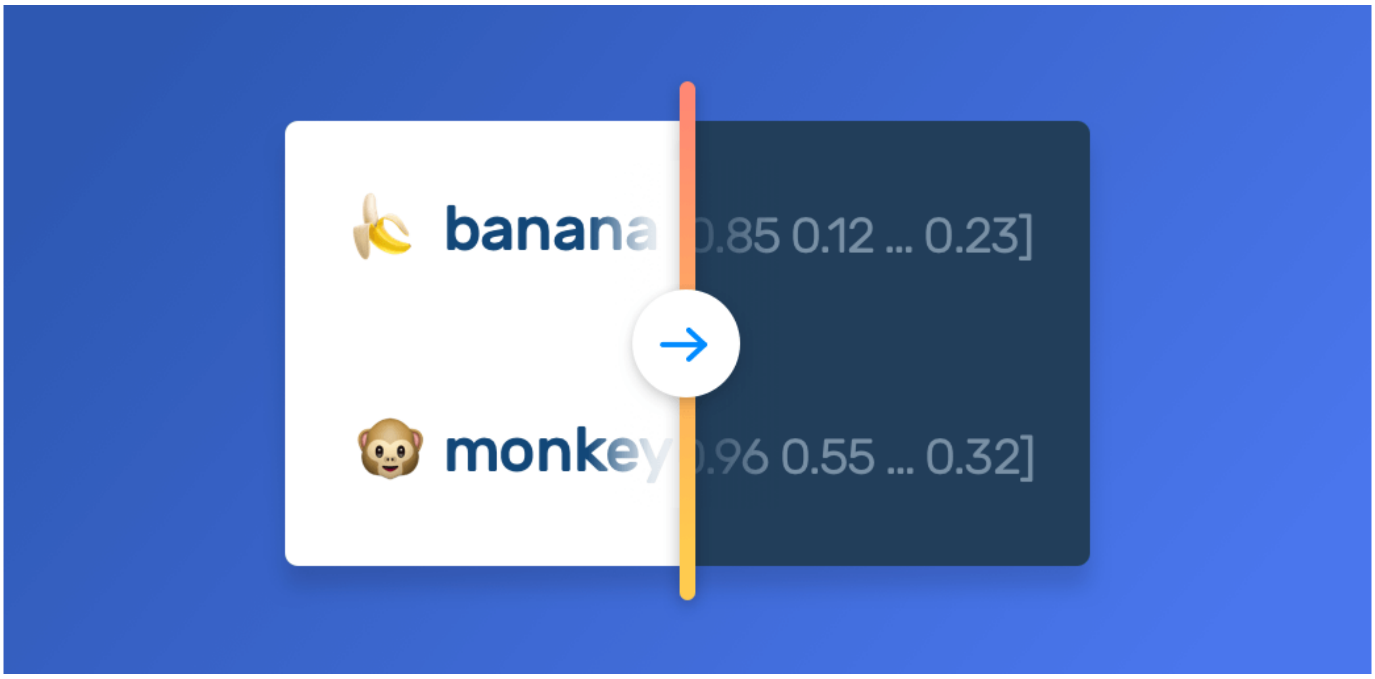
Assignment 3: Converting Words into vectors (Advance techniques)



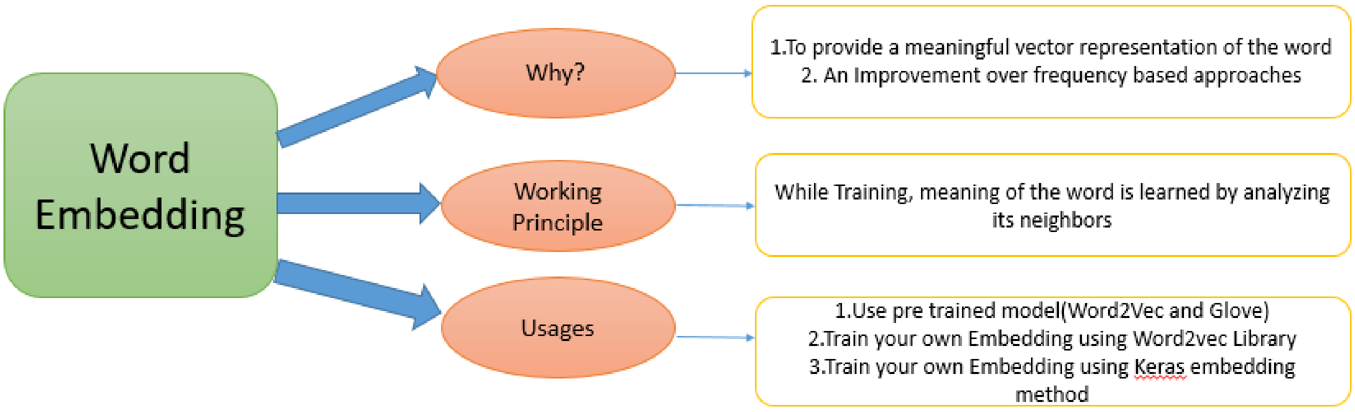
Topics Covered in this Assignment:

1. Words Embedding
2. Gensim
3. Word2Vec

Welcome to the Assignment 3 of the series of NLP!!
This chapter will dive deep into converting words into vectors. So before beginning this section, do revise the previous assignment as this is the continuation of the same.

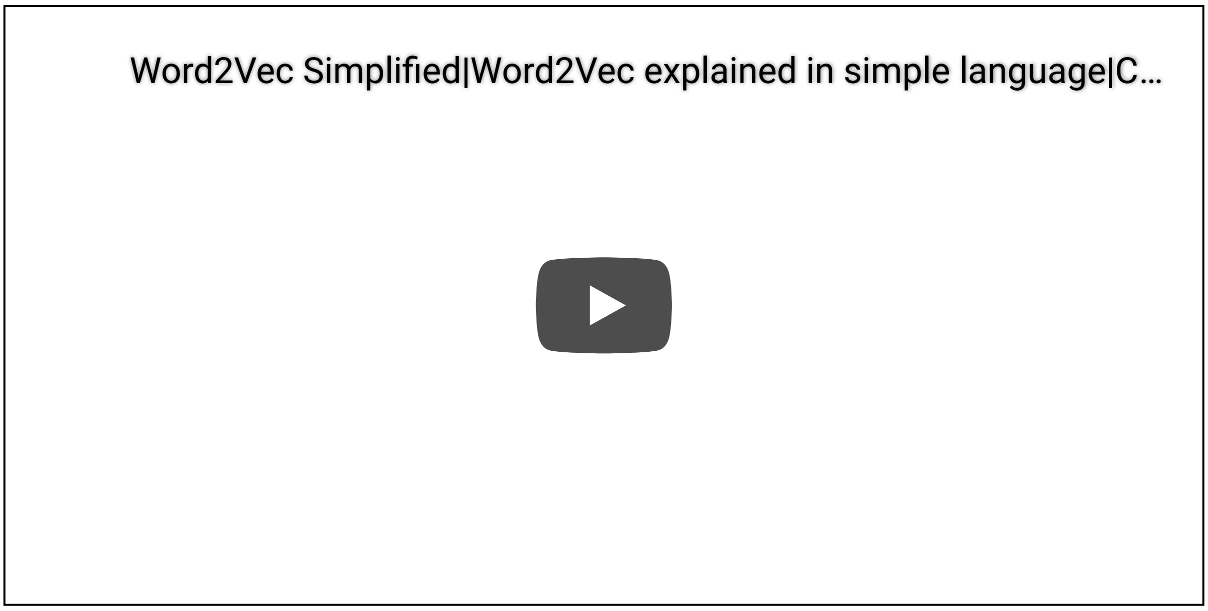
1.Words Embedding

Word Embedding is a language modeling technique used for mapping words to vectors of real numbers. It represents words or phrases in vector space with several dimensions. Word embeddings can be generated using various methods like neural networks, co-occurrence matrix, probabilistic models, etc.


Word Embedding
Why?
Working Principle
Usages
1.To provide a meaningful vector representation of the word
2. An Improvement over frequency based approaches
While Training, meaning of the word is learned by analyzing its neighbors
1.Use pre trained model(Word2Vec and Glove)
2.Train your own Embedding using Word2vec Library
3.Train your own Embedding using Keras embedding method

Reference: https://en.wikipedia.org/wiki/Word_embedding

```
#### Refer Video
from IPython.display import YouTubeVideo
YouTubeVideo('lEzzgLh_SFA', width=600, height=300)
```


Word2Vec Simplified|Word2Vec explained in simple language|C…

Gensim

"Generate Similar" is a popular open source natural language processing library used for unsupervised topic modeling. It is designed to extract semantic topics from documents. It can handle large text collections. Hence it makes it different from other machine learning software packages which target memory processing. Gensim also provides efficient multicore implementations for various algorithms to increase processing speed. It provides more convenient facilities for text processing than other packages

Corpus: A collection of text documents.
Vector: Form of representing text.
Model: Algorithm used to generate representation of data.
Topic Modelling: It is an information mining tool which is used to extract semantic topics from documents.
Topic: A repeating group of words frequently occurring together.

GENSIM
topic modelling for humans

Refernece:https://en.wikipedia.org/wiki/Gensim

Word2Vec

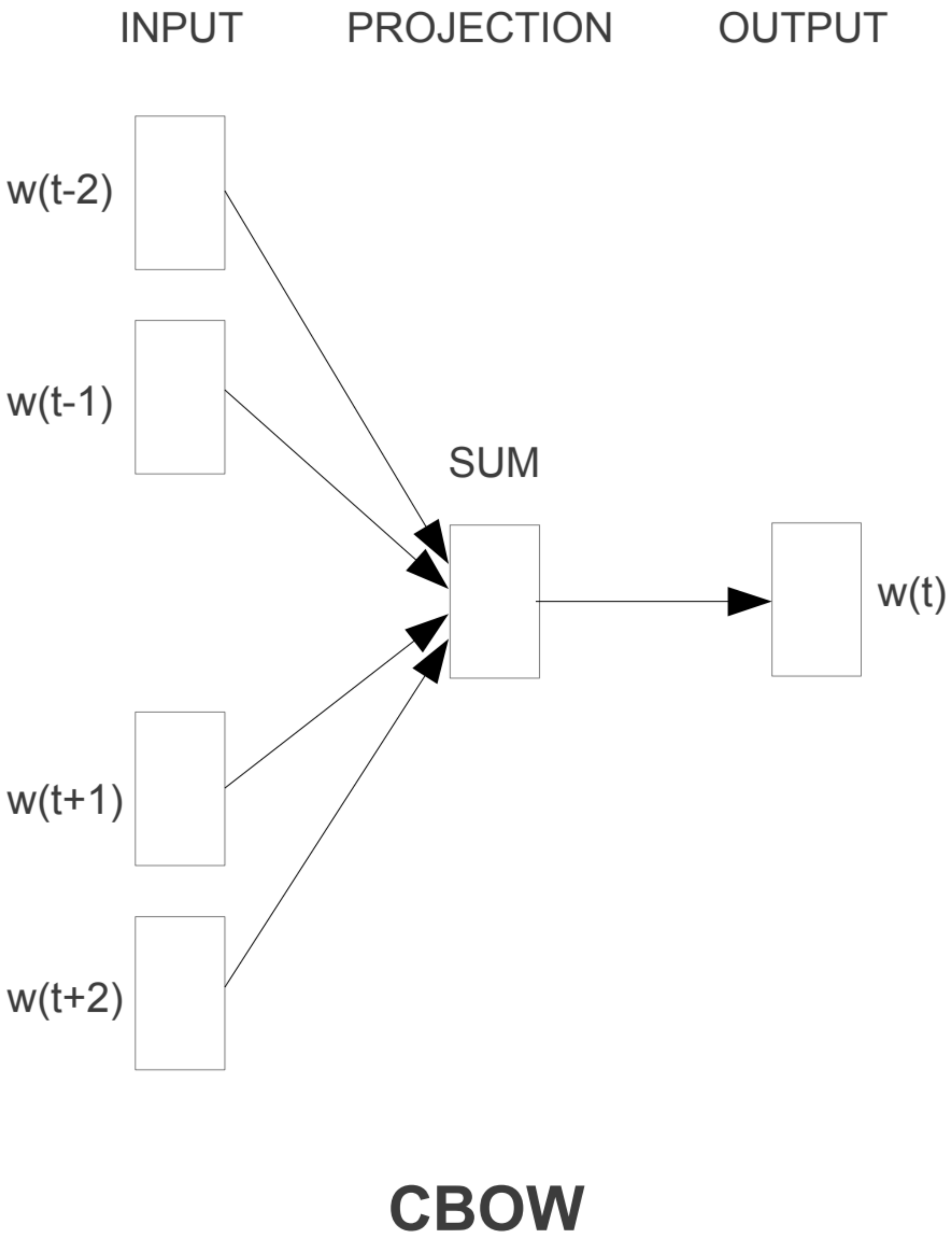
Word2Vec consists of models for generating word embedding. These models are shallow two layer neural networks having one input layer, one hidden layer and one output layer.

It has 2 types

1. CBOW (Continuous Bag of Words)
2. Skip Gram

CBOW (Continuous Bag of Words)

CBOW model predicts the current word given context words within specific window. The input layer contains the context words and the output layer contains the current word. The hidden layer contains the number of dimensions in which we want to represent current word present at the output layer.

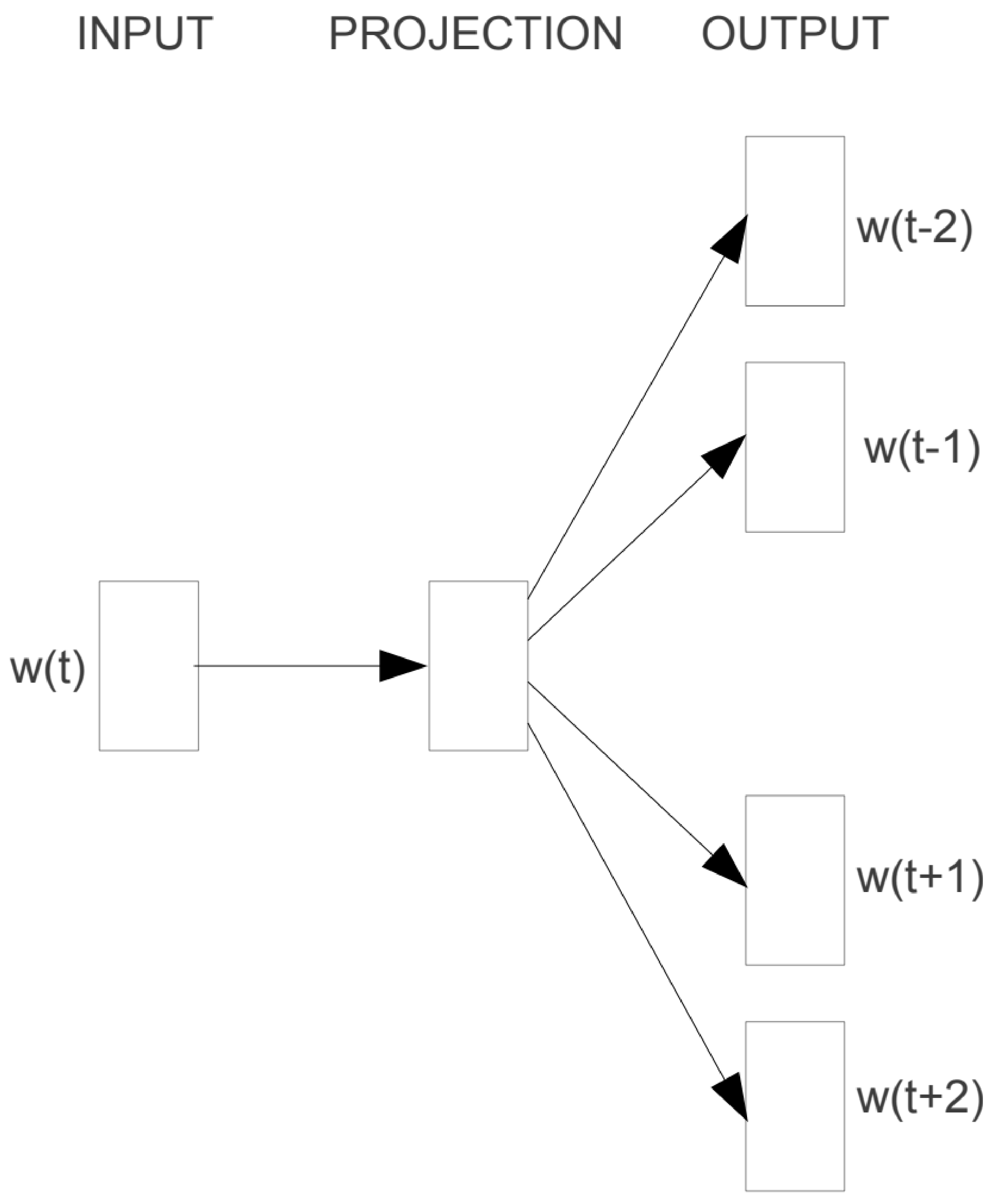


```
YouTubeVideo('uskth3b6H_A', width=600, height=300)
```

Word2vec: Continuous bag-of-words architecture Part-1

Skip Gram

Skip gram predicts the surrounding context words within specific window given current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer.

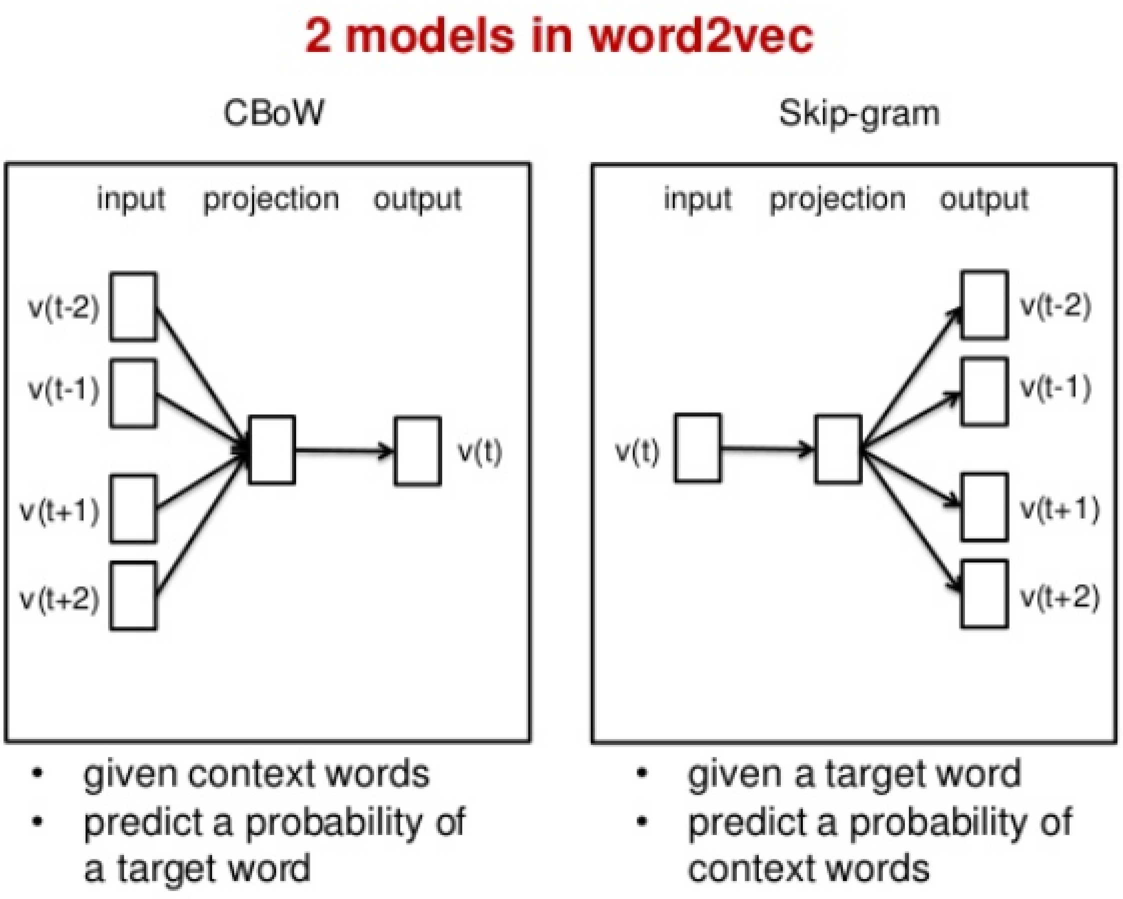


Skip-gram

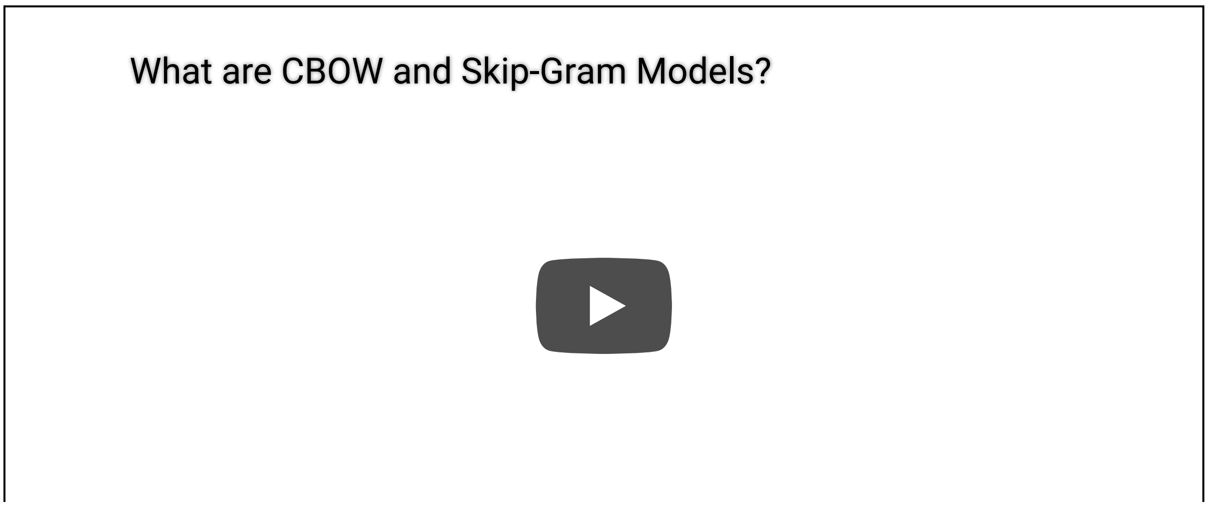
Reference: https://en.wikipedia.org/wiki/Word2vec

```
YouTubeVideo('Kfn0keI5TwQ', width=600, height=300)
```

Diiference between CBOW and SKIP-GRAM



```
YouTubeVideo('81kI-b5iPDE', width=600, height=300)
```



Let's start implementing vectorization of words using Word2Vec from gensim models. For that we first import nltk and then download punkt and stopwords from nltk.

```
#importing nltk and download punkt and stopwords
```

```
     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Unzipping tokenizers/punkt.zip.
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True
```

Now we import the Word2Vec library from gensim.models and stopwords from nltk.corpus. Re is also imported which will be used for preprocesisng the data.

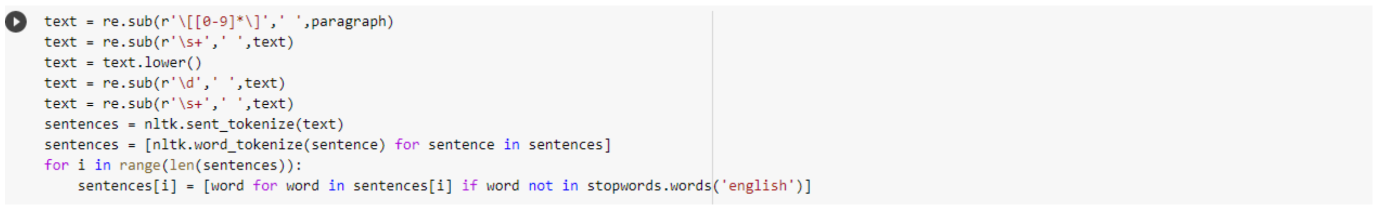
```
#import word2vec from gensim.models,import stopwords from nltk.corpus,import re
```

Below is the sample data that we will be using for vectorization. The words from this data will be converted into vectors. You may also use your own data or any random data as long as you follow the correct procedure.

```
paragraph = """I have three visions for India. In 3000 years of our history, people from a
               the world have come and invaded us, captured our lands, conquered our minds
               From Alexander onwards, the Greeks, the Turks, the Moguls, the Portuguese,
               the French, the Dutch, all of them came and looted us, took over what was o
               Yet we have not done this to any other nation. We have not conquered anyone
               We have not grabbed their land, their culture,
               their history and tried to enforce our way of life on them.
               Why? Because we respect the freedom of others.That is why my
               first vision is that of freedom. I believe that India got its first vision
               this in 1857, when we started the War of Independence. It is this freedom t
               we must protect and nurture and build on. If we are not free, no one will r
               My second vision for India’s development. For fifty years we have been a de
               It is time we see ourselves as a developed nation. We are among the top 5 n
               in terms of GDP. We have a 10 percent growth rate in most areas. Our povert
```

```
                     Our achievements are being globally recognised today. Yet we lack the self-
                     see ourselves as a developed nation, self-reliant and self-assured. Isn’t t
                     I have a third vision. India must stand up to the world. Because I believe
                     stands up to the world, no one will respect us. Only strength respects stre
                     strong not only as a military power but also as an economic power. Both mus
                     My good fortune was to have worked with three great minds. Dr. Vikram Sarab
                     space, Professor Satish Dhawan, who succeeded him and Dr. Brahm Prakash, fa
                     I was lucky to have worked with all three of them closely and consider this
                     I see four milestones in my career"""
```

Next step is to preprocess the above given sample data. The data is needed to be preprocessed to remove any extra whitespaces, unnecessary digits so that the data can be properly vectorized and extra things don't occupy necessary space. Also, preprocessing involves tokenization of the data as vectorization requires sentences to be converted into words. Stopwords removal also takes place to discard stopwords that are not necessary in sentences and give more importance to significant words in the data.



```
text = re.sub(r'\[[0-9]*\]',' ',paragraph)
text = re.sub(r'\s+',' ',text)
text = text.lower()
text = re.sub(r'\d',' ',text)
text = re.sub(r'\s+',' ',text)
sentences = nltk.sent_tokenize(text)
sentences = [nltk.word_tokenize(sentence) for sentence in sentences]
for i in range(len(sentences)):
    sentences[i] = [word for word in sentences[i] if word not in stopwords.words('english')]
```

We will be using regular expressions for preprocessing the data. Digits in the data are replaced with whitespaces and existing one or more whitespaces are replaced with a single whitespace. Know more about Regular Expressions: https://docs.python.org/3/library/re.html.

The data is converted into lowercase for ease in vectorization because in the computer's perspective, lowercase and uppercase characters are different(i.e. A and a are completely different). The paragraph is tokenized into sentences and also sentences are tokenized into words. English stopwords are removed from the processed data.

See the above image and write the code in below cell.

```
# Preprocessing the data
# we are preprocessing the data here we are replacing the data with space
```

Let's check the preprocessed data which is stored in the sentences variable. We print the second processed sentence by indexing it with '1'. You can print any other sentence by changing the index.

```
sentences[1]
```

```
     ['years',
      'history',
      ',',
      'people',
      'world',
      'come',
      'invaded',
      'us',
      ',',
      'captured',
      'lands',
      ',',
      'conquered',
      'minds',
      '.']
```

```
#print any sentence index as above
```

We can see that the data is now processed and all the extra things are removed including stop words. After the data is preprocessed, we will vectorize the data using Word2Vec. We are vectorizing the data on word2vec with min_count=1. min_count parameter ignores all words with total frequency lower than the given value.

```
# Training the Word2Vec model
```

Here we have trained data on Word2Vec with min_count=1

Now we can make model vocabulary using the wv object. This object essentially contains the mapping between words and embeddings. After training, it can be used directly to query those embeddings in various ways. The vocab function creates the vocabulary of the model and this vocabulary is stored in the words variable.

```
#Creating words using vocab
```

Now we find a word vector for verifying that our model is performing well. We simply pass a word into the model through the wv object(which contains mappings between words and its embeddings). We also check the dimensions of the produced vector using the shape function.

```
# Finding Word Vectors

#print vector

#printing dimensions of vector using .shape.
```

```
     [-3.5179397e-03 -3.4589237e-03 -2.7647335e-03 -1.8160340e-03
```

```
 -6.8303006e-04  9.3852001e-04  2.0732521e-03 -3.3545583e-03
 -2.3874394e-03  1.0649309e-03  3.6924358e-03 -1.8903443e-04
 -1.5188340e-03 -3.3402075e-03  2.6749428e-03 -3.5984409e-03
 -4.3519607e-04 -4.6389275e-03  2.6145631e-03  3.7788785e-05
 -4.9366630e-03 -1.6103391e-04 -6.3322007e-04  3.8645919e-03
  4.5789294e-03  2.3744109e-03  2.0943447e-04 -3.2039303e-03
 -4.3171537e-03  1.0167706e-03  3.8016431e-03  4.7078794e-03
  5.3632061e-04 -3.2078689e-03 -2.7323847e-03  2.4435124e-03
  1.0301910e-03 -5.6927645e-04 -4.5142956e-03  1.3771259e-03
 -4.8323791e-03  4.7151307e-03 -1.0395538e-03 -4.2740465e-03
  3.3769265e-03 -4.5292635e-04 -3.8562121e-05 -3.1071121e-03
  3.2217070e-03  4.6482431e-03 -5.6062109e-04 -1.6434408e-03
  2.2370180e-03  2.2035597e-03  1.2311791e-03 -2.6353446e-03
  2.5288074e-03 -3.9153011e-03  1.0606211e-03  4.1741366e-03
 -4.6601873e-03 -3.2402922e-03  6.2047725e-04 -1.0725461e-03
  2.1774473e-03  2.7224142e-03 -2.7022937e-03 -1.6388030e-03
  1.1355574e-03 -4.7190220e-04  1.2884039e-03 -4.9151839e-03
 -5.4327596e-05 -2.4248043e-04  3.7232062e-03  2.0909810e-03
 -3.0244836e-03  1.9090609e-03  1.3588530e-03  2.6150297e-03
  1.2196387e-03 -1.6418603e-03 -2.8449581e-03  9.3908416e-04
 -2.0692802e-03 -4.1547050e-03 -4.7332188e-03 -4.9970825e-03
  1.2269685e-03  2.9223498e-03  6.0328777e-04  1.5559405e-03
  3.6185232e-03 -4.0755267e-03 -2.2776488e-03 -2.4253055e-03
  3.1937618e-04  4.3549608e-05 -1.0948826e-03 -3.3010321e-03]
dimensions of the vector: (100,)
```

We see that the vector of the 'war' word is displayed. You can try it for any other word from the data. We can observe that for all the words the shape of the vector is 100. This is because while creating the Word2Vec object we had set the vector_size parameter to default. The default value of vector_size is 100. It can be changed according to different use cases. Understand about it more here: https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html

Now, we try to find the most similar words for the given word. The most_similar function is used to find the similar words.

```
#trying to find the Most similar words
#similar = model.wv.most_similar('vikram')
```

We have stored the most similar words in Similar variable. After storing the variable we will print the values.

```
#printing similar values using for loop
```

```
('.', 0.23603850603103638)
('nations', 0.23229455947875977)
('greeks', 0.21883772313594818)
('dutch', 0.20198717713356018)
('tried', 0.1898503452539444)
('independence', 0.18289512395858765)
('space', 0.18254604935646057)
```

```
('strong', 0.18048027157783508)
('prakash', 0.17268215119838715)
('today', 0.17113453149795532)
```

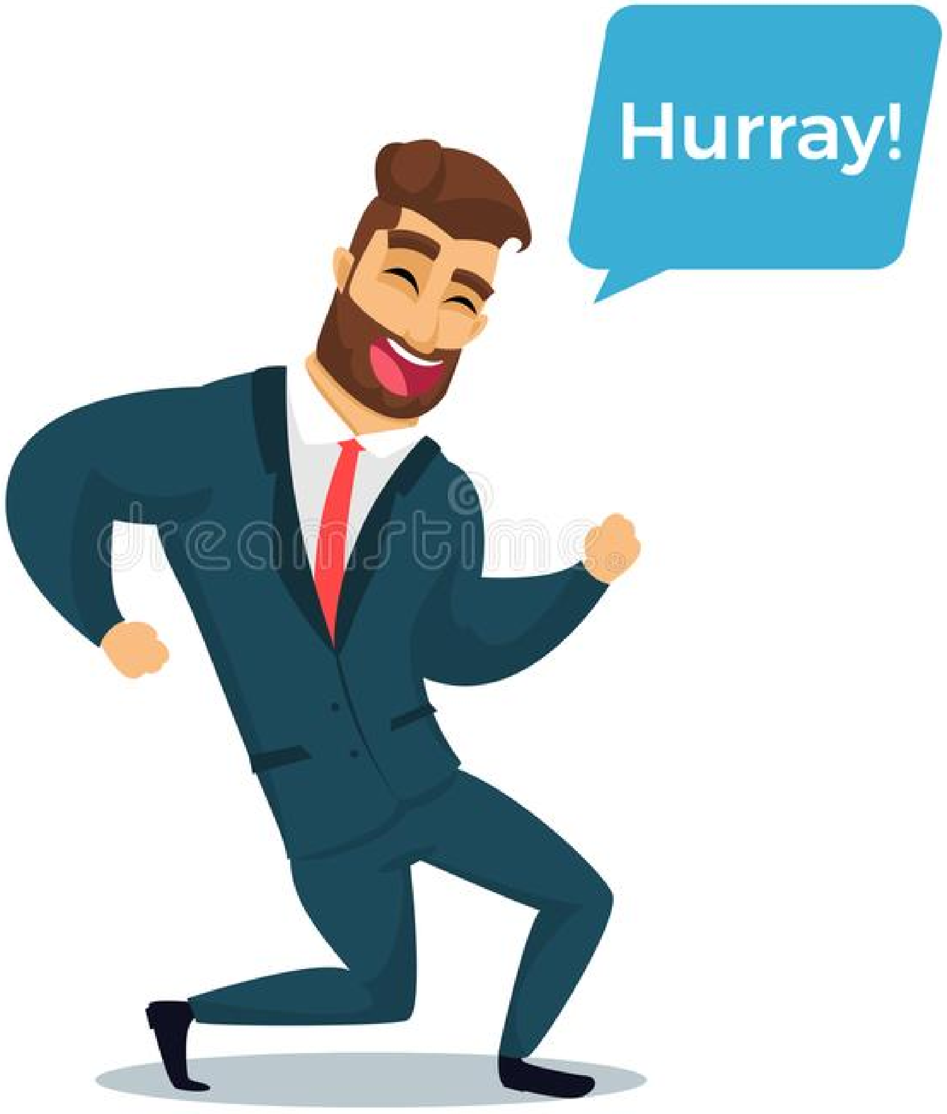
We found that most similar word for vikram based on the data which have trained on it.

In this assignment, we were able to code for Word2Vec using gensim library i.e an advance version of word embedding. -_-

Great job!! You have come to the end of this assignment. Treat yourself for this :))

Do fill this [feedback form]()

You may head on to the next assignment.


Hurray!