COMP4901K and MATH4824B Final Exam

December 17, 2020, Thursday

Time: 04:30PM - 07:30PM

Instructor: Yangqiu Song

Name: ________________ **Student ID:** ________________

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

Q1. Yes/No Questions (10 Points)

Indicate whether each statement is true (✓) or false ($\times$).

1. Named Entity Recognition (NER) belongs to the Syntactic Understanding part of NLP.
2. Using N-grams in Bag-of-words can help capture local dependency and order.
3. In the evaluation of binary classification, the Recall metric is the fraction of predicted positive documents that are indeed positive.
4. In Laplace smoothing, if we set the strength of the prior to zero, then we will obtain maximum-likelihood estimation.
5. ReLU ($f(x) = \max(0, x)$, Figure 1) can not be used as the activation function for the output layer in a binary classification task, where cross entropy loss is used.
6. In convolutional layer for stationary input, filters are used to capture different patterns in the input space, where different locations are processed by filters with different parameters.
7. Neural language models can be used to train word vectors.
8. It is easy to incorporate new words or documents when using Singular Value Decomposition for dimensionality reduction.
9. In general, the RNN decoder of a Seq2seq model is uni-directional, as the decoding process is uni-directional.
10. The self-attention module in the Transformer architecture can help capture positional information of words by itself.

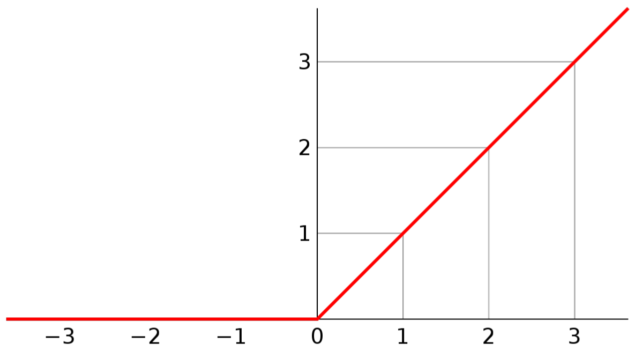


Figure 1: An illustration of ReLU function.

FTFTT FTFTF

Q2. Data Pre-processing (5 Points)

Consider the following corpus containing 3 sentences. It is stored in a Python *list* variable:

```
corpus = [
``In addition, wash your hands after touching something.'',
``The public should go out less and reduce social activities.'',
``Avoid touching your eyes, mouth and nose.'',
]
```

Write codes to perform tokenization, stop words removal, and stemming, based on the following Python codes.

```
import nltk
from nltk import word_tokenize # Tokenizer
from nltk.stem import PorterStemmer # Stemmer
stopwords = [``and'', ``is'', ``the'', ``a''] # The list of stop words
ps = PorterStemmer() # use ps.stem() to get the stemming of a certain word

num_doc = len(corpus)
tokenized_corpus = [___(1)___ for i in range(num_doc)]
tokenized_corpus_stop_filter = [___(2)___ for i in range(num_doc)]
tokenized_corpus_stop_filter_stemming = [___(3)___ for i in range(num_doc)]
```

a). Fill in the blanks of (1), (2), and (3) in the above codes to perform tokenization, stop words removal, and stemming. The interfaces of tokenizer, stop words, and stemmer are provided in the head of the codes.

The output of the whole process of pre-processing for an input sentence should be a list of processed word strings, e.g., `[``Natural'', ``language'', ``processing'']`. (3 points)

b). What is the output of `tokenized_corpus_stop_filter_stemming[2]` (The output of the 3rd sentence) of your pre-processing? (2 points)

a).
(1) word_tokenize(corpus[i])
(2) [w for w in tokenized_corpus[i] if w not in stopwords]
(3) [ps.stem(token) for token in tokenized_corpus_stop_filter[i]]
b). [“Avoid”, “touch”, “you”, “eye”, “,”, “mouth”, “nose”, “.”]
1 point for partially correct. 0 points for totally wrong.

Q3. Document Similarity (10 Points)

Consider the following document collection D = {D1, D2, D3} (given as one document per line):

```
D1: the beautiful flower the red flower the green leaf
D2: the red leaf
D3: the flower is small
```

Assume that the stopword list contains words “the” and “is”, and words are not stemmed. Please answer following questions relevant to *Document Vector Space Model*.

a). Write down the vector representations of D1, D2, and D3 using (base 2) log normalization term frequency (TF) weighting scheme ($log(1 + f_{t,d})$) (the dimensions should be listed alphabetically, i.e., the index for “leaf” should be precedent to “red”, and stopwords removal should be taken into consideration). (6 points)

b). Given a new document D4:

```
D4: the flower is red
```

compute the cosine similarity based on (base 2) log normalization TF weights between D4 and D1, D4 and D2, D4 and D3, respectively. (4 points)

a). After removing stops words, documents are as follows:

D1: beautiful flower red flower green leaf (0.5 point)
D2: red leaf (0.5 point)
D3: flower small (0.5 point)

The vocabulary is [“beautiful”, “flower”, “green”, “leaf”, “red”, “‘small”] (1.5 point)
So the BOW representation of each document is as follows:

D1: [1, log(3), 1, 1, 1, 0] (1 point)
D2: [0, 0, 0, 1, 1, 0] (1 point)
D3: [0, 1, 0, 0, 0, 1] (1 point)

(If the vocabulary is not in alphabetical order, then there are 1.5 points discounted (0.5 for each vector).)

b). The document 4’s vector is [0, 1, 0, 0, 1, 0] (1 point)

The cosine similarities are as follows:
between D1 and D4: 0.7163 (1 point)
between D2 and D4: 0.5 (1 point)
between D3 and D4: 0.5 (1 point)

Q4. Evaluation of Classification Results (10 Points)

Calculate Precision, Recall, and F1-score of the following model's predictions and document labels. Note: please compute the Precision, Recall, and F1-score of each class and take the average of each to report the result.

Document	Prediction	Label
0	Psychology	Business
1	Business	Technology
2	Entertainment	Psychology
3	Business	Business
4	Technology	Business
5	Technology	Technology
6	Entertainment	Entertainment
7	Technology	Business
8	Entertainment	Business
9	Psychology	Psychology
10	Business	Entertainment
11	Business	Technology
12	Psychology	Business
13	Business	Business
14	Entertainment	Technology
15	Technology	Business
16	Business	Business
17	Psychology	Business
18	Technology	Technology
19	Psychology	Psychology
20	Entertainment	Entertainment
21	Business	Entertainment
22	Technology	Business
23	Psychology	Psychology
24	Business	Business
25	Technology	Technology

8 points in total	Business	Technology	Psychology	Entertainment	Precision	F1
Business	4	2	0	2	1/2	2/5=0.4
Technology	4	3	0	0	3/7	6/13=0.462
Psychology	3	0	3	0	1/2	3/5=0.6
Entertainment	1	1	1	2	2/5	4/9=0.444
Recall	1/3	1/2	3/4	1/2		

for each class, 2 points if precision, recall, and f1 are all correct, 1.5 points if two are correct, 1 point if one is correct, 0 point if none is correct.

$$\text{average precision} = 1/4(1/2 + 3/7 + 1/2 + 2/5) = 16/35 = 0.4571$$
$$\text{average recall} = 1/4(1/3 + 1/2 + 3/4 + 1/2) = 25/48 = 0.5208$$
$$\text{average f1} = 1/4(2/5 + 6/13 + 3/5 + 4/9) = 223/468 = 0.4765$$

2 points if average precision, average recall and average f1 are all correct, 1.5 points if two are correct, 1 point if one is correct, 0 point if none is correct

Q5. Naive Bayes (10 Points)

Note: You need to apply the Laplace smoothing with strength of the prior = 0.5 to probabilities of each class and each word. Note that out of vocabulary words will **not** be considered in the test set.

We aim to use Naive Bayes model to determine whether a document is relevant to movie or not. There are two classes: relevant (+) and irrelevant (−), and take the following miniature training and test documents.

Table 1: training and test documents

	Class	Documents
Training	−	it tastes so good
	+	the hero is too powerful to be interesting
	−	the melody is so fantastic
	+	the plot is predictable and boring
	+	the most interesting film of this year
Test	?	the character is not interesting and the story is boring

Assume that the stopword list contains {"it", "so", "is", "too", "to", "be", "the", "and", "of", "this", "not"}.

Please compute:

a). the probability for each class, $P(y = +)$ and $P(y = -)$

$$P(y=+) = \frac{N_+ + 0.5}{N_{doc}+1} = \frac{7}{12}$$
$$P(y=-) = \frac{N_- + 0.5}{N_{doc}+1} = \frac{5}{12}$$

(2 points. 1 point for each probability.)

b). the probability for every word except stopwords and out of vocabulary words in the test document, $P(w_i|y = +)$ and $P(w_i|y = -)$

$$P(interesting|y=+) = \frac{2+0.5}{10+6.5} = \frac{5}{33}$$
$$P(interesting|y=-) = \frac{0+0.5}{4+6.5} = \frac{1}{21}$$
$$P(boring|y=+) = \frac{1+0.5}{10+6.5} = \frac{1}{11}$$
$$P(boring|y=-) = \frac{0+0.5}{4+6.5} = \frac{1}{21}$$

(4 points. 1 point for each probability.)

c). the **necessary** steps, and your final decision of the test document, relevant or irrelevant.

$$P(d|y=+) = \frac{5}{363} \text{ (1 point)}$$
$$P(d|y=-) = \frac{1}{441} \text{ (1 point)}$$
$$P(d|y=+)P(y=+) = \frac{35}{4356} \text{ (1 point)}$$
$$P(d|y=-)P(y=-) = \frac{5}{5292} \text{ (1 point)}$$

Because $P(d|y=-)P(y=-) < P(d|y=+)P(y=+)$, this test document is relevant. (0 points for final decision)

Q7. Convolutional Neural Network (10 Points)

Suppose we have a convolutional neural network consisting of the following components:

1. a word embedding layer with hidden dimension = 128;
2. a convolutional layer which has a filter K_1 of size 2×128, and another filter K_2 of size 4×128;
3. a max-pooling layer

Suppose we set the sequence length to be 20 and use valid padding (i.e., no padding), answer the following questions:

a). if the stride of both filters in the convolutional layer is 1, compute the dimensions of the feature maps generated by K_1 and K_2 respectively.

b). if the stride of K_1 is 2 and the stride of K_2 is 4, compute the dimensions of the feature maps generated by K_1 and K_2 respectively.

(a) $K_1 : (20 - 2)/1 + 1 = 19$, so the dimension is 19×1; (2 points)
$K_2 : (20 - 4)/1 + 1 = 17$, so the dimension is 17×1. (2 points)
(b) $K_1 : (20 - 2)/2 + 1 = 10$, so the dimension is 10×1; (3 points)
$K_2 : (20 - 4)/4 + 1 = 5$, so the dimension is 5×1. (3 points)

Q8. Neural Language Models (10 Points)

Feedforward Neural Network Language Model (NNLM) is an early architecture to train word vectors and language models jointly. The very basic idea of this model is to predict the next word given N precedent words using a feedforward neural network.

Here is the formal definition of the model. The vocabulary size of the corpus is V (including all the special tokens). Consider a sentence with m words $[w_1, w_2, \cdots, w_m]$. Let w_k be the current word, which is the k-th word in the sentence, and $\mathbf{e}(w_k)$ be it's corresponding embedding (word vector) with embedding size e. $\mathbf{y}(w_k)$ is the one-hot vector for the word w_k (e.g., if a word w_k is indexed the 3rd in the vocabulary, then $\mathbf{y}(w_k) = [0, 0, 1, 0, \cdots, 0] \in \mathbb{R}^V$). The model is trained using the following architecture to predict the current word w_k:

$$\mathbf{x}(w_k) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{e}(w_{k-i}) \tag{7}$$

$$\mathbf{h}(w_k) = g(\mathbf{W}\mathbf{x}(w_k) + \mathbf{b}) \tag{8}$$

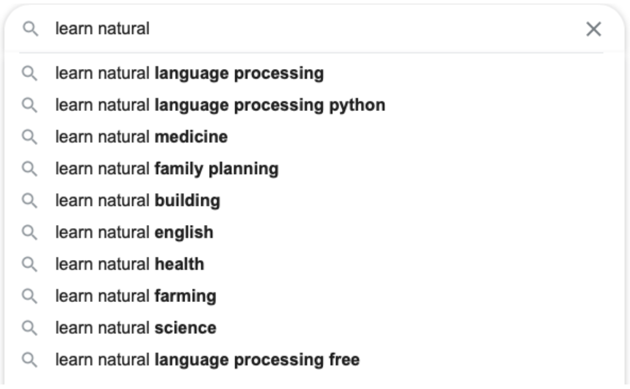
$$\hat{\mathbf{y}}(w_k) = f(\mathbf{U}\mathbf{h}(w_k) + \mathbf{d}) \tag{9}$$

$$\begin{aligned} J &= CrossEntropy(\mathbf{y}(w_k), \hat{\mathbf{y}}(w_k)) \\ &= -\sum_{i=1}^{V} \mathbf{y}(w_k)_i \cdot \log[\hat{\mathbf{y}}(w_k)_i] \end{aligned} \tag{10}$$

Here, $\mathbf{x}(w_k)$ is an e-dimensional vector, with the same size as the word embedding. $\mathbf{W}$ is an r-by-e matrix, and $\mathbf{b}$ is an r-dimensional bias vector. $\mathbf{U}$ is a V-by-r matrix, and $\mathbf{d}$ is a V-dimensional bias vector. Equation (10) is the cross-entropy loss function, where $\mathbf{y}(w_k)_i$ is the i-th element of vector $\mathbf{y}(w_k)$.

a). What should be the f in Equation (9)? Write down the name of the function.

b). An application of NNLM. Considering you are using a search engine, when you type in a few words it will automatically present you a list of candidate words that you may want to type in next.



learn natural

learn natural **language processing**
learn natural **language processing python**
learn natural **medicine**
learn natural **family planning**
learn natural **building**
learn natural **english**
learn natural **health**
learn natural **farming**
learn natural **science**
learn natural **language processing free**

Figure 2: An example of the search engine.

Now assume that this search engine uses NNLM as its backend algorithm for this next word prediction task. **Suppose you have typed in 3 words, "natural language processing"**, and the model tries to predict the next word using the above model. Here are the settings and parameters of the model. The vocabulary dictionary is in the Table 2. There are 5 words in the vocabulary in total.

Vocabulary words	natural	language	processing	python	example
Index	1	2	3	4	5

Table 2: Vocabulary dictionary

The parameters are listed below. The embedding matrix is **E**, a 2-by-5 matrix, where the embedding size is 2 and the i-th column (starting from 1) indicates the i-th word's embedding. **W** is a 2-by-2 matrix, while **b** is a 2-dimensional vector. **U** is a 5-by-2 matrix and **d** is a 2-dimensional vector. N is set to **3** here. In the calculation, Equation (7) take the average of N previous words and its shape is the same as the embedding size. In Equation (8), we set $\mathbf{h}(w_k) = g(\mathbf{W}\mathbf{x}(w_k) + \mathbf{b}) = \mathbf{W}\mathbf{x}(w_k) + \mathbf{b}$. If we take the most probable word predicted by the model, what will be predicted in this setting? Write down the steps of calculation. (*Hint*: you may not need to calculate the exact value of $\hat{\mathbf{y}}$. What you need is a ranking of all candidate words.)

$$\mathbf{E} = \begin{bmatrix} 0.0 & 0.5 & 0.4 & -0.2 & -0.8 \\ 1.0 & 0.3 & 0.8 & 0.2 & 0.2 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 2 & 0 \\ -1 & 2 \\ 0 & 1 \end{bmatrix}, \mathbf{d} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

c). Think about what you have learned in Word2vec. Compared to word2vec model, if we want to use the acquired embeddings in NNLM as word vectors for other downstream tasks like classification, what is the significant drawback of the NNLM model?

(a).
Softmax function. (1 points)
(b).

$$\mathbf{x} = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$$

$$\mathbf{h} = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.7 \\ 1.4 \end{bmatrix}$$

$$\hat{\mathbf{y}} = softmax(\begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 2 & 0 \\ -1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1.7 \\ 1.4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}) = softmax(\begin{bmatrix} 1.7 \\ -1.4 \\ 3.4 \\ 1.1 \\ 1.4 \end{bmatrix})$$

Take the word with the largest probability, which is "processing".
(2 points for **x**, 2 points for **h**, 2 points for $\hat{\mathbf{y}}$, 1 point for the final decision.)

Q9. Word Embeddings for Classification (10 Points)

Suppose we use word embeddings + 1-layer perceptron + softmax to solve a sentence classification problem. The word embedding matrix $\mathbf{E} \in \mathbb{R}^{3\times 8}$ is shown below, where 3 is the hidden dimension and 8 is the vocabulary size:

$$\mathbf{E} = \begin{bmatrix} 0.98 & 0.48 & 0.91 & 0.41 & 0.13 & 0.31 & 0.74 & 0.22 \\ 0.90 & 0.91 & 0.51 & 0.22 & 0.06 & 0.51 & 0.64 & 0.27 \\ 0.95 & 0.11 & 0.65 & 0.46 & 0.70 & 0.22 & 0.86 & 0.32 \end{bmatrix} \tag{11}$$

The vocabulary dictionary is {'_unk_': 0, '_pad_':1 , 'computer': 2, 'the': 3, 'i': 4, 'mathematics': 5, 'like': 6, 'course': 7}, where '_unk_' is the token for unknown word and '_pad_' is the token for padding. In this question, you don't need to deal with the stopwords. The input of the 1-layer perceptron is the unweighted average of the word embedding vectors of all tokens in input sentence, which should be a vector in $\mathbb{R}^3$. Suppose the parameters of the 1-layer perceptron are : weight matrix $\mathbf{W} = \begin{bmatrix} 2 & 0.5 & 2 \\ 1 & 2 & 0.5 \\ 0.5 & 1 & 1 \end{bmatrix}$, and bias $\mathbf{b} = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}$. Given an input vector $\mathbf{v}_i \in \mathbb{R}^3$, the output of the 1-layer perceptron is computed as $\mathbf{W}\mathbf{v_i} + \mathbf{b}$. Suppose there are three candidate labels.

Consider the forward pass of classifying the sentence 'i like the computer programming course', answer the following questions:

a). Compute the output of the word embedding layer, which should be a 3×6 matrix .

b). Compute the output of the softmax layer.

(a) Convert the input sentence into word index sequence : [4,6,3,2,0,7], (2 points)

so the output of the word embedding layer is $\begin{bmatrix} 0.13 & 0.74 & 0.41 & 0.91 & 0.98 & 0.22 \\ 0.06 & 0.64 & 0.22 & 0.51 & 0.90 & 0.27 \\ 0.70 & 0.86 & 0.46 & 0.65 & 0.95 & 0.32 \end{bmatrix}$ (2 points)

(b) Take the average of the output matrix of the embedding layer, we have $\mathbf{v_1} = \begin{bmatrix} 0.565 \\ 0.433 \\ 0.657 \end{bmatrix}$ (1 point)

The output of the 1-layer perceptron is

$$\mathbf{v_2} = W\mathbf{v_1} + \mathbf{b}$$
$$= \begin{bmatrix} 2.661 \\ 1.760 \\ 1.373 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.661 \\ 2.260 \\ 2.373 \end{bmatrix}$$

(3 points)

So the output of the softmax layer is $\begin{bmatrix} 0.413 \\ 0.277 \\ 0.310 \end{bmatrix}$ (2 points)