Information Retrieval Course

Fall 2023

IR-HW03

Koorosh Khavari Moghaddam

koorosh.k@aut.ac.ir

koorosh.khavari.m@gmail.com

Student ID: 401131023

Contents

Making Representation Vectors

```
# clean the text
def clean_text(text):
    return gsp.remove_stopwords(gsp.strip_tags(gsp.strip_punctuation(gsp.strip_numeric(text))))
D_train['review_text'] = D_train['review_text'].astype('string')
D_train['review_text'] = D_train['review_text'].fillna('')
D_train['review_text'] = D_train['review_text'].apply(clean_text)
# tokenize and transform the text
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
reviews = [torch.tensor([tokenizer.encode(sentence, add_special_tokens=True)])[:,:512] for sentence in list(D_train['review_text'])]
with torch.no_grad():
    results = []
    for review in tqdm(reviews):
        try: results += [model(review).last_hidden_state.mean(dim=1).squeeze()]
        except: print(f"[error] : {review}")
    reviews_vec = torch.stack(results)
# store the texts

with open ("bert_review_vectors.pth", "wb") as f:
    pickle.dump(reviews_vec, f)
```

✓ 115m 28.6s

```
100%|██████████| 23038/23038 [1:54:29<00:00,  3.35it/s]
```

Preprocessing Data

In order to tokenize the comments inside the dataset, we must clean the texts and reduce them to 512 tokens, said procedure is done using gensim library, also the processed representations are stored in the bert_review_vectors.pth file.

[1:6] we cleared the text of the comments using gensim library by iterating through all of them.

[7:16] we tokenize the cleansed texts and represent them as described in the homework problem (centroid of the transformed tokens in the latent space of pretrained BERT model)

[17:20] storing the result of the model for the later use.

Matrix Factorization

Learn U & I Matrices (As Ground Truth For Evaluation)

```
1  Latent_Dimension = 64
2  λ = 1e-3
3  τ = 1e-3
4  U = np.random.uniform(low=0, high=1, size=(Users.shape[0],Latent_Dimension))
5  I = np.random.uniform(low=0, high=1, size=(Items.shape[0],Latent_Dimension))
6  dU, dI = np.zeros_like(U), np.zeros_like(I)
7  N = D_train.shape[0]
8  def E(R,I,U): return np.sum(R*((R-U@I.T)**2))/N+λ*(np.linalg.norm(U)+np.linalg.norm(I))
9  def _dU(R,I,U): return np.sum(-2*(R-U@I.T)@I)/N+λ*2*U
10 def _dI(R,I,U): return np.sum(-2*(R-U@I.T).T@U)/N+λ*2*I
11 _Δ = []
12 for i in tqdm(range(100)):
13     # calculate loss
14     Δ = E(R,I,U)
15     _Δ += [Δ.item()]
16     # calculate gradient vectors
17     dU = _dU(R,I,U)
18     dI = _dI(R,I,U)
19     # calculate descent direction
20     dU = dU/np.linalg.norm(dU)
21     dI = dI/np.linalg.norm(dI)
22     # bisection to find step size
23     while Δ<E(R,I,U-dU): dU=dU/2
24     while Δ<E(R,I-dI,U): dI=dI/2
25     # update factors
26     U -= dU
27     I -= dI
28     # early stopping
29     if (np.linalg.norm(dI)<τ)&(np.linalg.norm(dU)<τ): break
30
31 plt.plot(list(range(len(_Δ))), _Δ)
32 plt.show()
33
34 # real vs approximate utility matrix at (0,0)
35 R1 = U@I.T
```

✓ 21.1s

```
99%|█████████▉| 99/100 [00:20<00:00,  4.72it/s]
```

I changed the matrix naming to [U]ser and [I]tem instead of Q and P for simpler reading.

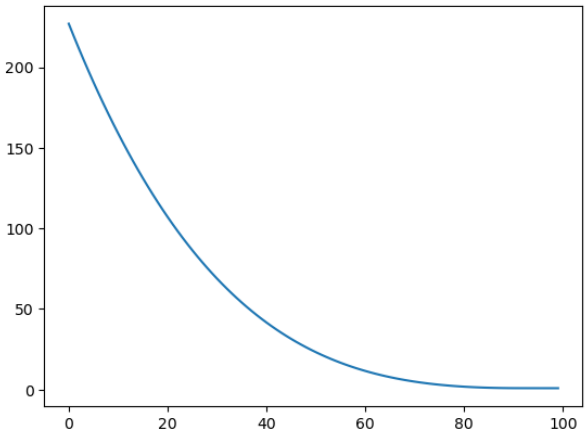
[1:7] we defined the requreid variables for the training procedure. τ parameter is an threshold used to early stop for the training procedure. dU and dI will used to store the gradient of the loss with respect to each of the User and Item Matrices. N is a normalization variable equal to the size of the trainig dataset, used in the calculation of the error term to make it stable.

[8:10] required functions are defined, E is the error term as defined in the homework problem, _dU and _dI is the gradient of the error term with respect to matrices U and I respectively.

[11] _Δ is a variable used to store the history of the loss during the training.

[12:29] main training procedure, here we first calculate and stored the error[13:15], then calculated the descent direction by normalizing the resulted gradient vector of the loss with respect to each of the U and I matrices [16:21], then we find the appropriate step size using bisection method [22:24] and update current U and I matrices towards it [25:27], we also checked the value of the error to early stop the training process if needed in [28-29].

The final training plot is shown in the side figure.



Item Representation Using BERT

Part 3.A:: Replacing I Matrix

```
# load representation vectors
with open ("bert_review_vectors.pth", "rb") as f:
    reviews_vec = pickle.load(f).numpy()
# make 64 dimensional representation of comments
pca = PCA(n_components=64)
pca.fit(reviews_vec)
reviews_pca_vec = pca.transform(reviews_vec)
# make Item Comment matrix
Item_Comment = []
PCA_Item_Comment = []
for item in Items: PCA_Item_Comment += [np.mean(reviews_pca_vec[[(D_train.loc[D_train['item_id'] == item].index)[0].astype(int)],:], axis=0)]
for item in Items: Item_Comment += [np.mean(reviews_vec[[(D_train.loc[D_train['item_id'] == item].index)[0].astype(int)],:], axis=0)]
bert_I = np.array(Item_Comment)
pca_I = np.array(PCA_Item_Comment)
R2 = U@pca_I.T
```

✓ 4.6s

In the second part of the homework, we used the BERT model to represent each item in the latent space.

[1:3] we load the stored representation which are prepared in the first part.

[4:7] we transform each comment to 64-dimensional space using PCA module of the sklearn.

[8:15] for each item, we represent it as the centroid of the all-related comments in the 64-dimensional space. Therefor the result is the new Item Matrix with the same shape of the older I matrix, and as told in the homework, we used the U matrix of the matrix factorization part alongside it.

Concatenation Of Bert & MF

Part 3.B:: Concatinating I Matrix

```
concatinated_I = np.concatenate((I, bert_I), axis=-1)
pca = PCA(n_components=64)
pca.fit(concatinated_I)
pca_concatinated_I = pca.transform(concatinated_I)
R3 = U@pca_concatinated_I.T
```

✓ 0.4s

[1] we concatenated the matrix factorization item matrix with the BERT representation which resulted in 832-dimentional Item matrix.

[2:5] we used the PCA to reduce the item matrix dimension to the 64-dimensional space.

Evaluation

```
I0,I1 = np.where(R > 0)
ground_truth, bert_method, concatinate_method = [],[],[]

for I,r1 in enumerate(np.argsort(-R1,axis=-1)):
    ground_truth += [r1[np.in1d(r1, I1[np.where(I0==I)], invert=True)][:20]]
for I,r2 in enumerate(np.argsort(-R2,axis=-1)):
    bert_method += [r2[np.in1d(r2, I1[np.where(I0==I)], invert=True)][:20]]
for I,r3 in enumerate(np.argsort(-R3,axis=-1)):
    concatinate_method += [r3[np.in1d(r3, I1[np.where(I0==I)], invert=True)][:20]]

ground_truth = np.array(ground_truth)
bert_method = np.array(bert_method)
concatinate_method = np.array(concatinate_method)
```

✓ 0.8s

Given the test dataset, we can't find out the user ratings for the 20 items which is required to calculate the value of the mentioned evaluation metrics inside the homework, because there are different number of ratings of each user for the items which are fewer than required 20 items.

Important Not Valid but Necessary Assumption: I used the Rating Matrix resulted from the Matrix Factorization section as the ground truth. After dropping the locations filled using the training values, I sorted each user rating for the items and kept only the 20 top rated ones. We are going to use these lists as the ground truth lists to calculate each of the evaluation metrics in the homework.

[1] location of the training data inside training set

[2] three matrices used to store the rating matrices of the Matrix Factorization part, Pure BERT representation part and concatenated BERT and MF part

[4:9] calculating lists of top 20 rating items for each user given each of the methods.

[11:13] converting the lists to the NumPy arrays

Evaluation Methods

Evaluation

```python
def Recall(y_true,y_pred,k): return np.intersect1d(y_true, y_pred).shape[0]/k
def DCG(X): return X/(np.log2(1+np.arange(X.shape[0]))+np.eye(1,X.shape[0]))
def NDCG(y_true,y_pred,k): return np.sum((DCG(y_pred)/DCG(y_true))[:k])
def Rank_Correlation(y_true,y_pred): return spearmanr(y_true, y_pred)
def evaluate(y_true, y_pred, label, y_true_matrix, y_pred_matrix):
    _R,_N,_C = [],[],[],
    for ind,y in enumerate(zip(y_true,y_pred)):
        y_t,y_p = y[0],y[1]
        _R += [Recall(y_t,y_p,20)]
        _N += [NDCG(y_true_matrix[ind,y_t],y_pred_matrix[ind,y_p],20)]
        _C += [Rank_Correlation(y_t,y_p)]
    _R = np.array(_R).mean()
    _N = np.array(_N).mean()
    _C = np.array(_C).mean()
    print(f"Evaluation Results Of {label}:")
    print(f"[Recall]          : {_R:0.2f}")
    print(f"[NDCG]            : {_N:0.2f}")
    print(f"[Rank_Correlation]: {_C:0.2f}")
```

✓ 0.0s

Each of the mentioned method inside the homework is implemented in this section as a function that receives two sorted sets and returns the evaluation metric.

[1] Recall metric: will simply assumes whole True set is the top 20 rating of the matrix factorization part and counts the intersection of the provided list with this list as the True Positive value, then it will normalize this value by the size of the True set (which is 20) and return the Recall value

[2:3] NDCG metric: it required two sorted sets. We used the Matrix Factorization rating matrix as the ground truth list, after calculating the DCG value for the true list and predicted list, the method will return the sum of normalized DCG up to k'th element (which is 20 as told in the homework)

[4] Rank Correlation Method: we used the implementation of the Spearman's correlation provided in the SciPy library. Given presented mismatch in the two list, the result of this list is not reliable to begin with.

BERT Model Evaluation

BERT Method Result

```
1  evaluate(ground_truth,bert_method,"Pure BERT Representation", R1, R2)
```

✓ 0.7s

```
Evaluation Results Of Pure BERT Representation:
[Recall]          : 0.03
[NDCG]            : 23.00
[Rank_Correlation]: 0.25
```

```
1  evaluate(ground_truth,concatinate_method,"Concatinated Representation", R1, R3)
```

✓ 0.8s

```
Evaluation Results Of Concatinated Representation:
[Recall]          : 0.03
[NDCG]            : 23.57
[Rank_Correlation]: 0.26
```

Assuming the Matrix Factorization ratings matrix as the ground truth, the metric results are shown above. It is not a reliable result though, because incorporating the MF ratings with the BERT model's representation in the concatenation part will definitely injects more information from the MF part inside the rating matrix and hence make the representation more like the MF part which is CONSIDERED to be ground truth but not the real one, therefor we can't say improvement in these metrics will make the concatenation method better.