Semantic Textual Similarity (STS)

Semantic text similarity refers to the degree of similarity or relatedness between two pieces of text based on their meaning or semantics rather than their exact lexical similarity. In other words, semantic text similarity measures how closely two texts convey similar ideas, concepts, or intents, even if the specific words used in the texts are different.

To find the similarity score, there are three main steps:

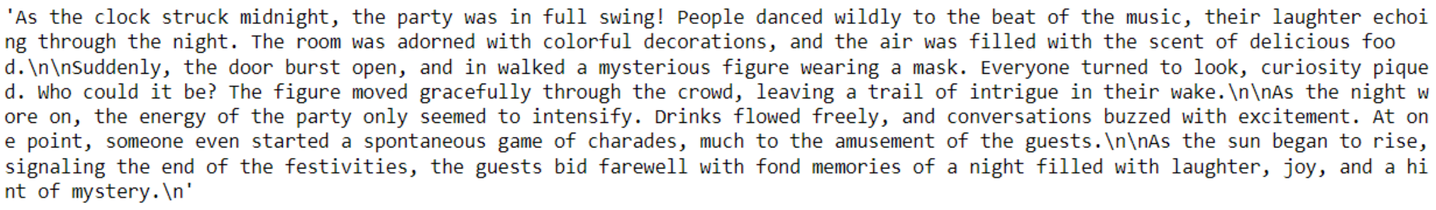
1. Preprocess the texts
2. Convert the text to a vector
3. Find the similarity between the vectors

1. Preprocessing:

a. Removing the punctuation, digits, and stopwords as they do not affect the semantic meaning except some like “no”, “nor”, “not”.

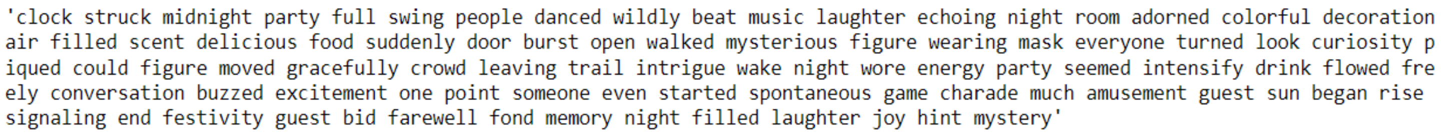
b. Tokenization and lemmatisation to bring each and every word in their base form so it would be easy to check the similarity between the two texts.

Before:



```
'As the clock struck midnight, the party was in full swing! People danced wildly to the beat of the music, their laughter echoi
ng through the night. The room was adorned with colorful decorations, and the air was filled with the scent of delicious foo
d.\n\nSuddenly, the door burst open, and in walked a mysterious figure wearing a mask. Everyone turned to look, curiosity pique
d. Who could it be? The figure moved gracefully through the crowd, leaving a trail of intrigue in their wake.\n\nAs the night w
ore on, the energy of the party only seemed to intensify. Drinks flowed freely, and conversations buzzed with excitement. At on
e point, someone even started a spontaneous game of charades, much to the amusement of the guests.\n\nAs the sun began to rise,
signaling the end of the festivities, the guests bid farewell with fond memories of a night filled with laughter, joy, and a hi
nt of mystery.\n'
```

After:

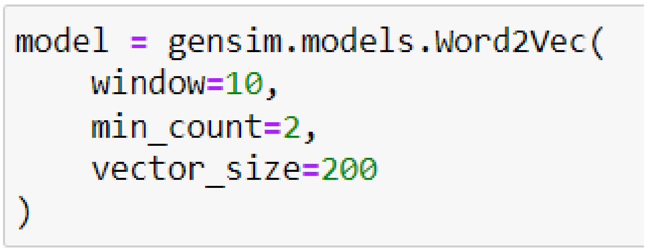


```
'clock struck midnight party full swing people danced wildly beat music laughter echoing night room adorned colorful decoration
air filled scent delicious food suddenly door burst open walked mysterious figure wearing mask everyone turned look curiosity p
iqued could figure moved gracefully crowd leaving trail intrigue wake night wore energy party seemed intensify drink flowed fre
ely conversation buzzed excitement one point someone even started spontaneous game charade much amusement guest sun began rise
signaling end festivity guest bid farewell fond memory night filled laughter joy hint mystery'
```

2. Training a word2Vec model to make embeddings:

Using gensim library training the word2vec model which returns the word embeddings in the vector of size 200.

After we get the word embedding of each word in a text then we take the average of all the embedding at each index (applying mean in 2d array in axis =1)

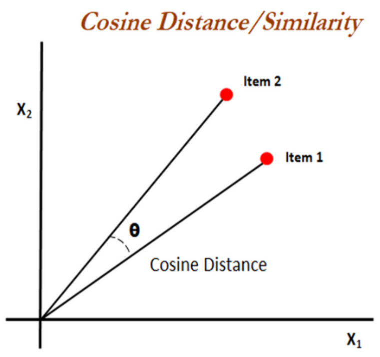


```
model = gensim.models.Word2Vec(
    window=10,
    min_count=2,
    vector_size=200
)
```

3. Finding similarity:

For this, we use the cosine similarity, which finds the cos(angle) between the 2 vectors.

The more the value of cos(angle), the more the silimarity. This helps us to quantify the similarity between the two texts.



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

Deployment Of the model

We first save the model built in .pkl file using pickle module.

Then also save the functions which will be needed to preprocess and convert text to embeddings in functions.py file

In main.py, I made a FastAPI object that takes text1 and text2 as input and calculates the similarity between the two texts.