Midterm Exam

601.471/671 NLP: Self-Supervised Models

Spring 2023

Johns Hopkins University

March 2023

Complete all questions.

Use additional paper if needed.

Time: 70 minutes.

Name of student: _____

Q1. Word Embeddings *20 points*

1. (8 points) Describe a summary of training Word2Vec embeddings (less than five sentences).

   Word2Vec is a neural network model that learns word embeddings, which are vector representations of words in a high-dimensional space. The training process involves feeding a large corpus of text into the model, which then learns to predict the probability of a word given its context words or vice versa. The resulting word embeddings can capture semantic and syntactic relationships between words and are commonly used in natural language processing tasks such as text classification and language translation. The training process can be optimized using techniques such as negative sampling or hierarchical softmax.

2. (8 points) What are "one-hot" vectors and why are they not good choices for representing word meanings (mention two of them.)

   One-hot vectors are sparse vectors where each dimension represents a word in a vocabulary, and only the dimension corresponding to a particular word is set to 1, with all other dimensions set to 0. One of the major drawbacks of one-hot vectors for representing word meanings is that they are high-dimensional and sparse, which makes them computationally expensive and memory-intensive. This can become problematic when working with large vocabularies or datasets. Additionally, one-hot vectors are unable to capture the relationships and similarities between words, as they treat each word as independent and unrelated to other words. Therefore, one-hot vectors are not effective at representing word meanings in natural language processing tasks.

3. (3 points) Give three examples of how we can evaluate word vectors (just name them briefly). For each example, please indicate whether it is intrinsic or extrinsic.

   Here are three examples of how we can evaluate word vectors:Word similarity: intrinsic; Part-of-speech (POS) tagging: extrinsic; Sentiment analysis: extrinsic

4. (1 point) Indicate the statement(s) that are correct about Skip-grams (or indicate if none are correct).
   ☐ Its objective function is to build a good predictive model of the surrounding context words given a center word.
   ☐ The final word embedding for a word is the average or sum of the input (center) embedding v and output (outside) vector u corresponding to that word.
   ☐ It makes use of global co-occurrence statistics.

   None are correct.

Q2. Neural Nets and Backpropagation *35 points*

1. (3 points) The eXclusive OR (XOR) function is a boolean logic operation widely used in cryptography, which is given by

$$y = \text{XOR}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = (0,1) \text{ or } \mathbf{x} = (1,0) \\ 0 & \text{if } \mathbf{x} = (1,1) \text{ or } \mathbf{x} = (0,0) \end{cases}$$

Can the following two-layer network represent an XOR function on $\mathbf{x} = [x_1, x_2]$? Briefly explain your reasoning in one sentence.

$$\mathbf{h} = \mathbf{W}_1 \cdot \mathbf{x} + \mathbf{b}_1$$
$$\hat{y} = \mathbf{W}_2 \cdot \mathbf{h} + \mathbf{b}_2$$

☐ Yes ☐ No

Reasoning:

No. We need activation functions to let this model learn nonlinear functions such as XOR.

2. (6 points) What is Softmax function? How do we use the Softmax function in a sequence classification model?

Given the score of each class, the softmax estimates the probabilities of p_i associated with each class.

$$\frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$$

In a sequence classification model, we first estimate the score of each token with a neural network and then use softmax to produce probabilities for each token. Finally, we can estimate the cross-entropy between our prediction and ground-truth to measure the quality of our prediction.

3. (6 points) Explain why we need activation functions. What would happen if we don't use activation functions in a multi-layer perceptron (MLP)? Give one example of an activation function and list one limitation.

Activation functions introduce non-linearity to the output of a neuron or a layer of neurons, allowing the neural network to model complex, nonlinear relationships between inputs and outputs.
An example activation is the ReLU function, which is given by

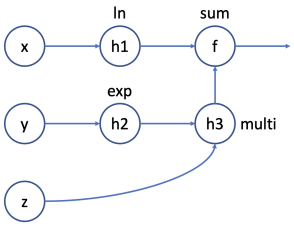
$$f(x) = \max(0, x)$$

A limitation of ReLU activation is that the gradients are zero when $x < 0$, effectively killing the neurons.

4. (20 points) Draw the computation graph for $f(x, y, z)$. Each node in the graph should correspond to only one simple operation (addition, multiplication, exponentiation, etc.). Then we will follow the forward and backward propagation described in class to estimate the value of f and partial derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$ at $[x, y, z] = [1, 3, 2]$.

$$f(x, y, z) = \ln x + \exp(y) \cdot z$$

(a) Draw the computation graph for $f(x, y, z) = \ln x + \exp(y) \cdot z$. The graph should have three input nodes for x, y, z and one output node f. Label each intermediate node h_i.



(b) Run the forward propagation and evaluate f and h_i ($i = 1, 2, \dots$) at $[x, y, z] = [1, 3, 2]$.

Solution redacted because it's part of our HW.

(c) Run the backward propagation and give partial derivatives for each intermediate operation, i.e., $\frac{\partial h_i}{\partial x}$, $\frac{\partial h_j}{\partial h_i}$, and $\frac{\partial f}{\partial h_i}$. Evaluate the partial derivatives at $[x, y, z] = [1, 3, 2]$.

Solution redacted because it's part of our HW.

(d) Aggregate the results in (c) and evaluate the partial derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$ with chain rule. Show your work.

Solution redacted because it's part of our HW.

Q3. Language Modeling *17 points*

Consider the following vocabulary, $V = \{\texttt{BOS}, \texttt{EOS}, \texttt{who}, \texttt{framed}, \texttt{roger}, \texttt{rabbit}, \texttt{the}\}$ where `BOS` is the dummy token indicating the beginning of a sentence, and `EOS` indicates the end of a sentence. Consider the following training data:

```
BOS who EOS
BOS who framed roger EOS
BOS roger rabbit EOS
BOS who framed roger rabbit EOS
BOS roger framed who EOS
BOS who framed the rabbit EOS
```

1. (9 points) Compute the following probabilities: $P(\texttt{rabbit})$, $P(\texttt{rabbit}|\texttt{roger})$, $P(\texttt{EOS}|\texttt{rabbit})$.

   Solution redacted because it's part of our HW.

2. (8 points) Briefly explain what the sparsity problem is in n-gram language models? Briefly explain what enables neural language models to overcome this issue.

   Solution redacted because it's part of our HW.

Q4. Transformers *28 points*

1. (14 points) Define the Self-Attention operation and compute its computational complexity when applied to a sequence n inputs $\mathbf{x}_1, ..., \mathbf{x}_n$ where $\mathbf{x}_i \in \mathbb{R}^d$.

   Let the input tokens be $\mathbf{X}$, the key, query, and value are given by

   $$\mathbf{K} = \mathbf{X}\mathbf{W}^k, \mathbf{Q} = \mathbf{X}\mathbf{W}^q, \mathbf{V} = \mathbf{X}\mathbf{W}^v$$

   Then the self-attention is given by

   $$\mathbf{S} = softmax\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_q}}\right)\mathbf{V}$$

   The computational complexity of self-attention is quadratic, i.e., $O(n^2 d)$.

2. (8 points) List two advantages of Transformers (which use self-attention) over Recurrent Neural Language Models (RNNs).

   First transformers can capture longer-term dependencies. Instead of relying on the sequential flow of information with hidden states, transformers directly associate relevant tokens with the self-attention mechanism.
   Moreover, transformers perform parallel computation. RNNs process one token at a time while transformers process all elements simultaneously.

3. (6 points) Briefly explain what positional embeddings are. What would happen if we don't use positional embeddings in a Transformer model?

   Positional embedding is a set of given or learned embeddings that are added to the input embeddings of a sequence, which allows Transfomer models to model the absolute or relative order of the input tokens.
   Without positional embeddings, sentences will be regarded an unordered set of text tokens and we cannot accurately capture the semantics of the sentence due to ambiguity.