Count-based

Co-occurrence Matrix Build word-context co-occurrence matrix which is sparse and symmetric. In order to save the memory, we store the co-occurrence frequency only in upper triangular matrix. It saves half of the memory and speed up the computation.

The importance between target word and each context word is not the same. The closer to target word the more important it is. We adopt linear distance weighting in (3.1). Weight decreases as distance increases from target word.

$$[\ldots, target, 1, \frac{d-1}{d}, \frac{d-2}{d}, \ldots, \frac{1}{d}] \quad \text{d:distance to target word} \tag{3.1}$$

Weighted Co-occurrence Matrix Instead of using raw co-occurrence frequency, we use pointwise mutual information (PMI) [80] in (3.2) to weight.

$$PMI(x,y) = \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{NC(x,y)}{C(x), C(y)} \tag{3.2}$$

$p(x)$: occurrence probability of word x
$p(x,y)$: co-occurrence probability of word x and y
$C(x)$: occurrence of x
$C(x,y)$: co-occurrence of x and y
N : total number of words in corpus

It calculates how much more x and y co-occur. The range of PMI is from $-\infty$ to $min[-\log p(x), -\log p(y)]$. The value is 0 if x and y is independent (co-occur by chance) in corpora. The value is maximized when x and y are perfectly associated (always co-occur), and the value is minimized when x and y barely co-occur. Words with strong association have higher PMI value (high C(x,y)). PMI has different variants, such as positive PMI (PPMI) [81], PMI^k [82], Normalized PMI (NPMI) [83] and shifted PPMI(SPPMI) [84]. PPMI sets negative PMI value to 0. It makes sense to mark low correlation(tend not to co-occur) and uncorrelated (never co-occur) word pair to 0. The weighted matrix becomes more sparse since negative values are removed, and the computation cost is lower than PMI. PMI^k and NPMI were proposed to make PMI less sensitive to rare words. Levy and Goldberg [84] found that skip-gram with negative sampling (SGNS) is implicitly factorizing a shifted PMI co-occurrence matrix, hence SPPMI is the original PMI shifted by $\log k$ (k > 0).

$$SPPMI(x, y) = Max(PMI(x, y) - \log k, 0) \tag{3.3}$$

Dimensionality Reduction The size of our weighted matrix is 249571×249571 which is extremely large. We use truncated Singular Value Decomposition (truncated SVD) to reduce high-dimensional matrix. The formula of SVD and truncated SVD is in (3.4). $X_{m\times n}$ is a $m \times n$ co-occurrence matrix. $U_{m\times m}$ and $V^T_{n\times n}$ is left and right singular vectors. $\Sigma_{m\times n}$ is a diagonal matrix containing non-negative real number (singular values) on the diagonal in descending order. Truncated SVD discards values except first r largest singular values, they contain most of

information in original matrix. It ensures the minimal loss of information and dimensionality reduction as well.

$$X_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \rightarrow U_{m\times r}\Sigma_{r\times r}V^T_{r\times n} \tag{3.4}$$

Weighting Exponent Caron p-transform [85] adjusts the scale of singular values in diagonal matrix Σ by weighting exponent p, where $X = U\Sigma^p$

Principal Components Removal Another method similar to Caron p-transform is principal components removal (PC-removal) [86]. It removes the first k dimensions of Σ. High variance dimensions may contain more unuseful information to lexical semantic tasks in contrast to low variance dimensions. If mapping the first k dimensions back to the co-occurrence space, most contributing words are people's names, "and" and "or". Figure 3.4[7] shows the the performance of dimensions removal in different levels of random noise. The performance falls off slowly (some information lost) if removing dimensions from matrix with a small amount of noise. It shows significant effect of noise reduction if removing dimensions from matrix with large amount of noise. Optimal value of weighting exponent p and the number of removed first k dimensions depending on tasks and corpus.

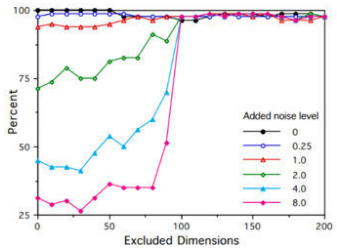

Figure 3.4: Dimensions removal.

[7] This figure is from [86]'s experiments.

Matrix Normalization Normalize the truncated SVD matrix to speed up the computation. Each row vector is a unit vector (length = 1), therefore, the cosine similarity is dot product of two word embeddings.

After weighted co-occurrence matrix and dimensionality reduction, the count-based word embedding can be formulated in (3.5)

$$EM = [SVD_p(WCM)]_{m \times k:r} \tag{3.5}$$

EM : embedding matrix
WCM : weighted co-occurrence matrix
p : weighting exponent
m : number of rows (words)
k : remove first k dimensions
r : number of columns after truncated SVD

As to prediction-based word embedding, we use GENSIM [8] to train our models which are skip-gram and CBOW (Continuous Bag of Words). The detailed implementations are in their official website, and it won't be introduced here.

[8] https://radimrehurek.com/gensim/