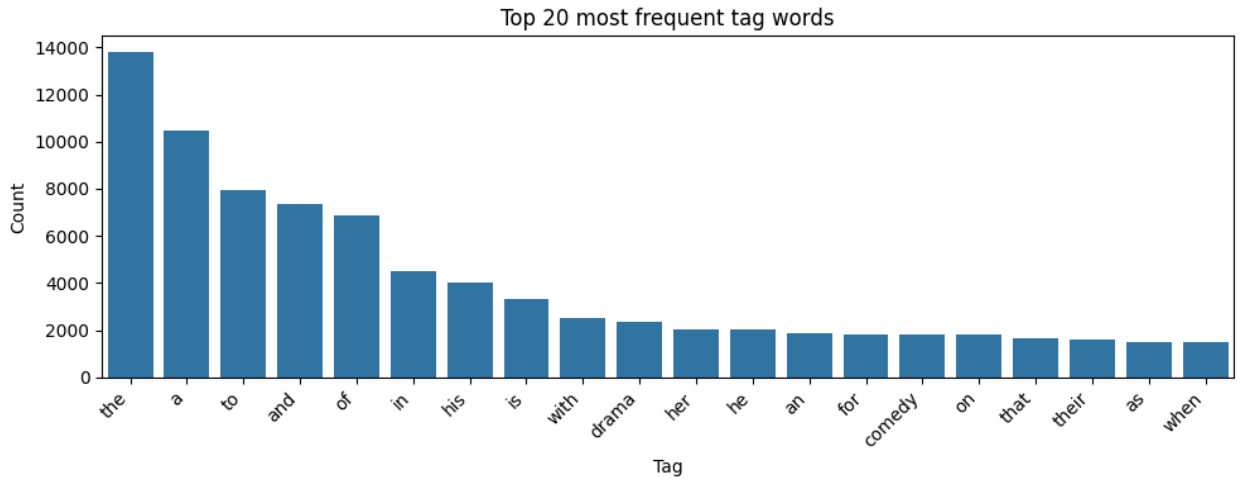


با مشاهده این نمودار می‌توانیم بفهمیم:

1. بیشتر فیلم‌ها دارای تعداد متوسطی از کلمات در بخش تگ خود هستند، به این معنی که توضیحات نه خیلی کوتاه و نه خیلی بلند دارند.
2. تعداد کلمات در تگ می‌تواند نشان‌دهنده میزان جزئیات و اطلاعات مرتبط با هر فیلم باشد.
3. قله نمودار نشان می‌دهد که چه تعداد کلمه در تگ برای اکثر فیلم‌ها رایج است و انحراف از آن قله می‌تواند فیلم‌هایی با توضیحات بسیار کم یا بسیار زیاد را نشان دهد.

۲۰ کلمه پرتکرار ستون برچسب

این نمودار ۲۰ کلمه رایج در ستون «برچسب‌ها» را در تمام فیلم‌ها فهرست می‌کند. این کلمات منعکس‌کننده مضامین و ژانرهای غالب در مجموعه داده‌ها (مثلاً اکشن، ماجراجویی، کمدی) هستند و مروری سریع بر موضوع کاتالوگ ارائه می‌دهند.

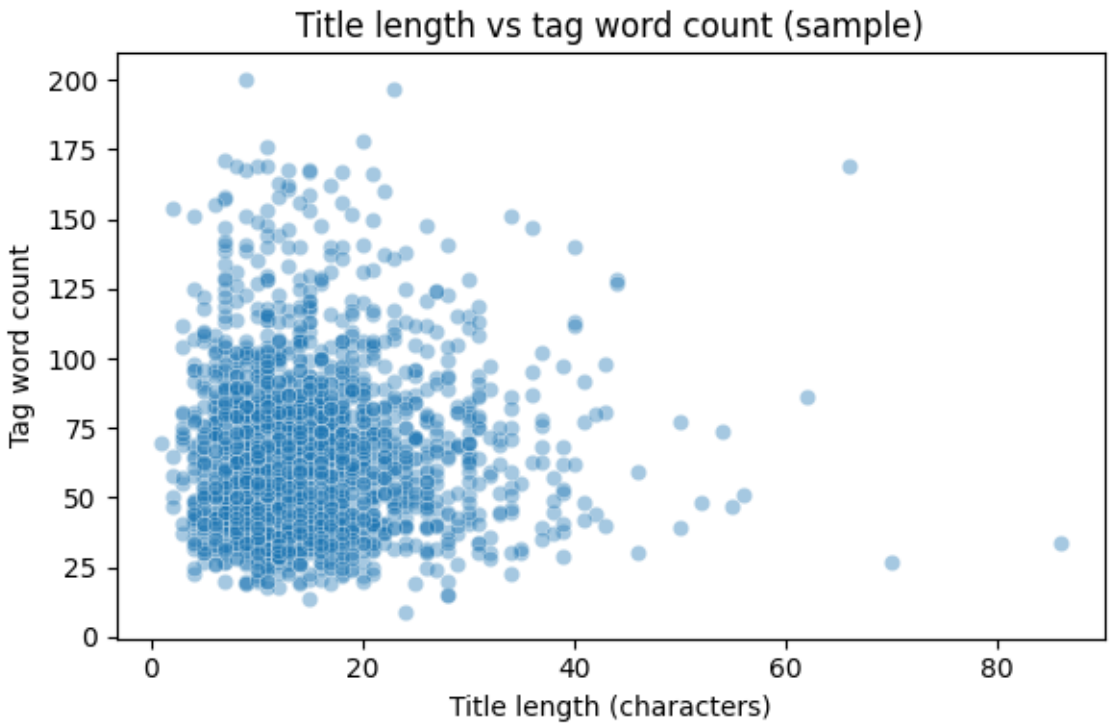


قادریم از روی نمودار بفهمیم:

1. بیشتر کلمات پرکاربرد، حروف اضافه، ضمایر و افعال رایج زبان انگلیسی هستند مانند ('the', 'a' 'to', 'and', 'of') این کلمات معمولاً اطلاعات معنایی خاصی در مورد ژانر یا محتوای فیلم نمی‌دهند.
2. کلمات کلیدی ژانر و موضوع: با این حال، کلماتی مانند 'drama' و 'comedy' نیز در میان ۲۰ کلمه برتر قرار دارند که نشان‌دهنده ژانرهای بسیار رایج در مجموعه داده فیلم‌ها هستند.
3. پاکسازی داده‌ها: حضور کلمات بسیار رایج و بدون معنای خاص (stop words) در لیست کلمات برتر، نشان می‌دهد که برای استفاده مؤثر از تگ‌ها در تحلیل یا مدل‌سازی (مانند سیستم‌های پیشنهاد دهنده)، نیاز به مرحله‌ای برای حذف این کلمات و تمرکز بر کلمات کلیدی معنایی‌تر داریم. این کار دقت مدل‌ها را افزایش می‌دهد.

رابطه بین طول عنوان و تعداد کلمات برچسب

این نمودار پراکندگی، طول عنوان فیلم (بر حسب کاراکتر) را با تعداد کلمات موجود در برچسب‌ها مقایسه می‌کند. این نمودار نشان می‌دهد که آیا عناوین طولانی‌تر تمایل به داشتن متن برچسب بیشتر (یعنی متادیتای بیشتر) دارند یا خیر. از یک نمونه تصادفی از فیلم‌ها استفاده می‌شود تا نمودار خوانا باقی بماند.



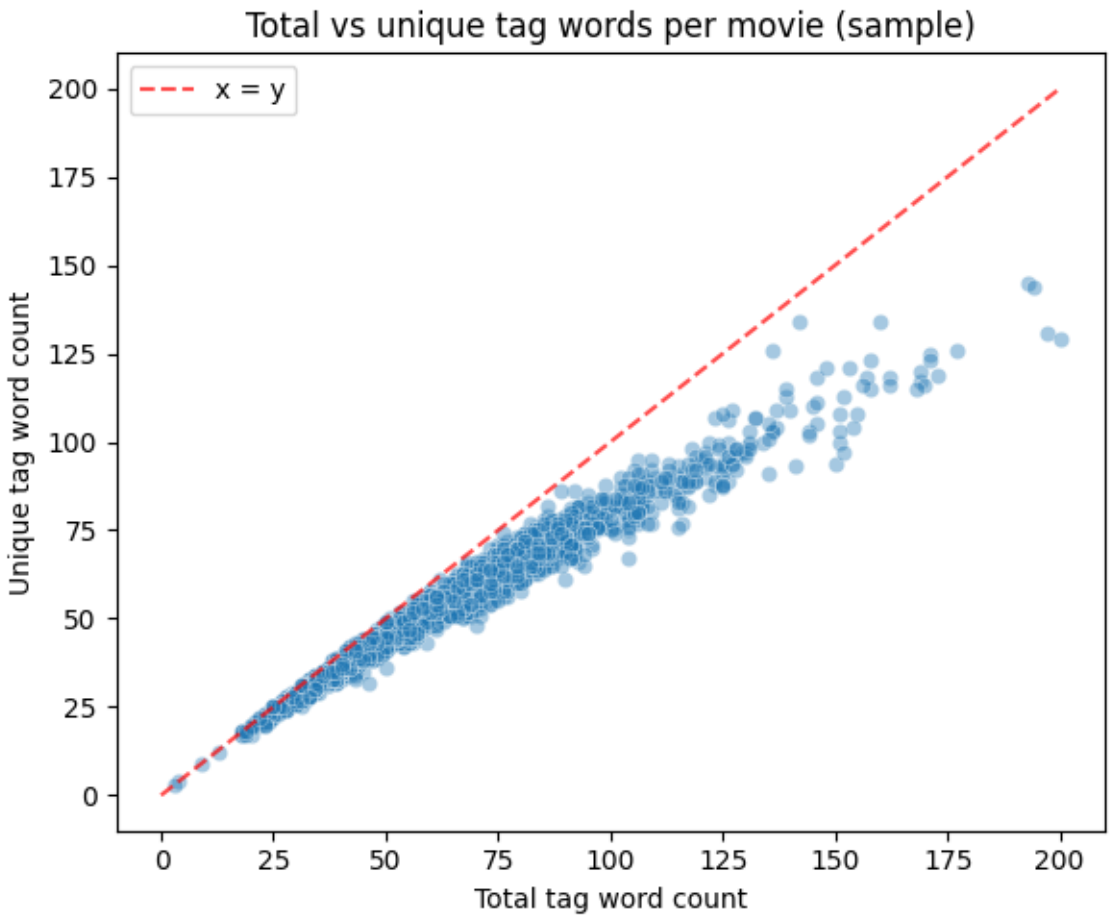
مواردی که می توان از نمودار بدست آورد:

1. عدم وجود همبستگی قوی: به نظر نمی‌رسد که رابطه خطی قوی بین طول عنوان فیلم و تعداد کلمات تگ وجود داشته باشد. یعنی فیلم‌هایی با عنوان‌های بلند لزوماً تگ‌های بیشتر یا کمتری ندارند و بالعکس.
2. توزیع نقاط: اکثر نقاط در قسمت‌های پایین‌تر نمودار متمرکز شده‌اند که نشان می‌دهد بیشتر فیلم‌ها هم عنوان‌های با طول متوسط و هم تگ‌های با تعداد کلمات متوسط دارند.
3. نقاط دورافتاده (Outliers) : ممکن است نقاطی وجود داشته باشند که عنوان بسیار کوتاه یا بلندی دارند اما تعداد تگ‌های آن‌ها بسیار متفاوت است (یا بالعکس). این نقاط می‌توانند فیلم‌های خاصی باشند که ارزش بررسی بیشتر را دارند.

این نمودار به ما کمک می‌کند تا بفهمیم آیا عنوان فیلم می‌تواند نشانه‌ای از میزان جزئیات متادیتا (تگ‌ها) باشد یا خیر. در این حالت، به نظر می‌رسد که این دو ویژگی مستقل از یکدیگر هستند.

رابطه بین تعداد کل کلمات برچسب و منحصر به فرد در هر فیلم

این نمودار تعداد کل کلمات برچسب را با تعداد کلمات منحصر به فرد در هر فیلم مقایسه می‌کند. نقاط نزدیک به قطر ($x \approx y$) به معنای تکرار کم و نقاط بسیار پایین‌تر از آن به معنای کلمات تکرار شده زیاد است. این تکرار می‌تواند بر نحوه وزن‌دهی به عبارات در مدل توصیه تأثیر بگذارد.



از این نمودار می‌توانیم بفهمیم:

1. میزان تکرار کلمات: هر چقدر یک نقطه از خط قرمز پایین‌تر باشد، به این معنی است که کلمات تگ آن فیلم بیشتر تکرار شده‌اند. برای مثال، اگر یک فیلم 20 کلمه در تگ‌هایش داشته باشد (محورx)، اما فقط 10 کلمه از آن‌ها منحصربه‌فرد باشند (محورy)، این نقطه پایین‌تر از خط x=y قرار می‌گیرد.
2. کیفیت متادیتا: تکرار زیاد کلمات ممکن است نشان‌دهنده filler words یا توصیفات تکراری باشد که اطلاعات جدیدی اضافه نمی‌کنند. در مقابل، تعداد بالای کلمات منحصربه‌فرد می‌تواند نشان‌دهنده متادیتای غنی‌تر و متنوع‌تر باشد.
3. تأثیر بر مدل‌های پیشنهاد دهنده: در مدل‌های پیشنهاد دهنده، تکرار کلمات می‌تواند بر وزن‌دهی کلمات (مثلاً با TF-IDF) تأثیر بگذارد. این نمودار به ما کمک می‌کند تا بفهمیم چقدر تکرار کلمات در مجموعه داده ما شایع است و آیا نیاز به روش‌هایی برای کاهش اثر تکرار داریم یا خیر.

مدل پایه

مدل پایه مورد استفاده در این پروژه شامل مراحل زیر است:

ریشه‌یابی واژگان (Stemming)

از Porter Stemming جهت کاهش اشکال مختلف یک واژه به ریشه مشترک آن استفاده شد.

Porter Stemmer یکی از روش‌های کلاسیک در پردازش زبان طبیعی است که با حذف پسوندهای متداول، واژگان را به فرم پایه تقلیل می‌دهد.

```
ps=PorterStemmer()
def stem(text):
    y=[]
    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)
new_df["tags"]=new_df['tags'].apply(stem)
```

پس از اعمال stemming بر روی ستون تجمیع‌شده تگها، داده‌ها جهت بردارسازی با استفاده از CountVectorizer آماده شدند.

تبدیل متن به بردارها

با استفاده از CountVectorizer در این مرحله الف) تعداد ویژگی‌ها به حداکثر ۵۰۰۰ ویژگی پرتکرار محدود شده است ب) Stop words حذف شدند.

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=5000 , stop_words='english')
vectors = cv.fit_transform(new_df['tags']).toarray()
```

این ماتریس در حقیقت به اندازه تعداد فیلم ها ردیف و به تعداد ۵۰۰۰ کلمه منحصربفرد موجود در فیلد تگ، ستون دارد و در درایه ها مقادیر تکرار این کلمات در هر ردیف قرار گرفته است. این ماتریس به مدل اجازه می‌دهد تا **فیلم‌هایی با موضوع مشابه** را کشف کند.

محاسبه شباهت

برای هر فیلم، شباهت **Cosine Similarity** با بقیه فیلم‌ها براساس بردارهای عددی بدست آمده از ستون تگ مربوط به هر فیلم محاسبه می‌شود:

```
from sklearn.metrics.pairwise import cosine_similarity
similarity=cosine_similarity(vectors)
```

این معیار زاویه بین دو بردار ویژگی را محاسبه می‌کند و عددی بین ۰ تا ۱ ارائه می‌دهد که نشان‌دهنده میزان شباهت است.

یافتن فیلم‌های مشابه

- یافتن ایندکس مربوط به عنوان فیلم دریافت شده
- استخراج بردار شباهت مربوط به آن ایندکس
- مرتب‌سازی شباهت بصورت نزولی
- انتخاب 5 فیلم برتر (به جز خود فیلم)

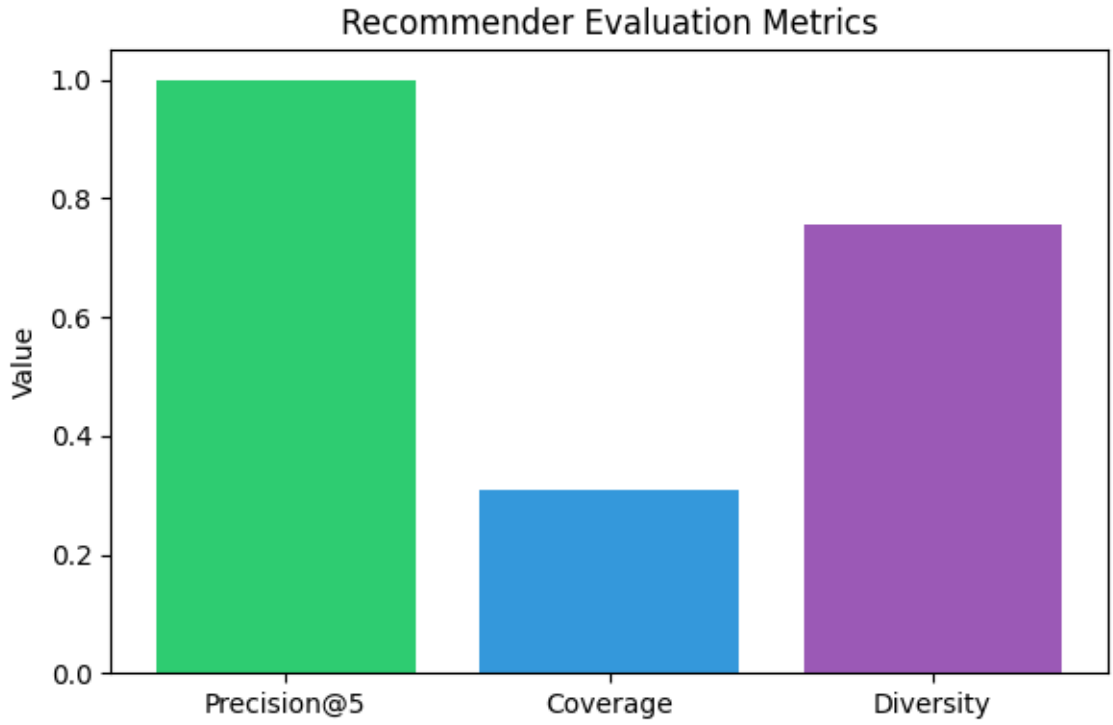
```
def recommend(movie):
    movie_index=new_df[new_df['title']==movie].index[0]
    distances=similarity[movie_index]
    movie_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:6]
    movies[]=
    for i in movie_list:
        movies.append(new_df.iloc[i[0]].title)
    return movies
```

نمایش به کاربر

نتایج به صورت لیست عناوین به کاربر در رابط Tkinter داده می‌شود.

ارزیابی

سیستم توصیه‌گر بر اساس تگ و شباهت کسینوسی روی نمونه‌ای تصادفی از ۵۰۰ فیلم از بین ۴٬۸۰۵ فیلم کاتالوگ ارزیابی شد و برای هر فیلم ۵ توصیه برتر تولید شد. دقت در ۵K= (Precision@۵) برابر ۱.۰۰ است؛ یعنی در هر لیست ۵تایی، همهٔ توصیه‌ها حداقل یک تگ با فیلم جستجو مشترک دارند و از نظر مرتبط‌بودن با تگ عالی عمل می‌کند. پوشش کاتالوگ (Catalog Coverage) حدود ۳۰.۹۱٪ است، یعنی تقریباً یک‌سوم کل فیلم‌ها در حداقل یک لیست توصیه ظاهر می‌شوند و توصیه‌ها فقط روی تعداد کمی فیلم متمرکز نیست. تنوع درون‌لیستی (Intra-list Diversity) با مقدار ۰.۷۶ نشان می‌دهد آیتم‌های هر لیست نسبتاً متنوع‌اند و شبیه هم نیستند. در مجموع، سیستم هم از نظر مرتبط‌بودن با تگ عالی است، هم پوشش قابل‌قبول دارد و هم تنوع لیست‌ها خوب است.



رابط گرافیکی(Tkinter)

رابط کاربری شامل(بصورت خلاصه):

- یک **فیلد ورودی** برای دریافت نام فیلم
- دکمه‌ای برای گرفتن پیشنهادات
- نمایش لیست پیشنهادی در بخش پایین یا جعبه متن

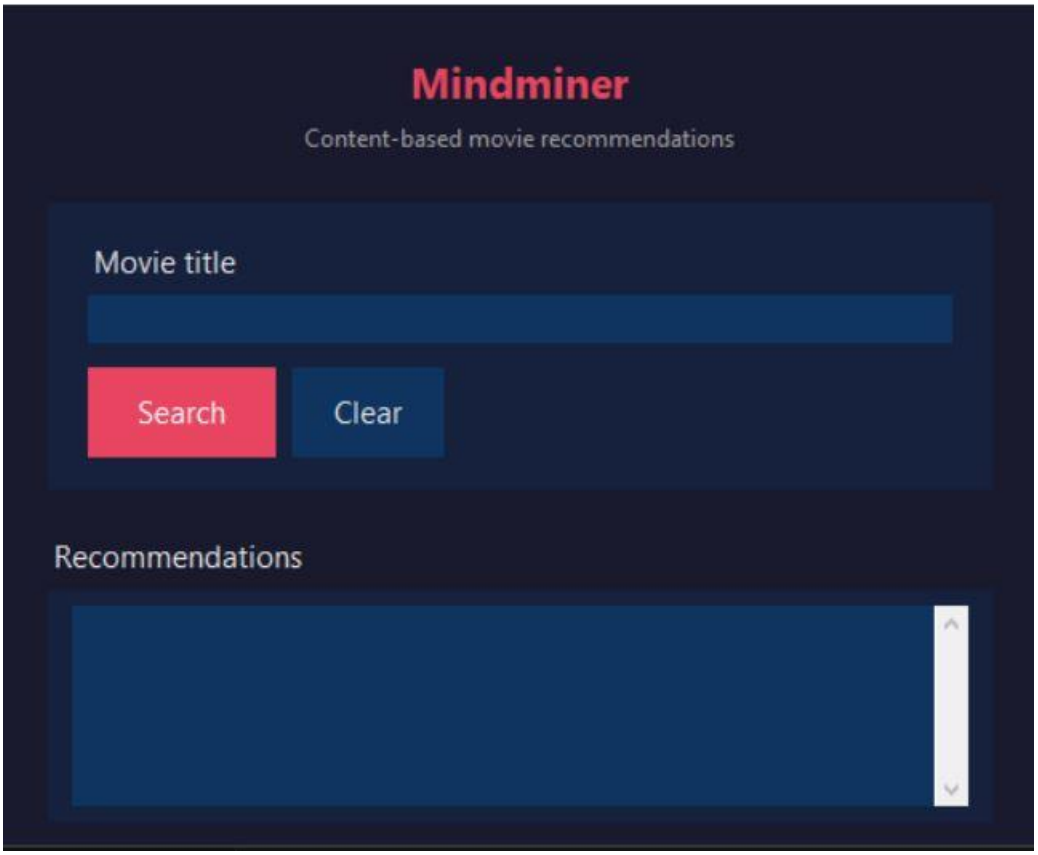
به صورت شماتیک:

```
import tkinter as tk
root = tk.Tk()
entry = tk.Entry(root)
suggest_btn = tk.Button(btn_frame, text="Search", command=suggest_movies)
listbox = tk.Listbox(root)
```

تعریف تابع **suggest_movies** و صدا زدن تابع **recommend** :

```
def suggest_movies():
    query = entry.get().strip()
    if not query:
        results_list.delete(0, tk.END)
        results_list.insert(tk.END, "Enter a movie title above and press Search or Enter.")
        return
    status_var.set("Searching...")
    root.update_idletasks()
    try:
        suggested = recommend(query)
        results_list.delete(0, tk.END)
        for m in suggested:
            results_list.insert(tk.END, m)
        status_var.set(f"Found {len(suggested)} result(s). Search is case-insensitive.")
    except Exception as e:
        results_list.delete(0, tk.END)
        results_list.insert(tk.END, f"Error: {e}")
        status_var.set("Error occurred.")
    finally:
        root.update_idletasks()
```

تصویر رابط کاربری:



منابع

۱. مستندات رسمی کتابخانه‌های:

- Pandas
- Scikit-learn
- NLTK
- Tkinter

۲. منابع آموزشی مرتبط با سیستم‌های توصیه‌گر

۳. دیتاست فیلم‌های عمومی منتشرشده برای اهداف آموزشی

جمع‌بندی

در این پروژه یک سیستم توصیه‌گر فیلم مبتنی بر محتوا طراحی و پیاده‌سازی شد. با استفاده از پیش‌پردازش داده‌های متنی، بردارسازی و محاسبه شباهت کسینوسی، امکان پیشنهاد فیلم‌های مشابه فراهم گردید.

این پروژه نشان داد که حتی با استفاده از مدل‌های پایه و روش‌های کلاسیک پردازش متن، می‌توان سیستمی کاربردی و قابل استفاده طراحی نمود.