In [1]:
```
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from nltk.corpus import wordnet as wn
```

In [2]:
```
file = open('Downloads/glove-global-vectors-for-word-representation/glove.6B.100d.txt')
```

In [4]:
```
vocabulary = []
vector_matrix = []
word_embeddings = {}

for line in file:
    lines = line.split()
    vocabulary.append(lines[0])
    vectors = np.asarray([float(i) for i in lines[1:]])
    word_embeddings.update({lines[0]:vectors})
    vector_matrix.append(vectors)
```

In [13]:
```
def evaluate(word, relation, k = 4, return_list = False):
    if [i for i in [word, relation[0], relation[1]] if i not in word_embeddings]:
        raise ValueError("Word must be in vocabulary")
    result_vector = word_embeddings[relation[0]] - word_embeddings[relation[1]] + word_embedding
s[word]
    nearest_indices = k_nearest_vectors(k, vector_matrix, [result_vector])[0]
    closest_words = [vocabulary[i] for i in nearest_indices if vocabulary[i] != word and vocabul
ary[i] not in relation]
    if return_list:
        return closest_words
    else:
        return closest_words[0]
```

In [8]:
```
def k_nearest_vectors(k, mtx, candidate_vector):
    """Takes in an integer value(k), a matrix (2D list) of all the vectors, and the vector (list
) we want to compare.
    Returns an array of length k for indices of the most similar word vectors, and the cosine si
milarities of these vectors """
    cos_similarities = cosine_similarity(mtx, candidate_vector).flatten()
    k_sorted = np.flip(np.argsort(cos_similarities)[-k:], axis = 0)
    cos_sorted = np.flip(np.sort(cos_similarities), axis = 0)[:k]
    return k_sorted, cos_sorted
```

In [11]:
```
vocabulary.index('woman') == k_nearest_vectors(5, vector_matrix, [word_embeddings['woman']])[0][
0]
```

```
True
```

Evaluation

In [14]:
```
evaluate('king', ['woman', 'man'])
```

```
'queen'
```

In [15]:
```
evaluate('they', ['his', 'he'])
```

```
'their'
```

In [16]:
```
evaluate('japan', ['europe', 'germany'])
```

```
'asia'
```

In [17]:
```
evaluate('nurse', ['man', 'doctor'])
```

```
'woman'
```

In [18]:
```
evaluate('terrorist', ['christianity', 'lawful'])
```

```
'islamic'
```