CSCI-GA-2590: Natural Language Processing (Spring 2023)

Midterm (5:00pm–7:00pm, March 7)

- You should finish the exam within **1 hours and 45 minutes** and submit through Gradescope by **7pm EST**.
- You can refer to textbooks, lecture slides, and notes. However, searching answers online and collaborating with others are not allowed.
- Please write your solution on a separate sheet or the released exam sheet clearly, then upload the photo of your handwriting or the annotated pdf to Gradescope.
- Make sure you leave enough time to upload the exam. **Gradescope will terminate the submission window at 7:00pm promptly. There will be no grace period**.

1 True or False

1. (1 point) Naive Bayes models assume that the features (words) in the text are conditionally independent given the class, while logistic regression does not make this assumption.

   [✓ **True**] ◯ False

2. (1 point) Naive Bayes models are not suitable for multi-class text classification.

   ◯ True [✓ **False**]

3. (1 point) Using pretrained word embeddings to represent input tokens handles the problem of out-of-vocabulary words.

   ◯ True [✓ **False**]

   Solution: Each word embedding corresponds to a word in the vocabulary which is fixed before training, so it cannot be extended to OOV words at inference time.

4. (1 point) The loss function of the skip-gram model is convex.

   ◯ True [✓ **False**]

5. (1 point) RNNs can be used to encode variable-length text.

   [✓ **True**] ◯ False

6. (1 point) In multihead attention, the computation of different attention heads can be parallelized.

   [✓ **True**] ◯ False

7. (1 point) In beam search, let the beam size be k, the vocab size be v and the output sequence length be n. Then, assuming no parallelization, the time complexity of beam search is $O(nkv)$.

   ◯ True [✓ **False**]

   Solution: At each of the n time steps, we need to find the top-k states out of kv states, this cost is not constant. Thus, the time complexity must be larger than $O(nkv)$. The exact cost depends on the algorithm used to find the top-k elements.

8. (1 point) A decoder model can be used to generate text sequentially.

   [✓ **True**] ◯ False

9. (1 point) Solving coreference resolution may require commonsense knowledge.

   [✓ **True**] ◯ False

10. (1 point) The next sentence prediction objective used by BERT is cruicial for its performance.

    ◯ True [✓ **False**]

    Solution: Later works (e.g., Roberta) showed that the NSP objective is not necessary.

2 Multiple Choices

Note: There can be more than one correct answers; select all that apply! **No partial credits: all correct answers must be checked**.

1. (2 points) Which of the following is/are true about logistic regression?

    [✓ **It works with arbitrary features.**]
    ◯ It has a closed-form MLE (maximum-likelihood estimation) solution.
    ◯ Its output is binary (0 or 1).
    ◯ Each feature is assumed to be independent to any other feature.

    Solution: The output $p(y = 1 \mid x)$ is a probability, which is not binary.

2. (2 points) Consider the following set of documents:

    D1: The sky is dark
    D2: The star is shining
    D3: The star in the sky is shining

    Represent each word by its frequency in every documents (i.e. rows of the word $\times$ document matrix). What is the word that is closest to "star" using cosine distance?

    ◯ sky
    ◯ the
    [✓ **shining**]
    ◯ in

3. (2 points) Which of the following can mitigate gradient vanishing when training an RNN model?

    ◯ gradient clipping
    [✓ **truncated backpropagation**]
    [✓ **residual connections**]
    ◯ small learning rate

Solution: Any operation that can reduce the number of recurrence (which causes repeated multiplication in the gradient) would mitigate gradient vanishing.

4. (2 points) Alice and Bob are trying to solve a sequence generation task where the input has length n and the output has length m. To model the output distribution given the input, Alice proposes to use a Transformer encoder-decoder. Bob proposes to use a single Transformer decoder that first reads in the input (without outputing the next word distribution) and then generates the output. They are wondering which model is more efficient.

   Let K_{encdec} and K_{dec} be the total number of attention scores computed for each example by the encoder-decoder model and the decoder-only model during training. (Recall that in Transformers an attention score is the scaled dot product between a query and a key vector.) Assume that all Transformer encoder and decoder have a single layer. Can you help them figure out the relation between K_{encdec} and K_{dec}?

   ○ $K_{\text{encdec}} > K_{\text{dec}}$
   [✓ $K_{\textbf{encdec}} < K_{\textbf{dec}}$]
   ○ $K_{\text{encdec}} = K_{\text{dec}}$
   ○ The relation depends on specific values of m and n.

Solution:

- Encoder-decoder: $n^2 + m^2 + mn$ (encoder attention + decoder attention + encoder-decoder attention)
- Decoder only: $(m + n)^2$
- Note that the masked attention weights still need to be computed during training.

5. (2 points) Recall that BLEU-n denotes the BLEU score computed based on precisions of unigrams up to n-grams. Given outputs of a machine translation system and their references, Alice decides to compute BLEU-2 while Bob decides to computed BLEU-3. Which score do you think will be higher?

   [✓ **BLEU-2 $\geq$ BLEU-3**]
   ○ BLEU-3 $\geq$ BLEU-2
   ○ BLEU-3 = BLEU-2
   ○ The relation between BLEU-3 and BLEU-2 depends on the specific outputs and references.

Solution: Let p_1, p_2, p_3 be the unigram, bigram and trigram precision defined in BLEU. It is easy to see that $p_3 \leq p_2 \leq p_1$ since each correct trigram implies a correct unigram and a correct bigram, but not the other way around.

$$\begin{aligned}
&\log(p_1 p_2)^{1/2} - \log(p_1 p_2 p_3)^{1/3} \\
=&\log \frac{(p_1 p_2)^{1/2}}{(p_1 p_2 p_3)^{1/3}} \\
=&\log \left(\frac{p_1 p_2}{p_3^2}\right)^{1/6} \\
=&\frac{1}{6}(\log p_1 + \log p_2 - 2\log p_3) \geq 0
\end{aligned}$$

Therefore, BLUE-2 is at least as large as BLEU-3.

6. (2 points) Which of the following is/are structured prediction tasks?

   ✓ **POS tagging**
   ✓ **Semantic parsing**
   ✓ **Coreference resolution**
   ◯ Natural language inference

7. (2 points) Consider a distribution over sentences $p(x)$. Let s be the sequence produced by greedy decoding from p. Which of the following decoding strategy is most likely to produce s?

   ◯ top-p sampling with $p = 1$
   ✓ **top-k sampling with $k = 1$**
   ◯ tempered sampling with $T = 1$
   ◯ tempered sampling with $T = 0.1$

Solution: A and C are equivalent to standard sampling. B is equivalent to greedy sampling. D is close (though not equivalent) to greedy sampling.

8. (2 points) Which of the following technique(s) is(are) NOT used in pretraining large models like BERT?

   ◯ stochastic gradient descent
   ✓ **beam search**
   ◯ layer normalization
   ✓ **negative sampling**

9. (2 points) In an LSTM, suppose the forget gate f_t is always 0, the input gate i_t is always 1, and the output gate is always 1. Which of the following is/are true?

◯ The memory cell is never updated.
◯ The LSTM has no gradient vanishing/explosion problem.
$\surd$ **The LSTM is equivalent to an RNN (i.e. they output the same hidden states).**
◯ The LSTM has no recurrence (i.e. the output from the previous step has no effect on the output at the current step).

3 Written Questions

You can either type your answers in the text box or write it on paper and upload an image.

1. You are using logistic regression to classify the following (toy) dataset:

x	y
not good	-1
good	$+1$
bad	-1

You decide to use the bag-of-words representation as the feature vector such that

$$\phi(x) = \begin{bmatrix} \mathbb{I}(\text{good}) \\ \mathbb{I}(\text{bad}) \\ \mathbb{I}(\text{not}) \end{bmatrix}$$

where the indicator function $\mathbb{I}(\cdot)$ returns 1 if the word is in the input and 0 otherwise.

Recall that the logistic regression model outputs

$$p(y = 1 \mid x) = \frac{1}{1 + \exp(-w \cdot \phi(x) - b)}$$

Let's assume b is zero in the following questions.

(a) (3 points) Give the value of w that would achieve zero training error. (Training error is the number of misclassified examples in the training set divided by the total number of training examples)

Solution: The answer is not uniqued; one solution is $w = (1, -1, -2)$

(b) (2 points) Is your solution in the previous question unique? Why or why not?

Solution: No. Any positive scaling of w would also achieve zero training error.

(c) (2 points) Can you find a weight vector that achieves lower loss than your solution in (a)? If so, give such a weight vector; if not, explain why.

Solution: Yes. cw where $c > 1$.

Note that loss refers to the logistic loss, which is different from error rate (number of misclassified examples divided by the total number of examples).

(d) (1 point) Using your solution in (a), what's the model prediciton on the input "not bad"?

Solution: -1

(e) (3 points) Does there exist a weight vector that would achieve zero training error and classify "not bad" as +1? If so, give such a vector; if not, explain why.

Solution: To achieve zero training error, we have

$$w_1 > 0, w_2 < 0, w_3 + w_1 < 0 \implies w_3 < 0$$

To classify "not bad" as +1, we have

$$w_2 + w_3 > 0 \implies w_3 > 0$$

Therefore, such a w doesn't exist.

2. Recall that to compute word vectors using the skip-gram method, given a target word, we predict words in its context window. Let $\mathcal{V}$ be the vocabulary, $u_w \in \mathbb{R}^d$ be the embedding of a word $w \in \mathcal{V}$ in the context, and $v_w \in \mathbb{R}^d$ be the embedding of a target word $w \in \mathcal{V}$. Then, the skip-gram model can be written as

$$p(c \mid w) = \frac{\exp\left(u_c \cdot v_w\right)}{\sum_{c' \in \mathcal{V}} \exp\left(u_{c'} \cdot v_w\right)}$$

(Note that we have simplied the notation a bit from the lecture slides.)

(a) (2 points) Write down the log-likelihood for a single example (c, w) under the skip-gram model.

$$\ell(c, w) =$$

Solution:

$$\ell(c, w) = u_c \cdot v_w - \log\left(\sum_{c' \in \mathcal{V}} \exp\left(u_{c'} \cdot v_w\right)\right)$$

(b) (3 points) Write down the partial derivative of $\ell(c, w)$ with respect to the word vector v_w:

$$\frac{\partial \ell}{\partial v_w} =$$

Solution:

$$\ell(c, w) = u_c \cdot v_w - \log\left(\sum_{c' \in \mathcal{V}} \exp\left(u_{c'} \cdot v_w\right)\right)$$

$$\therefore \frac{\partial \ell(c, w)}{\partial v_w} = u_c - \frac{\left(\sum_{c' \in \mathcal{V}} u_{c'} \cdot \exp\left(u_{c'} \cdot v_w\right)\right)}{\left(\sum_{c' \in \mathcal{V}} \exp\left(u_{c'} \cdot v_w\right)\right)}$$

(c) (2 points) Based on your results in (b), explain why training skip-gram word vectors is expensive.

Solution: Comoputing the derivative of the log likelihood requires us to find the dot product $\sum_{c' \in \mathcal{V}} \exp(u_{c'} \cdot v_w)$. Since c' spans the entire vocabulary, we would require to find the dot product with the embedding of all the words which is expensive.

(d) (2 points) To reduce the computation, we convert the word prediction problem into a binary classification problem using negative sampling, where we classify whether (c, w) is a real pair of context and target words appearing in the dataset or not. Write down the **loss** for a single example (c, w, y) where $y \in \{1, 0\}$.

Solution:

$$\ell(c, w, y) = -\Big(y \cdot \log(p(y = 1 \mid w, c)) + (1 - y) \cdot \log(1 - p(y = 1 \mid w, c))\Big)$$

(e) (2 points) Briefly explain why negative sampling is faster to train than the original skip-gram objective.

Solution: In negative sampling, complexity of gradient computation is bounded by the number of negative samples we choose. However, in the original skip-gram objective, gradient computation is bounded by the size of vocabulary since we end up finding probability distribution over the entire vocabulary.

3. Consider a single head self-attention module. Let the input values, queries, and keys be

$$V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, K = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}, Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}.$$

In addition, assume that $k_i, q_i, v_i \in \mathbb{R}^3$ and

(a) (1 point) What's the dimension of the output matrix

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix}$$

Solution: 3x3

(b) (2 points) Write down an expression for the output o_3.

Solution:

$$\sum_{i=1}^{3} \frac{e^{q_i \cdot k_i}}{\sum_{i=1}^{3} e^{q_i \cdot k_i}} v_i$$

(c) (3 points) Suppose $v_1 = [1, 0, 0], v_2 = [0, 0, 1], v_3 = [1, 1, 0]$ and $K = V$. Now, we would like o_3 to "copy" the value v_1, i.e. $o_3 \approx v_1$. Give a value of q_3 that satisfies this condition. (HINT: The value is not unique; as long as o_3 is closer to v_1 than other values, it is fine. Think about what should be the relation of the attention weights with respect to q_3.)

Solution: Any value that satisfies $q_3 \cdot v_1 \gg q_3 \cdot v_2$ and $q_3 \cdot v_1 \gg q_3 \cdot v_3$, e.g., $[100, -100, 0]$

(d) (3 points) Following the previous question, now we would like o_3 to "summarize" the previous two inputs, i.e. $o_3 \approx v_1 + v_2$. [CORRECTION: $o_3 \approx (v_1 + v_2)/2$] Give a value of q_3 that satisfies this condition.

Solution: Any value that satisfies $q_3 \cdot v_1 = q_3 \cdot v_2 \gg q_3 \cdot v_3$, e.g., $[100, -100, 100]$

(e) (2 points) Note that in the current implementation, assuming q_i's are independent, then o_i's are also independent to each other. Describe one scenario where we would like to model dependency between the outputs.

Solution: Many answers can be correct. One example is that in POS tagging, we may need to know the previous tag to disambiguite whether a word (e.g., "run") is a noun or a verb.

(f) (2 points) Which of the following modification of the self-attention module can model dependency among o_1, o_2, o_3? (select all that applies)

◯ Adding position embeddings to the input.
√ **Adding another self-attention layer on top of it.**
◯ Adding L2 regularization during learning.
◯ Increase embedding size.

Solution: A common mistake is A. Here we are looking for some operation that allows o_i to influence o_j, and position embedding doesn't achieve this.