به نام خدا

پروژه سوم درس بازیابی اطلاعات

سامانه‌های توصیه‌گر

استاد درس: دکتر ممتازی

نام: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

زمستان ۱۴۰۲

فهرست مطالب

فصل اول - سامانه‌های توصیه‌گر مبتنی بر Matrix Factorization

۱-۱ مقدمه

تجزیه ماتریس (matrix factorization) یک روش یادگیری ماشینی است که برای کاهش ابعاد داده‌ها و استخراج ویژگی‌های پنهان از آنها استفاده می‌شود.در این پروژه از این روش برای سیستم‌های توصیه‌گر استفاده می شود.

۱-۲ پیش پردازش

در این پروژه داده‌ها به دو قسمت train , test تقسیم شده‌اند و به فرمت text میباشند، داده‌های ارائه شده با استفاده از تابع read_data به فرمت dataframe تبدیل شده‌اند و با توجه به اینکه reviw_text در این قسمت بررسی نمیشود، دیتای آن ذخیره نشده است.

خروجی تابع read_data به صورت زیر می‌باشد:

	user_id	item_id	rate
0	A2YKWYC3WQJX5J	B00106AC06	1
1	A2LXC5ZHHP0WXP	B00AE07BMQ	1
2	A3HLTHHLPKLRQA	B00AIQOKDY	1
3	A6N1DC5AMPLSK	B000F6RFX4	1
4	ALNFHVS3SC4FV	B0020122ZS	1
...	...	...	...
23033	A11I1I9QLMAM1A	B005TI7NPI	1
23034	AGFGY4EJ37VS2	B002PBHQP4	1
23035	A3MUSWDCTZINQZ	B009VGNYFM	1
23036	A1FWFCJU2G7TRA	B00D6EDGYE	1
23037	A38YU2G73LYCU2	B001JKTTVQ	1

۳-۱ پیاده سازی Matrix Factorization

با بررسی داده‌های مسئله تنها rate=1 را شامل می شوند و داده هایی که کاربر به آنها امتیازی نداده در دیتاست وجود ندارد. ماتریس R به ابعاد nm_user x num_item می‌باشد و اگر کاربر به آیتمی امتیاز داده باشد در ماتریس عدد ۱ قرار می‌گیرد و در غیر اینصورت عدد ۰ در ماتریس ذخیره می‌شود.

```
R={}
R = np.zeros((num_users, num_items))

for row in train_data.itertuples():
     i = map_item[row.item_id]
     u = map_user[row.user_id]
     R[u, i] = 1
```

کلاس MatrixFactorization برای پیاده سازی Matrix Factorization پیاده سازی شده است. در ماتریس اولیه P , Q مقادیری میان ۰-۱ به صورت رندوم قرار می‌گیرد.

```
        self.P = np.random.rand(self.num_users, self.num_factors)
        self.Q = np.random.rand(self.num_items, self.num_factors)
```

تابع هزینه به صورت زیر در نظر گرفته شده است:

$$E = \left(\sum_{(u,i) \in training} (r_{iu} - q_i . p_u^T)^2 + \lambda \left(\sum_i \|q_i\|_2^2 \sum_u \|p_u\|_2^2 \right) \right)$$

برای آموزش مدل در هر تکرار داده‌ها shuffle می‌شوند و ماتریس P,Q به صورت زیر آپدیت می‌شود:

```
        for n in range(self.num_iterations):
            np.random.shuffle(self.samples)
            for u, i, r in self.samples:
              # Computer prediction and error
              prediction = self.predict_rating(i, u)
              grad = (r - prediction)
              # Update user and item latent feature matrices
                   self.Q[i, :] -= self.learning_rate * (((-2) * grad * self.P[u, :]) + (2 *
self.regularization_rate * self.Q[i, :] * (self.P[u, :] * self.P[u, :].T)))
                   self.P[u, :] -= self.learning_rate * (((-2) * grad * self.Q[i, :]) + (2 *
self.regularization_rate * self.P[u, :] * (self.Q[i, :] * self.Q[i, :].T)))
                  # self.Q[i, :] -= self.learning_rate * (((-2) * grad * self.P[u, :]) + (2 *
self.regularization_rate * self.Q[i, :] ))
                  # self.P[u, :] -= self.learning_rate * (((-2) * grad * self.Q[i, :]) + (2 *
self.regularization_rate * self.P[u, :] ))
```

هدف از این آموزش min کردن تابع خطا در به روزرسانی P , Q است، برای انجام این کار مشتق تابع خطا نسبت به P, Q محاسبه می‌شود و براساس آن آپدیت می‌شود.

$$Q_{i+1} = Q_i - \frac{\partial E}{\partial Q}$$
$$P_{i+1} = P_i - \frac{\partial E}{\partial P}$$

امتیاز پیش‌بینی هر کاربر u برای آیتم i به صورت زیر با استفاده از ماتریس P , Q به دست می آید:

```
    def predict_rating(self, i, u):
        """
        Predict the rating for item i and user u.
        Args:
            i (int): Item index.
            u (int): User index.
        Returns:
            float: Predicted rating.
        """
        return self.Q[i, :].dot(self.P[u, :].T)
```

مقادیر اولیه در نظر گرفته شده برای اجرا:

```
num_factors = 64
regularization_rate = 0.001
num_iterations = 100
learning_rate = 0.01
```

۴-۱ پیاده سازی معیارهای ارزیابی

معیارهای ارزیابی خواسته شده به صورت زیر پیاده سازی شده است:

۱-۴-۱ پیاده سازی recall

Recall برابر است با تقسیم تعداد مواردی که توسط مدل درست تشخیص داده‌اند شده بر تعداد کل مواردی که توسط مدل ایجاد شده‌اند، این معیار برای همه کاربران محاسبه شده و در نهایت میانگین آن در خروجی گزارش می‌شود:

```
    def calculate_recall(self,users_list, groundTruth_list, topk=20):
        num_users = len(users_list)
        total_recall = 0
        for u in users_list:
            ground_truth = groundTruth_list[u]
            predicted_ratings = np.dot(self.Q , self.P[u, :].T)
            sorted_indices = np.argsort(predicted_ratings)[::-1]
            top_items = sorted_indices[:topk]
            intersection = set(top_items).intersection(ground_truth)
            # Recall = TruePositives / (TruePositives + FalseNegatives)
            recall = len(intersection) / len(ground_truth)
            total_recall += recall
```

```
    average_recall = total_recall / num_users

    return average_recall
```

۲-۴-۱ پیاده سازی ndcg

مقدار ndcg_score با استفاده از تابع زیر پیاده سازی می شود:

```
[ ]  import numpy as np

    def dcg_score(score, k=None):
        if k is None:
            k = len(score)

        dcg=[]
        dcg.append(score[0])
        i=1
        while i < k:
          s = (score[i]/np.log2(i+1))
          dcg.append(s)
          i+=1

        return np.sum(dcg)
    def ndcg_score(y_true, y_score, k=None):

        ideal_dcg = dcg_score(y_true, k)
        pred_dcg = dcg_score(y_score, k)

        if ideal_dcg == 0:
            return 0
        else:
            return pred_dcg / ideal_dcg
```

مقدار ndcg_score با استفاده از تابع زیر برای همه کاربران محاسبه شده و در نهایت میانگین گرفته میشود:

```
def calculate_ndcg(self,users_list, groundTruth_list, k=20):

    num_users = len(users_list)
    ndcg =0

    for u in users_list:

       ground_truth = groundTruth_list[u]

       predicted_ratings = np.dot(self.Q , self.P[u, :].T)
       sorted_indices = np.argsort(predicted_ratings)[::-1]
       top_items = sorted_indices[:k]

       pred_score = []
       true_score = []

       for t in top_items:
         if t in ground_truth:
          pred_score.append(1)
         else:
          pred_score.append(0)

       if(len(ground_truth) >= k):
          true_score = [1]*k
       else:
          true_score = [1]*len(ground_truth)+[0]*(k-len(ground_truth))

       ndcg+=ndcg_score(np.asarray(true_score), np.asarray(pred_score), k=k)

    average_ndcg = ndcg / num_users

    return average_ndcg
```

۳-۴-۱ پیاده سازی rank correlation

رابطه محاسبه rank correlation به صورت زیر می باشد:

$$\rho = 1 - \frac{6\Sigma\, \mathrm{d}_i^2}{n(n^2-1)}$$

این مقدار برای هر کاربر محاسبه می‌شود و در نهایت میانگین آن گزارش میشود:

```
def calculate_rank_correlation(self,users_list, groundTruth_list):

    rc =0
    num_users = len(users_list)
    s = (self.num_items * (pow(self.num_items,2)-1))
```

```
    for u in users_list:

        ground_truth = groundTruth_list[u]
        predicted_ratings = np.dot(self.Q , self.P[u, :].T)

        d2=0

        for i,val in enumerate(predicted_ratings):
          if i in ground_truth:
           val = val-1
          d2+=pow(val,2)

        rc+=1-((6*d2)/s)

    return rc / num_users
```

۵-۱ نتیجه

کیفیت عملکرد سیستم توصیه گر پیاده سازی شده با استفاده از توابع NDCG، Recall و correlation rank به صورت زیر است:

Result: recall=0.001 ndcg=0.002 rank_correlation = 0.99993

فصل دوم - ترکیب مدل بخش قبل با بردار های معنایی عصبی

۲-۱ مقدمه

هدف از این پخش غنی سازی Matrix Factorization با استفاده از اطلاعات جانبی که حاوی نظر کاربر نسبت به آن آیتم است.

۲-۲ پیش پردازش

در این پروژه داده‌ها به دو قسمت train , test تقسیم شده‌اند و به فرمت text میباشند، داده‌های ارائه شده با استفاده از تابع read_data به فرمت datafram تبدیل شده‌اند.

خروجی تابع read_data به صورت زیر می‌باشد:

	user_id	item_id	rate	review_text
0	A2YKWYC3WQJX5J	B00106AC06	1	I usually love the Motions conditioners and ma...
1	A2LXC5ZHHP0WXP	B00AE07BMQ	1	Axe messy look styling gum is a product that w...
2	A3HLTHHLPKLRQA	B00AIQOKDY	1	I have always found liquid soap to be as much ...
3	A6N1DC5AMPLSK	B000F6RFX4	1	I've tried plenty of products that claim to he...
4	ALNFHVS3SC4FV	B0020122ZS	1	Suave Kids is one of my favorite brands of sha...
...	...	...	...	...
23033	A11I1I9QLMAM1A	B005TI7NPI	1	I've never really used toner before because it...
23034	AGFGY4EJ37VS2	B002PBHQP4	1	This cream is too thick and too stinky. It sme...
23035	A3MUSWDCTZINQZ	B009VGNYFM	1	My boyfriend took a shower and used this. He ...
23036	A1FWFCJU2G7TRA	B00D6EDGYE	1	I have VERY translucent skin-- even with the '...
23037	A38YU2G73LYCU2	B001JKTTVQ	1	I personally love all Shany's products! This ...

23038 rows × 4 columns

۲-۳ بازیابی نظرات کاربران

مدل آماده Bert با استفاده از کد زیر گرفته می‌شود:

```
)] from transformers import BertModel, BertTokenizer


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model =BertModel.from_pretrained("bert-base-uncased")
```

با استفاده از کد زیر embedding مدل برت برای هر آیتم که نظری برای آن ثبت شده است محاسبه می‌شود:

```
import numpy as np

def get_bert_embedding(text):
    tokens = tokenizer(text, return_tensors='pt', truncation=True, padding=True)
    with torch.no_grad():
        outputs = model(**tokens)
    return outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
```

```
] from tqdm import tqdm

mean_text_embeding = {}

for item in tqdm(item_set):
  mean_text_embeding[item] = []
  filtered_train_data = train_data[(train_data.item_id == item)]
  review_texts = filtered_train_data['review_text'].unique()
  for t in review_texts:
     mean_text_embeding[item].append(get_bert_embedding(review_texts[0]))
```

میانگین embedding هر آیتم با استفاده از کد زیر محاسبه می شود:

```
[14] from tqdm import tqdm

     for item in tqdm(item_set):
       arr = np.array(mean_text_embeding[item])
       mean = np.mean(arr, axis=0)
       mean_text_embeding[item] = mean
```

با استفاده از کد زیر embedding به دست آمده برای هر آیتم که ۷۶۸ بعد میباشد با استفاده از pca به ۶۴ بعد کاهش می یابد:

```
[16] from sklearn.decomposition import PCA

     pca = PCA(n_components=64)
     pca_embeding = pca.fit_transform(embeding_list)
```

```
pca_embeding.shape
```

```
(733, 64)
```

۲-۳ پیاده سازی

پیاده سازی معیارهای ارزیابی و Matrix Factorization برای این سوال، مشابه سوال قبل میباشد.

۲-۴ جایگزینی

در این روش ماتریس کاهش بد یافته که از میانگین embedding نظرات برای هر آیتم به دست آمده بود با ماتریس Q در Matrix Factorization جایگزین میشود.

۲-۴-۱ نتیجه

نتایج معیارهای ارزیابی به صورت زیر است:

Result: recall=0.02 ndcg=0.03 rank_correlation=0.99998

۲-۵ االحاق

بردار ۷۶۸ بعدی به دست آمده برای هر آیتم از مدل BERT و بردار متناظر آن آیتم از ماتریس Q در Matrix Factorization با یکدیگر ترکیب شده و سپس کاهش بعد به ۶۴ با استفاده از pca داده میشود و سپس با ماتریس Q در Matrix Factorization جایگزین میشود.

۲-۵-۱ نتیجه

نتایج معیارهای ارزیابی به صورت زیر است:

Result: recall=0.020 ndcg=0.025 rank_correlation = 0.99997

۲-۶ نتیجه نهایی

ترکیب مدل Matrix Factorization با بردارهای معنایی عصبی خروجی بهتری را ارائه میکند و در میان دو روش پیاده سازی شده برای ترکیب Matrix Factorization با بردارهای معنایی عصبی، به صورت ناچیز روش جایگزینی عملکرد بهتری دارد.