

```
In [ ]: import nltk
        from gensim.models import Word2Vec
        from nltk.corpus import stopwords
        import re
```

```
In [ ]: f = open('Text 1.txt', 'r')
        paragraph = f.read()
        print(paragraph)
        f.close()
```

```
In [ ]: # Preprocessing 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)
```

```
In [ ]: # Preparing the dataset
        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('en
```

```
In [ ]: # Training the Word2Vec model
        model = Word2Vec(sentences, min_count=1)
        words = model.wv.vocab
```

```
In [ ]: # Visualise the embedding
        from gensim.models import Word2Vec
        from sklearn.decomposition import PCA
        from matplotlib import pyplot

        # fit a 2d PCA model to the vectors
        X = model[words]
        pca = PCA(n_components=2)
        result = pca.fit_transform(X)
        # create a scatter plot of the projection
        max_visual = 20
        pyplot.scatter(result[:max_visual, 0], result[:max_visual, 1])
        words = list(model.wv.vocab)
        for i, word in enumerate(words):
            if i < max_visual:
                pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
        pyplot.show()
```

```
In [ ]: # Finding Word Vectors
        vector = model.wv['queen']
        print(vector)
```

```
In [ ]: # Most similar words
        similar = model.wv.most_similar('queen')
```

```
In [ ]: print(similar)
```

```
In [ ]: # Solving Today's Objective  
result = model.wv.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)  
print(result)
```