COMP4332 and RMBI4310 Final Exam

June 2, 2020, Tuesday

Part 1: 1:30pm-2:50pm

Part 2: 3:10pm-4:30pm

Instructor: Yangqiu Song

Question	**Score**	**Question**	**Score**
1	/ 10	6	/ 10
2	/ 10	7	/ 10
3	/ 8	8	/ 7
4	/ 12	9	/ 9
5	/ 10	10	/ 15
Total:	**/ 100**		

Note:

1. In your answer book, please indicate clearly which question you are answering.

2. Please do know leave the meeting even you have finished all questions.

3. If there are different definitions in this paper from the lecture notes, you should follow this paper.

Q1. Yes/No Questions (10 points)

Indicate whether each statement is true (✓) or false (×).

1. The number of parameters to be estimated in an RNN model is dependent on the maximum length among all the input sequences.
2. The Word2vec model could solve word sense disambiguation problem (e.g., distinguishing difference meanings of words based on the context) well.
3. Cold start problem in recommender system is caused by data sparsity.
4. One basic assumption of User-based CF is that User preferences may evolve over time.
5. If we use adjusted cosine similarity for two items i and j in Item-based CF, we need to compute the average ratings of both i and j.
6. The input features for deep component of Wide and Deep Learning model include the continuous features and the cross product transformation of categorical features.
7. In recommender systems, wide models usually have better generalization capability compared with deep models.
8. Adopting data mining approaches for heterogeneous information networks in recommender system cannot alleviate the data sparsity issue.
9. Heterogeneous relationships can complement each other.
10. Network embedding is slower than traditional matrix factorization based methods.

Q2. n-gram (10 points)

Consider three sentences (all words have been turned to lower cases):

Document 1: *text categorization is to classify documents into different semantic topics .*

Document 2: *rnn could be a useful tool to classify documents in text categorization .*

Document 3: *we tend to use rnn to extract features of documents .*

(a) Write down all bi-grams of the first sentence.

(b) Calculate the TF-IDF score for the following uni-grams and bi-grams: {*'rnn', 'documents', 'extract', 'classify-documents'*} for all 3 sentences. Please use raw frequency for TF weighting and IDF weighting $\log \frac{N}{n_i}$ where n_i is the frequency of word i shown in a document counted over the set and N is the number of documents.

	Document 1	Document 2	Document 3
rnn			
documents			
extract			
classify-documents			

Q3. LSTM (8 points)

Consider a single unit in LSTM network. $\mathbf{x}_t \in \mathbb{R}^D$ ($\mathbb{R}^D$ means a D-dimensional vector) is the input of step t, and $\mathbf{h}_t \in \mathbb{R}^d$ is the hidden state of step t:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{u}_t &= \tanh(\mathbf{W}_u\mathbf{x}_t + \mathbf{U}_u\mathbf{h}_{t-1} + \mathbf{b}_u) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \mathbf{u}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

Calculate the total number of parameters to be estimated in the LSTM (based on the dimensionalities of all vectors and matrices). Here we assume input $\mathbf{x}_t$ is a fixed embedding vector, so please do not count it into the parameters.

Q4. CNN+RNN for text classification (12 points)

Consider a form of CNN+RNN model for text classification. An input sentence is tokenized as $[w_1, w_2, \cdots, w_l]$, and we use d-dimensional pre-trained word2vec embedding to represent the words as $\mathbf{S} = [\mathbf{e}(w_1), \mathbf{e}(w_2), \cdots, \mathbf{e}(w_l)]$, $\mathbf{S} \in \mathbb{R}^{d \times l}$ ($\mathbb{R}^{d \times l}$ means it is a d by l matrix).

In the network, we firstly apply convolution and pooling for the input embeddings. The input $\mathbf{S}$ is filtered by m convolution filters $\mathbf{H_i} \in \mathbb{R}^{d \times w}, i \in \{1, \cdots, m\}$, where w is the window size. The stride is set to be 1 and the padding method is 'VALID'.

The filtered output of the i-th filter is denoted as $\mathbf{p}_i$. As the first dimension of the filter is equivalent with the first dimension of the input $\mathbf{S}$ (both are d), and the padding is 'VALID', the output $\mathbf{p}_i$ could be viewed as a vector by squeezing its first dimension. The output after convolution is the concatenation of the m filtered outputs :

$$\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m]$$

Max pooling is used here with a stride of 2, filter size of 2, and 'VALID' padding. The output of the convolutional layer is denoted as a concatenation of m filtered output:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m]$$

Next we apply an RNN layer to matrix $\mathbf{Q}$.

$$\mathbf{h}_t = f(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b})$$

Here, $\mathbf{x}_t$ is the t-th row of $\mathbf{Q}$. The output of RNN $\mathbf{r} \in \mathbb{R}^n$ is selected as the last hidden layer, and is deemed as the encoding of the whole sentence.

(a) What is the size of each one of $\mathbf{p}_i$, $(i = 1, \cdots, m)$. And what is the size of matrix $\mathbf{Q}$?

(b) Design an output layer using Fully Connected Neural Network with Softmax Output for a k-class classification task. Write down the basic formulations and specify the size of parameters that you use for the output layer.

Q5. User CF (10 points)

A table recording the ratings of 6 users and 6 items is as follows:

	Item1	Item2	Item3	Item4	Item5	Item6
User1	8	7	6	5	?	9
User2	?	5	3	?	9	?
User3	8	?	7	1	?	9
User4	?	8	?	3	3	2
User5	7	?	4	10	4	?
User6	6	10	4	?	?	10

You are asked to predict the ratings of **User3** for **Item2** through **User-based Collaborative Filtering**. The similarity function to be used in this problem **should** be the **Jaccard Similarity**:

$$sim(u, v) = \frac{|I(u) \cap I(v)|}{|I(u) \cup I(v)|}$$

where u, v are two users, $I(u)$ is the set of items rated by user u, $|S|$ is the cardinality of set S.

The prediction function **should** be :

$$pred(u, i) = \Sigma_{v \in S(u,K) \cap N(i)} sim(u, v) r_{v,i}$$

where u and i denotes a user and an item respectively, $S(u, 3)$ is the top 3 similar users to user u, $N(i)$ is the set of users who have rated item i, $sim(u, v)$ is the similarity score between user u and user v, $r_{v,i}$ is the rating of user v for item i.

Q6. Neural CF (10 points)

Consider the following Neural CF model.

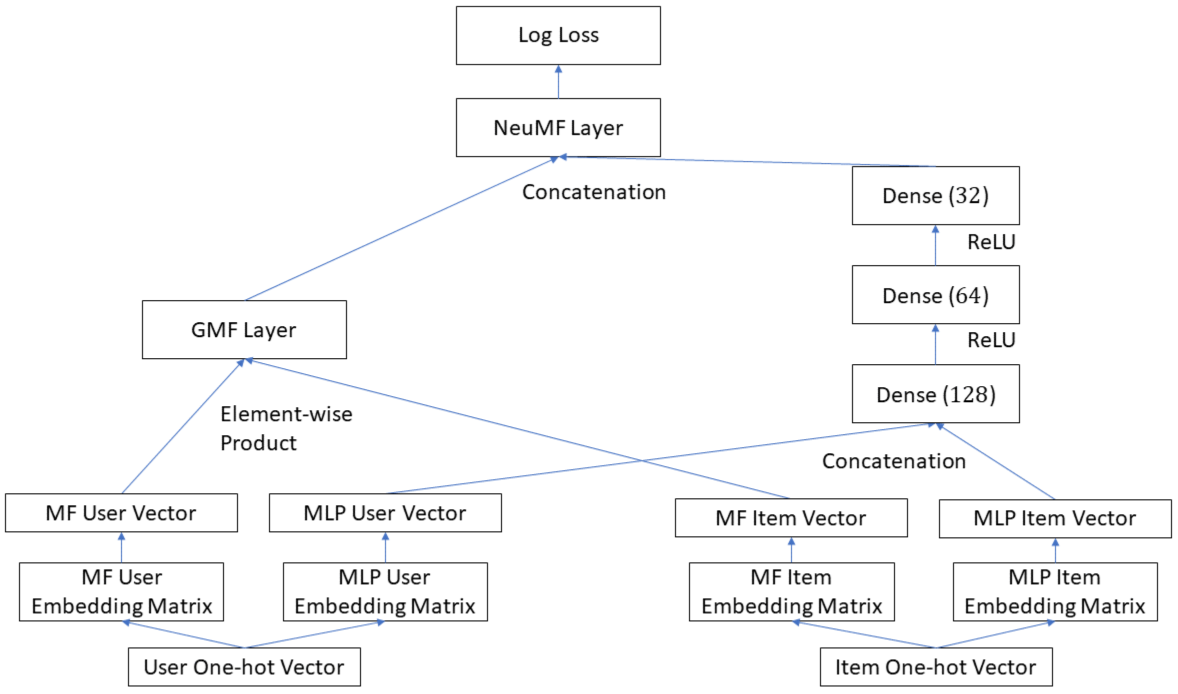


Figure 1: NCF Architecture

Assume that:

1. The dimensions of MF user and item embeddings are both 100. Both embeddings are trainable.
2. The dimensions of MLP user and item embeddings are both 128. Both embeddings are trainable.
3. There are 1000 unique users and 2000 unique items in the training set.
4. All the bias terms can be ignored.
5. Output size of three dense layer are 128, 64, and 32 respectively. Activation function is ReLU.
6. Final output dimension is 1.

Compute the number of all **trainable** parameters of this model.

Q7. Heterogeneous Information Networks (10 points)

Given a network schema in Figure 2 and a network instance in Figure 3, compute the **PathCount similarity matrix** (each element is the count of path instances in the network) between all users and businesses for the given meta-path in Figure 4.

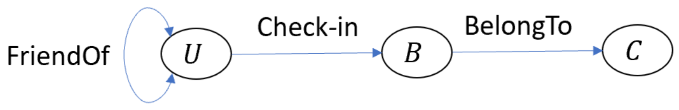


Figure 2: Network Schema

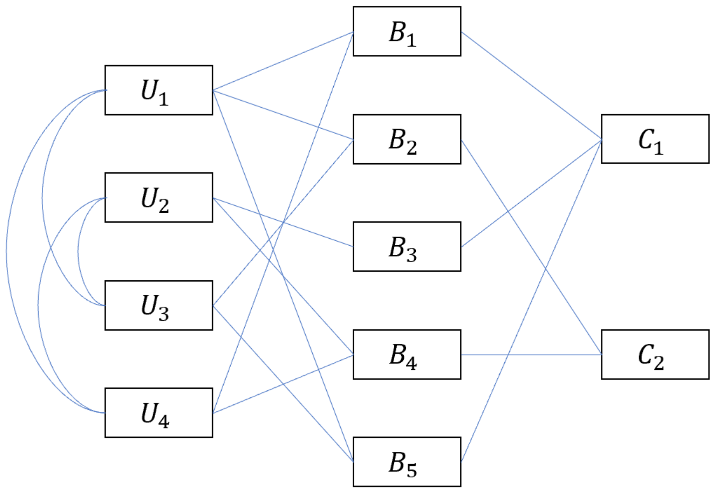


Figure 3: Network Instance

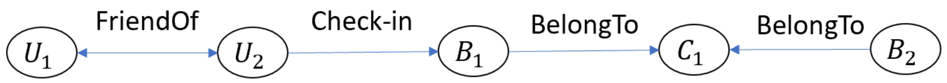


Figure 4: Meta-path

Q8. Node2vec (7 points)

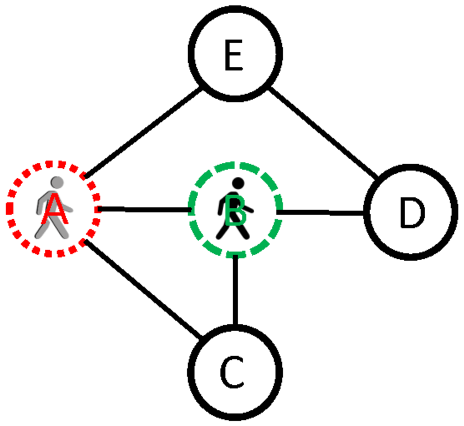


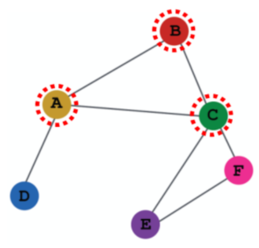
Figure 5: Node2vec

In this question, you are required to list all the paths of three nodes generated from the biased second-order random walk process. Assume that B is the last node which is indicated with the green dashed circle, and A is the second last node which is indicated with the red dotted circle. Assume p is 0.5 (controlling the BFS), q is 2 (controlling the DFS), and the remaining walk length is 2. You should list all possible paths and their associated probabilities.

Please write down all the necessary steps to compute the probabilities. The probability should be in the format of fraction rather than decimal (e.g., 3/4 rather than 0.75).

Hints: your path should start with node B and contain 3 nodes, e.g., $B \to D \to B$, and the sum of all probabilities should be equal to 1.

Q9. Graph Neural Networks (9 points)



(a) Input Graph

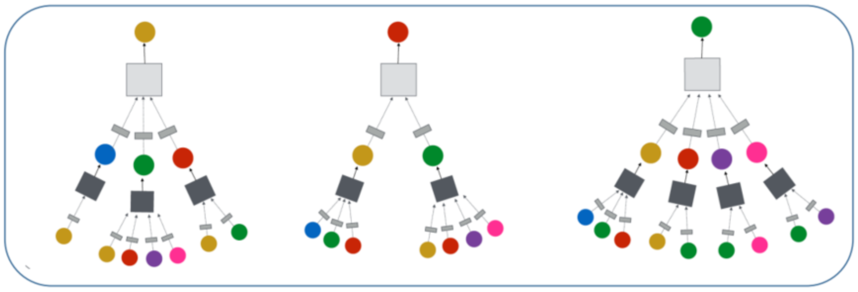

(b) Graph Convolution Operations for A,B,C

Figure 6: Examples for graph convolution operations

Graph data are anywhere in our real-life and graph representation is a very hot topic in data mining. Recently, many researchers design graph neural networks with the help of graph convolution operations to aggregate neighbourhood information followed by a neural network. Figure 6 shows the graph convolution operations of a 2-layer graph convolutional network (GCN). The detailed computation can be defined as follows:

$$\begin{aligned}
\boldsymbol{h}_v^0 &= \boldsymbol{x}_v \\
\boldsymbol{h}_v^1 &= \sigma(\boldsymbol{W}_1 \sum_{u \in \mathcal{N}(v)} \frac{\boldsymbol{h}_u^0}{|\mathcal{N}(v)|} + \boldsymbol{B}_1 \boldsymbol{h}_v^0) \\
\boldsymbol{h}_v^2 &= \sigma(\boldsymbol{W}_2 \sum_{u \in \mathcal{N}(v)} \frac{\boldsymbol{h}_u^1}{|\mathcal{N}(v)|} + \boldsymbol{B}_2 \boldsymbol{h}_v^1) \\
\boldsymbol{z}_v &= \boldsymbol{h}_v^2
\end{aligned}$$

1. If the node features $\boldsymbol{X} \in \mathbb{R}^{m \times |V|}$ (m is the feature dimension and $|V|$ is the number of nodes) and the hidden dimension for each layer is d. How many **trainable** parameters are required for the above GCN?

2. The GCN aggregates the neighbour messages by taking their average. GraphSAGE makes the aggregation more general:

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}_k \cdot \text{CONCAT}(\text{AGGREGATE}_k(\{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}), \boldsymbol{h}_v^{k-1}))$$

   Assume we use 2-layer GraphSAGE variants with the *Mean* aggregator

$$\text{AGGREGATE}_k = \sum_{u \in \mathcal{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{|\mathcal{N}(v)|}, \forall u \in \mathcal{N}(v),$$

   the *Mean-Pooling* and the *Max-Pooling* aggregator

$$\text{AGGREGATE}_k = \gamma(\{\sigma(\boldsymbol{Q}_k \boldsymbol{h}_u^{k-1}), \forall u \in \mathcal{N}(v)\})$$

   where σ is an activation function (e.g., ReLU) and $\gamma()$ is an element-wise function, such as mean function or max function, $\boldsymbol{Q}_k$ maps $\boldsymbol{h}_u^{k-1}$ to d dimensions. Please count the number of parameters for the three variants?

Note: you do not need to include biases in your computation.

Q10. Aggregators of Graph Neural Networks (15 points)

As shown in Q9, we can use different aggregators to aggregate neighbourhood information and update node representations. Let's consider a 1-layer GraphSAGE network and use three different aggregators to compute the node representations:

$$\begin{aligned}
\text{MEANAGGREGATE}_k &= \text{mean}(\{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \\
\text{MAXAGGREGATE}_k &= \max(\{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \\
\text{SUMAGGREGATE}_k &= \text{sum}(\{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})
\end{aligned}$$

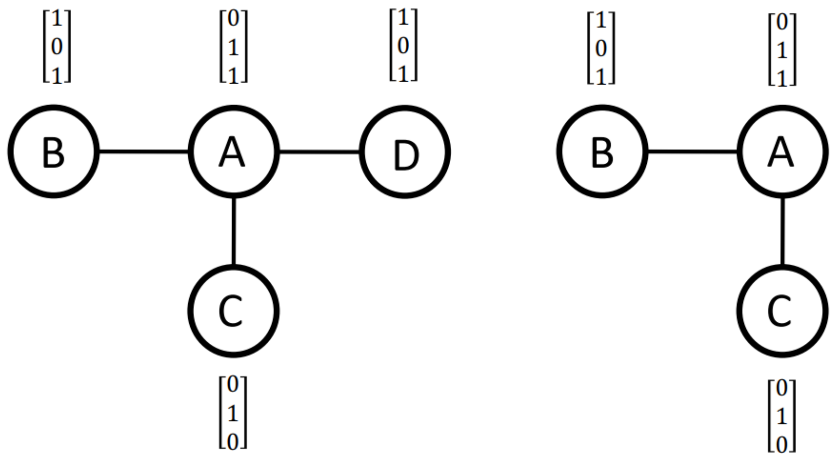
They apply different element-wise operations on the input. For instance, $\text{mean}(\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}) = \begin{bmatrix}0.5\\0.5\\0\end{bmatrix}$, $\max(\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}) = \begin{bmatrix}1\\1\\0\end{bmatrix}$, and $\text{sum}(\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}) = \begin{bmatrix}1\\1\\0\end{bmatrix}$.

Assume we have three different cases and each case has two graphs. Each node v associates with a 3-dimension $\boldsymbol{x}_v$ feature vector, e.g., the feature vector of the node A in the first graph of "Case 1" is $\boldsymbol{h}_{v_A}^0 = [0, 1, 1]$. Please use the above three aggregators and the matrix $\boldsymbol{W}_1 = \begin{bmatrix}-1 & 2 & 3 & 1 & -3 & 4\\ 2 & -2 & 1 & 5 & 2 & -5\end{bmatrix}$ to compute **18** representations $\boldsymbol{z}_A$ for node A:

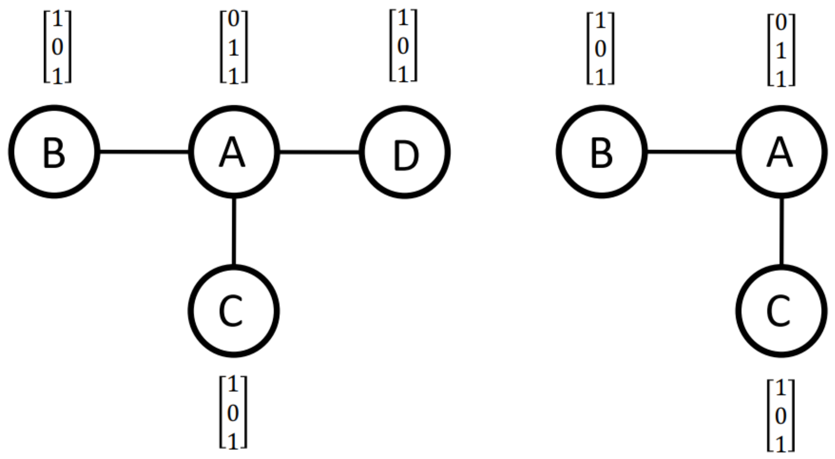
$$\boldsymbol{h}_v^1 = \boldsymbol{W}_1 \cdot \text{CONCAT}(\text{AGGREGATE}_k(\{\boldsymbol{h}_u^0, \forall u \in \mathcal{N}(v)\}), \boldsymbol{h}_v^0)$$

From the result, can you distinguish two nodes A of each case by their final representations? Which aggragator would be a better one for graph representation learning?

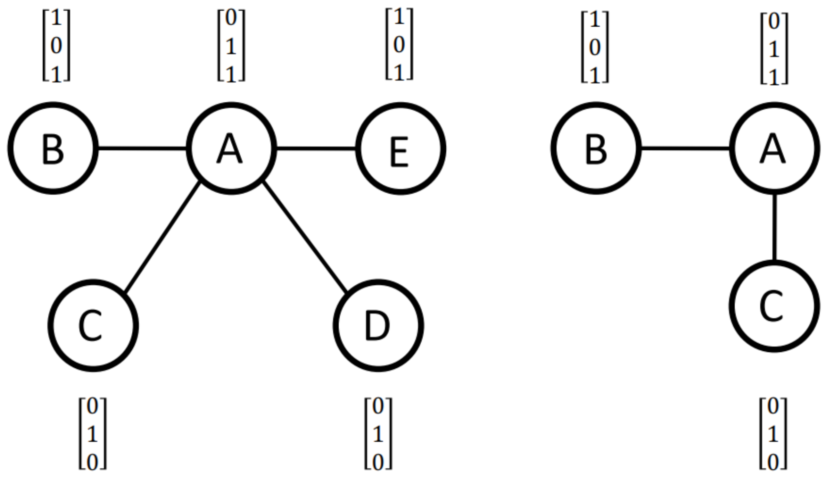
All numbers should be in the format of fraction rather than decimal (e.g., 3/4 rather 0.75).



(a) Case 1



(b) Case 2



(c) Case 3