Hi, this is Sandeep, working as Software Engineer involved in Python, NLP, Machine Learning, Deep Leaning with TensorFlow 2.0 etc....

In this article I am exploring into I am exploring into Word Embeddings in Natural Language Processing.

## Word Embeddings:

Word embedding is a learned representation for text where words that have the same meaning have a similar representation.

What is pretrained Embedding.

---

*Pretrained Embedding are the embeddings learned in one task and that are used for solving similar kind of tasks.*

---

It is a form of "**Transfer Learning**".

**Why do we need Pretrained word Embeddings?**

1. Pretrained Word Embedding are capture semantic and Syntactics meaning of word which learned on Large Datasets
2. They can boost the performance of NLP model.

Cool, Why don't we use or create our own Word Embeddings.

1. Sparsity of training Data.
2. Large able number of Trainable Parameters.

**Different Types of word Embeddings.**

- ⇨ *Google Word2Vec*
- ⇨ *Glove from stand ford.*

**Google Word2Vec:**

Google Word2Vec is most popular and trained on Google News Dataset.

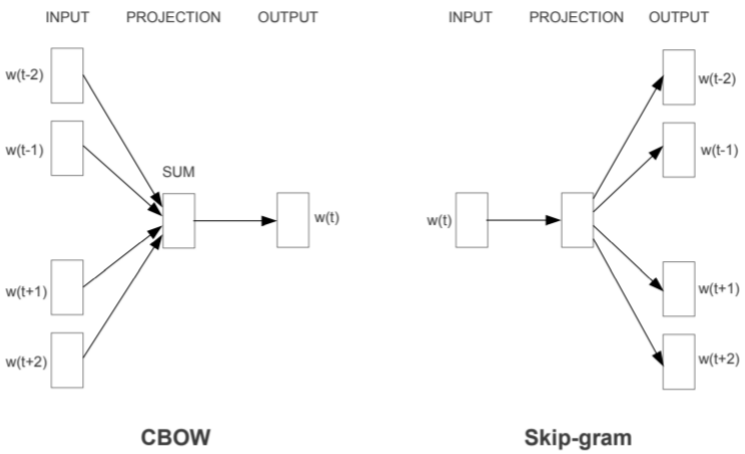
Architecture of Google WZV is Simple Feed Forward Neural Network. with just one hidden layer.

**Continuous Bag of word:**

Learn the focus word by given the Neighboring words.

**Skip gram model:**

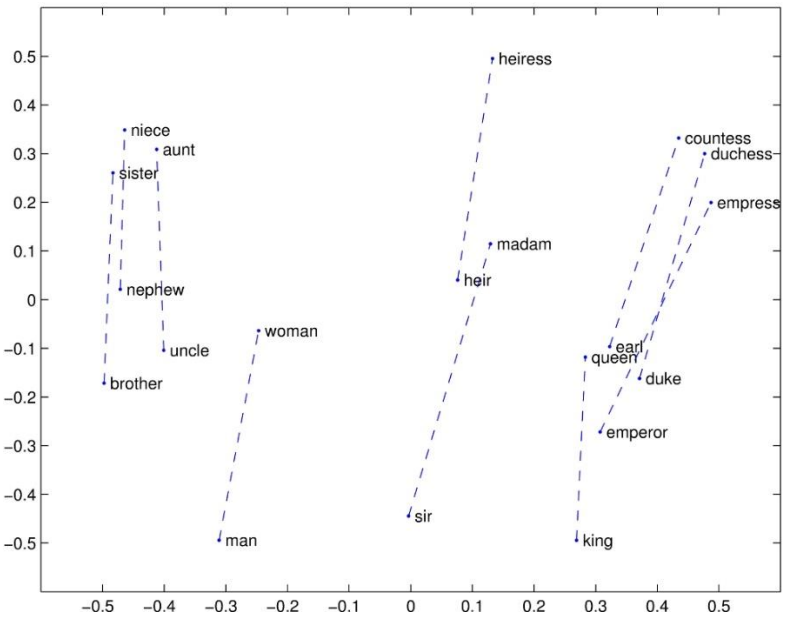
Learn the Neighboring words gives the focus word.




**Glove Embedding: -**

**Glove Embedding is to derive the relationship b/w the words from Global Statistics.**

**Simplest Way to look this in a "Co-Occurrence Matrix".**




https://sandeepkumar16nlp.github.io/sandeepmyportfolion.github.io/

**Below example code snippet is to understand how Word2Vec find the similarity between words.**

```
In [15]: #### Word2Vec WIth Spacy

In [*]: import spacy

In [17]: nlp = spacy.load('en_core_web_lg')

In [18]: nlp(u'Hello Sandeep welcome to NLP').vector.shape

Out[18]: (300,)

In [19]: nlp(u'Sandeep').vector.shape

Out[19]: (300,)

In [20]: ###  Identify similar vectors

In [21]: tokens = nlp(u'Man woman king lion')
         token2 = nlp(u'love like hate')

In [22]: for tk1 in token2:
             for tk2 in token2:
                 print(tk1.text,tk2.text,tk1.similarity(tk2))

love love 1.0
love like 0.65790397
love hate 0.6393099
like love 0.65790397
like like 1.0
like hate 0.6574652
hate love 0.6393099
hate like 0.6574652
hate hate 1.0
```