

Practical 6

Name: Shivam Tawari

Roll no: A-58

Subject: NLP

Aim: To write a python program to detect similar sentence from given paragraph.

Theory:

Sentence Similarity:

Sentence similarity or semantic lecture similarly is a measure of how similar of two pieces of text are as to what degree they express the same meaning.

Release task include paragraph or duplicate identification search and matching application. Sentence similarity is normally calculated by the following two steps:

- ① ~~Comparing~~ Calculating the embedding of the sentence.
- ② Taking the cosine similarity between them.

* Cosine Similarity:

Cosine similarity calculate similarity by measuring the cosine of angle between two vectors.

$$\text{Similarity} = \cos(\theta) = \frac{AB}{(|A| \times |B|)}$$

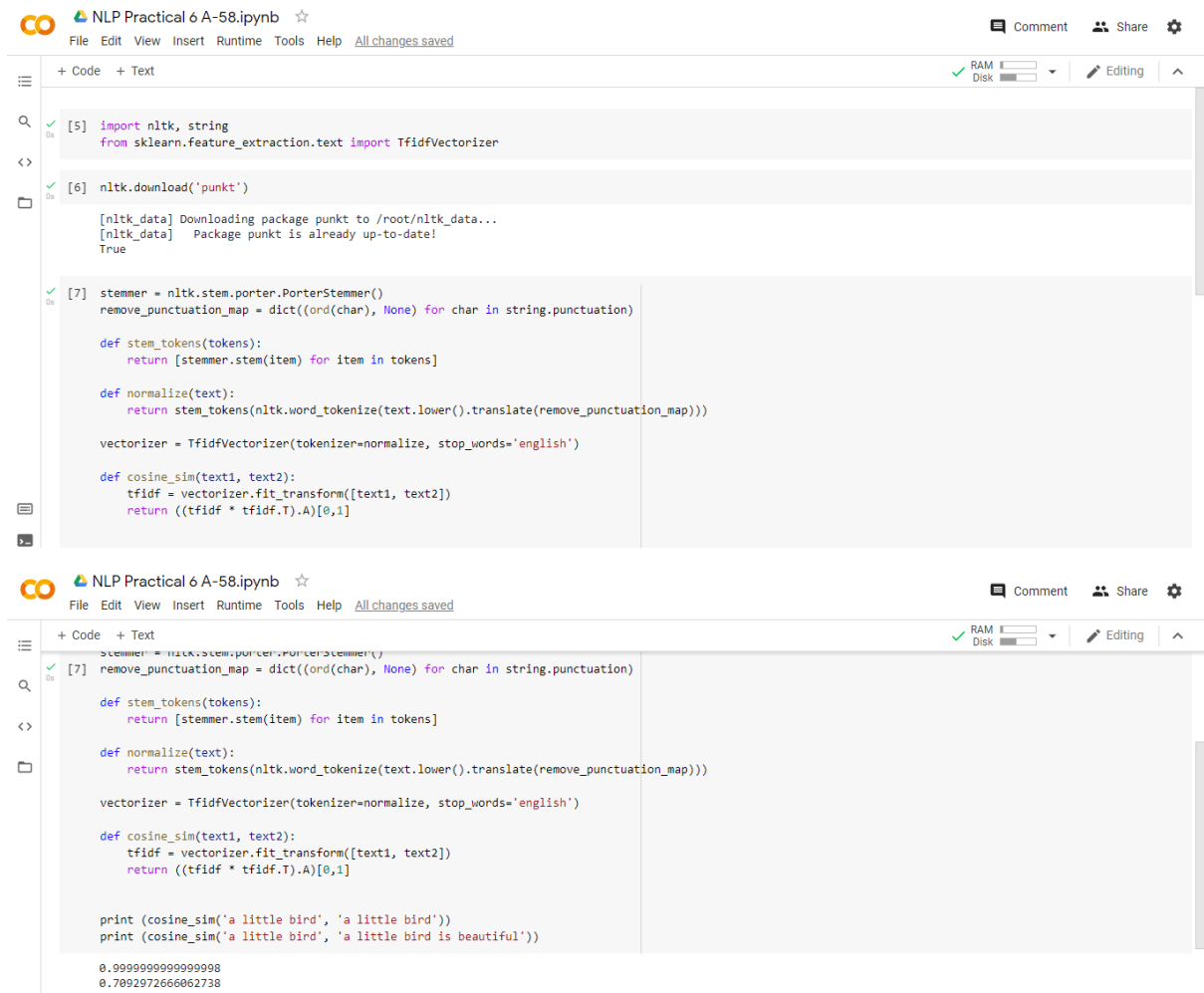
Mathematically cosine similarity is measure of similarity between two non-zero vector of an inner product space that measure the cosine angle between them. It is advantageous because even if the two similar documents are far apart by euclidean distance, they may still oriented closer together, smaller the angle, higher the cosine similarity.

Conclusion: Hence, successfully created and implemented a python program to detect similar sentence from a given paragraph.

Practical – 6

Name: Shivam Tawari

Roll no: A – 58



The image displays two screenshots of a Jupyter Notebook titled "NLP Practical 6 A-58.ipynb". The interface includes a top bar with the Colab logo, file editing options, and a status bar showing RAM and disk usage. The notebook is in "Editing" mode.

Top Screenshot: Shows the first three code cells. Cell [5] imports `nlTK` and `TfidfVectorizer`. Cell [6] downloads the `punkt` tokenizer. Cell [7] defines a `PorterStemmer`, a punctuation removal map, a `stem_tokens` function, a `normalize` function, a `TfidfVectorizer`, and a `cosine_sim` function.

```
[5] import nlTK, string
    from sklearn.feature_extraction.text import TfidfVectorizer

[6] nlTK.download('punkt')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
True

[7] stemmer = nlTK.stem.porter.PorterStemmer()
    remove_punctuation_map = dict((ord(char), None) for char in string.punctuation)

    def stem_tokens(tokens):
        return [stemmer.stem(item) for item in tokens]

    def normalize(text):
        return stem_tokens(nlTK.word_tokenize(text.lower().translate(remove_punctuation_map)))

    vectorizer = TfidfVectorizer(tokenizer=normalize, stop_words='english')

    def cosine_sim(text1, text2):
        tfidf = vectorizer.fit_transform([text1, text2])
        return ((tfidf * tfidf.T).A)[0,1]
```

Bottom Screenshot: Shows the continuation of the code from the previous cell, including the `cosine_sim` function definition and the execution of the function with two sample sentences. The output shows a cosine similarity of approximately 0.9999999999999998 for identical sentences and 0.7092972666062738 for similar sentences.

```
print(cosine_sim('a little bird', 'a little bird'))
print(cosine_sim('a little bird', 'a little bird is beautiful'))

0.9999999999999998
0.7092972666062738
```