__/__/__

NLTK ?
natural language toolkit

tools

→ Useful for working with human language data

- Tokenization
- Stopword removal
- Lemmatization
- POS tagging
- Parsing

punkt : Segments text into sentences using an unsupervised algorithm

Working :
→ Learns abbreviations, ~~collectio~~ collocations, and sentence boundaries from raw text.

Eg :

```
x = " ABC. He uses NLTK hamesha!"
print (sent_tokenize (x))
o/p: ["ABC.", "He uses NLTK hamesha!"]
```

uses Punkt (pre-trained model) to identify sentence boundaries

Use Case : preprocessing in tasks like sentiment analysis, summarization, or any sentence-level NLP

Stopwords : common words that are usually filtered out before processing

Use case : improves signal-to-noise ratio in text classification, TF-IDF vectorization, topic modeling

क्यों हटाये ?

words like "the", "is", "and" don't carry much semantic weight and can introduce noise in models

Eg :

```
from nltk.corpus import stopwords
hatao_bc = set (stopwords.words('english'))
tokens = ["this", "is", "a", "sample", "sentence"]
filter = [ word for word in tokens if word not_in hatao_bc]
print (filter)          # o/p: ['sample', 'sentence']
```

WordNet : Used for lemmatization and semantic analysis
→ A large English lexical db where:
- noun, verbs, adjectives are grouped into sets of synonyms

Eg:
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print (lemmatizer.lemmatize ("running", pos="v"))
#o/p: run
```
↑ verb

Q) Lemmatization क्या है?
→ Text normalization technique in NLP, ~~convert~~
→ Converts a word into its base or dictionary form (its lemma)
→ This will result to a meaningful root word like "good" from "better"

Eg:
- running → run (verb)
- better → good
- was → be

Purpose : reduce variations of words to a ~~com~~ common base ~~for~~ form, making text data more uniform and reducing its complexity for analysis.

Semantic analysis : Understanding the meaning, intent and emotional tone of text by examining the relationships and context b/w words and phrases.

अब दोनों को साथ मे use कैसे ~~करे~~ करते है?

Lemmatization $\xrightarrow{\text{converts}}$ "running" to "run"
Now this will help semantic analysis model recognize them as the same concept. This consistency improves the model's ability to correctly interpret the overall meaning of sentence.

punkt_tab : internally used by Punkt tokenizer to store learned sentence boundary parameters.

To train own Punkt tokenizer on a custom corpus

TF-IDF (Term Frequency-Inverse Document Frequency)

→ Frequency of a word in a document.

→ Penalizes common words across documents.

O/P: Sparse matrix where each row is a document and each column is a weighted word feature

$$TF(t,d) = \frac{(\text{No. of occurance of term } t \text{ in document } d)}{(\text{Total no. of terms in the document } d)}$$

$$IDF(t,D) = \log_e \frac{(\text{Total no. of documents in the corpus})}{(\text{no. of documents with term } t \text{ in them})}$$

$$TF-IDF(t,d,D) = [TF(t,d)] \times [IDF(t,D)]$$

Word indexes: {'geeks': 1, 'for': 0, 'r2j': 2}

Doc ~~index~~ → ~~d0~~ d ~~d2~~

tf-idf value:

(0, 0) 0.5493 - -
(0, 1) 0.8355 - - → tf-df value of a word having index 1 i.e. geeks in document index 0
(1, 1) 1.0
(2, 2) 1.0

(Document index, word index)

Eg: Doc1 : The Cat sat on the mat.
Doc2 : the dog played in the park.
Doc3 : Cats and dogs are great pets.

In doc 1, word "cat" appears 1
total no. of terms in doc 1 $\rightarrow$ 6
so, $TF(cat, Doc1) = 1/6$

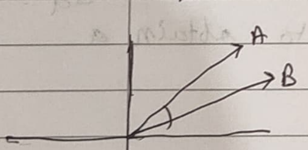
In doc 2, $TF(cat, Doc2) = 0$
In doc 3, $TF(cat, Doc3) = 1/6$

Total no. of doc in the corpus $= (D) : 3$
no. of doc containing term "cat" : 2 (Doc1 & 3)

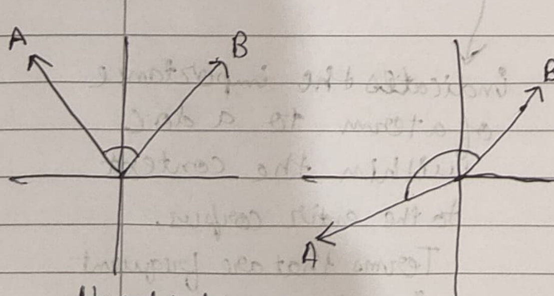
$$IDF(cat, D) = \log\left(\frac{3}{2}\right) \approx 0.176$$

Cosine similarity : measure similarity b/w two vectors using cosine of the angle b/w them

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

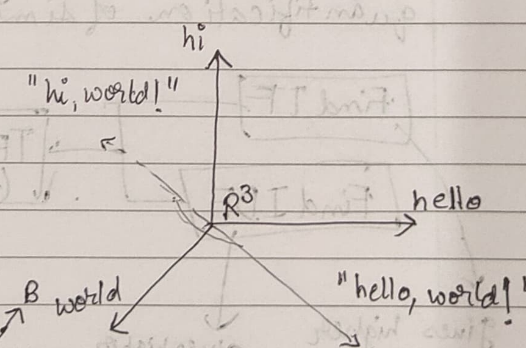


Similar



Unrelated

Opposite



Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically it calculates the cosine of the angle b/w two vectors

Advantage

even if the two similar doc are far part by the Euclidean distance ~~bc~~ coz of the size (eg: word "yeah" could appear 50 times in one doc and 10 times in another) they could still have a smaller angle b/w them.

Smaller the angle = ↑ similarity

Working

~~words~~ | words/phrases → convert → vector form

convert → using TF-IDF

→ The vector presentation of the doc can then be used within cosine similarity formula to obtain a quantification of similarity

Find TF

Find IDF

→ TF-IDF = (TF) X (IDF)
(Score)

Find TF → gives higher score to words that are central to doc's theme

Find IDF → gives higher score to term that are uncommon across the whole dataset

TF-IDF → indicates the importance of a term to a doc within the context to the entir corpus. Terms that are frequent in a doc but rare in the overall dataset recieve highest TF-IDF score

→ For each doc, a vector is created where each dimension represents a unique word from the entire vocabulary of the corpus.

→ The value at each dimension (pos) in the vector is the TF-IDF score for that specific word in that doc

→ This process results in a numerical vector for each doc, allowing mathematical comparison.