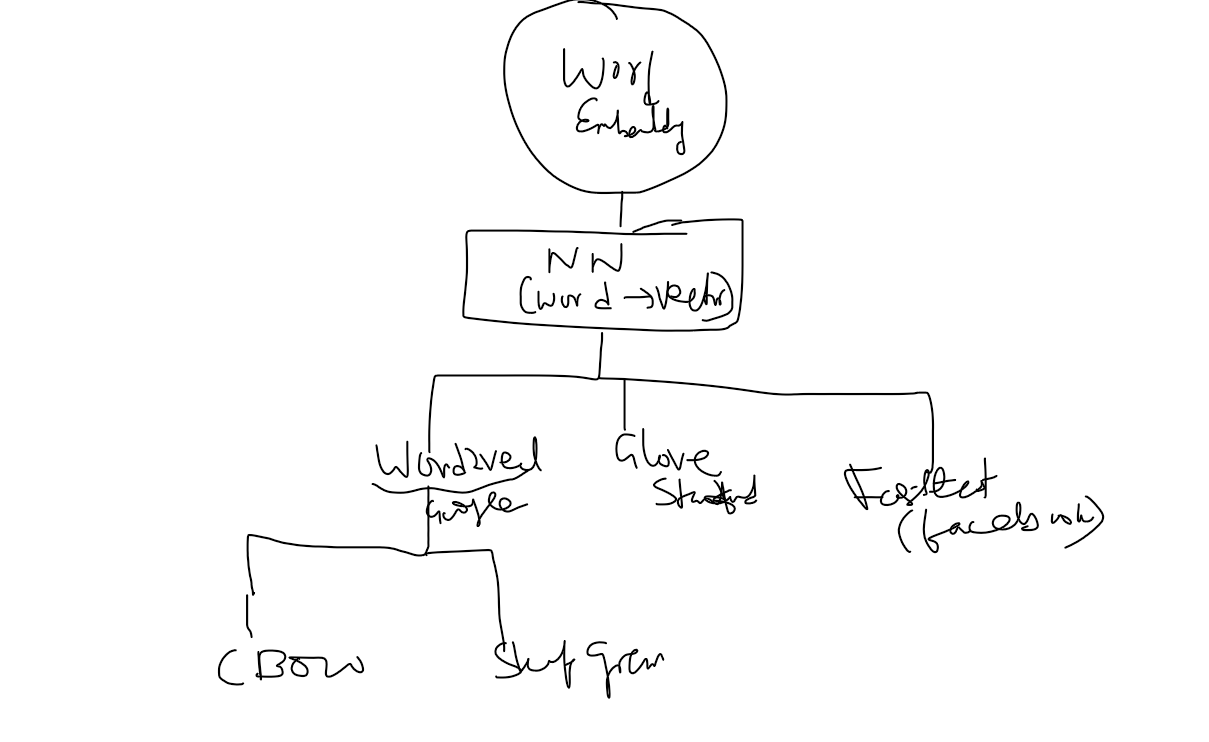
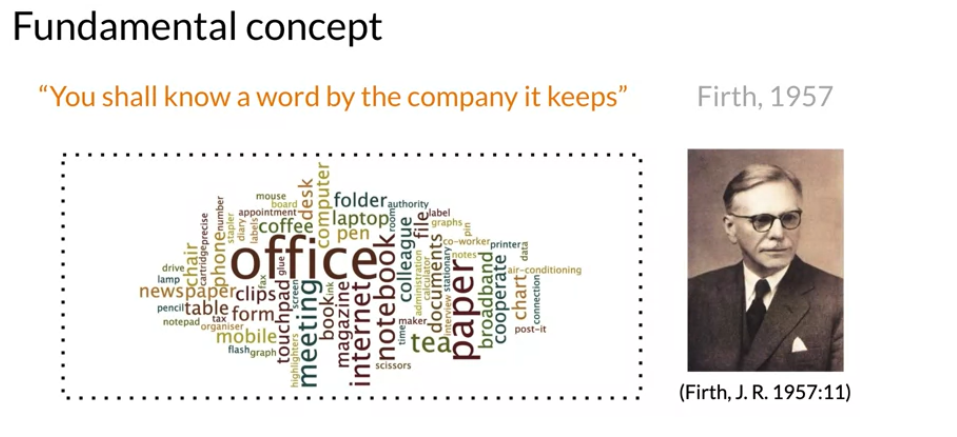
Word Embeddings



It is a mapping of words into vectors of real numbers using the neural network, probabilistic model, or dimension reduction on word co-occurrence matrix.Word embedding is a way to perform mapping using a neural network. There are various word embedding models available such as word2vec (Google), Glove (Stanford) and fastest (Facebook).



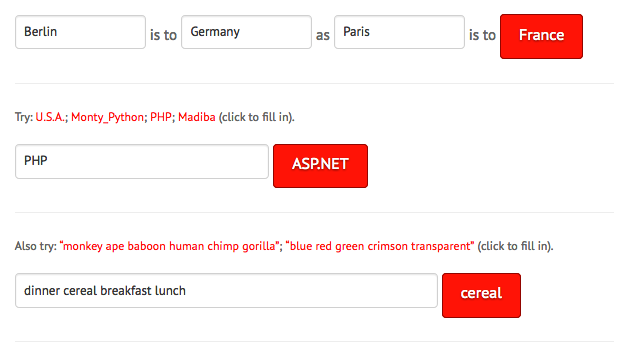
**Word2Vec**

In 2013, a seminal work by Mikolov et al. [[7](https://learning.oreilly.com/library/view/practical-natural-language/9781492054047/ch03.html#footnote_3_3)] showed that their neural network–based word representation model known as “Word2vec,” based on “distributional similarity,” can capture word analogy relationships such as:

* King – Man + Woman ≈ Queen

“it derives the meaning of a word from its context: words that appear in its neighborhood in the text. So, if two different words (often) occur in similar context, then it’s highly likely that their meanings are also similar. Word2vec operationalizes this by projecting the meaning of the words in a vector space where words with similar meanings will tend to cluster together, and words with very different meanings are far from one another.

Word2vec is the technique/model to produce word embedding for better word representation



There are two architectures used by word2vec

1. Continuous Bag of words (CBOW)
2. skip gram

**CBOW**

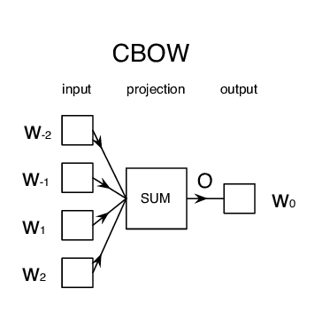
CBOW, the primary task is to build a language model that correctly predicts the center word given the context words in which the center word appears

It is a (statistical) model that tries to give a probability distribution over sequences of words. Given a sentence of, say, m words, it assigns a probability Pr(w1, w2, ….., wn) to the whole sentence

*High probability to “good” sentences and low probabilities to “bad” sentences. By good, we mean sentences that are semantically and syntactically correct. By bad, we mean sentences that are incorrect—semantically or syntactically or both. So, for a sentence like “The cat jumped over the dog,” it will try to assign a probability close to 1.0, whereas for a sentence like “jumped over the the cat dog,” it tries to assign a probability close to 0.0*

he CBOW model tries to understand the context of the words and takes this as input. It then tries to predict words that are contextually accurate. Let us consider an example for understanding this. Consider the sentence: ‘It is a pleasant day’ and the word ‘pleasant’ goes as input to the [neural network](https://analyticsindiamag.com/how-neural-network-can-be-trained-to-play-the-snake-game/). We are trying to predict the word ‘day’ here. We will use the one-hot encoding for the input words and measure the error rates with the[one-hot encoded](https://analyticsindiamag.com/comparing-label-encoding-and-one-hot-encoding-with-python-implementation/) target word. Doing this will help us predict the output based on the word with [least error](https://analyticsindiamag.com/decoding-most-used-confused-abused-jargons-in-machine-learning/).

### **The Model Architecture**



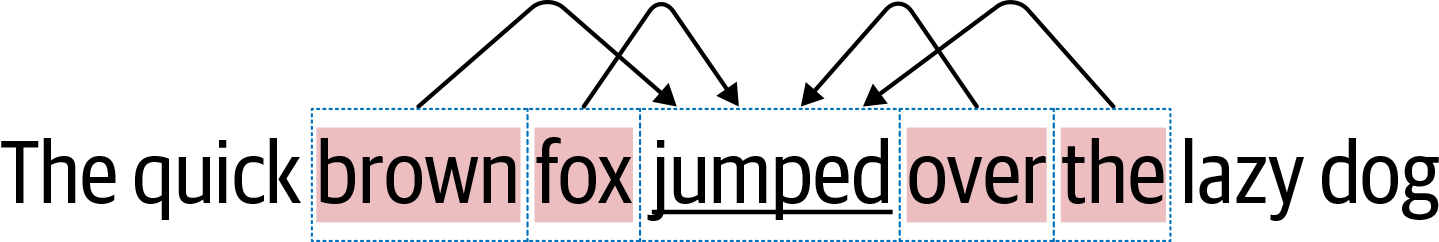
The CBOW model architecture is as shown above. The model tries to predict the target word by trying to understand the context of the surrounding words. Consider the same sentence as above, ‘It is a pleasant day’.The model converts this sentence into word pairs in the form (contextword, targetword). The user will have to set the window size. If the window for the context word is 2 then the word pairs would look like this: ([it, a], is), ([is, pleasant], a),([a, day], pleasant). With these word pairs, the model tries to predict the target word considered the context words.

If we have 4 context words used for predicting one target word the input layer will be in the form of four 1XW input vectors. These input vectors will be passed to the hidden layer where it is multiplied by a WXN matrix. Finally, the 1XN output from the hidden layer enters the sum layer where an element-wise summation is performed on the vectors before a final activation is performed and the output is obtained.



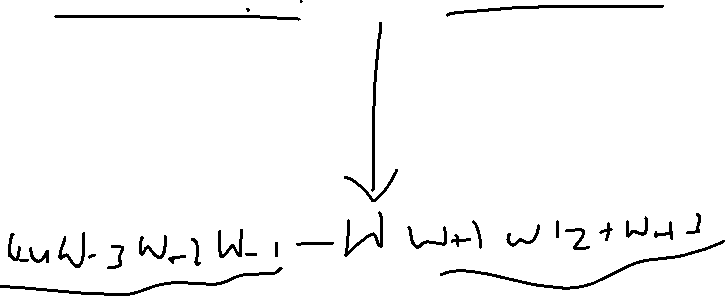
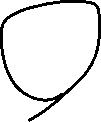
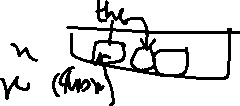
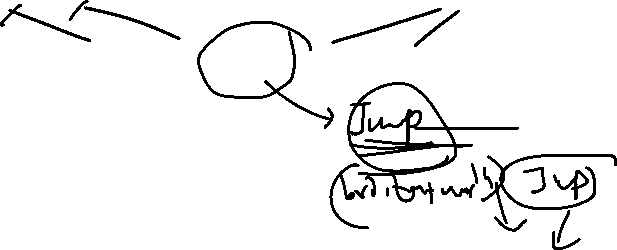
*Example 2*

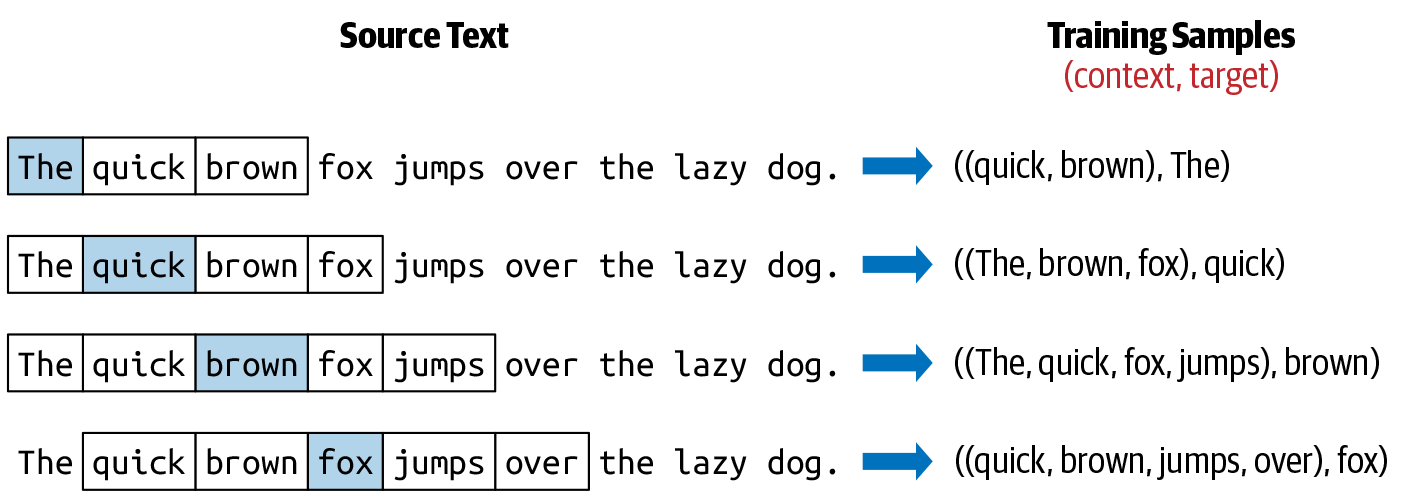


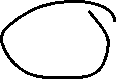
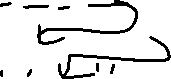
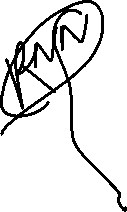
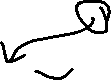
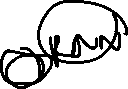
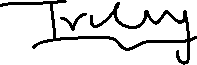


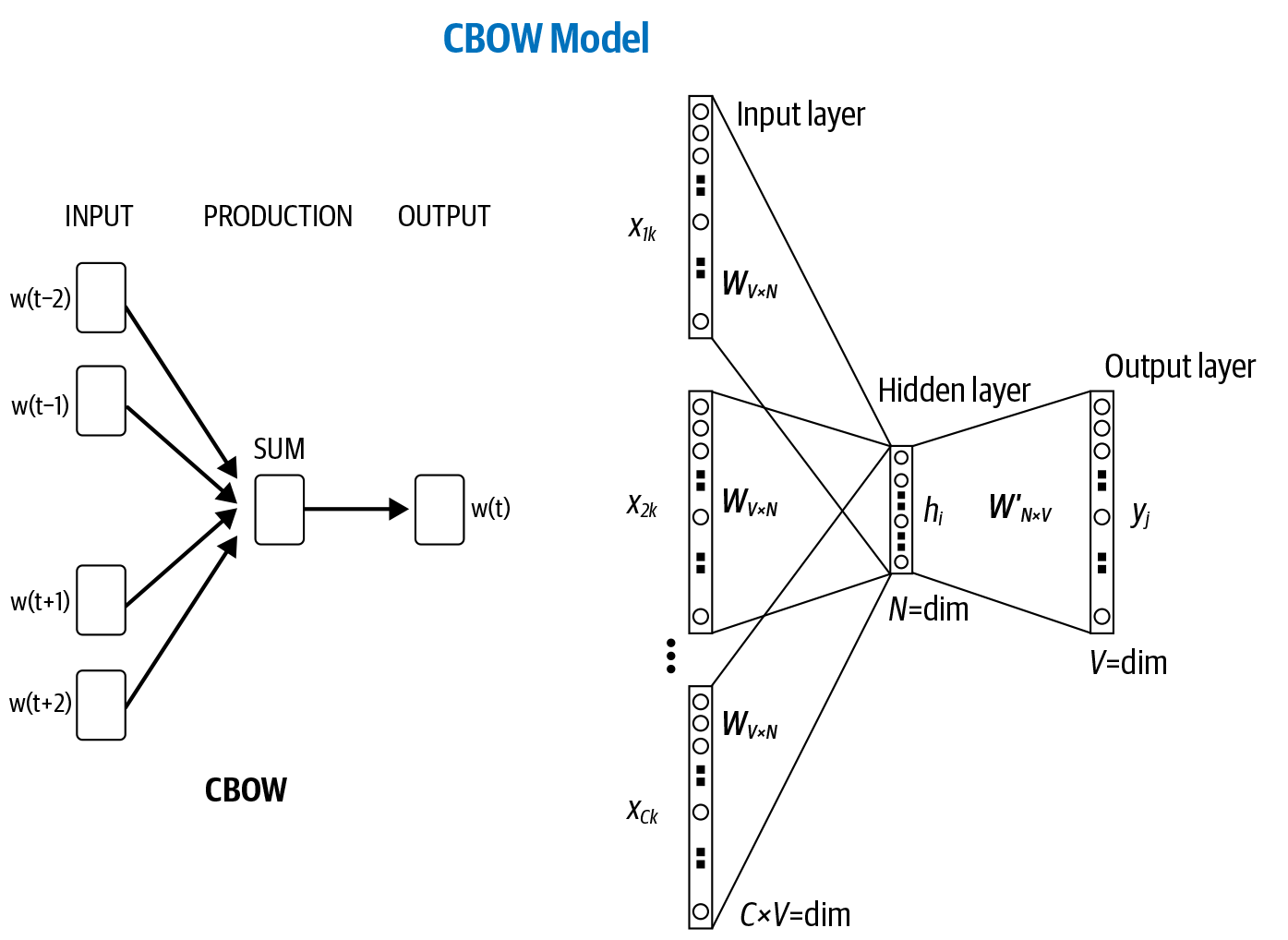


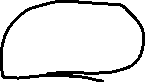
k as 2





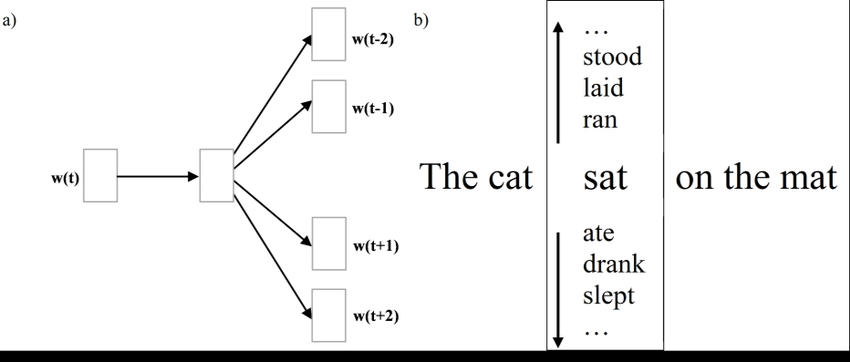




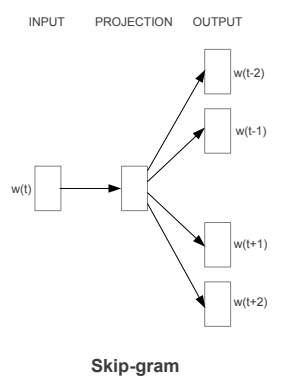


Skip-Gram

Skip-gram is used to predict the context word for a given target word. It’s reverse of CBOW algorithm. Here, target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult.

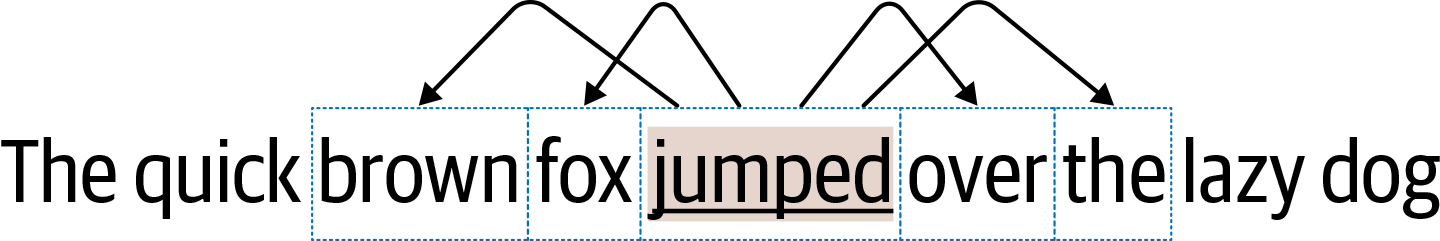


The word sat will be given and we’ll try to predict words cat, mat at position -1 and 3 respectively given sat is at position 0 .

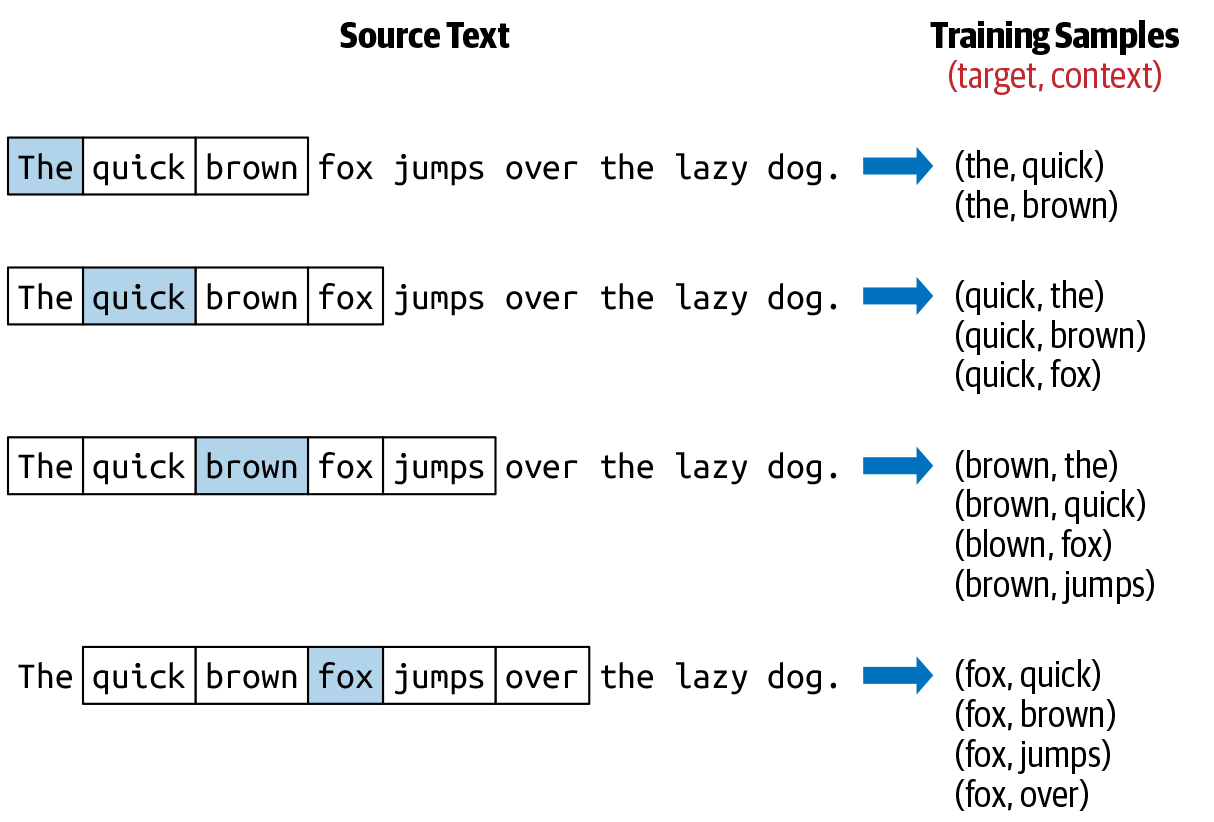


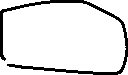
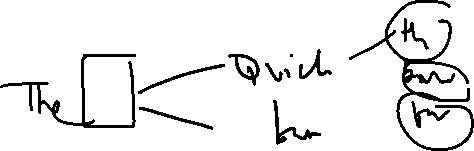
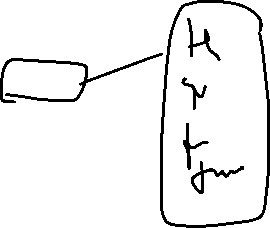
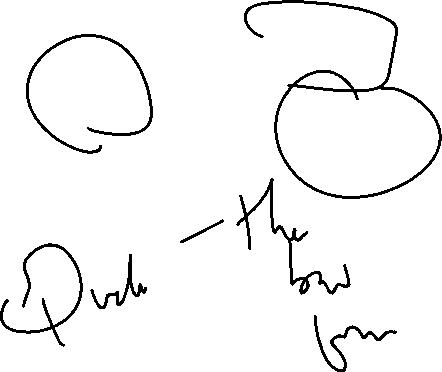
As we can see w(t) is the target word or input given. There is one hidden layer which performs the dot product between the weight matrix and the input vector w(t). No activation function is used in the hidden layer. Now the result of the dot product at the hidden layer is passed to the output layer. Output layer computes the dot product between the output vector of the hidden layer and the weight matrix of the output layer. Then we apply the softmax activation function to compute the probability of words appearing to be in the context of w(t) at given context location.

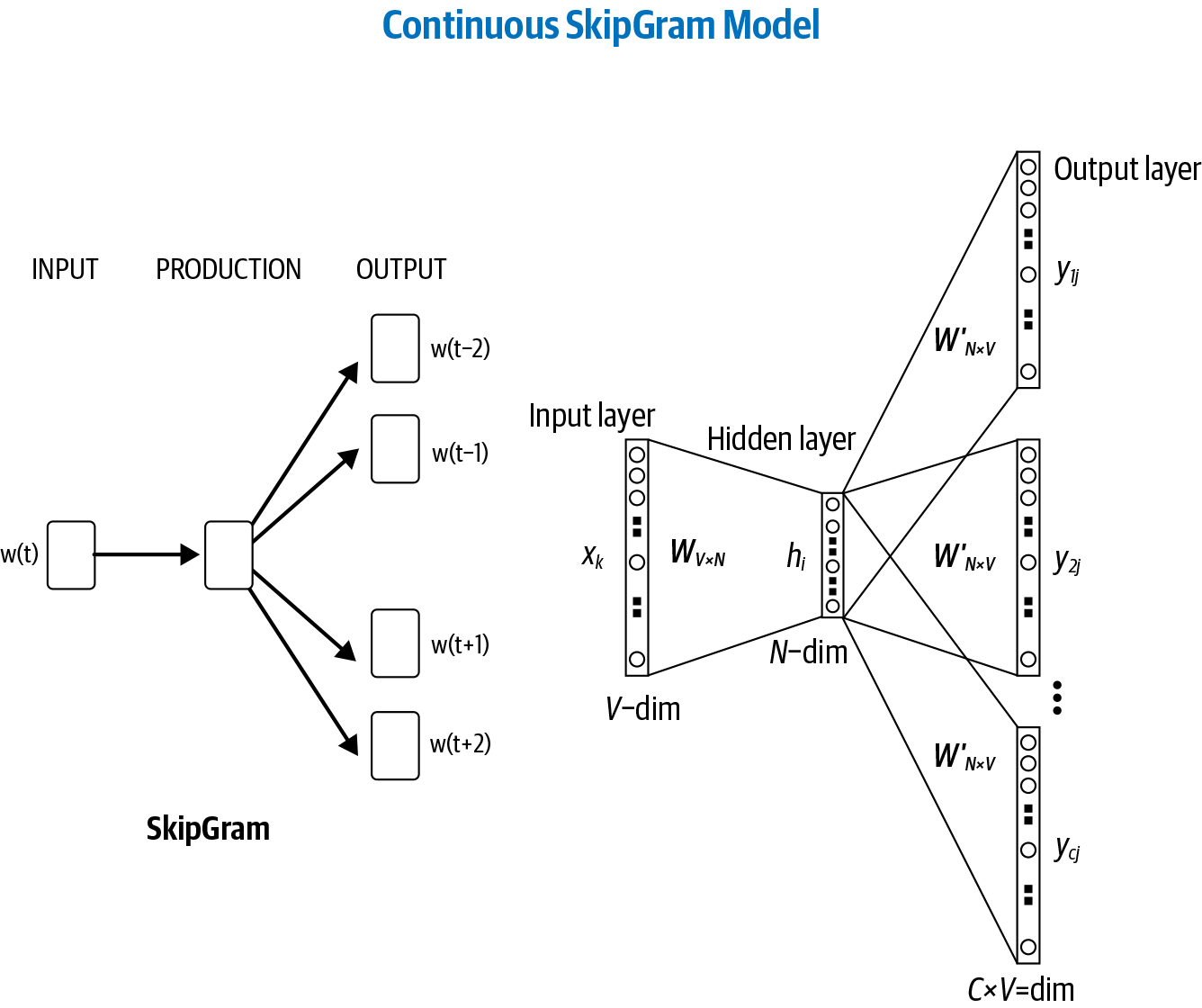
SkipGram, the task is to predict the context words from the center word. For our toy corpus with context size 2, using the center word “jumps,” we try to predict every word in context—“brown,” “fox,” “over,” “the”



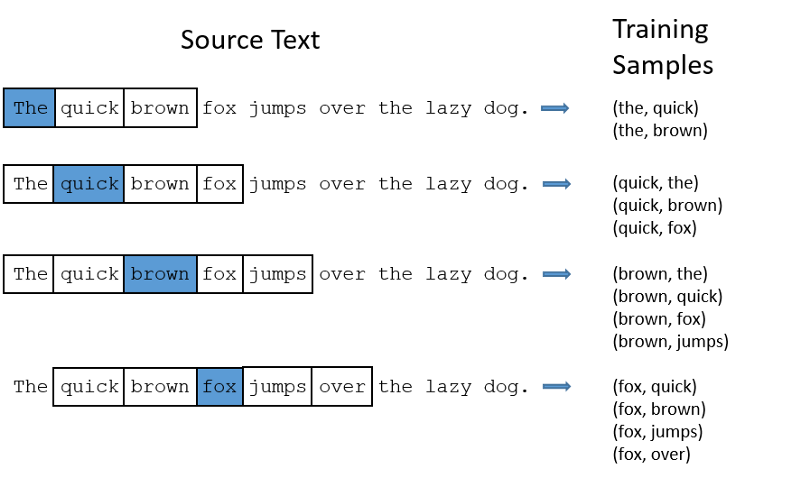






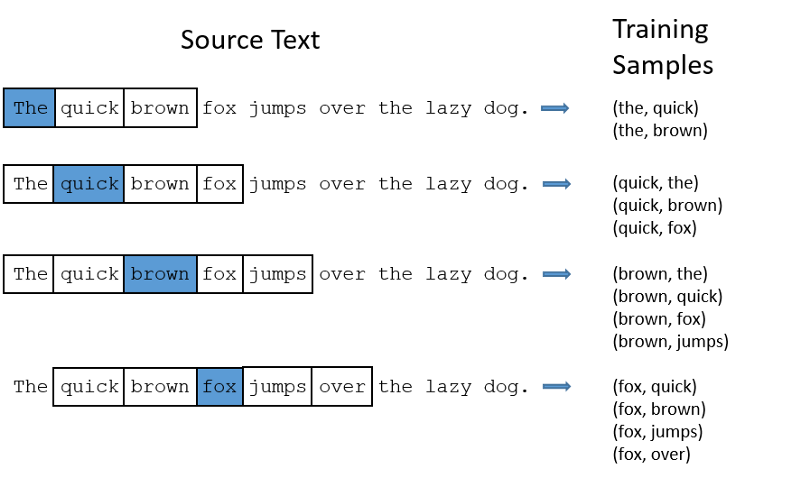


**CBOW** is learning to predict the word by the context. Here the input will be the context **#neighboring words** and output will be the target word. The limit on the number of words in each context is determined by a parameter called “**window size**”.



Skip Gram

Skip Gram is learning to predict the context by the word. Here the input will be the word and output will be the target context #neighboring words. The limit on the number of words in each context is determined by a parameter called “window size”



Example: The quick brown fox jumps over the lazy dog #yes the same example :-)

Model: Skip Gram

INPUT Layer: blue box word

TARGET Layer: White box content Window Size: 5