Ex No: 5 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 TDF-IDF vectorizer 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 txt and the ranked list of relevant documents.

O/P :

AI - propn

driven → Verb

platforms → noun

personalize → verb

learning → verb

Score : 0.16 → AI helps automatic grading and administrative task in schools.

Score : 0.10 → Intelligent tutoring system adapt to each students learning style.

Program:

```
package to install
pip install 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:15} -> {token.pos_}")

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine_similarity.

documents = [
    "AI tools analyze student performance and
                                    provide
     real_time feedback".

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

"Virtual classrooms by AI chance 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[-1],
                tfidf_matrix[:-1]).flatten()
ranked_docs = sorted(zip(similarities,
                documents), reverse=True)
print("\n Top relevant documents:\n")
for score, doc in ranked_docs:
    print(f"Score: {score:.2f} -> {doc}")
```

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