CS 4650/7650 Fall 2020: Homework 3

September 17, 2020

Instructions

1. This homework has two parts: questions 1–4 are theory questions, and Q5 is a programming assignment with some written components within a Jupyter Notebook.

   We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under HW3 Writing), make sure to mark which page(s) correspond to each problem or subproblem. For instance, Q2 has three subproblems, so the solution to each must start on a new page.

   (b) For the coding problem (Q5), please attach a pdf export of 'word_embedding.ipynb', including outputs, to your writeup.

   (c) Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.

2. LaTeX solutions are strongly encouraged (a solution template is available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.

3. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on the submission site.

Questions

1. Consider the term-document matrix for four words in three documents shown in Table 1. The whole document set has $N = 20$ documents, and for each of the four words, the document frequency df_t is shown in Table 2.

term-document	Doc1	Doc2	Doc3
car	28	5	25
insurance	4	19	0
auto	0	34	30
best	15	0	18

Table 1: Term-document Matrix

	df
car	13
insurance	7
auto	11
best	17

Table 2: Document Frequency

(a) Compute the *tf-idf* weights according to the definitions in Jurafsky and Martin's Textbook (equations 6.12-14) for each of the words *car*, *auto*, *insurance* and *best* in Doc1, Doc2, and Doc3. [5 pts]

(b) Use the *tf-idf* weight you got in (a) to represent each document with a vector, and calculate the cosine similarities between these three documents. [3 pts]

(c) **[CS4650 only]** Suppose we train the word vectors of four words "stocks", "currency", "can", "will" using *word2vec* and with *tf-idf*. Which one of the following two word pairs, ("stocks", "currency") and ("can", "will"), may have a high *tf-idf* cosine similarity but a low *word2vec* cosine similarity? And which one may have a low *tf-idf* cosine similarity but a high *word2vec* cosine similarity? Explain why. [2 pts]

(d) **[CS7650 only]** *Stemming* refers to reducing a word to its root word by removing its suffix, e.g. "learned" to "learn". Which one of the methods, *word2vec* and *tf-idf*, do you think would benefit from *stemming*? Explain why. [2 pts]

2. The distributional hypothesis — popularly stated as "a word is characterized by the company it keeps" — suggests that words that occur in similar contexts should be similar in meaning. The distributional hypothesis forms the basis for the Skipgram model of Mikolov et. al., which is an efficient way of learning the meaning of words as dense vector representations from unstructured text. The skipgram objective is to learn the probability distribution $P(C|T)$ where given a target word w_t, we estimate the probability that a context word w_c lies in the context window of w_t. The distribution of the probabilistic model is parameterized as follows:

$$P(C = w_c|T = w_t) = \frac{\exp(\mathbf{u}_{w_c}^\top \cdot \mathbf{v}_{w_t})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{u}_{w'}^\top \cdot \mathbf{v}_{w_t})}, \tag{1}$$

where vectors $\mathbf{u}_{w_c}$ and $\mathbf{v}_{w_t}$ represent the context word w_c and the target word w_t respectively. Notice the use of softmax function and how the problem of learning embeddings in this model has been cast as a classification problem. The vectors for all the words in the vocabulary $\mathcal{V}$ can be succinctly represented in two matrices $\mathbf{U}$ and $\mathbf{V}$, where the vector in the j^{th} column in $\mathbf{U}$ and $\mathbf{V}$ corresponds to the context and target vectors for the j^{th} word in $\mathcal{V}$. Note that $\mathbf{U}$ and $\mathbf{V}$ are the parameters of the model. Answer the following questions about the Skipgram model.

   (a) The cross entropy loss between two probability distributions p and q, is expressed as:

$$\text{CROSS-ENTROPY}(p, q) = -\sum_m p_m \log(q_m). \tag{2}$$

   For a given target word w_t, we can consider the ground truth distribution $\mathbf{y}$ to be a one-hot vector of size $|\mathcal{V}|$ with a 1 only at the true context word w_c's entry, and 0 everywhere else. The predicted distribution $\hat{\mathbf{y}}$ (same length as $\mathbf{y}$) is the probability distribution $P(C|T = w_t)$. The j^{th} entry in these vectors is the probability of the j^{th} word in $\mathcal{V}$ being a context word. Write a simplified expression of cross entropy loss, $\text{CROSS-ENTROPY}(\mathbf{y}, \hat{\mathbf{y}})$, for the Skipgram model on a single pair of words w_c and w_t. Your answer should be in terms of $P(C = w_c|T = w_t)$). [2 pts]

   (b) Find the gradient of the cross entropy loss calculated in (a) with respect to the target word vector $\mathbf{v}_{w_t}$. Your answer should be in terms of $\mathbf{y}$, $\hat{\mathbf{y}}$ and $\mathbf{U}$). [4 pts]

   (c) Find the gradient of the cross entropy loss calculated in (a) with respect to each of the context word vectors $\mathbf{u}_{w_c}$. Do this for both cases $C = w_c$ (true context word) and $C \neq w_c$ (all other words). Your answer should be in terms of $\mathbf{y}$, $\hat{\mathbf{y}}$ and $\mathbf{v}_{w_t}$. [4 pts]

3. Sobchak Industries, a dog-grooming supplies production company, is planning to train a word embedding model which will help power its support chatbot. To this end, it intends to run a **skipgram with negative sampling** model over a corpus of ten million words containing 100,000 unique word types. Larry, the chief designer of the system, set the hyperparameters at 2 training epochs, 300 hidden dimensions, and a window size of 5.

   Donny, a junior research engineer, is worried about the **out-of-vocabulary problem**. He tells Larry that if they add a **subword** component to their embeddings, the model will be better able to approximate vectors for new words it encounters during the chat sessions.

   (a) Donny's suggestion is the following: each non-space character in the corpus (there are 52 letters and 10 other characters) gets its own target embedding as a 'word part' (so, e.g. the character *a* has a separate embedding from the word *a*). During training, each pass over a target word w_t is augmented by an identical and independent pass over each of the characters it contains, predicting the same context words as for w_t (so no context characters are considered). During downstream inference (chatbot), a new word's embedding is initialized as the average of its characters' 'word part' embeddings.

   The average character length of a word in the corpus is 5.5. Answer the following: what is the percentage increase of training Donny's suggested model in terms of **space**? In terms of **time**? If we reduce the hidden dimensions parameter to 200 and only train one epoch, and want to change the **window size** so that the time costs are closest to the originally planned model, what window size should we choose? [2+2+3 pts]

   Note: You may ignore window effects of document boundaries; assume these are negligible.

   (b) Maude, another engineer, points out that language isn't built simply on characters which carry meaning. She proposes an *affix-based* method: collect the 1,000 most common prefixes (of any length) in the vocabulary, and the 1,000 most common suffixes in the vocabulary, and during the pass over the corpus, any word ending in one of the suffixes and/or starting with one of the prefixes is *averaged* with the affix embedding(s) and the cross-entropy loss is used to update all components of the embedding. This time, the change is applied to both target and context embeddings. List **all** statistics which would help calculate the added space and time complexities of this variant. Assume each word can only be associated with one prefix, one suffix, or one of each. Use unambiguous terminology (or, better yet, mathematical notation / pseudo-code functions, with clear variable definitions). [5 pts]

   (c) Why is an exact calculation of the added complexities in Maude's formulation impossible, even given the entire training corpus? Can changing the training regime to hierarchical softmax or CBOW remove this uncertainty? [3 pts]

4. (a) ELMo (Peters et al., 2018) is a method of getting **contextualized embeddings** for downstream tasks, using the concatenation of independently-trained left-to-right and right-to-left LSTMs.

   i. Underneath the LSTMs, words are represented through a character-level CNN, which means ELMo is capable of handling the out-of-vocabulary problem well. Explain why. [1 pt]

   ii. Even if ELMo used a closed word vocabulary instead of the character-level CNN, inference would still be possible given some out-of-vocabulary words. Explain why. [1 pt]

   (b) OpenAI GPT (Radford et al., 2019) is another method for training contextualized embeddings, using a left-to-right Transformer. The Transformer architecture requires **positional embedding** in order to represent the order of tokens in the sentence. One way of getting positional embeddings is:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

   where pos is the position, i is the dimension, and d is the model dimension size. In this way, the model can easily learn to attend by relative positions, since for any fixed offset k, $PE_{pos}(j+k)$ can be represented as a linear function of $PE_{pos}(j)$.
   Assume the case where $d = 2$:

$$PE(x) = [\sin x, \cos x]^\top$$

   then, we have:

$$PE(x+k) = \boldsymbol{A} PE(x)$$

   where k is a fixed offset (constant) and $\boldsymbol{A}$ is a matrix. Calculate what $\boldsymbol{A}$ should be, in terms of k. [5 pts]

   (c) One major difference between BERT (Devlin et al., 2019) and GPT is that they use different objectives. GPT uses an autoregressive language modeling as objective, while BERT uses masked language modeling as its pretraining objective. Explain what these two objectives are, and list advantages that masked language modeling has over autoregressive language modeling in natural language understanding tasks. [3 pts]

5. In this assignment, we will be exploring word embeddings, specifically looking at word2vec, a popular technique to generate word vectors. You will not be training your own word embeddings, instead you would be using pre-trained word embeddings from GenSim. This programming assingment is designed to provide you a better understanding of the vector space the word embeddings lie in. Specifically, you will be looking at similarities, semantics, analogies, biases and visualization. Download and complete the notebook, following the instructions provided therein-

   'word_embedding.ipynb'.

   Export 'word_embedding.ipynb' with output to a pdf and attach to your writeup. Your writeup should be uploaded to HW3 Writing.