COMS 4995 - Project Proposal - WordEmb

Yang, Yifan
yy3185@columbia.edu

Huang, Erik
th2925@columbia.edu

November 22, 2021

1 Introduction

For our final project, we are looking into a fundamental component for many natural language processing tasks, word embeddings. More specifically, we are revisiting, from the perspective of functional parallel processing, the static word embeddings derived from word co-occurrence matrix with a fixed-size context window and the truncated SVD method.

Word embeddings, or word vectors, aim to encode a word's meaning in a fixed-size vector, whose dimension is typically magnitudes smaller than the size of the vocabulary. The intuition behind the use of co-occurrence matrix is the distributional hypothesis [2]: the meaning of a word can be determined from the context it appears in. And words with similar meaning would occur in similar context.

To represent a word's context, we slide a fixed-size context window over the corpus and count all pairs of words that occurred in each other's context. The result is a co-occurrence matrix M, where element M_{ij} represents how many times a word i occurred in the context of the word j. For example, for the following corpus:

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

A context window of size 1 would produce the following co-occurrence matrix.

$$\mathbf{X} = \begin{array}{cc} & \begin{array}{cccccccc} \textit{I} & \textit{like} & \textit{enjoy} & \textit{deep} & \textit{learning} & \textit{NLP} & \textit{flying} & . \end{array} \\ \begin{array}{c} \textit{I} \\ \textit{like} \\ \textit{enjoy} \\ \textit{deep} \\ \textit{learning} \\ \textit{NLP} \\ \textit{flying} \\ . \end{array} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 1: Example co-occurrence matrix taken from [4]

Building upon the co-occurrence matrix, studies show that raw counts on occurrences are often not a great measure of association between words as they tend to be very skewed [3]. Therefore, to obtain the vector representation of a word, we compute what is called a Positive Pointwise Mutual Information matrix from the co-occurrence matrix, using the equation below.

$$\mathrm{PPMI}(\mathrm{word}_1, \mathrm{word}_2) = \max(\log_2(\frac{P(\mathrm{word}_1, \mathrm{word}_2)}{P(\mathrm{word}_1)P(\mathrm{word}_2)}), 0) \tag{1}$$

Finally, because the vector representations we have obtained from the co-occurrence matrix has very high dimension ($d = |V|$, where V is the vocabulary set), we want to obtain an approximate representation using fewer dimensions. One way to do so is the truncated Singular Value Decomposition method, in which we truncate a matrix M to the top k singular values.

$$\underset{|V|\times|V|}{\begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix}} = \underset{|V|\times k}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k\times|V|}{\begin{bmatrix} & C & \end{bmatrix}}$$

Figure 2: Visualization of truncated SVD matrix [3]

The $|V| \times k$ matrix W would be the final word embedding, where k is the dimension of the embedding. The obtained embedding can then be used for downstream NLP tasks such as sentiment analysis, named entity recognition, etc.

2 Objective

As outlined in the introduction, we aim to implement a parallel algorithm in computing the static word embedding. The procedure that we will be mainly focus on are:

1. Parallelizing the computation of co-occurrence matrix, which involves how do we partition the corpus and combine intermediate results. This would be challenging because the co-occurence matrix would be incredibly sparse, and threads/workers cannot afford maintain an entire co-occurrence matrix in the memory.

2. Parallelizing the computation of the PPMI matrix. This poses similar challenges as above.

If time permitted, we will also attempt implementing our own parallelized SVD solver as outlined in [1]. However, solving SVD is by itself a heavy and complicated task, and most users would seek to use existing solver such as the Numpy or the Scipy SVD solver.

References

[1] M. Berry. *Parallel Algorithms for the Singular Value Decomposition.* 2005. URL: `https://www.irisa.fr/sage/bernard/publis/SVD-Chapter06.pdf`.

[2] Association for Computational Linguistics. *Distributional Hypothesis.* Dec. 2010. URL: `https://aclweb.org/aclwiki/Distributional_Hypothesis`.

[3] D. Jurafsky. *Vector Semantics.* 2019. URL: `https://web.stanford.edu/~jurafsky/li15/lec3.vector.pdf`.

[4] C. Maning. *CS224n: Natural Language Processing with Deep Learning.* 2019. URL: `http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes01-wordvecs1.pdf`.