In [1]:
```
#Install gensim and bokeh
#!pip install gensim
#!pip install bokeh
```

Dataset : https://code.google.com/archive/p/word2vec/

Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases.

In [1]:
```
Path=r'C:\Users\Arun\Desktop\Nipu\GoogleNews-vectors-negative300.bin'
```

In [3]:
```
import gensim
#train word2vec model
# Load Google's pre-trained Word2Vec model.
model = gensim.models.KeyedVectors.load_word2vec_format(Path, binary=True)
#Vocabulary size
words = list(model.wv.vocab)
print('Here is the Vocabulary Size.. %d' % len(words))
```

```
Here is the Vocabulary Size.. 3000000
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DeprecationWarning: Call to deprecated `wv` (Attribute will be removed in 4.0.0, use self instead).
```

Word2vec makes each word a vector. We are using the 300-number vector, which can be seen for the word "king".

In [17]:
```
w = model['king']
print("Array length of Word 'King':-",len(w))
```

```
Array length of Word 'King':- 300
```

In [19]:
```
print("'King' 300 dimension array values:-")
print(w)
```

```
'King' 300 dimension array values:-
[ 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 -2.88984375e-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
  1.96093750e-02 -2.72216797e-02  1.96533203e-02  4.27246094e-02
 -2.46093750e-01  6.39648438e-02 -2.25585938e-01 -1.68945312e-01
  2.09916992e-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
 -2.24804688e-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  4.88935547e-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 -2.88007812e-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 -4.66018945e-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]
```

The following code shows which words are most similar to the given one.

In [7]:
```
model.most_similar('king')
```

Out[7]:
```
[('kings', 0.7138046026229858),
 ('queen', 0.6510957479476929),
 ('monarch', 0.6413194537162781),
 ('crown_prince', 0.6204220056533813),
 ('prince', 0.6159093141810474),
 ('sultan', 0.5864823460578918),
 ('ruler', 0.5797567367553711),
 ('princes', 0.5646551847457886),
 'Prince_Paras', 0.5432944297790527),
 ('throne', 0.5422104597091675)]
```

Kings,queen,monarch,crown_prince,sultan...etc these words are similar to word 'king'

In [8]:
```
model.most_similar('queen')
```

Out[8]:
```
[('queens', 0.7399442791938782),
 ('princess', 0.7070531249046326),
 ('king', 0.6510956883430481),
 ('monarch', 0.6383601427078247),
 ('very_pampered_McElhatton', 0.6357026696205139),
 ('Queen', 0.6163408160209656),
 ('NYC_anglophiles_aflutter', 0.6060680150985718),
 ('Queen_Consort', 0.5923796296611969),
 ('princesses', 0.5908075571060181),
 ('royal', 0.5637185573577881)]
```

queen,princess,king,monarch,queen...etc these words are similar to word 'queen'

The following code shows the similarity between two words.

In [9]:
```
model.similarity('king', 'queen')
```

Out[9]:
```
0.6510957
```

65% have probability both have similar words

The code below shows the distance between two words.

In [10]:
```
import numpy as np

w1 = model['king']
w2 = model['queen']

dist = np.linalg.norm(w1-w2)

print("Distance Between King & queen:-",dist)
```

```
Distance Between King & queen:- 2.4796925
```

This shows the classic word2vec equation of queen = (king - man) + female

In [11]:
```
model.most_similar(positive=['woman', 'king'], negative=['man'])
```

Out[11]:
```
[('queen', 0.7118192315101624),
 ('monarch', 0.6189674735069275),
 ('princess', 0.5902431011199951),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321243286133),
 ('kings', 0.5236844420433044),
 ('Queen_Consort', 0.5235946178436279),
 ('queens', 0.5181133747100083),
 ('sultan', 0.5090593235015869),
 ('monarchy', 0.5087411999702454)]
```

71% have probability 'queen' is the correct value of equation " queen = (king - man) + female"

The following code shows which item does not belong with the others.

In [12]:
```
model.doesnt_match("man king queen apple".split())
```

```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\models\keyedvectors.py:877: FutureWarning: arrays to stack must be passed as a "sequence" type such as list or tuple. Support for non-sequence iterables such as generators is deprecated as of NumPy 1.16 and will raise an error in the future.
  vectors = vstack(self.word_vec(word, use_norm=True) for word in used_words).astype(REAL)
```

Out[12]:
```
'apple'
```

Visualization of vocabalary in 2-D Space

In [53]:
```
# Importing bokeh libraries for showing how words of similar context are grouped together
import bokeh.plotting as bp
import pandas as pd
from bokeh.models import HoverTool, BoxSelectTool,LabelSet,ColumnDataSource
from bokeh.plotting import figure, show, output_notebook
voc_size=3000
#Defining the chart
output_notebook()
plot_chart = bp.figure(plot_width=700, plot_height=600, title="A map/plot of 3000 word vectors",
    tools="pan,wheel_zoom,box_zoom,reset,hover",
    x_axis_type=None, y_axis_type=None, min_border=1)

#Extracting the list of word vectors, limiting to 1000, each is of 200 dimensions
word_vectors = [model[w] for w in list(model.wv.vocab.keys())[:voc_size]]

# Reducing dimensionality by converting the vectors to 2d vectors
from sklearn.manifold import TSNE
tsne_model = TSNE(n_components=2, verbose=1, random_state=0)
tsne_w2v = tsne_model.fit_transform(word_vectors)

# Storing data in a dataframe
tsne_df = pd.DataFrame(tsne_w2v, columns=['x', 'y'])
tsne_df['words'] = list(model.wv.vocab.keys())[:voc_size]

# Corresponding word appears when you hover on the data point.
plot_chart.scatter(x='x', y='y',source=tsne_df)
hover = plot_chart.select(dict(type=HoverTool))
hover.tooltips={"word": "@words"}
show(plot_chart)
```

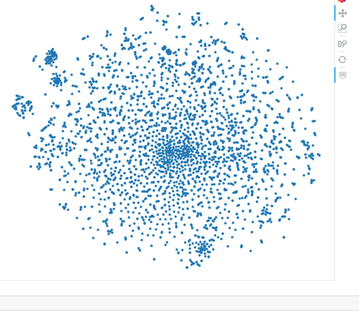
BokehJS 2.0.1 successfully loaded.

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:14: DeprecationWarning: Call to deprecated `wv` (Attribute will be removed in 4.0.0, use self instead).
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 3000 samples in 0.424s...
[t-SNE] Computed neighbors for 3000 samples in 9.688s...
[t-SNE] Computed conditional probabilities for sample 1000 / 3000
[t-SNE] Computed conditional probabilities for sample 2000 / 3000
[t-SNE] Computed conditional probabilities for sample 3000 / 3000
[t-SNE] Mean sigma: 0.883493
[t-SNE] KL divergence after 250 iterations with early exaggeration: 110.179993
[t-SNE] KL divergence after 1000 iterations: 2.198179
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:23: DeprecationWarning: Call to deprecated `wv` (Attribute will be removed in 4.0.0, use self instead).
```

A map/plot of 3000 word vectors



In [ ]:
```
```