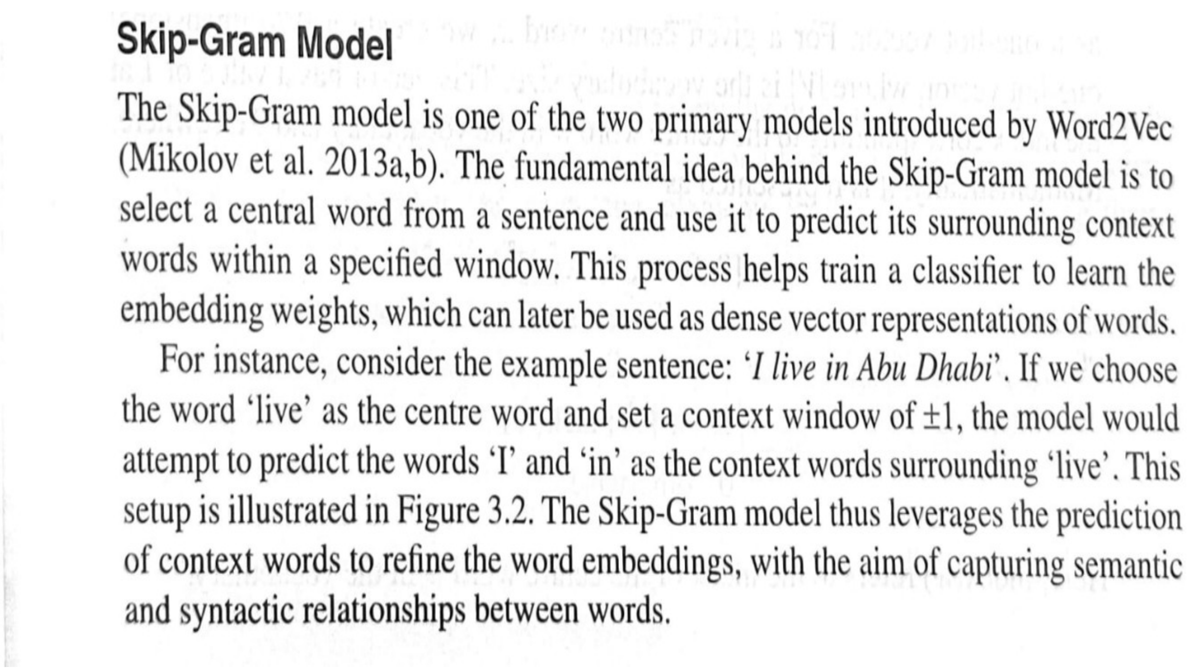
Word Embedding–based Generative Models, GloVe and Skip-Gram Models

The **Skip-gram model** is a neural network architecture used for learning word embeddings by predicting the surrounding context words from a given target word. It is the inverse of the CBOW model, where the input is the target word and the output is the probability of words appearing in its context window. This process allows the model to capture semantic relationships between words.



Skip-Gram Model

The Skip-Gram model is one of the two primary models introduced by Word2Vec (Mikolov et al. 2013a,b). The fundamental idea behind the Skip-Gram model is to select a central word from a sentence and use it to predict its surrounding context words within a specified window. This process helps train a classifier to learn the embedding weights, which can later be used as dense vector representations of words.

For instance, consider the example sentence: '*I live in Abu Dhabi*'. If we choose the word 'live' as the centre word and set a context window of ±1, the model would attempt to predict the words 'I' and 'in' as the context words surrounding 'live'. This setup is illustrated in Figure 3.2. The Skip-Gram model thus leverages the prediction of context words to refine the word embeddings, with the aim of capturing semantic and syntactic relationships between words.

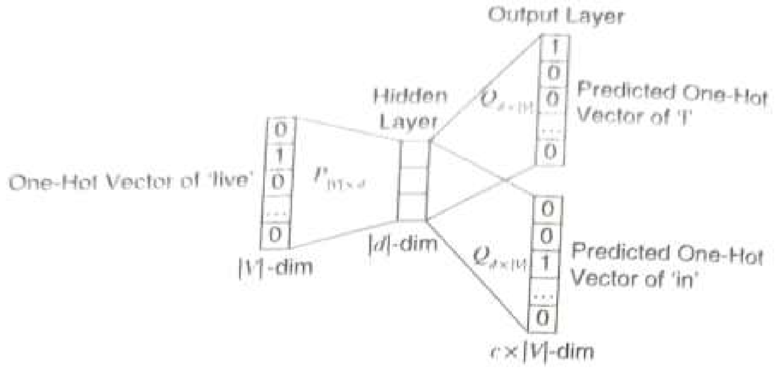


FIGURE 3.2 An illustration of the Skip-Gram model to predict the context words 'I' and 'in' given the input centre word 'life'. The centre word is converted to a one-hot encoder vector of dimension $|V|$, and the output final layers are the one-hot encoding vectors for the two predicted outputs. The model learns the weight matrices, P and Q, which are used as the embedding. Each row of the P matrix is a d-dimensional embedding.

Preliminaries. Let us begin with some foundational concepts. In the Skip-Gram model, words in a sentence are represented as one-hot vectors, which are used to classify whether any word in the vocabulary is a neighbour of the centre word. The centre word is represented by a one-hot vector x, while the context words are denoted by y^j. To facilitate this process, we introduce two matrices: $P \in \mathbb{R}^{|V|\times d}$ and $Q \in \mathbb{R}^{d\times|V|}$. Here, P represents the input word matrix and Q represents the output word matrix. The variable d stands for the embedding dimension, while $|V|$ denotes the vocabulary size, which corresponds to the size of the one-hot vectors. In the input matrix P, the word w_i is associated with a d-dimensional embedding vector located in the ith row of P. This row vector is denoted by p_i. Similarly, in the output matrix Q, the word w_j is associated with a d-dimensional embedding vector located in the jth column of Q, denoted by q_j. The Skip-Gram model aims to learn both the input (p_i) and output (q_j) word vectors for each word w_i. Essentially, the model seeks to learn two embedding vectors, p and q, for every word in the vocabulary. The following steps outline the process for learning these embedding weights:

1. **Generation of One-Hot Input Vector:** Initially, the centre word is represented as a one-hot vector. For a given centre word x, we create a $|V|$-dimensional one-hot vector, where $|V|$ is the vocabulary size. This vector has a value of 1 at the index corresponding to the centre word w in the vocabulary and 0 elsewhere. Mathematically, it is represented as:

$$x = [0, 0, \ldots, 1, \ldots, 0]^T$$

or

$$x = \begin{cases} 1 & \text{if } i = \text{index}(w) \\ 0 & \text{otherwise} \end{cases}$$

Here, index(w) refers to the index of the centre word w in the vocabulary.

2. **Get the Embedded Vector for the Centre Word:** Next, we multiply the one-hot vector x by the input word matrix P to obtain the embedding vector p_c for the centre word. This operation can be expressed as:

$$p_c = P^T x \in \mathbb{R}^d$$

Here, P^T is the transpose of the input word matrix P, and p_c is a d-dimensional vector that represents the embedding of the centre word. This multiplication effectively selects the column in P corresponding to the centre word's index, yielding its embedding vector.

3. **Getting the Similarity Between the Centre and Context Words:** The intuition behind the Skip-Gram model is that the likelihood of a word appearing close to a target word depends on the similarity of their embedding vectors. To measure this similarity, we can use the dot product, where a higher dot product indicates a greater degree of similarity between the words. To compute the similarity score, we multiply the centre word vector p_c by the output word matrix Q, resulting in the score vector z:

$$z = Q^T p_c \tag{3.6}$$

Each element in z represents the dot product score between the context word and the centre word, reflecting the model's prediction of how likely each word is to appear in the context of the centre word.

4. **Turn Similarity Scores into Probability:** The similarity scores obtained in the previous step are not sufficient for our prediction task for several reasons. First, these scores are not probabilities; they range unboundedly between $-\infty$ and $+\infty$. Second, the scores do not account for relative scores across all the words in the vocabulary. For our task, we need a concrete probability that represents the likelihood of a context word given a centre word.

   To achieve this, we use the *softmax function* to convert these scores into a probability distribution:

$$\hat{y} = \text{softmax}(z) = \frac{\exp(z)}{\sum_{j=1}^{|V|} \exp(z_j)}$$

Here, $\hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m}$ represent the probabilities of observing each context word within a window of size m around the centre word. These probabilities correspond to the respective elements in the $\hat{y}$ vector, making them suitable for the prediction task.

5. **Objective Function:** To learn the Q and P matrices, we need an objective function that measures how well the predicted probabilities $\hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m}$ match the true probabilities $y_{c-m}, \ldots, y_{c-1}, y_{c+1}, \ldots, y_{c+m}$. We use an independence assumption to simplify the model by assuming that, given the centre word, all output words are independent of each other. This allows us to break down the joint probability into a product of individual probabilities.

The objective function J that we aim to minimise is the negative log-likelihood of the true context words, given the centre word:

$$\text{minimise } J = -\log P(w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m} \mid w_c)$$

$$= -\log \prod_{j=0,j\neq m}^{2m} P(w_{c-m+j} \mid w_c)$$

$$= -\log \prod_{j=0,j\neq m}^{2m} P(q_{c-m+j} \mid p_c)$$

$$= -\log \prod_{j=0,j\neq m}^{2m} \frac{\exp(q_{c-m+j}^T p_c)}{\sum_{k=1}^{|V|} \exp(q_k^T p_c)}$$

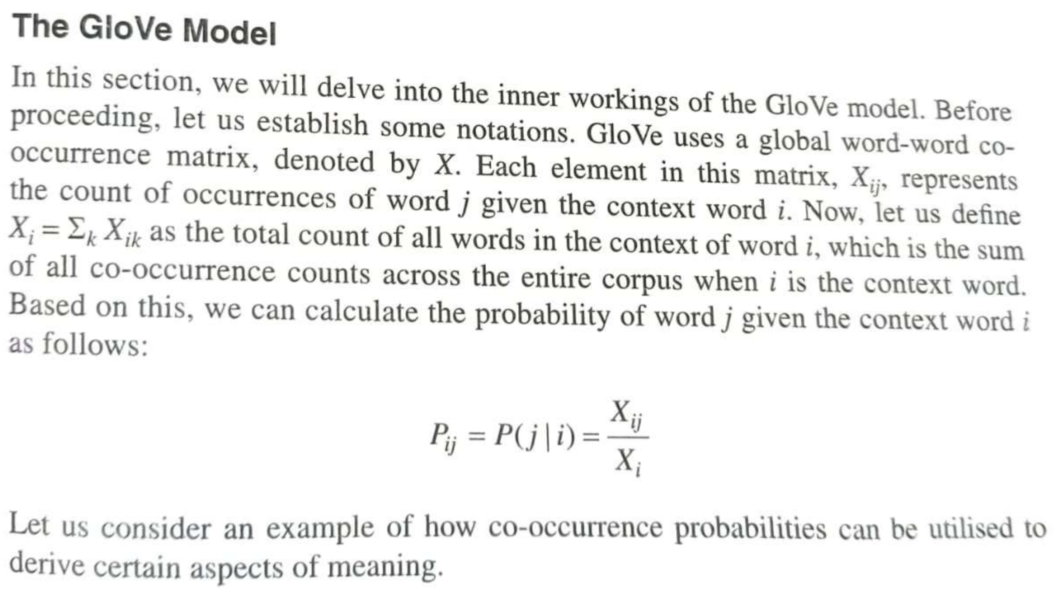
$$= -\prod_{j=0,j\neq m}^{2m} q_{c-m+j}^T p_c + 2m \log \exp(q_k^T p_c)$$

Now, Stochastic Gradient Descent (SGD) (as discussed in Chapter 2) can be used to update all the relevant word vectors q_c and p_c.

Global Vectors for Word Representation

We discussed word representation techniques using count-based methods (like co-occurrence matrices) and predictive methods (like Word2Vec). Count-based methods often fall short in tasks such as analogy, indicating that their vector representations are suboptimal.

Global Vectors for Word Representation (GloVe) combine the strengths of both approaches by leveraging the global statistical information captured by word-word co-occurrence matrices while also benefiting from the local context window approach used in Word2Vec.



The GloVe Model

In this section, we will delve into the inner workings of the GloVe model. Before proceeding, let us establish some notations. GloVe uses a global word-word co-occurrence matrix, denoted by X. Each element in this matrix, X_{ij}, represents the count of occurrences of word j given the context word i. Now, let us define $X_i = \Sigma_k X_{ik}$ as the total count of all words in the context of word i, which is the sum of all co-occurrence counts across the entire corpus when i is the context word. Based on this, we can calculate the probability of word j given the context word i as follows:

$$P_{ij} = P(j \mid i) = \frac{X_{ij}}{X_i}$$

Let us consider an example of how co-occurrence probabilities can be utilised to derive certain aspects of meaning.

.3 Let us consider two words, i and j, that represent movie genres: i = *comedy* and j = *horror*. We will examine the relationship of the words i and j by studying the ratio of their co-occurrence probabilities with some probe words k. In the example shown in Table 3.4, we use k = *laughter* for words related to *comedy* but not with *horror*, and the ratio of P_{ik}/P_{jk} or $P(laughter|comedy)/P(laughter|horror)$ is large (8.89) as expected. Similarly, k = *scary* will be more related to *horror*, and the ratio P_{ik}/P_{jk} is low (0.05). And, the words that are either related to both *comedy* or *horror* or totally unrelated will have a ratio close to one. This example demonstrates that a ratio of probabilities can effectively distinguish words relevant to one genre but not the other (*laughter, scary*) and words relevant to either both (*popcorn*) or none (*algebra*). In short, this approach of taking the ratio of co-occurrence probabilities provides a more effective and distinctive measure as against the raw probabilities.

TABLE 3.4 An illustration of co-occurrence probabilities and ratios for the target words *comedy* and *horror* with k probe words as context from a hypothetical corpus.

k	$P(k\|comedy)$	$P(k\|horror)$	$P(k\|comedy)/P(k\|horror)$
laughter	0.0080	0.0009	8.89
scary	0.0005	0.0100	0.05
popcorn	0.0050	0.0045	1.11
algebra	0.0001	0.0001	1.00

Objective Function. In Example 3.3, we noticed that the ratio of probabilities is a good starting point for learning word vectors as compared to the raw probabilities. We also came across how P_{ik}/P_{jk} score reflects when the words i and j are associated with the word k. We want to encode this insight into the vector space through a function F as follows:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{3.10}$$

where, $w \in \mathbb{R}^d$ is the word vector when the word is the centre or the main word, while and $\tilde{w} \in \mathbb{R}^d$ is the word vector for the context. Since vector spaces are linear, we can re-model F in Equation (3.10) with vector differences as in Equation (3.11):

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{3.11}$$

Now, the RHS of Equation (3.11) is a scalar, and the LHS is a vector. Hence, this needs to be addressed to maintain the linear structure by taking the dot product of the arguments as below:

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{3.12}$$

In the word-word co-occurrence matrices, the word and context can be interchangeable. So, we need to make F to be invariable even with the interchange of $w \leftrightarrow \tilde{w}$ and $X \leftrightarrow X^T$. For this, we require F to follow *homomorphism* between the groups $(\mathbb{R}, +)$ and $(\mathbb{R} > 0, \times)$, i.e.,

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \tag{3.13}$$

Equation (3.13) is solved using Equation (3.12) as:

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i} \tag{3.14}$$

Now, one solution of Equation (3.13) is $F = exp$, or,

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i) \tag{3.15}$$

In Equation (3.15), log (X_i) breaks the symmetry. However, it is independent of k and can be incorporated into a bias term b_i for the main word vector w_i. Finally, an additional bias $\tilde{b}_k$ for w_k is introduced to restore the symmetry,

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik}) \tag{3.16}$$

The GloVe modelling with Equation (3.16) has certain drawbacks, such as when two words never occur, the count is zero, and log (0) is undefined. A naïve approach would be to add a small integer value so that log X_{ik} becomes log $(1 + X_{ik})$. This maintains the sparsity of X while avoiding the mathematical divergence. However, this approach gives equal weightage to all the co-occurrences regardless of their frequency. To address this issue, Equation (3.16) can be reformulated to a weighted least square regression model. The final weighted objective function of the GloVe is given by:

$$J = \sum_{i,j=1}^{|V|} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2, \tag{3.17}$$

where $|V|$ is the vocabulary size and $f(X_{ij})$ is the weighing function. In the original GloVe paper (Pennington et al. 2014), the weighing function $f(X_{ij})$ is computed using:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise.} \end{cases}$$

Pennington et al. (2014) set $x_{\max} = 100$ and $\alpha = 3/4$.

Model Learning. Consider Skip-Gram, where the task is to model the probability distribution Q_{ij} for j to appear given the context i. Now, this probability is calculated using a softmax,

$$Q_{ij} = \frac{\exp(w_i^T \tilde{w}_j)}{\sum_{k=1}^{|V|} \exp(w_i^T \tilde{w}_k)}.$$

Skip-Gram attempts to maximise the log probability as a context window scans over the corpus, and the global objective function can be written as:

$$J = - \sum_{\substack{i \in \text{corpus} \\ j \in \text{context}(i)}} \log Q_{ij} \tag{3.18}$$

Computing the softmax for each term in Equation (3.18) is costly. Therefore, GloVe uses the precomputed co-occurrence matrix X to group terms with the same i and j values.

$$J = -\sum_{i=1}^{|V|} \sum_{j=1}^{|V|} X_{ij} \log Q_{ij} \tag{3.19}$$

This can be further deduced to:

$$J = -\sum_{i,j=1}^{|V|} X_i P_{ij} \log Q_{ij} \tag{3.20}$$

This cross-entropy loss itself is costly because it needs a normalised distribution of Q, which is expensive due to the summation over the entire vocabulary. Therefore, we can use the least square objective where the normalisation of P and Q can be skipped,

$$\hat{J} = \sum_{i,j} X_i (\hat{P}_{ij} - \hat{Q}_{ij})^2 \tag{3.21}$$

where $\hat{P}_{ij} = X_{ij}$ and $\hat{Q}_{ij} = \exp(w_i^T \tilde{w}_j)$ are the unnormalised distributions. Here, X_{ij} can take large values; so instead, logarithms of $\hat{P}$ and $\hat{Q}$ are used,

$$\hat{J} = \sum_{i,j} X_i (\log \hat{P}_{ij} - \log \hat{Q}_{ij})^2 = \sum_{i,j} X_i (w_i^T \tilde{w}_j - \log X_{ij})^2 \tag{3.22}$$

Here, the weighting factor X_i is not guaranteed to be optimal. Hence, a more general weighting function which depends on the context word can be used:

$$\hat{J} = \sum_{i,j} f(X_{ij})(w_i^T \tilde{w}_j - \log X_{ij})^2 \tag{3.23}$$

This is the final weighted least squared objective function of GloVe, which is used to train the word vectors w_i for the word i.