به نام خدا

دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر

استاد درس: دکتر محمد رحمتی

زمستان ۱۳۹۹

طراحی یک سیستم توصیه گر بر مبنای شاخصیت

سروش مهدی
شماره دانشجویی: ۹۹۱۳۱۰۵۰

گزارش پیاده سازی پروژه نهایی درس شناسایی اماری الگو

فهرست مطالب

۱ مقدمه

یک سیستم توصیه گر با تحلیل رفتار کاربر خود، اقدام به پیشنهاد مناسب‌ترین اقلام (داده، اطلاعات، کالا و...)می‌نماید با افزایش تجارت های الکترونیک نیاز و استفاده از این سیستم ها روز به روز بیشتر میشود و امروزه به یکی از مهم ترین کاربرد های هوش مصنوعی در جهان واقعی تبدیل شده بطوریکه یکی از اولین مسابقات مهم دنیای هوش مصنوعی طراحی یک سیستم توصیه گر برای توصیه فیلم از طرف نتفلیکس بود.
پالایش گروهی یا Collaborative filtering یک روش مهم و پرکاربرد برای سیستم های توصیه گر میباشند.اما در این روش ها هر چه ماتریس داده ها خلوت تر باشد نتایج به مراتب ضعیف تر میشوند.در این مقاله یک ایده که از دنیای روانشناسی الهام گرفته شده برای طراحی این سیستم ها استفاده میشود که مشکلات این روش ها به خصوص مشکل خلوت بودن داده ها را حل کند. در واقع سیستم های پالایش گروهی دو دسته هستند دسته اول که مبتنی بر کاربر هستند سعی میکنند تا کاربر های شبیه به کاربر هدف را پیدا کرده و از روی نظرات ان ها به کاربر هدف پیشنهاد دهند. دسته دوم که مبتنی بر ایتم هستند بر اساس ایتم هایی که کاربر هدف به ان ها رای داده ایتم هایی را که ممکن است کاربر هدف به ان ها علاقه مند باشد پیدا میکنند.
در علم روانشناسی دانشمندان متوجه شده اند که اشیا در دسته های مختلف درجه های شاخصیت متفاوت دارند. برای مثال مردم ممکن است یک گنجشک را بیشتر از یک پنگوئن پرنده در نظر بگیرند یا تایتانیک به عنوان یک فیلم رمانتیک شاخص در نظر گرفته میشود. به همین ترتیب اگر یک سیستم توصیه گر فیلم را در نظر بگیریم فیلم های مختلف میتوانند درجه شاخصیت و تعلق متفاوت در دسته های مختلف داشته باشند مثلا یک فیلم ممکن است درجه تعلق بالایی به فیلم های جنگی و پایینی به فیلم های رمانتیک داشته باشد. همچنین این قضیه برای کاربران مختلف نیز صدق میکند برای مثال یک کاربر ممکن است در دسته کسانی که فیلم های دلهره اور دوست دارند درجه تعلق بالایی داشته باشد اما در دسته کسانی که فیلم های رمانتیک دوست دارند درجه تعلق پایینی داشته باشد.

۲ راه حل پیشنهادی

با قرض گرفتن این ایده از دنیای روانشناسی ما میخواهیم یک سیستم توصیه گر طراحی کنیم. از اینجا به بعد فرض میکنیم که این سیستم توصیه گر برای توصیه فیلم به کاربران میباشد.
نحوه عملکرد این سیستم به این صورت است . ابتدا تمام ایتم ها را به دسته های مختلف خوشه بندی میکنیم که به این دسته ها Items group میگوییم . برای مثال میتوانیم تمام فیلم ها را بر اساس ژانر دسته بندی کنیم. سپس متناظر با هر یک از این دسته ها یک گروه از کاربر ها ایجاد میکنیم. دقت کنیم که هر یک از این دسته ها چه برای کاربر ها و چه برای ایتم ها مجموعه های فازی هستند و هر فیلم یا کاربر میتواند در دسته های مختلف درجه تعلق ها مختلف داشته باشد.سپس بر اساس درجه های تعلق کاربر ها به دسته های مختلف کاربر های مشابه را پیدا میکنیم و همسایه های یک کاربر را انتخاب میکنیم . سپس با استفاده از نظرات این همسایه ها سعی میکنیم برای کاربر مورد نظر نظراتش را پیشبینی کنیم.
این روش نسبت به متود های قبلی دقت بالاتری دارد . همچنین برتری مهم این روش به روش های قبلی حل مشکل خلوت بودن داده هاست که مخصوصا در سیستم های توصیه گر یک مشکل مرسوم میباشد. فرض کنیم دو نفر داریم که نفر اول به یک فیلم جنگی نمره بالایی داده است و به یک فیلم رمانتیک نمره پایینی داده است و نفر دوم نیز همینطور اما فیلم هایی که این دو نفر به ان ها رای داده اند مشترک نیستند. در روش های قبلی این دو کاربر به هیچ عنوان شبیه در نظر گرفته نمیشدند اما در روش این مقاله با توجه به خوشه بندی فیلم ها در مرحله اول این دو کاربر شباهت بالایی خواهند داشت. و در نتیجه تاثیر خلوت بودن داده ها بر عملکرد نهایی کمتر میشود.

۳ معیار شاخصیت

در روانشناسی شناختی شاخصیت یک شی یا ماهیت در یک دسته به این که چقدر این ماهیت در ان دسته میتواند یک نمونه به حساب بیاید بستگی دارد. که این معیار بستگی به ویژگی های چشمگیری که توسط اکثر اعضای ان دسته به اشتراک گذاشته میشوند بستگی دارد.که در حالت کلی این ویژگی ها میتوانند برای تعریف ان دسته یا موضوع غیرضروری نیز باشند.
در دیدگاه پروتوتایپی یک موضوع یا دسته میتواند با یک پروتوتایپ که ویژگی های چشمگیر ان دسته را دارد نمایش داده شود همچنین میزان شاخصیت یا تعلق یک شی به ان دسته نیز میتواند با میزان شبیه بودن ان شی به پروتوتایپ دسته اندازه گیری شود.برای مثال پروتوتایپ دسته پرندگان احتمالا ویژگی پرواز کردن را دارد پس پرنده ای که نمیتواند پرواز کند میزان کمتری به دسته پرندگان تعلق دارد نسبت به پرنده ای که پرواز میکند. میتوان گفت پروتوتایپ یک دسته به عنوان بهترین مثال ان دسته در نظر گرفته میشود.

۴ جزییات راه حل

فرض کنید مجموعه کاربر ها را با U و مجموعه ایتم ها را با O نمایش بدهیم. ایتم ها که در این مورد فیلم ها هستند میتوانند به گروه ها ی مختلف خوشه بندی شوند برای مثال میتوان فیلم ها را بر اساس ژانر دسته بندی کرد و هر فیلم با درجه متفاوت به دسته های مختلف تعلق دارد.برای این منظور در این پیاده سازی من از دیتاست genome tag استفاده کردم . در این دیتاست به ازای فیلم های مختلف و برچسب های مختلف ما متوانیم درجه ارتباط بین فیلم و برچسب را مشاهده کنیم . در واقع هر برچسب نماینده یک خوشه میباشد و هر فیلم در هر خوشه یک درجه تعلق دارد.این دیتاست از دیتاست های movielens میباشد.

در مقاله اشاره شده که این خوشه بندی به محدوده کار بستگی دارد به هر خوشه یک item group میگوییم که یک مجموعه فازی هست و هر ایتم در این مجموعه یک درجه تعلق بین صفر و یک دارد.
کابرانی که نظرات مشابهی درباره یکی از این خوشه های فیلم ها دارند نیز میتوانند یک مجموعه فازی تشکیل دهند که به این مجموعه ها user group میگوییم و هر کاربر یک درجه تعلق برای هر کدام از این مجموعه ها دارد که بین صفر و یک هست. سپس برای هر کاربر یک بردار شاخصیت تشکیل میدهیم که اعضای ان میزان تعلق کاربر به هر یک از این user group ها هستند. و با قرار دادن این بردار ها به صورت سطری ماتریس شاخصیت کاربر ها را تشکیل میدهیم که ان را M می نامیم. برای محاسبه درجه تعلق کاربر به هر یک از user group دو مقدار زیر را محاسبه میکنیم.

$$\frac{\sum_{y=1}^{n} w_{x,y}.R_{i,y}}{n.R^{max}} \tag{1}$$

در این عبارت n تعداد ایتم هایی از x item group ام هست که کاربر i ام به ان ها رای داده است و w میزان تعلق هر ایتم به این دسته هست و R نیز رای این کاربر به هر یک از ایتم های این دسته است. همچنین R در مخرج نیز میزان حداکثر مقدار رای مجاز است.

$$\frac{N_{x,i}}{N_i} \tag{2}$$

در این عبارت صورت برابر تعداد ایتم هایی است که کاربر i ام در x item group ام به ان ها رای داده و مخرج برابر تعداد تمامی رای های کاربر i ام هست. سپس برای بدست اوردن شاخصیت یک کاربر در یک دسته از این دو مقدار میانگین میگیریم.
برای اندازه گیری شباهت بردار های کاربر ها ما از سه معیار Distance-based-similarity و Cosine-based-similarity و Correlation-based-similarity بهره میگیریم

برای تعیین همسایه های یک کاربر ما یک حد استانه که ان را با gamma نشان میدهیم نعیین میکنیم و بردار های که شباهتشان به بردار مورد نظر بزرگ تر از گاما باشد را به مجموعه بردارهای همسایه بردار مورد نظر اضافه میکنیم.
در نهایت برای پیش بینی امتیاز یک کاربر بر روی یک ایتم از فرمول زیر استفاده میکنیم.

$$R(U_i, O_j) = \frac{\sum_{U_x \in N_i} R(U_x, O_j).Sim(U_x, U_i)}{\sum_{U_x \in N_i} Sim(U_x, U_i)} \quad (3)$$

که N مجموعه همسایه های کاربر i ام میباشد.

۵ بررسی و توضیح کدها

این مقاله برای تست راه حل خود از دیتاست های ۱۰۰ هزارتایی MovieLens و دیتاست ست نتفلیکس استفاده کرده بود. بنده از دیتاست movieLens-latest-small استفاده کردم که نسخه اپدیت شده دیتاست اول میباشد. همچنین برای استفاده از تگ ها از دیتاست tag-genome که توسط movielens منتشر شده استفاده کرد.
دو قسمت اول کد در فایل main هستند

```
tag_popularity_trashhold = 80
tag_relevance_trashhold = 0.08
test_size = 0.1

#data preproccesing
ratings = pd.read_csv('ml-latest-small/ratings.csv')
ratings = ratings.drop(columns = ['timestamp'])

tags = pd.read_csv('tag-genome/tags.dat' , header = None , sep = '\t' , names = ['TagID', 'Tag' , 'TagPopularity'
#only keep tags for item groups with tag popularity greater than trashhold
tags = tags[tags['TagPopularity']>tag_popularity_trashhold]

tag_relevance = pd.read_csv('tag-genome/tag_relevance.dat' , header = None , sep = '\t' , names = [ 'MovieID' ,'T
tag_relevance = tag_relevance[tag_relevance['TagID'].isin(tags['TagID'])]
#set tag relevence smaller than trashhold to 0
tag_relevance.loc[tag_relevance['Relevance']<tag_relevance_trashhold , ('Relevance')] = 0
tag_relevance = tag_relevance[tag_relevance['MovieID'].isin(ratings['movieId'].unique())]

ratings = ratings[ratings['movieId'].isin(tag_relevance['MovieID'].unique())]

#splitting train and test
good_split = False
while not good_split:
    test_ratings=ratings.sample(frac=test_size)
    train_ratings=ratings.drop(test_ratings.index)
    if set(train_ratings['userId'].unique()) >= set(test_ratings['userId'].unique()):
        good_split = True
```

Listing :۱

در این قسمت خواندن دیتاست ها و جدا کردن مجموعه اموزش و تست انجام میشود

```
#user ids and movie ids
users = train_ratings['userId'].unique()
movies = tag_relevance['MovieID'].unique()

#number of rates of each user
users_all_rates_num = train_ratings['userId'].value_counts().sort_index().to_numpy()
users_all_rates_num = users_all_rates_num.reshape(users_all_rates_num.shape[0],1)

#creating itemgroups and rates matrix
item_groups = create_item_groups(tag_relevance , 'TagID' , 'MovieID' , 'Relevance')
train_rates = create_rates(train_ratings , 'userId' , 'movieId' , 'rating' , movies)
#test ratings
test_ratings = test_ratings.to_numpy().astype('int64')
#typicality matrix
M = create_user_typicality_matrix(train_rates , item_groups , users , users_all_rates_num , 5.0)
#similarity matrix
sim  = sim_matrix(M , 'distance based' , .1)
#predicting rates
predicts = predict(test_ratings , sim , train_rates , movies , users)
#actual retes from test rating array
actual =np.ravel(test_ratings[:,2]).reshape(test_ratings.shape[0],1)
```

Listing :۲

در این قسمت ایدی کاربر ها و فیلم ها مشخص میشود و با استفاده از توابعی که در ادامه معرفی میشود عناصر مختلف محاسبه میشوند. کد های قسمت های بعدی در فایل funcs هستند

```python
def create_item_groups(tag_relevance , tag_id_key , movie_id_key , relevance_key):
    """a function for creating item groups

    Args:
        tag_relevance ([pandas dataframe]): [this dataframe in each row has tag id and movie id and relevance
        between these 2 this dataframe has to be sorted first based on movieid and next based on tagid and for
        each pair of movies and tags we
        this dataframe has a relevance]
        tag_id_key ([str]): [tag id key in tag_relevance datafram]
        movie_id_key ([str]): [movie id key in tag_relevance datafram]
        relevance_key ([str]): [relevance id key in tag_relevance datafram]

    Returns:
        [numpy matrix]: [its a tags_num * movies_num matrix whcih in each cell ij has relvence between
        movie j and tag i]
    """
    #item group is a matrix which row i is Ki
    tags = tag_relevance[tag_id_key].unique()
    tags_num = len(tags)
    movies_num = len(tag_relevance[movie_id_key].unique())
    item_groups = np.zeros((tags_num , movies_num))
    for i in range(tags_num):
        item_groups[i] = tag_relevance[tag_relevance[tag_id_key]==tags[i]][relevance_key].to_numpy()
        #tag relevance has to be sorted based on movie id
    return item_groups
```

Listing :۳

این تابع با استفاده از دیتاست tag-genome اقدام به ساخت `item group` ها میکند

```python
def create_rates(ratings , user_id_key , movie_id_key , rating_key , movies):
    """create a numpy matrix which in cell ij we have reta of user i on movie j
        if a user havent rate a movie we put zero in corresponding cell

    Args:
        ratings ([dataframe]): [in each row has userid and movieid and rate , all rates should be greater
        than zero ratings should be sorted first on userid and next on movieid]
        user_id_key ([str]): [user id key in ratings dataframe]
        movie_id_key ([str]): [movie id key in ratings dataframe]
        rating_key ([str]): [rating key in ratings dataframe]
        movies ([numpy array]): [an array with movie ids in order]

    Returns:
        [numpy matrix]: [in cell ij we have reta of user i on movie j]
    """
    #data set must have not zero rate and we set zero for rate of items
    #that a user has not rated
    users = ratings[user_id_key].unique()
    users_num = len(users)
    movies_num = len(movies)
    rates = np.zeros((users_num , movies_num))
    for i in range(users_num):
        user_ratings = ratings[ratings[user_id_key]==users[i]]
        user_ratings = user_ratings[[movie_id_key , rating_key]]
        user_ratings.set_index(movie_id_key, drop=True, inplace=True)
        user_ratings = user_ratings.to_dict()[rating_key]
        for j in user_ratings.keys():
            rates[i][np.where(movies==j)]=user_ratings[j]
    return rates
```

Listing :۴

این تابع با استفاده از دیتاست نظرات یک ماتریس درست میکند که نظرات در ان ذخیره شده و سطر ها نشان دهنده کاربران و ستون ها نشان دهنده فیلم ها میباشند.

```python
def create_user_typicality_matrix(rates , item_groups , users, users_all_rates_num , Rmax):
    """create user typicality matrix

    Args:
        rates ([numpy matrix]): [output of create_rates funcs on training dataset]
        item_groups ([numpy matrix]): [output of create_rates funcs on tag_relevance]
        users ([numpy array]): [sorted user ids]
        users_all_rates_num ([numpy array]): [for each user have number of rates that user has given]
        Rmax ([int]): [max value of ratings]

    Returns:
        [numpy matrix]: [typicality matrix which in cell ij we have typicality of user i in item group j]
    """
    users_num = rates.shape[0]
    tags_num = item_groups.shape[0]
    user_rates_num = np.zeros((users_num , tags_num))
    for i in range(users_num):
        for j in range(tags_num):
            user_rates_num[i][j] = np.count_nonzero(rates[i]*item_groups[j])
    user_rates_num[user_rates_num==0] = 1
    S_r = rates @ item_groups.T
    S_r = S_r / (user_rates_num * Rmax)
    S_f = user_rates_num / users_all_rates_num
    M = (S_r + S_f)/2
    return M
```

Listing :۵

این تابع با استفاده از نظرات و item group ها ماتریس M را میسازد.

```python
def sim_matrix(M , sim_type , gamma):
    """create similarity matrix for rows of matrix M
        and set gamma as trashhold

    Args:
        M ([numpy matrix]): [typicality matrix]
        sim_type ([str]): [type of similarity (Distance based or Cosine based or Correlation based)]
        gamma ([int]): [trash hold for similarities]

    Returns:
        [numpy matrix]: [in cell ij it has similarity between user i and j]
    """
    if sim_type == 'Distance based':
        sim = np.exp(-pairwise_distances(M))
    if sim_type == 'Cosine based':
        sim = cosine_similarity(M)
    if sim_type == 'Correlation based':
        sim = np.corrcoef(M)
    sim[sim<gamma] = 0
    return sim
```

Listing :۶

این تابع برای هر سطر ماتریس **M** شباهت این سطر را با سطر های دیگر بر اساس معیار شباهت محاسبه میکند

```python
def predict(test_ratings , sim_matrix ,train_rates , movies , users):
    """predict ratings based on rates matrix and similarity matrix if it can not predict
       any rate it put rating to zero

    Args:
        test_ratings ([numpy array]): [in each row has user id and movie id and rate]
        sim_matrix ([numpy matrix]): [similarity matrix]
        train_rates ([numpy matrix]): [rates matrix]
        movies ([numpy array]): [movie ids in order]
        users ([numpy array]): [user ids in order]

    Returns:
        [numpy array ]: [predicted rates]
    """
    predict = np.zeros((test_ratings.shape[0],1))
    k = 0
    for i in test_ratings:
        R = train_rates[: , np.where(movies==i[1])]
        R = R.reshape(R.shape[0],R.shape[1])
        Sim = sim_matrix[np.where(users==i[0])]
        Sim[R.T == 0]=0
        if np.sum(Sim)>0:
            predict[k] = (Sim @ R) / np.sum(Sim)
        else:
            predict[k] = 0
        k+=1

    return predict
```

Listing :۷

این تابع مقادیر پیش بینی شده نظرات کاربران را محاسبه میکند.

```python
def MAE(predicts , actual):
    """calculate MAE

    Args:
        predicts ([numpy array]): [predicted rates]
        actual ([numpy array]): [actual rates]

    Returns:
        [int]: [MAE value]
    """
    diff = predicts - actual
    return np.sum(np.abs(diff))/np.count_nonzero(predicts!=0)

def coverage(predicts):
    """calculate coverage of recommender

    Args:
        predicts ([numpy array]): [predicted rates]

    Returns:
        [int]: [value of coverage]
    """
    return np.count_nonzero(predicts) / len(predicts)
```

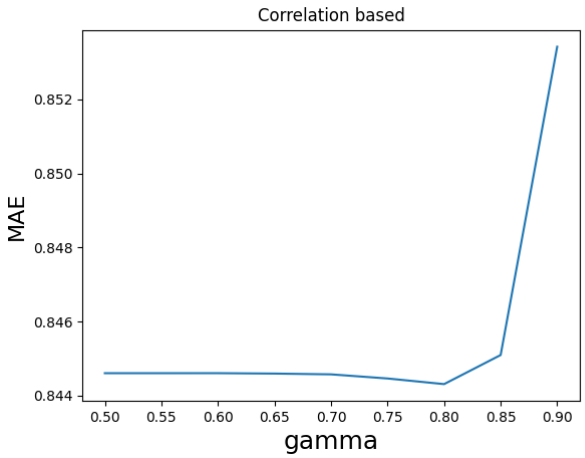
Listing ۸:

این دو تابع برای ارزیابی مقادیر پیش بینی شده نسبت به مقادیر واقعی میباشند.

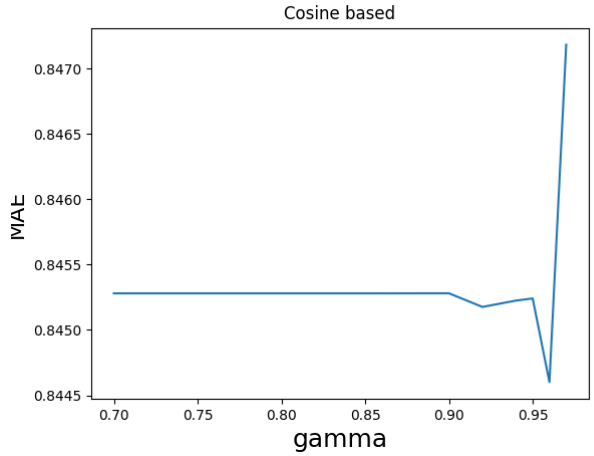
۶ تحلیل نتایج

۱.۶ بررسی معیار های شباهت

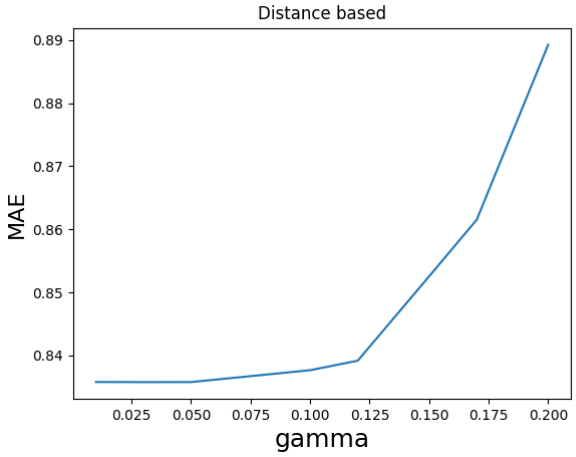
ابتدا بررسی میکنیم به ازای کدام یک از معیار های شباهت نتایج بهتری میگیریم. در این ازمایش ها ۱۰ درصد داده ها را به عنوان تست در نظر گرفتیم همانطور که مشاهده میشود در کل با افزایش گاما خطا نیز افزایش پیدا میکند. همچنین مشاهده میشود بهترین خطا مربوط به معیار `distance based` میباشد. پس از اینجا به بعد در ازمایش هایی که این معیار مشخص نشده به صورت پیش فرض این معیار را برابر `distance based` در نظر میگیریم.



شکل ۱: بررسی تاثیر پارامتر گاما و معیار شباهت



شکل ۲: بررسی تاثیر پارامتر گاما و معیار شباهت



شکل ۳: بررسی تاثیر پارامتر گاما و معیار شباهت

۲.۶ بررسی تاثیر پارامتر گاما و اندازه مجموعه اموزش

TABLE 1
Sensitivity of n on MAE with Different Train/Test Ratios

	γ	χ=0.1	χ=0.3	χ=0.5	χ=0.7	χ=0.9
n=5	0.8	0.8106	0.771	0.7478	0.7436	0.7361
n=10	0.7	0.8115	0.7771	0.7546	0.7451	0.7394
n=15	0.6	0.8117	0.7774	0.7563	0.7502	0.7393
n=20	0.6	0.8125	0.7757	0.7568	0.7481	0.735
n=25	0.5	0.8136	0.777	0.7576	0.7515	0.739
n=30	0.5	0.8129	0.7726	0.7536	0.7438	0.7349
AVG		0.8121	0.7751	0.7544	0.747	0.7373

TABLE 2
Sensitivity of n on Coverage with Different Train/Test Ratios

	γ	χ=0.1	χ=0.3	χ=0.5	χ=0.7	χ=0.9
n=5	0.8	0.9401	0.965	0.9711	0.9699	0.9773
n=10	0.7	0.9637	0.9794	0.9795	0.9792	0.9847
n=15	0.6	0.9764	0.9874	0.9891	0.9895	0.9896
n=20	0.6	0.9774	0.9877	0.9889	0.9862	0.986
n=25	0.5	0.9798	0.9909	0.9918	0.9923	0.9934
n=30	0.5	0.9739	0.9882	0.9902	0.986	0.9882
AVG		0.9685	0.9831	0.9851	0.9838	0.9865

شکل ۴: نتایج مقاله

نتایج بنده به صورت زیر است جدول اول مربوط به مقدار MAE و جدول دوم مربوط به مقدار coverage میباشد.

gamma	X=0.3	X=0.5	X=0.7	X=0.9
0.01	1.0138	0.9058	0.8492	0.8308
0.02	1.0149	0.9058	0.8492	0.8308
0.1	1.0416	0.9130	0.8533	0.8318
0.15	1.1503	0.9484	0.8708	0.8417
0.2	1.4495	1.0659	0.9266	0.8780

gamma	X=0.3	X=0.5	X=0.7	X=0.9
0.01	0.9346	0.9617	0.9737	0.9795
0.02	0.9343	0.9617	0.9737	0.9795
0.1	0.9275	0.9595	0.9724	0.9790
0.15	0.9020	0.9502	0.9674	0.9757
0.2	0.8381	0.9204	0.9532	0.9654

مشاهده میشود که مقادیر coverage به مقادیر مقاله نزدیک تر میباشند همچنین کاهش گاما و افزایش اندازه مجموعه اموزش هر دو این مقدار را زیاد میکنند. در مورد مقادیر MAE اختلاف کمی بین نتایج بنده و مقاله مشاهده میشود که احتمالا ناشی از تفاوت در روش های خوشه بندی فیلم ها میباشد. در کل کاهش پارامتر گاما و افزایش اندازه مجموعه اموزش باعث کاهش خطا میشود.

۳.۶ بررسی تاثیر اندازه مجموعه تست بر پارامتر coverage

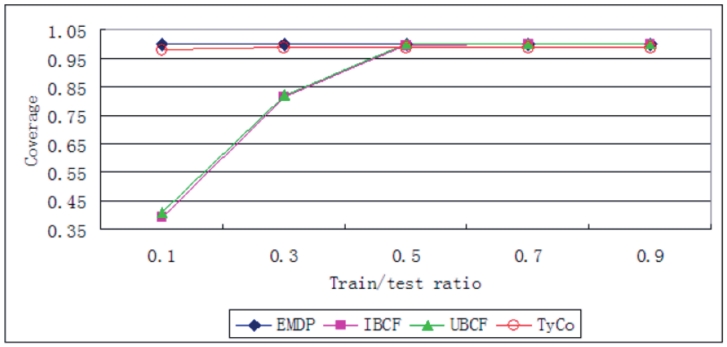
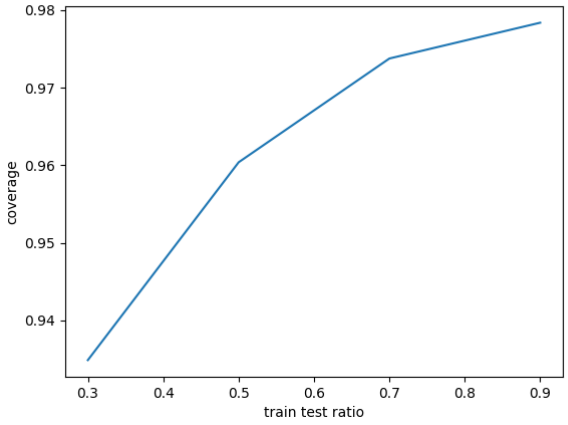


Fig. 5. Comparison on Coverage using ML data set.

شکل ۵: نتیجه مقاله



شکل ۶: نتیجه بدست امده

مشاهده میشود که هر دو نتیجه مشابه هستند و این الگوریتم در هر نسبتی از مجموعه های اموزش و تست عملکرد خوبی درباره این معیار دارد. این معیار در واقع نسبت مواردی که الگوریتم میتواند برای ان ها امتیاز را پیش بینی کند به کل موارد تست نشان میدهد.

۴.۶ بررسی معیار PE

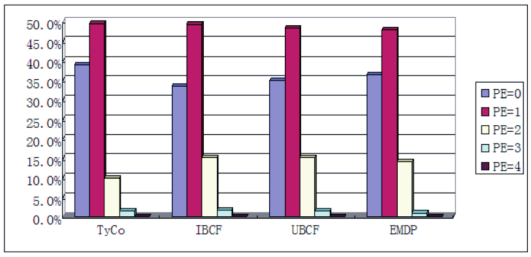
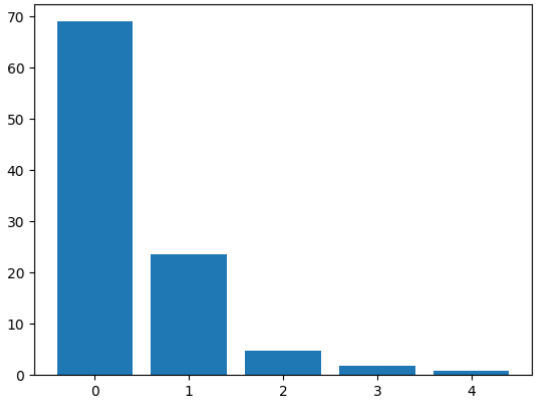


Fig. 6. Comparison on prediction errors.

شکل ۷: نتیجه مقاله



شکل ۸: نتیجه بدست امده

در این نمودار هر ستون نشان دهنده تعدادی از موارد است که اختلاف مقدار پیش بینی شده و مقدار واقعی به اندازه شماره ستون است اختلاف بین نتایج احتمالا به خاطر تفاوت در نوه خوشه بندی میباشد.

۵.۶ مقایسه تاثیر معیار های شباهت و اندازه مجموعه اموزش بر خطا

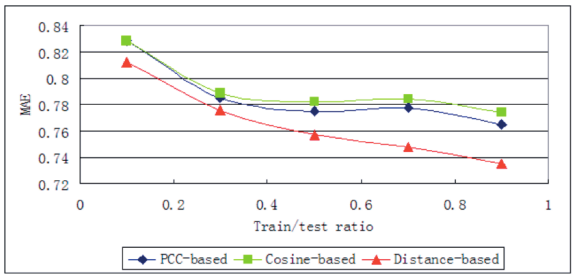
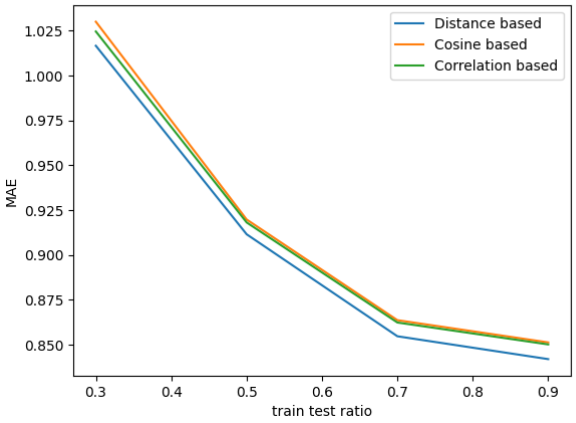


Fig. 8. Impact of different similarity functions on MAE.

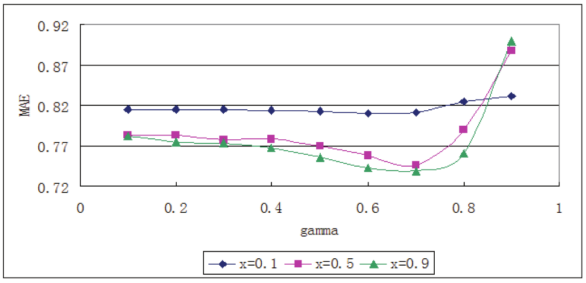
شکل ۹: نتیجه مقاله



شکل ۱۰: نتیجه بدست امده

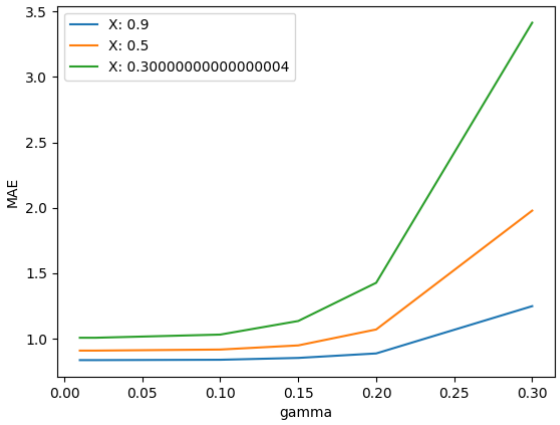
در این بخش مشاهده میشود نتایج مشابه هستند. در کل با افزایش نسبت مجموعه اموزش به تست خطا کمتر میشود و همچنین در بین معیار های شباهت مختلف `distance based` بهترین عملکرد را دارد.

۶.۶ بررسی تاثیر پارامتر گاما و اندازه مجموعه اموزش بر خطا



(a) γ vs. MAE while $n = 10$

شکل ۱۱: نتیجه مقاله



شکل ۱۲: نتیجه بدست امده

مشاهده میشود که در کل با افزایش گاما خطا افزایش میابد.