

NLP Task: POS Tagging and Document Ranking using TF-IDF and cosine similarity

Aim:

- To perform part of speech (POS) tagging on a given text using Spacy.
- To rank a set of documents based on their relevance to a user query using TF-IDF.

Procedure:

- Install and load the spacy NLP pipeline to perform pos tagging on each word.
- Take a user query asking how AI supports students in learning.
 - combine the doc and the query into a single corpus.
 - use TfidfVectorizer to transform the text corpus into numerical vectors.
 - compute cosine similarity to transform the text corpus into numerical vectors.
 - compute Rank the documents based on similarity scores in descending order.
 - display the pos tags for the input text and the ranked list of relevant documents.

AI - propn

driven \rightarrow Verb

platform \rightarrow noun

personalize \rightarrow verb

learning \rightarrow verb

Score

0.16 \rightarrow AI helps automatic grading
and administrative task in schools.

Score : 0.10 \rightarrow

Intelligent tutoring system
adapt to each students learning
"style".

Program:

```
package to install  
pip uninstall spacy  
python -m spacy download en-core-web-sm  
import spacy  
nlp = spacy.load("en-core-web-sm")  
text = "AI-driven platforms personalize learning  
path and help students grasp  
concepts faster".
```

```
doc = nlp(text)
```

```
for token in doc:
```

```
    print(f" {token.text} : isy -> {token.pos_tag} ")
```

```
from sklearn.feature_extraction.text import Tfidf  
vectorizer
```

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
documents = [
```

```
    "AI tools analyze student performance and  
    provide
```

```
    real-time feedback".
```

```
    "Intelligent tutoring systems adapt to  
    each student's learning style".
```

```
    "Virtual classrooms by AI enhance student engagement"]
```

```
query = "How does AI support students in  
learning?".
```

```
corpus = documents + [query]
```

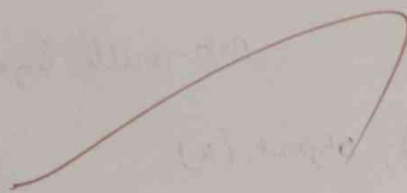
```
vectorizer = TfidfVectorizer()
```

```

tfidf-matrix = vectorizer.fit_transform (corpus)
similarities = cosine_similarity (tfidf-matrix [:],
                                tfidf-matrix [:])
ranked-docs = sorted (zip (similarities,
                           documents), reverse=True)

print ("In Top relevant documents: \n")
for score, doc in ranked-docs:
    print ("score: {score:2f} -> {doc}")

```



Result: ~~The~~ The system accurately tags each word in the input text with the grammatical role, enhancing understanding of sentence structure successfully.