CS291A Deep Learning NLP Assignment 1: Word Embeddings

Due: 2/13, 23:59pm PT via CodaLab
Instructor: William Wang

1 Policy on Collaboration among Students

We follow UCSB's academic integrity policy from UCSB Campus Regulations, Chapter VII: "Student Conduct and Discipline"):

"It is expected that students attending the University of California understand and subscribe to the ideal of academic integrity, and are willing to bear individual responsibility for their work. Any work (written or otherwise) submitted to fulfill an academic requirement must represent a student's original work. Any act of academic dishonesty, such as cheating or plagiarism, will subject a person to University disciplinary action. Using or attempting to use materials, information, study aids, or commercial "research" services not authorized by the instructor of the course constitutes cheating. Representing the words, ideas, or concepts of another person without appropriate attribution is plagiarism. Whenever another person's written work is utilized, whether it be a single phrase or longer, quotation marks must be used and sources cited. Paraphrasing another's work, i.e., borrowing the ideas or concepts and putting them into one's "own" words, must also be acknowledged. Although a person's state of mind and intention will be considered in determining the University response to an act of academic dishonesty, this in no way lessens the responsibility of the student."

More specifically, we follow Stefano Tessaro and William Cohen's policy in this class:

- You cannot copy the code or answers to homework questions or exams from your classmates or from other sources;
- You may discuss course materials and assignments with your classmate, but you cannot write anything down.
- You must write down the answers / code independently.
- The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment.

Specifically, each assignment solution must start by answering the following questions:

1. Did you receive any help whatsoever from anyone in solving this assignment? Yes / No.
   If you answered 'yes', give full details: _____
   (e.g. "Jane explained to me what is asked in Question 3.4")
2. Did you give any help whatsoever to anyone in solving this assignment? Yes / No.
   If you answered 'yes', give full details: _____
   (e.g. "I pointed Joe to section 2.3 to help him with Question 2".)

Academic dishonesty will be reported to the highest line of command at UCSB. When you are not sure, ask the teaching staff before you do so. Students who engage in plagiarism activities will receive an F grade automatically.

2 Programming Assignment (100%)

In this homework assignment, you will need to implement the Word2Vec (Skipgram) model and the Glove model in Python. Note that you need to implement the stochastic gradient descent training by yourself, rather than calling third party machine learning toolkits such as GenSim, TensorFlow, Theano, MXnet, Caffe, (Py)Torch etc.

2.1 Word2Vec (Skipgram) (50%)

In the first part of the assignment, you will need to implement the Skipgram model from Word2Vec. Recall that the Skipgram model that we covered in class: the main idea is to use the target word to predict the context word. More specifically, we use a one-hot input vector and a input embedding matrix to find the target hidden word embedding. Then, we use a softmax function to predict a probabilistic distribution over the context output words.

Given the input text such as: *"the quick brown fox jumped over the lazy dog"*. You will need to generate the training instances in the form of *(quick, the), (quick, brown), (brown, quick), (brown, fox), ...* Note that in addition to the positive training examples, you will also need to sample words to form negative examples, for example *(he, the)*.

You will need to implement a Stochastic Gradient Descent algorithm (https://en.wikipedia.org/wiki/Stochastic_gradient_descent) to optimize the word embeddings for this problem. Remember that we have already derived the gradient updates for you in class: all you need to do, is to randomly initialize the target word embeddings, perform multiple passes over the dataset, set up a convergence criteria (e.g., the delta in negative log-likelihood), and select appropriate adaptive learning rates.

For the full description of the Skipgram model, please refer to Mikolov et al., 2013[1] and our lecture slides on the Piazza resource page.

2.2 Glove (50%)

The GloVe model combines the idea of global statistics from word co-occurrences model and the prediction based word embedding learning idea: the model learns word vectors by examining word co-occurrences within a text corpus. So the first thing we will need to do, is to construct a co-occurrence matrix X, where a cell X_{ij} is a frequency count which shows how often the word i appears in the context of the word j. We can scan the corpus once to construct the co-occurrence matrix X, and from then on use this co-occurrence data in place of the actual corpus. We will construct our model based only on the values collected in X. In your implementation, you should consider choosing an efficient representation for the co-occurrence matrix.

After the matrix X is constructed, now our task is to obtain the values in continuous space for each in-vocabulary word we observe in the corpus. Note that the Glove model imposes a soft constraint for each word pair of word i and word j. Then, we can set up an objective function to solve a weighted least square regression problem.

Please refer to the original Glove paper[2] for the details.

2.3 Data Set, Input, and Output

1. For this assignment, you should download the training data from http://mattmahoney.net/dc/text8.zip. Here is a description of the text8 dataset: http://mattmahoney.net/dc/textdata

2. The input to your program should be the above zipped text file, and this vocabulary file: https://drive.google.com/file/d/0B76iNC1BIl5GdFdFZjVrdXNlYm8/view?usp=sharing

3. The output should be named as "vectors.txt", which **only includes the word embeddings for the words within the given vocabulary**, in the following space-separated format:

   apple 0.234 -0.213 0.314 ... 0.524
   banana 0.233 -0.203 0.982 ... 0.928

[1] https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[2] https://nlp.stanford.edu/pubs/glove.pdf

Note that you are not allowed to find additional training and testing datasets for this assignment.

2.4 CodaLab Environment

For this assignment, we publish a CodaLab competition (https://competitions.codalab.org/competitions/18107?secret_key=7aaa88ae-d29a-427a-866e-02ce295a68f9), where you can submit your code, your generated word vectors, and get evaluation scores that will be listed on a leader-board. If you are not familiar with CodaLab Competitions, check out here: participating in a competition.

Remember to register a CodaLab Competitions account using your umail account, so that the username will be your UCSBNetID. After that, log into CodaLab Compectitions and set up your team name (whatever nickname you like). To protect your privacy, only the team names will be shown on the leader-board and your usernames will be anonymous. After your submission finishes running, please choose to submit it to the leader-board. Note that here team name is equivalent to your nickname, and it is still an independent homework assignment. You must implement the two algorithms according to the instructions above.

If you are using your late days, please submit at a separate CodaLab late submission URL (https://competitions.codalab.org/competitions/18109?secret_key=8093a3c0-c163-4a4a-ba64-b97fc1dc6e2d).

2.5 Implementation & Submission

You must avoid calling APIs from existing machine learning toolkits such as Gensim, TensorFlow, Theano, MXnet, Caffe, (Py)Torch, and scikit-learn. Using numpy is allowed. There are two phases, *Word2Vec* and *Glove*, representing the two parts of the assignment. You must submit one zip file per phase via the corresponding submission page. Your code must be written in Python.

Submission format The final submission format should be:
word2vec.zip

- run.py
- vectors.txt
- other python scripts you wrote

glove.zip

- run.py
- vectors.txt
- other python scripts you wrote

Note that to create a valid submission, zip all the file with 'zip -r zipfilename *' starting from this directory.

- **DO NOT zip the directory itself, just its contents.**
- **Also, DO NOT submit the corpus: only submit your learned vectors.txt and your code.**
- **DO NOT submit the word vectors for all words from text8 in vectors.txt: we only need the embeddings for words from the input vocabulary file.**

Here is a sample 'run.py' file: https://www.cs.ucsb.edu/~william/courses/s17_292f/run.py. You must submit the 'run.py' file with the exactly same format. For each phase, you have a total of 30 possible submissions.

2.6 Evaluation Criteria

To evaluate the quality of your word embeddings, we will select a set of your learned word embeddings to measure their pairwise cosine similarities, and correlate the word similarity from your embeddings with human judgments. The correlation metric is Spearman's Rho (https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient). On the scoreboard, "Found" and "Not-Found" means how many in-vocabulary words and how many out-of-vocabulary words we have counted when evaluating your program. These are only for your debugging purposes, and they are not a part of the evaluation metric. Note that even though your submitted word vectors and your code are automatically evaluated on CodaLab Competitions, we will run a Plagiarism detection program (Stanford Moss) on them.