Homework 1: Adventures with Word Embeddings

Leila Safari, Mohammad Mahmoodi

Fall 2023

Advance Natural Language Processing

Due Monday 1402-01-28 at 11:59pm (Upload at the provided link in LMS)

Wednesday(1402-01-30 at 11am (Presenting in person)

The aim of this assignment is to furnish you with a solid practical and theoretical understanding of the inner workings of word embeddings. In the first part of the assignment, you will explore the effects of the various hyperparameters in word embedding algorithms may be tuned with. In the second part, you will use word embeddings in a classification task.

General instructions

Please post all clarification questions about this homework to zaibar2928@gmail.com email or the class Telegram group. You should budget at least a day just for your full set of experiments to run successfully. This likely means several days' worth of runtime in the debugging phase. We strongly recommend that you start coding as soon as possible, but at minimum a week in advance of the deadline.

1 Parameter search

This part of the assignment will involve a hands-on exploration of the effects of various hyperparameters on the information learned by different word embedding models. In overview, you will be training a set of models across a range of algorithms and hyperparameters, while evaluating the information they learn via three different tasks. The hyperparameters you will be exploring are:

Context window size (2, 7, 15) **Dimension (25, 100, 300)**
Number of negative samples (1,7, 20)

The models you will be testing them with are:

word2vec skip-gram with negative sampling

This comes out to 27 unique settings of parameters and model total. You will evaluate each of these using the provided script evaluate.py and report and analyze results in your writeup.

Writeup

Your submission for this part of the assignment will consist of

(1) A table containing the results of your parameter search;

(2) A written analysis of your results.

Each row of the table should contain numerical results for one choice of algorithm and parameter setting. The columns should include algorithm, context window, dimension, number of negative samples, correlation on WordSim353, accuracy for at least three BATS categories (you pick the ones you think are most interesting from among the 9 low-level categories, the 4 high-level categories, or the total score), and accuracy on the win353 paraphrase corpus. The table should look something like this:

Algorithm	Win.	Dim.	N. s.	WordSim	BATS 1	BATS 2	BATS 3	win353
word2vec	2	100	7	47.05	0.01	0.02	0.03	36
…	…	…	…	…	…	…	…	…

Table 1: An example results table.

Your writeup should at least address the following prompts, but you are also encouraged to include other interesting observations or hypotheses you have made.

1. Does larger dimensionality always equate to better performance? In which categories and for which models? Why do you think this is?
2. Does better performance on one task mean better performance on the others? Provide a hypothesis as to why or why not.
3. Was performance roughly similar across all analogy categories? If different, how did it vary? Why do you think you observed this variation? Perform a brief error analysis and compare errors across the BATS categories you selected for your table.

You do not have to answer these in order, but you should explicitly indicate where you answer each of them in your writeup.

1.1 Training corpus

You will train your embedding models on the Brown corpus (You can download it from Kaggle). The file is formatted with one sentence per line, tokens space separated. You may experiment with different methods of pre-processing, but it is recommended that you at least lowercase because some of the evaluation tasks only use lowercased words.

1.2 Implementation details

We recommend that you train your word2vec models using the Gensim package. You are welcome to code up your own implementation (e.g. in PyTorch) if you like, but if you choose to do this, you are responsible for ensuring that your implementation is correct.

1.3 Bonus

For up to five points of extra credit, you may also do an additional evaluation and analysis. You can evaluate the above parameter settings on GloVe (in this case we recommend using the implementation of GloVe available from its website); or you may load a single pre-trained BERT model (see Hugging Face's Transformers library) and evaluate it on the same tasks. If you choose to use BERT, you will need to implement the evaluation yourself, as the provided evaluation script only runs on static models.

Alternatively, you may implement a novel modification to word2vec or GloVe and evaluate it for up to five points of extra credit.

2 Word embedding usage

One of the most popular usages of word embeddings is in classification tasks. In this part, you should clone **https://github.com/gramai/Review-Classification** git repo. The repo is about review classification which is a subtasks of text classification. H/Se (owner of repo) used bag-of-words method for feature extraction. You should use semantic word embedding instead of bag-of-words. To satisfy this task you need to do the following steps:

- Use the best word2vec model which trained in the previous part.
- Use the trained word2vec model over the repo dataset.
- Use pretrain GloVe and word2vec models.
- Use pretrain FastText model (Bonus).

For each step you should report accuracy, Precision, Recall and F1-Score results and put them in a table. Also, you should run the basic scripts and compare your results with the basic one and analyze results.

Good luck!