Movie recommender systems

Sunday, October 26, 2025 6:37 PM

Based on my past browsing history, movie's are recommended to me. This kind of recommender systems are used everywhere. (End to End).

Types of recommender Systems :-

1. Content Based : Based on similarity of content - It would recommend. (If listening to love songs, you would be recommended lovesongs. Diff tags are made for for Different tags.
2. Collaborative filtering based:- Based on User Interest, We recommend System. Say User A & B. We say, A & B have very similar movie interests. So if A likes 'M2'. Hence B would also be recommended M2.
3. Hybrid :- Mixes both of them.

In this project we are gonna do 1. Content Based.

PROJECT FLOW :-

DATA → PRE-PROCESSING → MODEL BUILDING → CONVERT INTO WEBSITE → DEPLOY

VECTORIZATION (COMES UNDER PRE-PROCESSING)

This is the main concept which we are going to learn.

After a lot of transformation has been done, now we are going to work on this Dataset where the column headers are id, title, tags.

(1) First we do stemming. This is done to create 2 separate arrays of same meaning word. For example, "missed" and "missing" both mean the same thing. So we stem it to stem.

Code for this:

```
import nltk
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def stem(text):
    y = []
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
df['tags'].apply(stem)
```

df [tags] ...

(2) The Next Important thing is to convert the 'tags' column into Vectors. The reason for this would be explained later.
There are Different ways we can do Vectorizat", the way we are going to apply is by using 'Bag of words'.
Current the shape of Dataset is (4806 x 3).
We will be mixing all the tags (ie all 4806 rows of tags) and create a very huge set of words.
From here we would be choosing top n repeating words (n depends on us.)
Say we choose n = 5000.
Now what the Algo does it goes thru each tag and creates a vector of 5000 where it tells in that tag how many times it has repeated.

	w_1	w_2	w_3 ..	w_{5000}
T_1	0	1	2	0
T_2	1	1	0	5
T_3	0	0	0	6
⋮				
T_{4806}				

Say w_1 → Action then in tag1 it has repeated 0 times and 1 time in T_2.
We also have to make sure that we do not include stop words. ie, and, are, of etc.
CODE:
... Count Vectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000, stop_words='english')
vectors = cv.fit_transform(new_df['tags']).toarray()
```

Convert a collection of text documents to a matrix of token counts. This would produce a sparse representation of "scipy.sparse.csr_matrix."