# CS 7650: Natural Language Processing (Spring 2024)
# Problem Set 2

Instructor: Wei Xu
TAs: Andrew Li, Anton Lavrouk, Marcus Ma
Piazza: https://piazza.com/gatech/spring2024/cs7650a
Gradescope: https://www.gradescope.com/courses/697909

Due: Monday, March 11, 11:59 PM ET

## 1 Understanding Word2Vec

Given a sequence of words $w_1, \ldots, w_T$ and context size $c$, the training objective of skip-gram is:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log P\left(w_{t+j} \mid w_t\right)$$

where $P\left(w_o \mid w_t\right)$ is defined as:

$$P\left(w_o \mid w_t\right) = \frac{\exp\left(\mathbf{u}_{w_t}^{\top}\mathbf{v}_{w_o}\right)}{\sum_{k\in V}\exp\left(\mathbf{u}_{w_t}^{\top}\mathbf{v}_k\right)}$$

where $\mathbf{u}_k$ represents the "target" vector and $\mathbf{v}_k$ represents the"context" vector, for every $k \in V$.

(a) (**3 pts**) Derive the following gradient (probability w.r.t context vector):

$$-\frac{\partial \log P\left(w_o \mid w_t\right)}{\partial \mathbf{v}_{w_o}}$$

(b) (**2 pts**) Imagine that we train the model on a large corpus (e.g. English Wikipedia). Describe the effects of context size $c$ to the resulting word vectors $\mathbf{u}_w$, i.e. what if we use context size $c = 1, 5$, or 100?

## 2 Hidden Markov Models and the Viterbi Algorithm

We have a toy language with 2 words - "cool" and "shade". We want to tag the parts of speech in a test corpus in this toy language. There are only 2 parts of speech — NN (noun) and VB (verb) in this language. We have a corpus of text in which we the following distribution of the 2 words:

| | NN | VB |
|---|---|---|
| cool | 15 | 5 |
| shade | 6 | 9 |

Assume that we have an HMM model with the following transition probabilities (* is a special start of the sentence symbol).

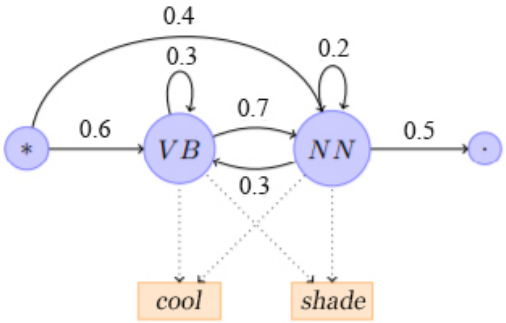


Figure 1: HMM model for POS tagging in our toy language.

(a) (**2 pts**) Compute the emission probabilities for each word given each POS tag.

(b) (**3 pts**) Draw the Viterbi trellis for the sequence "cool shade." (Do include the period!). Highlight the most likely sequence. Here is an example of Viterbi trellis.

# 3 LSTMs

The update equations for a LSTM at timestep $i$ are given in the following equations. Eisenstein Chapter 6.3 may be useful in answering this question.

$$\mathbf{f_i} = \sigma(\mathbf{W}^{(f)}\mathbf{h_{i-1}} + \mathbf{U}^{(f)}\mathbf{x_i} + \mathbf{B}^{(f)}) \quad \text{Forget gate}$$

$$\mathbf{i_i} = \sigma(\mathbf{W}^{(i)}\mathbf{h_{i-1}} + \mathbf{U}^{(i)}\mathbf{x_i} + \mathbf{B}^{(i)}) \quad \text{Input gate}$$

$$\mathbf{g_i} = tanh(\mathbf{W}^{(g)}\mathbf{h_{i-1}} + \mathbf{U}^{(g)}\mathbf{x_i}) \quad \text{Update candidate}$$

$$\mathbf{c_i} = \mathbf{f_i} \odot \mathbf{c}_{i-1} + \mathbf{i_i} \odot \mathbf{g_i} \quad \text{Memory cell update}$$

$$\mathbf{o_i} = \sigma(\mathbf{W}^{(o)}\mathbf{h_{i-1}} + \mathbf{U}^{(o)}\mathbf{x_i} + \mathbf{B}^{(o)}) \quad \text{Output gate}$$

$$\mathbf{h_i} = \mathbf{o_i} \cdot \tanh(\mathbf{c_i}) \quad \text{Output}$$

| **Weight** | **Value** |
|---|---|
| $\mathbf{W}^{(f)}$ | $[1, -2, -3]$ |
| $\mathbf{U}^{(f)}$ | $[0, -1, -2]$ |
| $\mathbf{B}^{(f)}$ | $0$ |
| $\mathbf{W}^{(i)}$ | $[0, 0, 1]$ |
| $\mathbf{U}^{(i)}$ | $[-1, -2, -2]$ |
| $\mathbf{B}^{(i)}$ | $1$ |
| $\mathbf{W}^{(g)}$ | $\begin{bmatrix} 0 & 1 & -3 \\ -3 & 1 & 0 \\ -2 & -1 & -3 \end{bmatrix}$ |
| $\mathbf{U}^{(g)}$ | $\begin{bmatrix} 1 & 0 & 0 \\ -2 & -3 & 0 \\ 1 & -1 & -2 \end{bmatrix}$ |
| $\mathbf{W}^{(o)}$ | $[1, 0, 1]$ |
| $\mathbf{U}^{(o)}$ | $[-1, 0, 1]$ |
| $\mathbf{B}^{(o)}$ | $-1$ |

Table 1: Weights for LSTM.

(a) (**1 pt**) In Table 1 we provide weight values and in Table 2 timestep inputs. Now we'll compute the value of $\boldsymbol{h}_i$ using Table 1 and Table 2:

| Vector | Value |
|---|---|
| $\boldsymbol{h}_{i-1}$ | $[3, 1, -1]^T$ |
| $\boldsymbol{c}_{i-1}$ | $[1, 0, -4]^T$ |
| $\boldsymbol{x}_i$ | $[-3, -2, -1]^T$ |

Table 2: Input/intermediate variables for LSTM.

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

**Note that this is just a toy example of an LSTM layer where the gate values are scalars. In a real LSTM such as the one in PyTorch, the gate values will be vectors with some hidden layer dimensionality.**

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 $\boldsymbol{h}_i$, describe the values that you would need to set the gates $\boldsymbol{f}_i, \boldsymbol{i}_i$ and $\boldsymbol{o}_i$ equal to. Only the values for these gates are necessary, do not change the equations for the update.

(b) (**1 pt**) Which variable from the list of intermediate variables in the given equations most closely resembles the hidden state of a standard Elman RNN? (Answer choices are $\boldsymbol{f}_i$, $\boldsymbol{i}_i$, $\boldsymbol{g}_i$, $\boldsymbol{c}_i$, $\boldsymbol{o}_i$, $\boldsymbol{h}_i$).

(c) (**2 pts**) In this problem, all the LSTM gates are scalars. What changes would have to be made to Table 1 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?

(d) (**2 pts**) What two problems in RNNs does the inclusion of the memory cell $\boldsymbol{c}_i$ improve? What property of its computation allows it to do this?

# 4 Beam Search Decoding

Consider the following bigram language model. The bigram probabilities are given in table 3. 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.

| Bigram | Prob. | Bigram | Prob. |
|---|---|---|---|
| P(, \| know) | 0.3 | P(important \| the) | 0.2 |
| P(the \| know) | 0.7 | P(response \| correct) | 0.4 |
| P(I \| ,) | 0.6 | P(answer \| correct) | 0.35 |
| P(it \| ,) | 0.4 | P(problems \| correct) | 0.25 |
| P(will \| I) | 0.2 | P(response \| important) | 0.1 |
| P(know \| I) | 0.8 | P(answer \| important) | 0.9 |
| P(was \| it) | 1.0 | P(answer \| exact) | 1.0 |
| P(correct \| the) | 0.3 | P(exact \| the) | 0.5 |

Table 3: Bigram Language Model probabilities for Beam Search.

1. (**3 pts**) 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.

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.

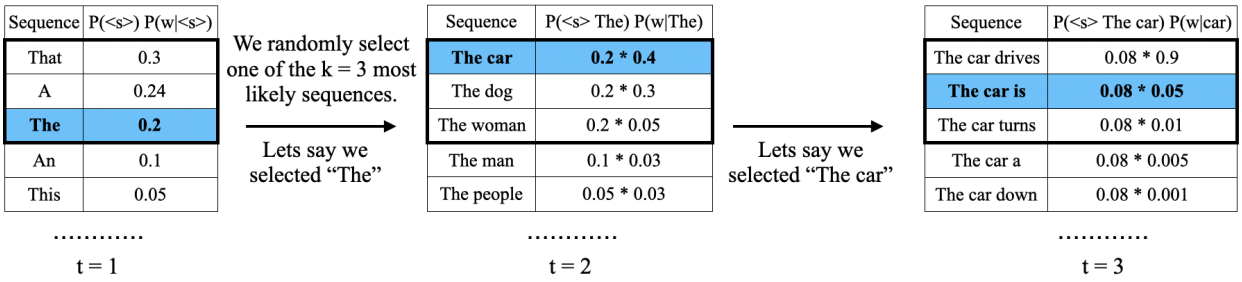


Figure 2: Example of top-k random sampling. At each timestep t, the model generates the probability of the possible next word. Then, we randomly sample from the $k$ most likely candidates from this distribution. Here, we consider $k = 3$. **Bold** sequences represent the sampled candidate at each timestep.

Once again, consider the bigram language model in Table 3.

(a) (**1 pt**) 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) (**1 pt**) What is the sequence with maximum probability in $S$? Report the sequence and the probability of the sequence.

(c) (**1 pt**) What is the sequence with minimum probability in $S$? Report the sequence and the probability of the sequence.

3. Let's say we are decoding a sentence of length $T$ using a language model with vocabulary size $V$. Report the time complexity in the big $O$ notation when the sentence is decoded using the following strategies:

   (a) (**1 pt**) Greedy search, where we store only the topmost sequence at each time step.

   (b) (**1 pt**) Beam search of width $k$, where we store the top-$k$ sequences at each time step:

   (c) (**1 pt**) Exhaustive search, where we store all the sequences at each time step: