Assignment 3: Word Vectorization

Course: Introduction to Natural Language Processing

Deadline: 25 March | 23:59

General Instructions

1. The assignment must be implemented in Python.
2. The assignment must be done using PyTorch. Usage of other frameworks will not be accepted.
3. Submitted assignment must be your original work. Please do not copy any part from any source including your friends, seniors, and/or the internet. If any such attempt is caught, then serious actions including an F grade in the course is possible.
4. A single .zip file needs to be uploaded to the Moodle Course Portal.
5. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
6. Please start early since no extension to the announced deadline would be possible.

1 About

Many NLP systems employ modern distributional semantic algorithms, known as word embedding algorithms, to generate meaningful numerical representations for words. These algorithms aim to create embeddings where words with similar meanings are represented closely in a mathematical space. Word

embeddings fall into two main categories: frequency-based and prediction-based. Frequency-based embeddings utilize various vectorization methods such as Count Vector, TF-IDF Vector, and Cooccurrence Matrix with a fixed context window. Prediction-based embeddings, exemplified by Word2vec, utilize models like Continuous Bag of Words (CBOW) and Skip-Gram (SG). However, the computational complexity of Word2vec's training algorithm can be high due to gradient calculations over the entire vocabulary. To tackle this challenge, variants such as Hierarchical Softmax output, Negative Sampling, and Subsampling of frequent words have been proposed. This task involves implementing a frequency-based modeling approach like Singular Value Decomposition (SVD) and comparing it with embeddings obtained using a Word2vec variant like Skip Gram with Negative Sampling. The analysis will focus on discerning differences in the quality of embeddings produced. While "word vectors" and "word embeddings" are used interchangeably, "embedding" specifically denotes encoding a word's meaning into a lower-dimensional space.

2 Training Word Vectors

2.1 Singular Value Decomposition

Implement a word embedding model and train word vectors by first building a Co-occurrence Matrix followed by the application of SVD. [15 Marks]

2.2 Skip Gram

Implement the word2vec model and train word vectors using the Skip Gram model with Negative Sampling. [15 Marks]

3 Corpus

Please train your model on the given csv files link here: Link to the corpus(News Classification Dataset)

Note that you have to only use the Description column of the train.csv for training your word vectors. You have to use the label/index column for the downstream classification task.

4 Downstream Task

After successfully creating word vectors using the above two methods, evaluate your word vectors by using them for the downstream classification task in the News Classification Dataset provided above. You are free to use any kind of RNN for the downstream task, but use the same RNN and RNN hyperparameters across vectorization methods for the downstream task. [10 + 10 Marks]

5 Analysis

Compare and analyze which of the two word vectorizing methods performs better by using performance metrics such as accuracy, F1 score, precision, recall, and the confusion matrix on both the train and test sets. Write a detailed report on why one technique might perform better than the other. Also, include the possible shortcomings of both techniques (SVD and Word2Vec). [10 Marks]

Hyperparameter tuning

Experiment with at least three different context window sizes. Explain why you chose those context window sizes. Report performance metrics for all three context window configurations. Mention which configuration performs the best and write about the possible reasons for it. [20 Marks]

Submission Format

Zip the following files into one archive and submit it through the Moodle course portal. The filename should be `<roll number>_assignment3.zip`, for example, `2021114017_assignment3.zip`.

- **Source Code**
  - `svd.py`: Train the word embeddings using SVD method and save the word vectors.

  - `skip-gram.py`: Train the word embeddings using Skip gram method (with negative sampling) and save the word vectors.

  - `svd-classification.py`: Train any RNN on the classification task using the SVD word vectors.

  - `skip-gram-classification.py`: Train any RNN on the classification task using the Skip-Gram word vectors.

- **Pretrained Models**

  - `svd-word-vectors.pt`: Saved word vectors for the entire vocabulary trained using SVD.

  - `skip-gram-word-vectors.pt`: Saved word vectors for the entire vocabulary trained using Skip-gram (using negative sampling).

  - `svd-classification-model.pt`: Saved model for the classification task trained using SVD word embeddings.

  - `skip-gram-classification-model.pt`: Saved model for the classification task trained using Skip-gram word embeddings.

- **Report (PDF)**

  - Hyperparameters used to train the model(s).
  - Corresponding graphs and evaluation metrics
  - Your analysis of the results.

- **README**

  - Instructions on how to execute the file, load the pretrained model, implementation assumptions etc.

Ensure that all necessary files are included in the zip archive. There should be, at least, four files in total.

Grading

Evaluation will be individual and based on your viva, report, and code review. During your evaluation, you will be expected to walk us through your code and explain your results. You will be graded based on the correctness of your code, accuracy of your results, and the quality of the code.

Implementation: 50 marks

Hyperparameter Tuning Report: 20 marks

Analysis: 10 marks

Viva during Evaluation: 20 marks

Resources

1. Efficient Estimation of Word Representations in Vector Space
2. Distributed Representations of Words and Phrases and their Compositionality
3. Stanford lecture notes - SVD and word2vec
4. A simple, intuitive explanation of word2vec, with Negative Sampling included
5. Skip Gram with Negative Sampling
6. word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method 5. On word embeddings in general
7. You can also refer to other resources, including lecture slides!