# What is a Word Embedding?

Word embeddings is a technique where individual words are transformed into a numerical representation of the word (a vector). Where each word is mapped to one vector, this vector is then learned in a way which resembles a neural network. The vectors try to capture various characteristics of that word with regard to the overall text. These characteristics can include the semantic relationship of the word, definitions, context, etc. With these numerical representations, you can do many things like identify similarity or dissimilarity between words.

Limitations : However, there are multiple limitations of simple embeddings such as this, as they do not capture characteristics of the word, and they can be quite large depending on the size of the corpus.

# Word2Vec

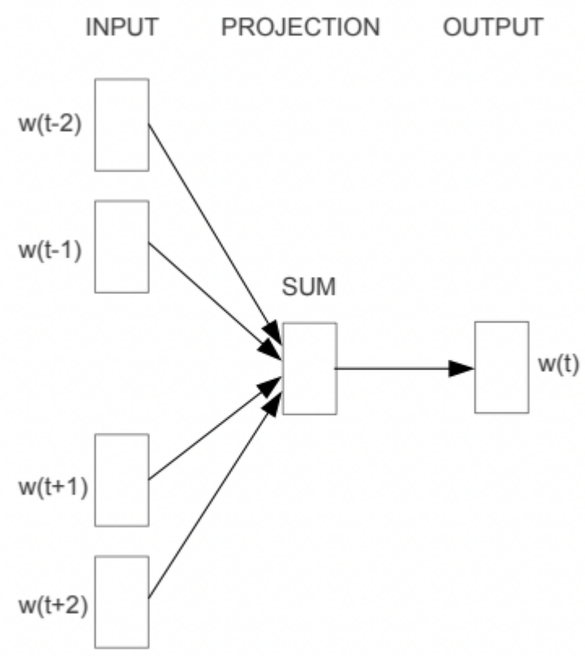
# Word2Vec Architecture

The effectiveness of Word2Vec comes from its ability to group together vectors of similar words. Given a large enough dataset, Word2Vec can make strong estimates about a word's meaning based on their occurrences in the text. These estimates yield word associations with other words in the corpus. For example, words like "King" and "Queen" would be very similar to one another. When conducting algebraic operations on word embeddings you can find a close approximation of word similarities. For example, the 2 dimensional embedding vector of "king" - the 2 dimensional embedding vector of "man" + the 2 dimensional embedding vector of "woman" yielded a vector which is very close to the embedding vector of "queen". Note, that the values below were chosen arbitrarily.

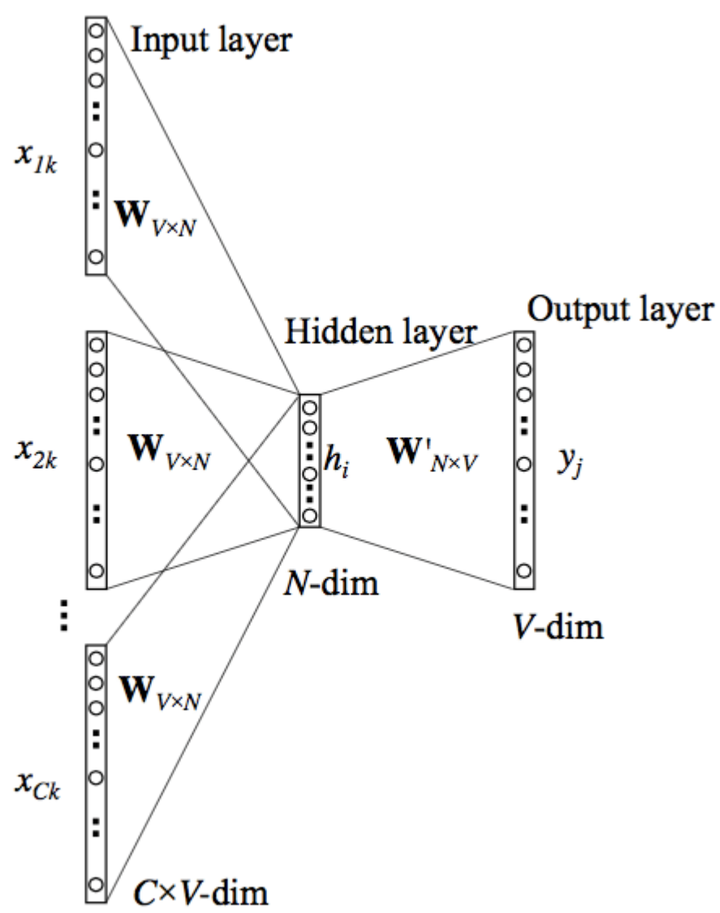
There are two main architectures which yield the success of word2vec.
1. CBOW architectures and
2. Skip Grams .

# CBOW (Continuous Bag of Words)

This architecture is very similar to a feed forward neural network.
This model architecture essentially tries to predict a target word from a list of context words. The intuition behind this model is quite simple: given a phrase "Have a great day" , we will choose our target word to be "a" and our context words to be ["have", "great", "day"]. What this model will do is take the distributed representations of the context words to try and predict the target word.



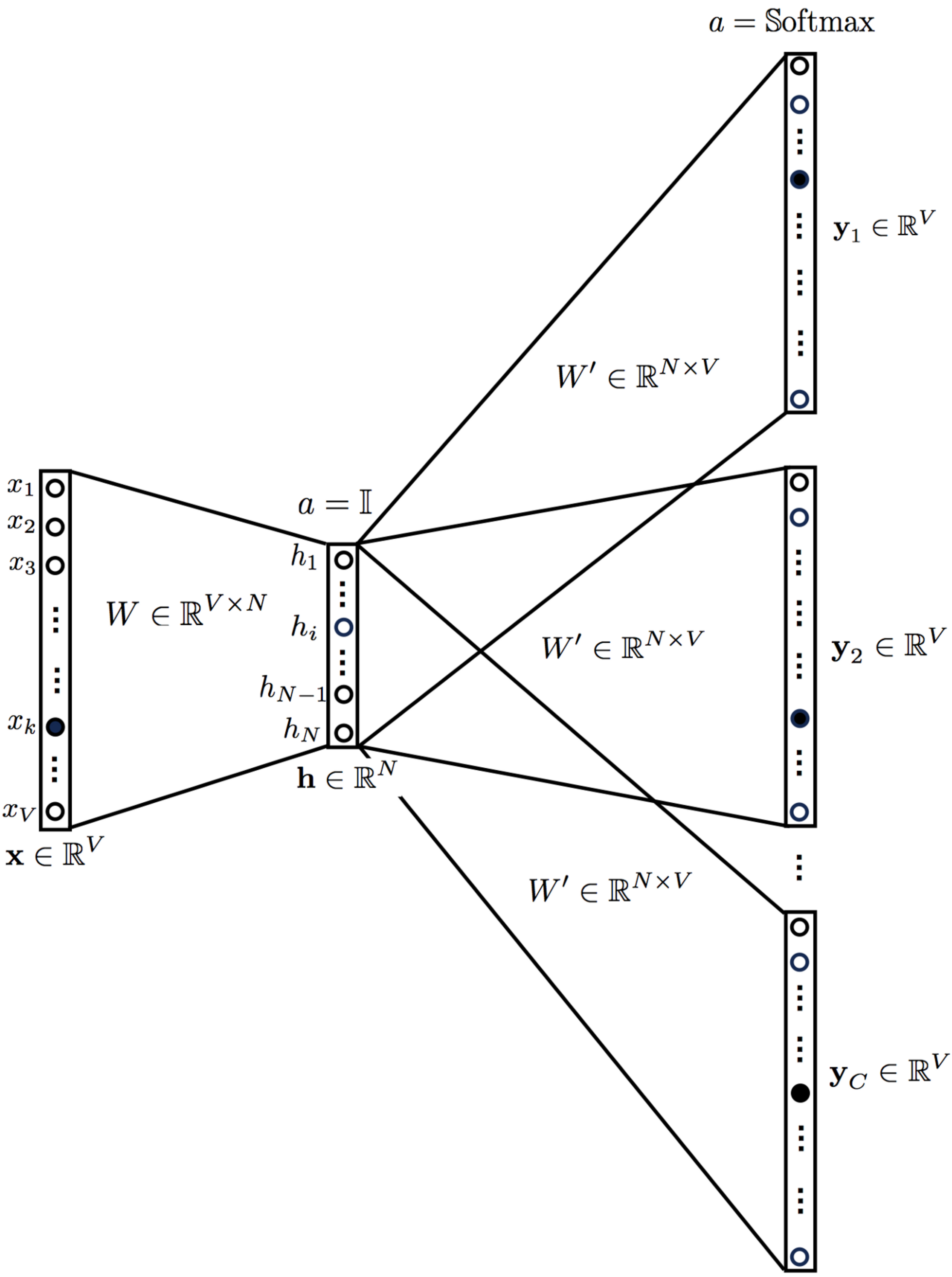

**CBOW**




```
the vocabulary size is V , and the hidden layer size is N. The units on adjacent layers are fully connected. The input is a one-hot encoded vector, which means for a given
input context word, only one out of V units, {x1, · · · , xV }, will be 1, and all other units
are 0. The weights between the input layer and the output layer can be represented by a
V × N matrix W. Each row of W is the N-dimension vector representation vw of the
associated word of the input layer.
```

# Skip-Gram Model

The skip-gram model is a simple neural network with one hidden layer trained in order to predict the probability of a given word being present when an input word is present. Intuitively, you can imagine the skip-gram model being the opposite of the CBOW model. In this architecture, it takes the current word as an input and tries to accurately predict the words before and after this current word. This model essentially tries to learn and predict the context words around the specified input word. Based on experiments assessing the accuracy of this model it was found that the prediction quality improves given a large range of word vectors, however it also increases the computational complexity. The process can be described visually as seen below.




The skip-gram model is introduced in Mikolov et al. Above figure 3 shows the skip-gram model. It is the opposite of the CBOW model. The target word is now at the input layer, and the context words are on the output layer.

In [ ]:

In [ ]:

## Gensim

> Gensim is an open source python library for natural language processing and it was developed and is maintained by the Czech natural language processing researcher Radim Řehůřek. Gensim library will enable us to develop word embeddings by training our own word2vec models on a custom corpus either with CBOW of skip-grams algorithms.

In [1]:
```
import gensim
```

In [2]:
```
from gensim.models import word2vec, KeyedVectors
```

In [3]:
```
import gensim.downloader as api
wv = api.load('word2vec-google-news-300')  # google news data set
vec_king = wv['king']
```

In [4]:
```
vec_king
```

Out[4]:
```
array([ 1.25976562e-01,  2.97851562e-02,  8.60595703e-03,  1.39648438e-01,
       -2.56347656e-02, -3.61328125e-02,  1.11816406e-01, -1.98242188e-01,
        5.12695312e-02,  3.63281250e-01, -2.42187500e-01, -3.02734375e-01,
       -1.77734375e-01, -2.49023438e-02, -1.67968750e-01, -1.69921875e-01,
        3.46679688e-02,  5.21850586e-03,  4.63867188e-02,  1.28906250e-01,
        1.36718750e-01,  1.12792969e-01,  5.95703125e-02,  1.36718750e-01,
        1.01074219e-01, -1.76757812e-01, -2.51953125e-01,  5.98144531e-02,
        3.41796875e-01, -3.11279297e-02,  1.04492188e-01,  6.17675781e-02,
        1.24511719e-01,  4.00390625e-01, -3.22265625e-01,  8.39843750e-02,
        3.90625000e-02,  5.85937500e-03,  7.03125000e-02,  1.72851562e-01,
        1.38671875e-01, -2.31445312e-01,  2.83203125e-01,  1.42578125e-01,
        3.41796875e-01, -2.39257812e-02, -1.09863281e-01,  3.32031250e-02,
       -5.46875000e-02,  1.53198242e-02, -1.62109375e-01,  1.58203125e-01,
       -2.59765625e-01,  2.01416016e-02, -1.63085938e-01,  1.35803223e-03,
       -1.44531250e-01, -5.68847656e-02,  4.29687500e-02, -2.46582031e-02,
        1.85546875e-01,  4.47265625e-01,  9.58251953e-03,  1.31835938e-01,
        9.86328125e-02, -1.85546875e-01, -1.00097656e-01, -1.33789062e-01,
       -1.25000000e-01,  2.83203125e-01,  1.23046875e-01,  5.32226562e-02,
       -1.77734375e-01,  8.59375000e-02, -2.18505859e-02,  2.05078125e-02,
       -1.39648438e-01,  2.51464844e-02,  1.38671875e-01, -1.05468750e-01,
        1.38671875e-01,  8.88671875e-02, -7.51953125e-02, -2.13623047e-02,
        1.72851562e-01,  4.63867188e-02, -2.65625000e-01,  8.91113281e-03,
        1.49414062e-01,  3.78417969e-02,  2.38281250e-01, -1.24511719e-01,
       -2.17773438e-01, -1.81640625e-01,  2.97851562e-02,  5.71289062e-02,
       -2.89306641e-02,  1.24511719e-02,  9.66796875e-02, -2.31445312e-01,
        5.81054688e-02,  6.68945312e-02,  7.08007812e-02, -3.08593750e-01,
       -2.14843750e-01,  1.45507812e-01, -4.27734375e-01, -9.39941406e-03,
        1.54296875e-01, -7.66601562e-02,  2.89062500e-01,  2.77343750e-01,
       -4.86373901e-04, -1.36718750e-01,  3.24218750e-01, -2.46093750e-01,
       -3.03649902e-03, -2.11914062e-01,  1.25000000e-01,  2.69531250e-01,
        2.04101562e-01,  8.25195312e-02, -2.01171875e-01, -1.60156250e-01,
       -3.78417969e-02, -1.20117188e-01,  1.15234375e-01, -4.10156250e-02,
       -3.95507812e-02, -8.98437500e-02,  6.34765625e-03,  2.03125000e-01,
        1.86523438e-01,  2.73437500e-01,  6.29882812e-02,  1.41601562e-01,
       -9.81445312e-02,  1.38671875e-01,  1.82617188e-01,  1.73828125e-01,
        1.73828125e-01, -2.37304688e-01,  1.78710938e-01,  6.34765625e-02,
        2.36328125e-01, -2.08984375e-01,  8.74023438e-02, -1.66015625e-01,
       -7.91015625e-02,  2.43164062e-01, -8.88671875e-02,  1.26953125e-01,
       -2.16796875e-01, -1.73828125e-01, -3.59375000e-01, -8.25195312e-02,
       -6.49414062e-02,  5.07812500e-02,  1.35742188e-01, -7.47070312e-02,
       -1.64062500e-01,  1.15356445e-02,  4.45312500e-01, -2.15820312e-01,
       -1.11328125e-01, -1.92382812e-01,  1.70898438e-01, -1.25000000e-01,
        2.65502930e-03,  1.92382812e-01, -1.74804688e-01,  1.39648438e-01,
        2.92968750e-01,  1.13281250e-01,  5.95703125e-02, -6.39648438e-02,
        9.96093750e-02, -2.72216797e-02,  1.96533203e-02,  4.27246094e-02,
       -2.46093750e-01,  6.39648438e-02, -2.25585938e-01, -1.68945312e-01,
        2.89916992e-03,  8.20312500e-02,  3.41796875e-01,  4.32128906e-02,
        1.32812500e-01,  1.42578125e-01,  7.61718750e-02,  5.98144531e-02,
       -1.19140625e-01,  2.74658203e-03, -6.29882812e-02, -2.72216797e-02,
       -4.82177734e-03, -8.20312500e-02, -2.49023438e-02, -4.00390625e-01,
       -1.06933594e-01,  4.24804688e-02,  7.76367188e-02, -1.16699219e-01,
        7.37304688e-02, -9.22851562e-02,  1.07910156e-01,  1.58203125e-01,
        4.24804688e-02,  1.26953125e-01,  3.61328125e-02,  2.67578125e-01,
       -1.01074219e-01, -3.02734375e-01, -5.76171875e-02,  5.05371094e-02,
        5.26428223e-04, -2.07031250e-01, -1.38671875e-01, -8.97216797e-03,
       -2.78320312e-02, -1.41601562e-01,  2.07031250e-01, -1.58203125e-01,
        1.27929688e-01,  1.49414062e-01, -2.24609375e-02, -8.44726562e-02,
        1.22558594e-01,  2.15820312e-01, -2.13867188e-01, -3.12500000e-01,
       -3.73046875e-01,  4.08935547e-03,  1.07421875e-01,  1.06933594e-01,
        7.32421875e-02,  8.97216797e-03, -3.88183594e-02, -1.29882812e-01,
        1.49414062e-01, -2.14843750e-01, -1.83868408e-03,  9.91210938e-02,
        1.57226562e-01, -1.14257812e-01, -2.05078125e-01,  9.91210938e-02,
        3.69140625e-01, -1.97265625e-01,  3.54003906e-02,  1.09375000e-01,
        1.31835938e-01,  1.66992188e-01,  2.35351562e-01,  1.04980469e-01,
       -4.96093750e-01, -1.64062500e-01, -1.56250000e-01, -5.22460938e-02,
        1.03027344e-01,  2.43164062e-01, -1.88476562e-01,  5.07812500e-02,
       -9.37500000e-02, -6.68945312e-02,  2.27050781e-02,  7.61718750e-02,
        2.89062500e-01,  3.10546875e-01, -5.37109375e-02,  2.28515625e-01,
        2.51464844e-02,  6.78710938e-02, -1.21093750e-01, -2.15820312e-01,
       -2.73437500e-01, -3.07617188e-02, -3.37890625e-01,  1.53320312e-01,
        2.33398438e-01, -2.08007812e-01,  3.73046875e-01,  8.20312500e-02,
        2.51953125e-01, -7.61718750e-02, -4.66308594e-02, -2.23388672e-02,
        2.99072266e-02, -5.93261719e-02, -4.66918945e-03, -2.44140625e-01,
       -2.09960938e-01, -2.87109375e-01, -4.54101562e-02, -1.77734375e-01,
       -2.79296875e-01, -8.59375000e-02,  9.13085938e-02,  2.51953125e-01],
      dtype=float32)
```

In [6]:
```
type(wv)
```

Out[6]:
```
gensim.models.keyedvectors.KeyedVectors
```

In [7]:
```
vec_king.shape
```

Out[7]:
```
(300,)
```

In [8]:
```
wv
```

Out[8]:
```
<gensim.models.keyedvectors.KeyedVectors at 0x2511f3e60a0>
```

In [ ]:
```
wv['man']
```

Out[8]:
```
array([ 0.32617188,  0.13085938,  0.03466797, -0.08300781,  0.08984375,
       -0.04125977, -0.19824219,  0.00689697,  0.14355469,  0.0019455 ,
        0.02880859, -0.25      , -0.08398438, -0.15136719, -0.10205078,
        0.04077148, -0.09765625,  0.05932617,  0.02978516, -0.10058594,
       -0.13085938,  0.001297  ,  0.02612305, -0.27148438,  0.06396484,
       -0.19140625, -0.078125  ,  0.25976562,  0.375     , -0.04541016,
        0.16210938,  0.13671875, -0.06396484, -0.02062988, -0.09667969,
        0.25390625,  0.24804688, -0.12695312,  0.07177734,  0.3203125 ,
        0.03149414, -0.03857422,  0.21191406, -0.00811768,  0.22265625,
       -0.13476562, -0.07617188,  0.01049805, -0.05175781,  0.03808594,
       -0.13378906,  0.125     ,  0.0559082 , -0.18261719,  0.08154297,
       -0.08447266, -0.07763672, -0.04345703,  0.08105469, -0.01092529,
        0.17480469,  0.30664062, -0.04321289, -0.01416016,  0.09082031,
       -0.00927734, -0.03442383, -0.11523438,  0.12451172, -0.0246582 ,
        0.08544922,  0.14355469, -0.27734375,  0.03662109, -0.11035156,
        0.13085938, -0.01721191, -0.08056641, -0.00708008, -0.02954102,
        0.30078125, -0.09033203,  0.03149414, -0.18652344, -0.11181641,
        0.10253906, -0.25976562, -0.02209473,  0.16796875, -0.05322266,
       -0.14550781, -0.01049805, -0.03039551, -0.03857422,  0.11523438,
       -0.0062561 , -0.13964844,  0.08007812,  0.06103516, -0.15332031,
       -0.11132812, -0.14160156,  0.19824219, -0.06933594,  0.29296875,
       -0.16015625,  0.20898438,  0.00041771,  0.01831055, -0.20214844,
        0.04760742,  0.05810547, -0.0123291 , -0.01989746, -0.00364685,
       -0.0135498 , -0.08251953, -0.03149414,  0.00717163,  0.20117188,
        0.08300781, -0.0480957 , -0.26367188, -0.09667969, -0.22558594,
       -0.09667969,  0.06494141, -0.02502441,  0.08496094,  0.03198242,
       -0.07568359, -0.25390625, -0.11669922, -0.01446533, -0.16015625,
       -0.00701904, -0.05712891,  0.02807617, -0.09179688,  0.25195312,
        0.24121094,  0.06640625,  0.12988281,  0.17089844, -0.13671875,
        0.1875    , -0.10009766, -0.04199219, -0.12011719,  0.00524902,
        0.15625   , -0.203125  , -0.07128906, -0.06103516,  0.01635742,
        0.18261719,  0.03588867, -0.04248047,  0.16796875, -0.15039062,
       -0.16992188,  0.01831055,  0.27734375, -0.01269531, -0.0390625 ,
       -0.15429688,  0.18457031, -0.07910156,  0.09033203, -0.02709961,
        0.08251953,  0.06738281, -0.16113281, -0.19628906, -0.15234375,
       -0.04711914,  0.04760742,  0.05908203, -0.16894531, -0.14941406,
        0.12988281,  0.04321289,  0.02624512, -0.1796875 , -0.19628906,
        0.06445312,  0.08935547,  0.1640625 , -0.03808594, -0.09814453,
       -0.01483154,  0.1875    ,  0.12792969,  0.22753906,  0.01818848,
       -0.07958984, -0.11376953, -0.06933594, -0.15527344, -0.08105469,
       -0.09277344, -0.11328125, -0.15136719, -0.08007812, -0.05126953,
       -0.15332031,  0.11669922,  0.06835938,  0.0324707 , -0.33984375,
       -0.08154297, -0.08349609,  0.04003906,  0.04907227, -0.24121094,
       -0.13476562, -0.05932617,  0.12158203, -0.34179688,  0.16503906,
        0.06176758, -0.18164062,  0.20117188, -0.07714844,  0.1640625 ,
        0.00402832,  0.30273438, -0.10009766, -0.13671875, -0.05957031,
        0.0625    , -0.21289062, -0.06542969,  0.1796875 , -0.07763672,
       -0.01928711, -0.15039062, -0.00106049,  0.03417969,  0.03344727,
        0.19335938,  0.01965332, -0.19921875, -0.10644531,  0.01525879,
        0.00927734,  0.01416016, -0.02392578,  0.05883789,  0.02368164,
        0.125     ,  0.04760742, -0.05566406,  0.11572266,  0.14746094,
        0.1015625 , -0.07128906, -0.07714844, -0.12597656,  0.0291748 ,
        0.09521484, -0.12402344, -0.109375  , -0.12890625,  0.16308594,
        0.28320312, -0.03149414,  0.12304688, -0.23242188, -0.09375   ,
       -0.12988281,  0.0135498 , -0.03881836, -0.08251953,  0.00897217,
        0.16308594,  0.10546875, -0.13867188, -0.16503906, -0.03857422,
        0.10839844, -0.10498047,  0.06396484,  0.38867188, -0.05981445,
       -0.0612793 , -0.10449219, -0.16796875,  0.07177734,  0.13964844,
        0.15527344, -0.03125   , -0.20214844, -0.12988281, -0.10058594,
       -0.06396484, -0.08349609, -0.30273438, -0.08007812,  0.02099609],
      dtype=float32)
```

In [ ]:
```
wv.most_similar('man') #
# Find the top-N most similar keys.
# Positive keys contribute positively towards the similarity, negative keys negatively.

# This method computes cosine similarity between a simple mean of the projection
# weight vectors of the given keys and the vectors for each key in the model.
# The method corresponds to the `word-analogy` and `distance` scripts in the original
# word2vec implementation.
```

Out[9]:
```
[('woman', 0.7664012908935547),
 ('boy', 0.6824870109558105),
 ('teenager', 0.6586930155754089),
 ('teenage_girl', 0.6147903800010681),
 ('girl', 0.5921714305877686),
 ('suspected_purse_snatcher', 0.5716364979743958),
 ('robber', 0.5585119128227234),
 ('Robbery_suspect', 0.5584409236907959),
 ('teen_ager', 0.5549196600914001),
 ('men', 0.5489763021469116)]
```

In [9]:
```
vec = wv['king']- wv['man'] + wv['woman']
vec
```

Out[9]:
```
array([ 4.29687500e-02, -1.78222656e-01, -1.29089355e-01,  1.15234375e-01,
        2.68554688e-03, -1.02294922e-01,  1.95800781e-01, -1.79504395e-01,
        1.95312500e-02,  4.09919739e-01, -3.68164062e-01, -3.96484375e-01,
       -1.56738281e-01,  1.46484375e-03, -9.30175781e-02, -1.16455078e-01,
       -5.51757812e-02, -1.07574463e-01,  7.91015625e-02,  1.98974609e-01,
        2.38525391e-01,  6.34002686e-02, -2.17285156e-02,  0.00000000e+00,
        4.72412109e-02, -2.17773438e-01, -3.44726562e-01,  6.37207031e-02,
        3.16406250e-01, -1.97631836e-01,  8.59375000e-02, -8.11767578e-02,
       -3.71093750e-02,  3.15551758e-01, -3.41796875e-01, -4.68750000e-02,
        9.76562500e-02,  8.39843750e-02, -9.71679688e-02,  5.17578125e-02,
       -5.00488281e-02, -2.20947266e-01,  2.29492188e-01,  1.26403809e-01,
        2.49023438e-01,  2.09960938e-02, -1.09863281e-01,  5.81054688e-02,
       -3.35693359e-02,  1.29577637e-01,  2.41699219e-02,  3.48129272e-02,
       -2.60009766e-01,  2.42309570e-01, -3.21777344e-01,  1.45416260e-02,
       -1.59179688e-01, -8.37402344e-02,  1.65039062e-01,  1.58691406e-03,
        3.09570312e-01,  3.16406250e-01,  7.38525391e-03,  2.41210938e-01,
        4.90722656e-02, -9.86328125e-02,  2.90527344e-02,  1.49414062e-01,
       -4.83398438e-02,  2.35595703e-01,  2.21191406e-01,  1.25488281e-01,
       -1.38671875e-01,  1.54296875e-01,  7.18994141e-02,  1.29882812e-01,
       -1.05712891e-01,  6.00585938e-02,  3.14697266e-01,  1.09619141e-01,
        8.49609375e-02,  7.71484375e-02, -2.17285156e-02,  6.11572266e-02,
       -1.89941406e-01,  2.07519531e-01, -1.63085938e-01,  1.13525391e-01,
        2.01171875e-01,  6.06689453e-02,  1.27929688e-01, -3.11279297e-01,
       -2.80151367e-01, -1.55883789e-01,  4.15039062e-02,  9.87854004e-02,
        1.69555664e-01, -3.49121094e-02,  2.08496094e-01, -9.89990234e-02,
        4.39453125e-03, -7.27539062e-02, -4.24804688e-02, -4.09179688e-01,
       -2.76367188e-01,  1.64062500e-01, -5.57617188e-01, -2.02199936e-01,
        2.12158203e-01, -9.81445312e-02,  2.30773926e-01,  2.75878906e-01,
        1.68092728e-01, -4.50439453e-02,  1.71615601e-01, -3.77075195e-01,
       -3.52478027e-03, -3.01513672e-01,  1.74224854e-01,  3.30078125e-01,
        2.00683594e-01,  1.17736816e-01, -1.37695312e-01, -1.07421875e-01,
        8.61816406e-02,  1.06445312e-01,  1.44531250e-01,  3.05175781e-03,
        1.80664062e-02,  3.73535156e-02,  7.32421875e-03,  1.32812500e-01,
        9.61914062e-02,  3.35998535e-01,  1.81152344e-01,  2.40905762e-01,
       -8.49609375e-02, -1.10107422e-01,  2.11914062e-01,  5.85937500e-03,
        1.62109375e-01, -4.15527344e-01,  1.39160156e-01,  1.01562500e-01,
        1.44531250e-01, -1.09375000e-01,  4.88281250e-02,  6.15234375e-02,
       -1.69921875e-01,  3.28369141e-02,  5.56640625e-02,  1.47460938e-01,
       -2.24609375e-02, -2.73925781e-01, -2.81982422e-01, -1.39160156e-01,
       -1.81884766e-01,  9.33532715e-02,  1.21093750e-01, -5.37109375e-03,
       -1.87500000e-01,  3.05175781e-04,  5.52734375e-01, -9.71679688e-02,
       -1.81640625e-01, -1.51855469e-01,  7.76367188e-02, -2.38281250e-01,
       -2.63977051e-02,  2.25555420e-01, -3.02734375e-01,  1.34765625e-01,
        3.23242188e-01,  1.25976562e-01,  3.51562500e-02, -2.04345703e-01,
        2.96142578e-01,  1.03149414e-01, -4.76074219e-03,  1.69189453e-01,
       -3.50585938e-01,  2.46887207e-02, -3.90502930e-01, -2.70507812e-01,
        1.85241699e-02,  1.04492188e-01,  2.84179688e-01,  1.35009766e-01,
       -5.95703125e-02,  1.88232422e-01,  8.88214111e-02,  3.24707031e-02,
       -8.98437500e-02,  5.45043945e-02,  5.65185547e-02,  1.56860352e-01,
       -9.70458984e-03, -7.08007812e-02,  5.71289062e-02, -3.08837891e-01,
       -1.91894531e-01,  4.83398438e-02,  5.22460938e-02, -1.59667969e-01,
       -4.49218750e-02, -7.37304688e-02,  5.51757812e-02,  2.12402344e-01,
        2.05322266e-01, -2.73437500e-02,  7.86132812e-02,  3.19091797e-01,
       -1.56982422e-01, -3.92822266e-01,  4.00390625e-02,  9.93652344e-02,
       -1.97372437e-02, -8.25195312e-02,  2.53906250e-02,  3.10668945e-02,
       -3.63769531e-02,  1.48925781e-02,  2.20703125e-01, -5.98144531e-02,
        6.15234375e-02, -7.14111328e-02, -4.00390625e-02, -1.03515625e-01,
        9.22851562e-02,  2.71789551e-01, -2.30224609e-01, -2.62695312e-01,
       -5.61523438e-01,  1.38549805e-02,  1.09863281e-01,  7.22656250e-02,
        4.58984375e-02, -3.31802368e-02, -8.03833008e-02, -6.10351562e-03,
        2.09960938e-01, -3.86840820e-01,  1.44645691e-01,  8.05664062e-02,
        2.96264648e-01, -1.17187500e-02, -2.34680176e-01,  1.32019043e-01,
        2.53906250e-01, -2.46826172e-01,  1.03759766e-01,  1.14013672e-01,
        1.71875000e-01, -5.61523438e-03,  2.05078125e-01,  6.34765625e-02,
       -4.51293945e-01, -2.26562500e-01, -1.03027344e-01, -1.31469727e-01,
        3.75976562e-02,  2.70996094e-01, -2.39257812e-01,  3.80859375e-02,
       -3.90625000e-02, -9.42382812e-02,  8.30078125e-03,  7.03125000e-02,
        2.75390625e-01,  3.31542969e-01, -1.07421875e-02,  3.72192383e-01,
       -1.24511719e-01,  1.94335938e-01, -1.35620117e-01, -3.09570312e-01,
       -2.36328125e-01, -1.26953125e-02, -2.76855469e-01,  1.57714844e-01,
        3.07617188e-01, -2.32910156e-01,  3.25439453e-01,  1.36718750e-02,
        1.99462891e-01, -2.61840820e-02, -8.08105469e-02, -7.50732422e-02,
       -4.11109924e-02,  1.95556641e-01, -5.64270020e-02, -2.79296875e-01,
       -2.75390625e-01, -4.04296875e-01, -1.75781250e-02, -5.85937500e-03,
       -7.71484375e-02,  1.33789062e-01,  2.36816406e-01,  2.01538086e-01],
      dtype=float32)
```

In [ ]:
```
wv.most_similar('cricket')  # semmantic meaning is captured
```

Out[15]:
```
[('cricketing', 0.8372225165367126),
 ('cricketers', 0.8165745735168457),
 ('Test_cricket', 0.8094818592071533),
 ('Twenty##_cricket', 0.8068488240242004),
 ('Twenty##', 0.7624266147613525),
 ('Cricket', 0.7541396617889404),
 ('cricketer', 0.7372579574584961),
 ('twenty##', 0.7316356897354126),
 ('T##_cricket', 0.7304614782333374),
 ('West_Indies_cricket', 0.698798656463623)]
```

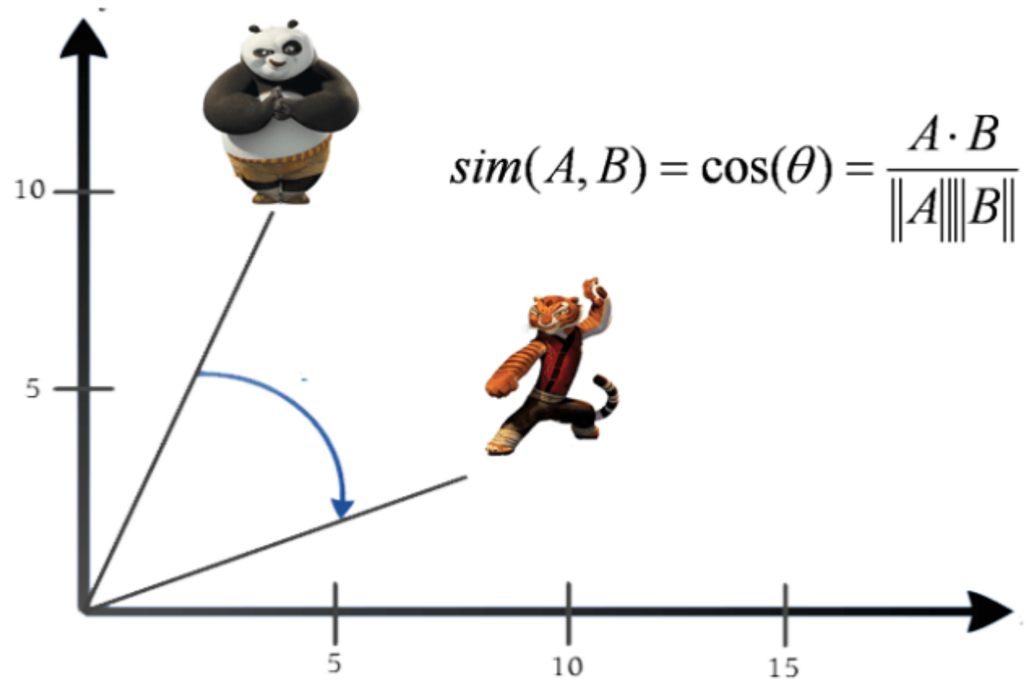
In [ ]:

## Cosine similarity :

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.

The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

# Cosine Similarity




$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

**Compute cosine similarity between two keys.**

In [12]:
```
wv.similarity('cricket','hocky') # how much similar is cricket to hocky
```

Out[12]:
```
0.20070238
```

In [ ]:
```
conclusion based on Cosine similarity :
    1. the value of cosine more towards the 1 , that menas two keys are very much similar.
    2. the value of cosine more towards the 0, that mens tow keys are not very much similar.

    eg. above example, the two keys 'cricket' and 'hocky' are 20% similiar to each other.
```

In [11]:
```
#### calculate distance between  tow keys.
dist = 1 - 0.20070238
dist
```

Out[11]:
```
0.7992976199999999
```

```
conclusion based on distance :
    1. distance more towards to Zero more the similar two keys or points are .
    2. distance more towards to 1, that means two points or keys are not much similar

    eg. distance 0.79 means two points are far away from each other.
```

In [ ]:
```
wv.similarity('cricket','sports') # cricket and sports are 40% similar to each other
```

Out[17]:
```
0.40087253
```

In [ ]:
```
wv.similarity('cricket','football') # cricket and football are 45% similar to each other
```

Out[18]:
```
0.45974636
```

In [ ]:
```
vec = wv['king'] - wv['man'] + wv['women']   # simple operations
```

In [ ]:
```
vec
```

Out[20]:
```
array([-0.33984375,  0.06298828, -0.00994873,  0.3305664 ,  0.10327148,
       -0.25854492,  0.19677734, -0.32476807, -0.15917969,  0.45752716,
       -0.39208984, -0.22460938,  0.02929688,  0.31982422, -0.0958252 ,
       -0.05932617,  0.01855469, -0.03945923,  0.01101685,  0.33984375,
        0.10351562,  0.14396667, -0.07641602,  0.06640625, -0.10839844,
       -0.1953125 , -0.3564453 , -0.02124023, -0.16503906, -0.16247559,
       -0.05519104, -0.1003418 ,  0.01464844,  0.23449707, -0.26611328,
       -0.07080078, -0.26904297, -0.00292969, -0.06738281, -0.02050781,
        0.04418945, -0.09326172,  0.12304688,  0.10626221,  0.10290527,
        0.01660156, -0.10791016, -0.0065918 ,  0.17578125,  0.10028076,
        0.22363281, -0.05761719, -0.31743622,  0.3922119 , -0.35498047,
       -0.23643494, -0.01074219, -0.01334572, -0.15283203, -0.00793457,
        0.08203125,  0.09985352,  0.003479  ,  0.11608887,  0.14550781,
       -0.125     , -0.11254883,  0.2548828 , -0.04345703,  0.34985352,
       -0.02880859,  0.15966797,  0.07226562,  0.17626953,  0.15979004,
       -0.20263672, -0.23864746,  0.3400879 ,  0.17700195,  0.15454102,
       -0.20385742,  0.05224609, -0.05224609,  0.3067627 , -0.14892578,
       -0.07910156, -0.10107422,  0.06176758,  0.08154297,  0.17700195,
        0.04589844, -0.41674805, -0.32019043, -0.16674805,  0.11083984,
       -0.02352905,  0.35290527, -0.30004883,  0.25634766, -0.16748047,
        0.21386719,  0.12304688, -0.27490234, -0.35009766, -0.4580078 ,
        0.38671875, -0.6265869 , -0.13139915,  0.34692383,  0.11553955,
        0.42016602,  0.32763672, -0.03381157,  0.08630371,  0.2419281 ,
       -0.35412598,  0.19325256, -0.36791992,  0.17056274,  0.1459961 ,
       -0.18945312,  0.28393555,  0.01953125, -0.18115234,  0.08374023,
       -0.01483154,  0.06591797,  0.21643066, -0.01757812,  0.18481445,
        0.01757812,  0.07226562,  0.01220703,  0.4343872 ,  0.2734375 ,
        0.20452881,  0.01586914,  0.01391602,  0.33740234, -0.30273438,
       -0.00439453, -0.5595703 ,  0.38867188, -0.1373291 ,  0.13574219,
       -0.20605469,  0.10742188,  0.02148438, -0.14648438,  0.2142334 ,
       -0.06054688,  0.23291016, -0.16430664, -0.4741211 , -0.13745117,
       -0.12353516, -0.26293945,  0.03125   ,  0.15332031, -0.08251953,
       -0.11865234, -0.10687256,  0.2421875 , -0.11425781, -0.05310059,
       -0.30371094, -0.06225586,  0.05029297, -0.07272339,  0.11499023,
       -0.2980957 ,  0.09692383,  0.41088867,  0.16601562,  0.3486328 ,
       -0.17211914,  0.33325195, -0.19030762,  0.14611816,  0.14916992,
       -0.33129883, -0.11791992, -0.39538574, -0.13183594, -0.07034302,
        0.03204346,  0.38623047, -0.07495117,  0.15112305,  0.38134766,
        0.2423706 , -0.11083984, -0.109375  , -0.08319092, -0.1003418 ,
        0.08483887,  0.34039307,  0.11035156,  0.01074219, -0.3815918 ,
       -0.1743164 ,  0.14331055,  0.2562256 , -0.2944336 ,  0.00488281,
       -0.04443359, -0.06347656,  0.03100586, -0.12084961,  0.41137695,
        0.078125  ,  0.34484863, -0.05615234, -0.2998047 ,  0.06054688,
        0.17797852, -0.01876068, -0.16064453,  0.09765625, -0.05682373,
       -0.2878418 , -0.05664062,  0.19726562, -0.22753906,  0.00268555,
       -0.17297363, -0.19726562, -0.15332031,  0.1899414 ,  0.32299805,
       -0.18261719, -0.06103516, -0.47460938,  0.1251831 ,  0.08056641,
       -0.05444336,  0.17797852,  0.00754547, -0.00805664, -0.18371582,
        0.23339844, -0.39074707,  0.18608856,  0.12158203,  0.2767334 ,
       -0.06054688, -0.28857422,  0.3359375 , -0.02954102, -0.12329102,
        0.09594727,  0.1965332 ,  0.15771484, -0.015625  ,  0.20019531,
       -0.07763672, -0.30419922, -0.21777344,  0.06689453,  0.11193848,
       -0.03125   ,  0.17773438, -0.05078125,  0.15014648, -0.19580078,
       -0.2319336 , -0.04980469,  0.16308594,  0.4963379 ,  0.64746094,
        0.03808594,  0.5743408 , -0.1274414 ,  0.16711426,  0.12188721,
       -0.43432617, -0.3585205 , -0.12646484, -0.04003906,  0.39794922,
        0.25683594, -0.45654297,  0.3178711 , -0.24658203,  0.16040039,
       -0.11010742, -0.14331055, -0.0291748 ,  0.02893066, -0.03295898,
       -0.05691528, -0.11279297, -0.18359375, -0.4482422 , -0.00830078,
       -0.12371826, -0.02783203,  0.26367188,  0.27148438,  0.28808594],
      dtype=float32)
```

In [15]:
```
wv.most_similar([vec])
```

Out[15]:
```
[('king', 0.8449392318725586),
 ('queen', 0.7300517559051514),
 ('monarch', 0.645466148853302),
 ('princess', 0.6156251430511475),
 ('crown_prince', 0.5818676352500916),
 ('prince', 0.5777117609977722),
 ('kings', 0.5613663792610168),
 ('sultan', 0.5376775860786438),
 ('Queen_Consort', 0.5344247817993164),
 ('queens', 0.5289887189865112)]
```

## Thank you for Reading :

***addition resource and the resources which I read :***

https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa (https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa)
https://arxiv.org/pdf/1411.2738.pdf (https://arxiv.org/pdf/1411.2738.pdf)
https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92 (https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92)
https://www.machinelearningplus.com/nlp/cosine-similarity/ (https://www.machinelearningplus.com/nlp/cosine-similarity/)

Best regards,
Pankaj Kumar Barman, MSc.CSMI
Ramakrishna Mission Vidyamandira, howrah, belur

In [ ]: