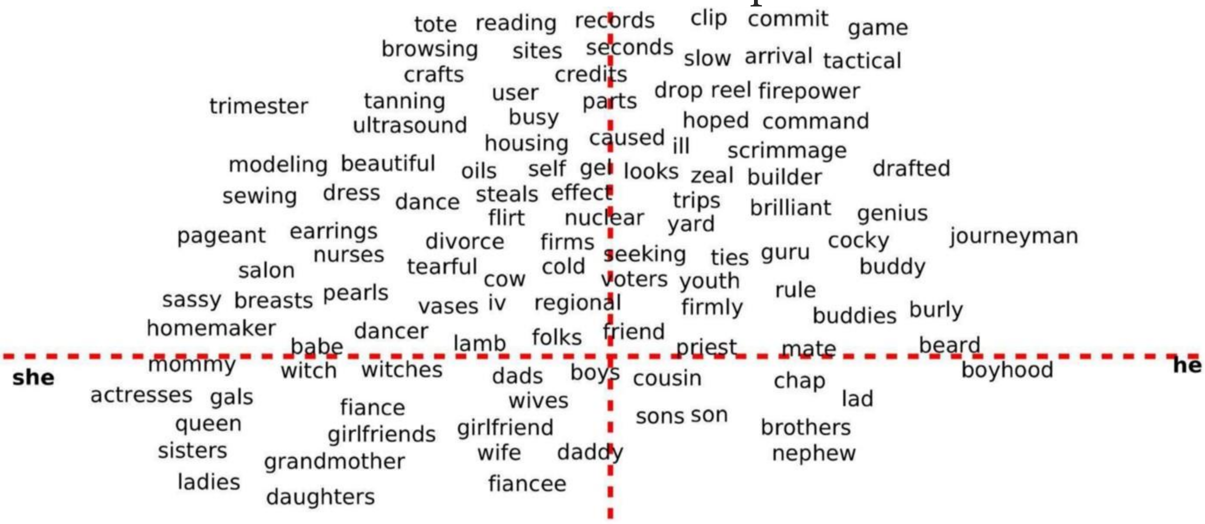
Skip-Gram: NLP context words prediction algorithm

NLP is a field of Artificial Intelligence in which we try to process human language as text or speech to make computers similar to humans. Humans have a large amount of data written in a very unorganized format. So, it's difficult for any machine to find meaning from raw text.

To make a machine learn from the raw text we need to transform this data into a vector format which then can easily be processed by our computers. This transformation of raw text into a vector format is known as word representation.

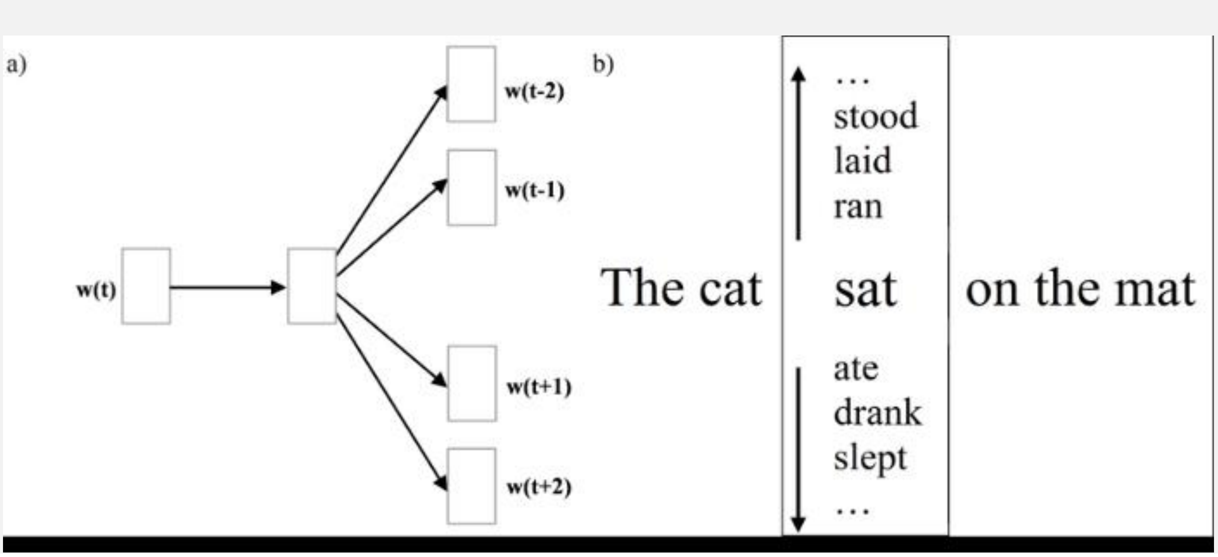


word embeddings

Word representation represents the word in vector space so that if the word vectors are close to one another means that those words are related to one other. In the given image, we can see various words which are mostly seen with a female are clustered on the left side while the words associated with males are clustered at the right side. So, if we pass the word like earrings the computer will relate it to the female gender which is logically correct.

As the vocabulary of any language is large and cannot be labeled by human and hence we require unsupervised learning techniques that can learn the context of any word on its own. Skip-gram is one of the unsupervised learning techniques used to find the most related words for a given word.

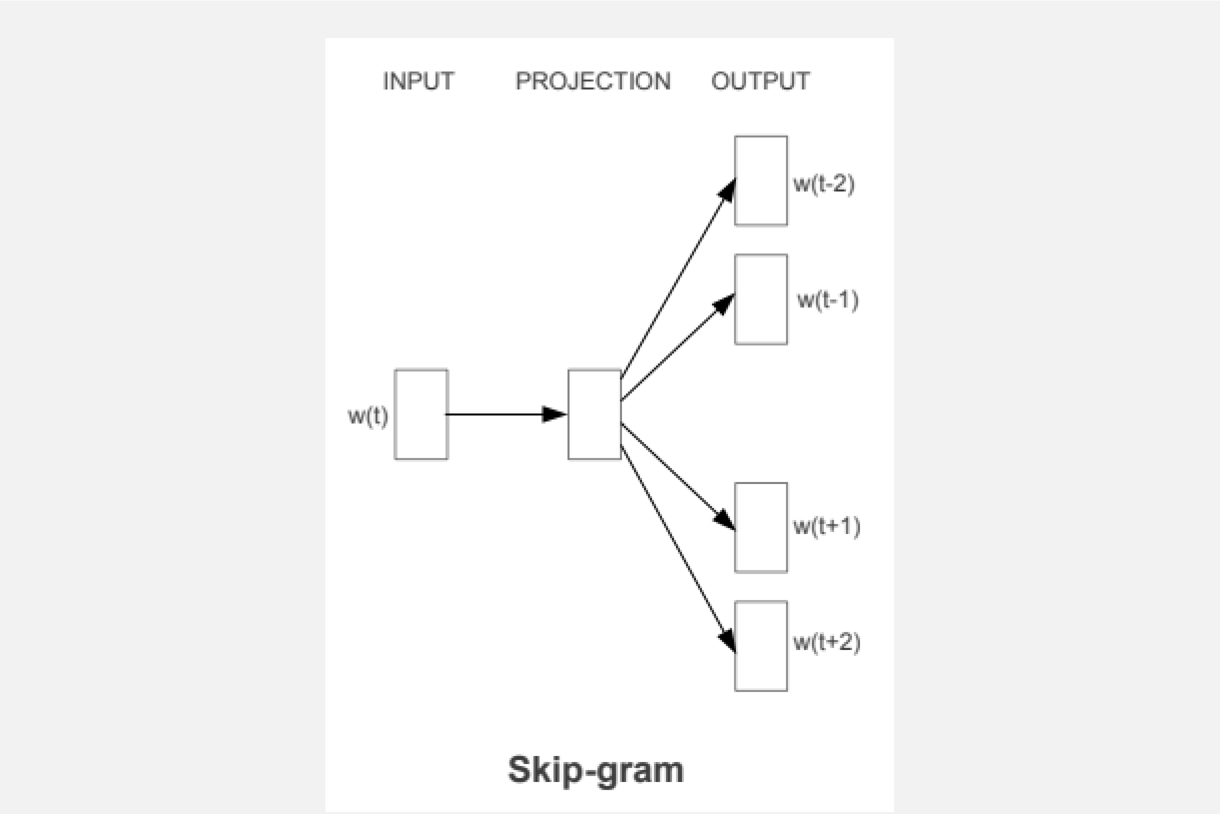
Skip-gram is used to predict the context word for a given target word. It's reverse of CBOW algorithm. Here, target word is input while context words are output. As there is more than one context word to be predicted which makes this problem difficult.



skip-gram example

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` . We do not predict common or stop words such as `the` .

Architecture



The Skip-gram model architecture (Source: https://arxiv.org/pdf/1301.3781.pdf Mikolov el al.)

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.

Variables we'll be using

1. The dictionary of unique words present in our dataset or text. This dictionary is known as vocabulary and is known words to the system. Vocabulary is represented by 'V'.

2. N is the number of neurons present in the hidden layer.

3. The window size is the maximum context location at which the words need to be predicted. The window size is denoted by c. For example, in the given architecture image the window size is 2, therefore, we will be predicting the words at context location (t-2), (t-1), (t+1) and (t+2).

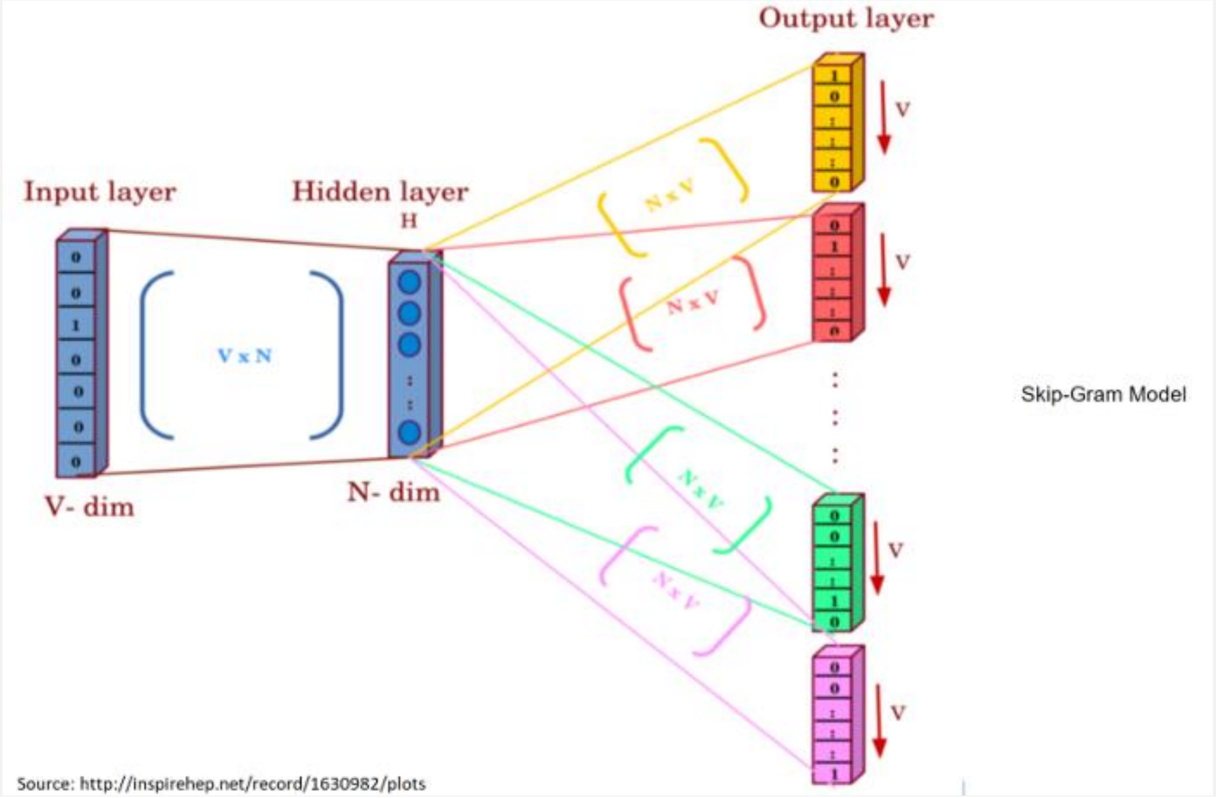
4. Context window is the number of words to be predicted which can occur in the range of the given word. The value of a context window is double the window size that is 2*c and is represented by k. For the given image the value of the context window is 4.

5. The dimension of an input vector is equal to |V|. Each word is encoded using one hot encoding.

6. The weight matrix for the hidden layer(W) is of dimension [|V|, N]. || is the modulus function which returns the size of an array.

7. The output vector of the hidden layer is H[N].

8. The weight matrix between the hidden and the output layer(W’) is of dimension [N,|V|].
9. The dot product between W’ and H gives us an output vector U[|v|].

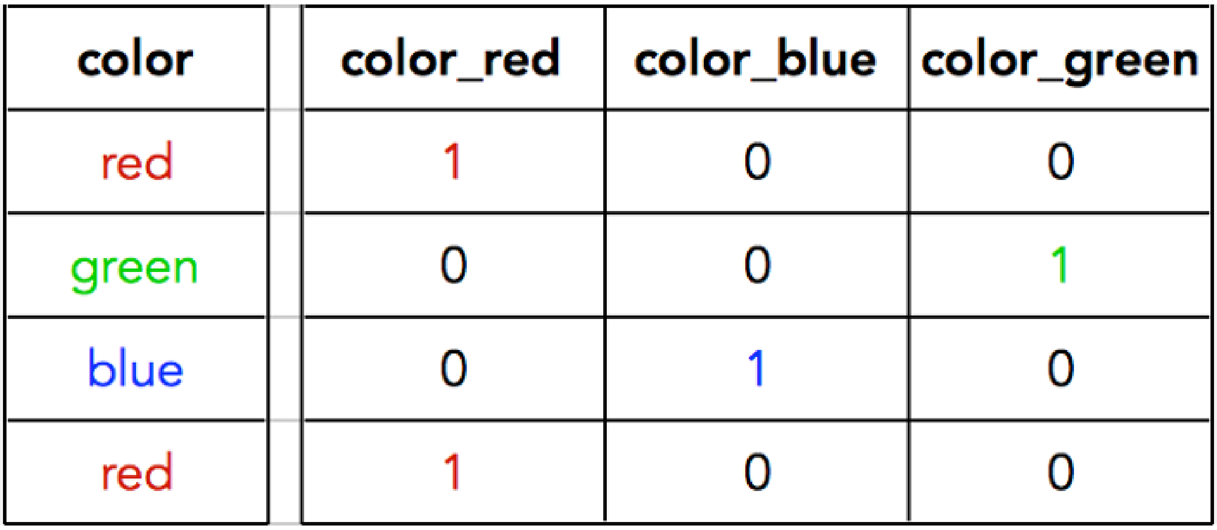


Source: http://inspirehep.net/record/1630982/plots

N = context window

Working steps

1. The words are converted into a vector using one hot encoding. The dimension of these vectors is [1,|v|].



color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

one hot encoding

2. The word w(t) is passed to the hidden layer from |v| neurons.

3. Hidden layer performs the dot product between weight vector W[|v|, N] and the input vector w(t). In this, we can conclude that the (t)th row of W[|v|, N] will be the output(H[1, N]).

4. Remember there is no activation function used at the hidden layer so the H[1,k]will be passed directly to the output layer.

5. Output layer will apply dot product between H[1, N] and W'[N, |v|] and will give us the vector U.

6. Now, to find the probability of each vector we'll use the softmax function. As each iteration gives output vector U which is of one hot encoding type.

7. The word with the highest probability is the result and if the predicted word for a given context position is wrong then we'll use backpropagation to modify our weight vectors W and W'.

This steps will be executed for each word w(t) present in vocabulary. And each word w(t) will be passed k times. So, we can see that forward propagation will be processed |v|*k times in each epoch.

Probability function

$$p(w_{c,j} = w_{O,c}|w_I) = \frac{\exp u_{c,j}}{\sum_{j'=1}^{V} \exp u_{j'}} \tag{2}$$

softmax probability

w(c,j) is the `j`th word predicted on the `c`th context position; w(O,c) is the actual word present on the `c`th context position; w(I) is the only input word; and u(c,j) is the `j`th value in the U vector when predicting the word for `c`th context position.

Loss function

$$\begin{aligned}\mathcal{L} &= -\log \mathbb{P}(w_{c,1}, w_{c,2}, \ldots, w_{c,C}|w_o) = -\log \prod_{c=1}^{C} \mathbb{P}(w_{c,i}|w_o) \\ &= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j^*})}{\sum_{j=1}^{V} \exp(u_{c,j})} = -\sum_{c=1}^{C} u_{c,j^*} + \sum_{c=1}^{C} \log \sum_{j=1}^{V} \exp(u_{c,j})\end{aligned}$$

Loss function

As we want to maximize the probability of predicting w(c,j) on the `c` th context position we can represent the loss function `L` .

Advantages

1. It is unsupervised learning hence can work on any raw text given.

2. It requires less memory comparing with other words to vector representations.

3. It requires two weight matrix of dimension [N, |v|] each instead of [|v|, |v|]. And usually, N is around 300 while |v| is in millions. So, we can see the advantage of using this algorithm.

Disadvantages

1. Finding the best value for `N` and `C` is difficult.

2. Softmax function is computationally expensive.

3. The time required for training this algorithm is high.