CS 7650 Midterm

Instructor: Dr. Wei Xu
TAs: Mounica Maddela, Nathan Vaska, and Sarah Wiegreffe

1 Word Embeddings

The WORD2VEC algorithm revolutionized the field of NLP by providing a high-quality, but cheaply computable means for producing continuous vector representations of words learned from a large, unlabelled corpus. Here, we will investigate the objectives used in the WORD2VEC algorithm. This question may require you to refer to Chapters 14.5, 14.6 of the Eisenstein readings.

Here is a sentence for which the algorithm will make a prediction for the missing word. The word embedding for each word in the context has been given.

Index Position	Word	Embedding
0	the	$[2, 1]$
1	quick	$[3, 2]$
2	brown	$[-2, 0]$
3	?	?
4	jumped	$[5, -1]$
5	over	$[2, -3]$
6	the	$[2, 1]$
7	lazy	$[-3, -1]$
8	dog	$[1, 2]$

Table 1: Word Embeddings for the Input Sentence.

1. Compute the Continuous Bag-of-Words (CBOW) vector representation of the missing word for a context window h of size 3. Show your work.

2. We've subset the vocabulary down to the words in Table 2. Fill in the scores of each word being the missing word in Table 2. Use the base-2 exponent and round to 2 decimal places.
   Hint: use dot products for this, not traditional vector-space similarity.

3. Which word would be predicted by the CBOW algorithm to be the missing word?

Word	Embedding	Unnormalized Score	Normalized Score (P(Word))
dog	$[1, 2]$		
horse	$[3, 4]$		
motorcycle	$[0, -1]$		
leopard	$[3, 0]$		
wolf	$[4, 0]$		

Table 2: A subset of the vocabulary of the CBOW model.

2 LSTMs

$$
\begin{aligned}
\boldsymbol{f}_{m+1} &= \sigma(\boldsymbol{\Theta}^{(h\to f)}\boldsymbol{h}_m + \boldsymbol{\Theta}^{(x\to f)}\boldsymbol{x}_{m+1} + \boldsymbol{b}_f) && \text{forget gate} \\
\boldsymbol{i}_{m+1} &= \sigma(\boldsymbol{\Theta}^{(h\to i)}\boldsymbol{h}_m + \boldsymbol{\Theta}^{(x\to i)}\boldsymbol{x}_{m+1} + \boldsymbol{b}_i) && \text{input gate} \\
\tilde{\boldsymbol{c}}_{m+1} &= \tanh(\boldsymbol{\Theta}^{(h\to c)}\boldsymbol{h}_m + \boldsymbol{\Theta}^{(w\to c)}\boldsymbol{x}_{m+1}) && \text{update candidate} \\
\boldsymbol{c}_{m+1} &= \boldsymbol{f}_{m+1} \odot \boldsymbol{c}_m + \boldsymbol{i}_{m+1} \odot \tilde{\boldsymbol{c}}_{m+1} && \text{memory cell update} \\
\boldsymbol{o}_{m+1} &= \sigma(\boldsymbol{\Theta}^{(h\to o)}\boldsymbol{h}_m + \boldsymbol{\Theta}^{(x\to o)}\boldsymbol{x}_{m+1} + \boldsymbol{b}_o) && \text{output gate} \\
\boldsymbol{h}_{m+1} &= \boldsymbol{o}_{m+1} \odot \tanh(\boldsymbol{c}_{m+1}) && \text{output.}
\end{aligned}
$$

Figure 1: LSTM update equations (Eisenstein Ch. 6.33).

The update equations for a LSTM at timestep $m+1$ are given in Figure 1. Eisenstein Chapter 6.3 may be useful in answering this question.

1. In Table 3 we provide weight values and in Table 4 timestep inputs. We'll now compute the value of $\boldsymbol{h}_{m+1}$ using Table 3 and Table 4:

   $\boldsymbol{f}_{m+1} = \sigma(4 + 4 + 0) = 1.0$
   $\boldsymbol{i}_{m+1} = \sigma(-1 + 9 + 1) = 1.0$
   $\tilde{\boldsymbol{c}}_{m+1} = \tanh([4, -8, -4]^T + [-3, 12, 1]^T) = \tanh([1, 4, -3]^T) = [0.76, 1.0, -1.0]^T$
   $\boldsymbol{c}_{m+1} = 1.0 \odot [1, 0, -4]^T + 1.0 \odot [0.76, 1.0, -1.0]^T = [1.76, 1.0, -5.0]^T$
   $\boldsymbol{o}_{m+1} = \sigma(2 + 2 - 1) = 1.0$
   $\boldsymbol{h}_{m+1} = 1.0 \odot \tanh([1.76, 1.0, -5.0]^T) = [\mathbf{0.94}, \mathbf{0.76}, \mathbf{-1.0}]^\mathbf{T}$

   The gates of this LSTM do not restrict the flow of any information. To effectively turn this LSTM into an Elman RNN at the current timestep, i.e., include **only** information from the current input and prior hidden state and **no** information from the prior memory cell in $\boldsymbol{h}_{m+1}$, describe the values that you would need to set the gates $\boldsymbol{f}_{m+1}, \boldsymbol{i}_{m+1}$ and $\boldsymbol{o}_{m+1}$ equal to.

2. Which variable from the list of intermediate variables in Figure 1 most closely resembles the hidden state of a standard Elman RNN? (Answer choices are $\boldsymbol{f}_{m+1}$, $\boldsymbol{i}_{m+1}$, $\tilde{\boldsymbol{c}}_{m+1}$, $\boldsymbol{c}_{m+1}$, $\boldsymbol{o}_{m+1}$, $\boldsymbol{h}_{m+1}$).

Weight	Value
$\boldsymbol{\Theta}^{(h\to f)}$	$[1, -2, -3]$
$\boldsymbol{\Theta}^{(x\to f)}$	$[0, -1, -2]$
$\boldsymbol{b}_f$	0
$\boldsymbol{\Theta}^{(h\to i)}$	$[0, 0, 1]$
$\boldsymbol{\Theta}^{(x\to i)}$	$[-1, -2, -2]$
$\boldsymbol{b}_i$	1
$\boldsymbol{\Theta}^{(h\to c)}$	$\begin{bmatrix} 0 & 1 & -3 \\ -3 & 1 & 0 \\ -2 & -1 & -3 \end{bmatrix}$
$\boldsymbol{\Theta}^{(w\to c)}$	$\begin{bmatrix} 1 & 0 & 0 \\ -2 & -3 & 0 \\ 1 & -1 & -2 \end{bmatrix}$
$\boldsymbol{\Theta}^{(h\to o)}$	$[1, 0, 1]$
$\boldsymbol{\Theta}^{(x\to o)}$	$[-1, 0, 1]$
$\boldsymbol{b}_o$	-1

Table 3: Weights for LSTM.

Vector	Value
$\boldsymbol{h}_m$	$[3, 1, -1]^T$
$\boldsymbol{c}_m$	$[1, 0, -4]^T$
$\boldsymbol{x}_{m+1}$	$[-3, -2, -1]^T$

Table 4: Input/intermediate variables for LSTM.

3. In this problem, all the LSTM gates are scalars. What changes would have to be made to Table 3 in order to create vector gates? (Specify which weights would change and what their new dimensions would be). What is the benefit of vector gates over scalars?

4. What two problems in RNNs does the inclusion of the memory cell $\boldsymbol{c}_{m+1}$ improve? What property of its computation allows it to do this?

3 Beam Search Decoding

Consider the following bigram language model. The bigram probabilities are given in table 5. Each probability is of the form $P(x_i|x_{i-1})$, where x_i corresponds to i^{th} word in a post / word sequence. Here, $\langle s \rangle$ denotes the start of a sentence.

1. Given a prefix string "$\langle s \rangle$ *I know*", what are the next 3 possible tokens. Run beam search with width $k = 2$ and generate the next 3 tokens.

   Assume $P(\langle s \rangle\ I\ know) = 1$. Make sure you show the probabilities for each step. Also, show the word sequences in the beam at the end of each step.

Bigram	Prob.	Bigram	Prob.
P(, | know)	0.4	P(important | the)	0.3
P(the | know)	0.6	P(response | correct)	0.6
P(I | ,)	0.8	P(answer | correct)	0.25
P(it | ,)	0.2	P(problems | correct)	0.15
P(will | I)	0.4	P(response | important)	0.5
P(know | I)	0.6	P(answer | important)	0.5
P(was | it)	1.0	P(answer | exact)	1.0
P(correct | the)	0.5	P(exact | the)	0.2

Table 5: Bigram Language Model probabilities for Beam Search.

2. Lets introduce some randomness into the standard beam search method used in question 3.1. At the end of each step, we randomly choose one of the top-k beam candidates and discard the rest. In other words, only the randomly chosen top-k candidate is expanded in the next step instead of all the top-k candidates. Figure 2 illustrates this sampling approach.

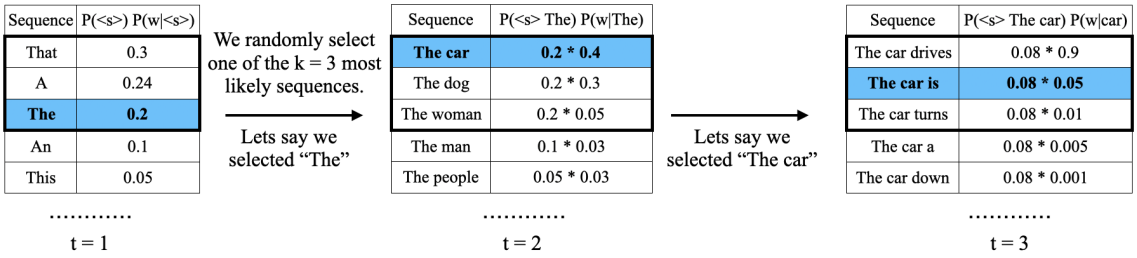

Figure 2: Example of top-k random sampling. At each timestep t, the model generates the probability of the possible next word. Then, we randomly sample from the k most likely candidates from this distribution. Here, we consider $k = 3$. **Bold** sequences represent the sampled candidate at each timestep.

Once again, consider the bigram language model in Table 5.

(a) Given the prefix string "$\langle s \rangle$ *I know*", run the top-k sampling approach for the next 3 tokens. Let S be the set of output sequences that this approach could possibly generate at the end of 3 steps. What is the size of S? Assume $k = 2$ and $P(\langle s \rangle \ I \ know) = 1$. You can just report the number.

(b) What is the sequence with maximum probability in S? Report the sequence and the probability of the sequence.

(c) What is the sequence with minimum probability in S? Report the sequence and the probability of the sequence.

4 Evaluation

1. Consider the following sentence with gold and predicted named entity tags.

Predicted NER Tags														
Barack	and	Michelle	Obama	attend	the	WHCD	event	at	the	Hilton	Hotel	in	Washington	
O	O	B-PER	I-PER	O	O	B-LOC	O	O	O	B-LOC	O	O	B-LOC	
Gold NER Tags														
Barack	and	Michelle	Obama	attend	the	WHCD	event	at	the	Hilton	Hotel	in	Washington	
B-PER	O	B-PER	I-PER	O	O	O	O	O	O	B-LOC	I-LOC	O	B-LOC	

Table 6: Predicted and gold NER tag sequences.

Compute the overall precision, recall, and F1 scores for the predicted entity tag sequence. Assume that the prediction is correct only when there is an exact match between the predicted entity and gold spans. You don't need to differentiate between the entity types. True positives will be the entity spans that match with the gold. False positives will be the entity spans in the predicted sequence that do not match or are not contained in the gold sequence. False negatives will be the entity spans in the gold sequence that do not match or are not contained in the predicted sequence.

2. *BLEU* score is the most common automatic evaluation metric for machine translation. Given a candidate translation $\mathbf{c}$ and human-written reference translations $\mathbf{r_1}, ..., \mathbf{r_k}$, we compute the *BLEU* score of $\mathbf{c}$ as follows:

   We first compute the modified n-gram precision p_n of $\mathbf{c}$ with $n = 1, 2, 3, 4$. Here n is the size of ngram:

$$p_n = \frac{\sum_{ngram \in c} min\Big(Max\_Ref\_Count(ngram), Count_c(ngram)\Big)}{\sum_{ngram \in c} Count_c(ngram)}$$
$$Max\_Ref\_Count(ngram) = max_{i=1..k} Count_{r_i}(ngram)$$

   For each of the n-grams in $\mathbf{c}$, we count the maximum number of times it appears in any one reference translation. Then, we clip this count by the number of times it appears in $\mathbf{c}$. We divide these clipped counts by the number of ngrams in $\mathbf{c}$.

   Next, we compute the brevity penalty BP. Let $len(\mathbf{c})$ be the length of c and let $len(\mathbf{r})$ be the length of the shortest reference translation.

$$BP = \begin{cases} 1 & \text{if } len(\mathbf{c}) \geq len(\mathbf{r}) \\ exp(1 - \frac{len(\mathbf{r})}{len(\mathbf{c})}) & \text{otherwise} \end{cases}$$

   Lastly, the *BLEU* score for candidate c with respect to $r_1, ..., r_k$ is:

$$BLEU = BP \ \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1}$$

where w_1, w_2, w_3, w_4 are weights that sum to 1. The log here is natural log.

Compute the $BLEU$ score for the following two candidate translations $\mathbf{c_1}, \mathbf{c_2}$ against two human-written references $\mathbf{r_1}, \mathbf{r_2}$:

Candidate Translation $\mathbf{c_1}$: fruits are good for health

Candidate Translation $\mathbf{c_2}$ vegetables are very important for good health

Reference Translation $\mathbf{r_1}$: eating fruits is good for health

Reference Translation $\mathbf{r_2}$: fruits and vegetables are essential for good health

Let $w_i = \frac{1}{3}$ for $i \in 1, 2, 3$ and $w_4 = 0$. In other words, we do not compute 4-grams. Show the computation of p_1, p_2, p_3, BP, and the final $BLEU$ score in your answer.

5 HMM

Consider the problem of POS Tagging using Hidden Markov Models. Assume that our vocabulary only contains the words "Paul's", "red", "pen", and "leaked". There are only 3 states corresponding to the tags Proper Noun(PN), Noun(N), and Not Noun(NN). The parameters of the HMM model are as follows:

	PN	N	NN
π	0.1	0.3	0.6

Table 7: Initial probabilities

from/to	PN	N	NN
PN	0	0.2	0.8
N	0.1	0	0.9
NN	0.2	0.4	0.4

Table 8: Transition probabilities

	Paul's	red	pen	leaked
PN	0.7	0.2	0.1	0
N	0.1	0.2	0.6	0.1
NN	0	0.4	0.1	0.5

Table 9: Emission probabilities for the states PN, N, and V

1. Compute the probability of the observed sequence "Paul's red pen leaked" by using the Forward Algorithm on the given HMM model.

2. Use the viterbi algorithm on the given HMM model to determine the best sequence of tags for the sentence "Paul's red pen leaked" and the probability of that sequence of tags. If there are ties, break them in this order NN < N < PN.

3. What is the time complexity of the Viterbi Algorithm in terms of the number of possible labels (Y) and the sequence length (T)? Justify your answer.

6 Transformer Self Attention

This section deals with transformer self attention. It may be helpful to read section 9.7 in the Speech and Language Processing textbook while answering these questions. **For all questions in this section, unless otherwise stated work must be shown in the form of matrix multiplications to receive full credit** (i.e. $C = AB^T$). For performing the computations, using Excel or other software is recommended to avoid computation errors. When writing your answers please round to 2 decimal places. You may use scientific notation to represent your answers if necessary.

Word			
Attention	1	2	4
is	-1	0	2
all	3	1	3
you	5	0	0
need	2	-2	-1

Table 10: Word embeddings

$$W^q = \begin{bmatrix} 1 & 1 \\ -3 & 1 \\ -2 & 3 \end{bmatrix}, \; W^k = \begin{bmatrix} -1 & 3 \\ -2 & -5 \\ -1 & -2 \end{bmatrix}, \; W^v = \begin{bmatrix} 3 & 0 \\ 2 & -4 \\ 4 & 0 \end{bmatrix}$$

$$W^o = \begin{bmatrix} -5 & 4 & -5 \\ 2 & -1 & 2 \\ 1 & -4 & 4 \\ 0 & 0 & 0 \\ 3 & 3 & -1 \\ -4 & -3 & -1 \\ 3 & 1 & 5 \\ 1 & 3 & 3 \end{bmatrix}$$

1. We will first consider a single attention head. Given the set of word embeddings, projection matrices, and a normalization factor of 48 instead of $\sqrt{d^k}$, fill out this table with the normalized query-key score for each possible pair of words.

Word					
Attention					
is					
all					
you					
need					

Table 11: Normalized Scores

2. Given the normalized scores, calculate the attention weights for each word with respect to the other words in the input sentence and fill in the table with your results. You do not need to show work for this question. Please do not mask any attention values.

Word					
Attention					
is					
all					
you					
need					

Table 12: Attention Values

3. Given the embeddings, the previously calculated attention values, and the value projection matrix, calculate the output embeddings of this attention head. Fill in the table with your results.

Word		
Attention		
is		
all		
you		
need		

Table 13: Self Attention Head Outputs

4. The outputs of three other self attention heads have been computed for you. Combine these values with the self attention embedding you calculated earlier in this question, and find the final output of this self attention layer using the output weight matrix. Fill in the table with your results.

Word		
Attention	1.5	-2.5
is	-4.78	0.15
all	1.75	-1.97
you	-3.96	-2.9
need	-0.53	4.61

Word		
Attention	1.51	0.07
is	-4.95	-3.47
all	2.33	-4.81
you	0.05	0.68
need	2.85	-1.91

Word		
Attention	-3.59	-3.18
is	3.38	-1.85
all	3.77	4.21
you	-0.15	1.46
need	-1.65	1.51

Table 14: Other Attention Head Outputs

Word			
Attention			
is			
all			
you			
need			

Table 15: Output of self attention layer

5. Given your understanding of the differences between recurrent networks and transformers, why do transformers tend to outperform recurrent networks in terms of computational efficiency? (1-2 sentences)

7 Extra Credit

For this open-ended question, we will ask you to help find some data for a research project.

1. Users on the Internet sometime may willingly or accidentally reveal information about themselves. Please find at least five posts on Reddit (https://www.reddit.com/) where users revealed information about themselves (**other than age or gender**). List the URL of the Reddit posts and the type(s) of information that was disclosed.

2. How did you find these posts? Explain the strategy you used.

3. Any thoughts or interesting observations you have as you search for these posts?