Introduction to Word2Vec:

Uzair Ahmad

In this tutorial, we'll use the popular `Gensim` library in Python to perform various tasks using word embeddings, specifically the pre-trained Word2Vec model.

Setup

First, let's get our environment set up. Install the required packages:

```
pip install gensim
```

Loading Pre-trained Word2Vec Model

For simplicity, we'll use the pre-trained Google News vectors:

```python
from gensim.models import KeyedVectors

# Load the Google News vectors
model = KeyedVectors.load_word2vec_format('https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz', binary=True, limit=100000)
```

Or You can manually download the Google News vectors from the [Gensim GitHub repository](). Once you download it, you can load it using the path to the downloaded file.

```python
import gensim.downloader as api

model = api.load("word2vec-google-news-300")

```

1. Finding Similarity

You can find the similarity between two words:

```python
similarity = model.similarity('king', 'queen')
print(f"Similarity between 'king' and 'queen': {similarity:.4f}")
```

```
man:woman::king:queen
```

2. Finding Analogies

The classic example is: man👩:king:?

```
result = model.most_similar(positive=['woman', 'king'], negative=['man'])
print(f"man:woman::king:{result[0][0]}")
```

```
man:woman::king:queen
```

3. Finding Synonyms (Most Similar Words)

Let's say we want the 5 most similar words to 'computer':

```
synonyms = model.most_similar('computer', topn=5)
for word, score in synonyms:
    print(f"{word}: {score:.4f}")
```

```
computers: 0.7979
laptop: 0.6640
laptop_computer: 0.6549
Computer: 0.6473
com_puter: 0.6082
```

4. Odd-One-Out

Find the word that doesn't belong:

```
odd_word = model.doesnt_match(['apple', 'banana', 'cherry', 'computer'])
print(f"The odd one out is: {odd_word}")
```

```
The odd one out is: computer
```

5. Visualizing Word Embeddings

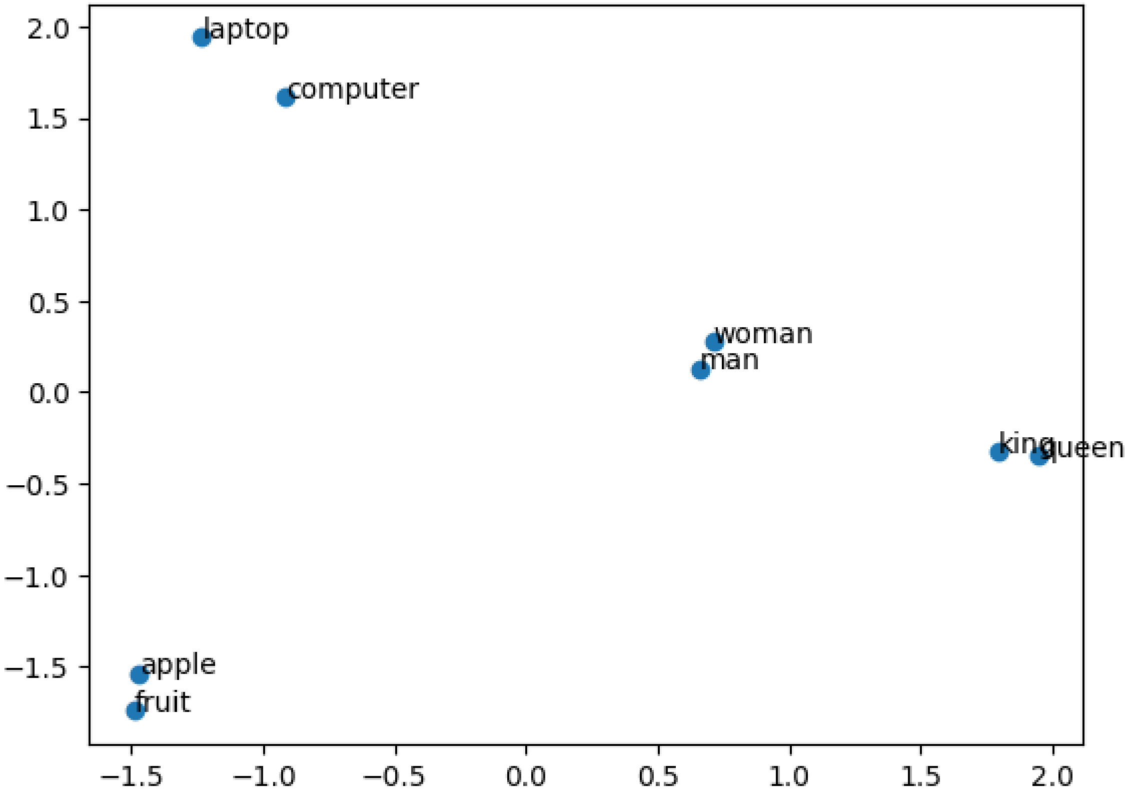
Using PCA (Principal Component Analysis), you can reduce the dimensionality of word vectors and plot them:

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

words = ['king', 'queen', 'man', 'woman', 'computer', 'laptop', 'fruit', 'apple']
vectors = [model[word] for word in words]

# Reduce dimensions
pca = PCA(n_components=2)
result = pca.fit_transform(vectors)

# Plotting
plt.scatter(result[:, 0], result[:, 1])
for i, word in enumerate(words):
    plt.annotate(word, xy=(result[i, 0], result[i, 1]))
plt.show()
```



This tutorial provided a brief overview of how you can work with word embeddings using the Gensim library. There's a lot more you can do with these embeddings, from using them in deep learning models to improving information retrieval systems. Happy coding!