Assignment 4

CSE 447 and 517: Natural Language Processing - University of Washington

Winter 2022

Please consult the course website for current information on the due date, the late policy, and any data you need to download for this assignment. This assignment is designed to advance your understanding of word embeddings and NLP models that make use of them. ★ **problems are for CSE 517 students only.** Other problems should be completed by everyone.

Submit: You will submit your writeup (a pdf) via Gradescope. Instructions can be found here.

1 Vector Embeddings – Eisenstein 14.6–9 (p. 332)

1. Download a set of pretrained English word embeddings (e.g., the GloVe embeddings available at `https://nlp.stanford.edu/projects/glove/`). In your writeup, tell us what embeddings you're working with. Use cosine similarity to find the most similar word to each of these words. Report the most similar word and its cosine similarity.
   - *dog*
   - *whale*
   - *before*
   - *however*
   - *fabricate*
2. Use vector addition and subtraction to compute target vectors for the analogies below. (This style of evaluation is explained in section 14.6 of the textbook. The textbook is, however, incorrect in the in-line formula that explains how to compute the target vector on page 322. In the textbook's notation, you want the word embedding most similar to $(-\mathbf{v}_{i_1} + \mathbf{v}_{j_1} + \mathbf{v}_{i_2})$. Note the sign changes. The same mistake appears in the 2021 video recording but has been corrected in the pdf version of the lecture slides.) After computing each target vector, find the top three candidates by cosine similarity. Report the candidates and their similarities to the target vector.
   - *dog* : *puppy* :: *cat* : ?
   - *speak* : *speaker* :: *sing* : ?
   - *France* : *French* :: *England* : ?
   - *France* : *wine* :: *England* : ?
3. Select a text classification dataset that is available to download for research (e.g., the Pang and Lee movie review data, currently available from `http://www.cs.cornell.edu/people/pabo/movie-review-data/`). If there is an official test dataset, set it aside; if not, use a random subset as the test set (at least 300 examples) and do not use it until part 4. Use the official development

set as development data, if it exists; if not, use a random subset and make sure your training set and development set do not overlap. Your writeup should explain what dataset you are using. Train a classifier on your training set, conforming to these requirements:

   (a) Use the pretrained word embeddings from part 1 as inputs; *do not* update them during training (they are "frozen").
   (b) For every word type in the training data that has no pretrained embedding, introduce a new word embedding, which is randomly initialized and updated during training. When you apply the model to development or test data, words that have no embedding (neither pretrained, nor added during training) should be ignored.
   (c) Train until accuracy on the development set stops improving.
   (d) You may use the development set to tune the model architecture (e.g., choosing a depth) and hyperparameters.
   (e) Which kind of network you use is up to you. You should briefly describe your network in the writeup.

4. Report accuracy, macro-averaged F_1 score (i.e., first, for each class, calculate F_1 with that class as target; then average those per-class F_1 scores), and training time. (You may also optionally report the training time summed across all the models you considered during architecture/hyperparameter tuning; note that doing this requires more planning.)

5. Repeat the above experiment—keeping everything the same (including tuning), except for one key change: update ("finetune") all word embeddings during training. Report the same measurements as in part 4.

2 Inequalities, Part 1 – Eisenstein 14.4 (pp. 331–2)

Hint: problems 2 and 3 are related to the "XOR" problem (2) from assignment 1.

A simple way to compute a vector representation of a sequence of words is to add up the vector representations of the words in the sequence. Consider a sentiment analysis model in which the predicted sentiment is given by:

$$\text{score}(w_1, \ldots, w_m) = \boldsymbol{\theta} \cdot \sum_{i=1}^{m} \mathbf{x}_{w_i} \tag{1}$$

where w_i is the ith word and $\mathbf{x}_{w_i}$ is the embedding for the ith word; the input is of length m (in word tokens). $\boldsymbol{\theta}$ are parameters. Prove that, in such a model, the following two inequalities cannot both hold:

$$\text{score}(\text{good}) > \text{score}(\text{not good}) \tag{2}$$
$$\text{score}(\text{bad}) < \text{score}(\text{not bad}) \tag{3}$$

Next, consider a slightly different model:

$$\text{score}(w_1, \ldots, w_m) = \frac{1}{m}\left(\boldsymbol{\theta} \cdot \sum_{i=1}^{m} \mathbf{x}_{w_i}\right) \tag{4}$$

Construct an example of a pair of inequalities similar to (2–3) that cannot both hold.

3 ★ Inequalities, Part 2 – Eisenstein 14.5 (p. 332)

Hint: problems 2 and 3 are related to the "XOR" problem (2) from assignment 1.

Continuing from problem 2, consider this model:

$$\text{score}(w_1, \ldots, w_m) = \boldsymbol{\theta} \cdot \text{ReLU}\left(\sum_{i=1}^{m} \mathbf{x}_{w_i}\right) \tag{5}$$

Show that, in this case, it *is* possible to achieve the inequalities (2–3). The recommended way to do this is to provide weights $\boldsymbol{\theta}$ and embeddings for the words *good*, *bad*, and *not*. The problem can be solved with four dimensions.

4 Extra Credit: Skip-Gram – Eisenstein 14.2 (p. 331)

In skipgram word embeddings, the negative sampling objective can be written as:

$$\mathcal{L} = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{C}} \text{count}(i, j) \cdot \psi(i, j) \tag{6}$$

$$\psi(i, J) = \log \sigma(\mathbf{u}_i, \mathbf{v}_j) + \sum_{i' \in \mathcal{W}_{neg}} \log(1 - \sigma(\mathbf{u}_{i'} \cdot \mathbf{v}_j)) \tag{7}$$

The notation in equation 7 is fully explained in the "Negative Sampling" discussion on pp. 320–1.

Suppose we draw the negative samples from the empirical unigram distribution $\hat{p}(i) = p_{unigram}(i)$. First, compute the expectation of $\mathcal{L}$ with respect to the negative samples, using this probability.

Next, take the derivative of this expectation with respect to the score of a single word/context pair $\sigma(\mathbf{u}_i \cdot \mathbf{v}_j)$, and solve for the pointwise mutual information of i and j. You should be able to show that, at the optimum, the PMI is a simple function of $\sigma(\mathbf{u}_i, \mathbf{v}_j)$ and the number of negative samples.

(This exercise is part of a proof that shows that skipgram with negative sampling is closely related to PMI-weighted matrix factorization.)