Movie Recommendation System

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Namespace Documentation

2.1 db_movie Namespace Reference

Functions

- def root_word (words)

  Takes a list of words and convert them to their root word.
- def search_story (rooted_word)

  Search in story line for given token and find best match movies possible.
- def search_across_token (lst)

  Find intersection across multiple list wherever possible.
- def search_title_actor_char (word)

  Searche in title actor and character seperately and return their search reslut.

Variables

- con = sqlite3.connect("IMDB.db")

  creating database and connecting to it
- cur = con.cursor()

  obtaining cursor on database to query database
- infile = open('data3.csv','r')

  creating table MoviesData
- **dr** = csv.reader(infile)
- list **to_db** = [(i[0], i[1] , i[2], i[3], i[4],i[5], i[6], i[7] , i[8], i[9], i[10], i[11], i[12], i[13], i[14], i[15], i[16],i[17], i[18], i[19] , i[20], i[21], i[22],i[23], i[24], i[25] , i[26], i[27], i[28],i[29], i[30], i[31], i[32],i[33],i[34],i[35])for i in dr]
- title = cur.execute(,(sys.argv[1],)).fetchall()

  inserting values into table MoviesData
- stop_words = set(stopwords.words('english'))

  Tokenizing search query and removing stop words.
- **word_tokens** = word_tokenize(sys.argv[1].lower())
- list **words** = [w for w in word_tokens if not w in stop_words]
- list **genre** = ['action','adult','adventure','animation','biography','comedy','crime','documentry','drama','family','fantasy', 'filmnoir','gameshow','history','horror','music','musical','mystry','news','realitytv','romance','scifi','short','sport','talkshow','thriller','w

- string **query** = 'select title, year from MoviesData where ('
- list **target_list** = [[ ],[ ],[ ],[ ],[ ]]
- list story = [ ]

  If more than 5 tokens then searching only in story-line.
- def **rooted_word** = root_word(words)
- word = int(words[i])

  Searching for each word in query seperately and storing their results in target_list.
- **l** = cur.execute(,(word,)).fetchall()
- **actor**
- **char**
- list res = [ ]

  finding intersection across all the list found for different words

2.1.1 Detailed Description

```
program to create database of movies and
to query for specific movies based on random search query
packages imported: sys, sqlite3, csv, nltk.stem, nltk.corpus, nltk.tokenize

Functions defined:
root_word
search_story
search_across_token
search_title_actor_char
```

2.1.2 Function Documentation

2.1.2.1 root_word()

```
def db_movie.root_word (
                words )
```

Takes a list of words and convert them
to their root word.

Parameters

words	list of words to be converted to their root word

Returns

rooted_word list of root words

Definition at line 23 of file db_movie.py.

```
23 def root_word(words):
24     ps = PorterStemmer()
25     rooted_word=[]
26     for w in words:
27         rooted_word.append(ps.stem(w))
28     return rooted_word
29
```

2.1.2.2 search_across_token()

```
def db_movie.search_across_token (
              lst )
```

Find intersection across multiple list wherever possible.

Parameters

lst	a 2-d list results corresponding to diffrent tokens in search query

Returns

res list of movies common across different tokens

Definition at line 51 of file db_movie.py.

```
51 def search_across_token(lst):
52     res=lst[0]
53     for i in range(1,len(lst)):
54         if lst[i]:
55             temp=[v for v in res if v in lst[i]]
56             if temp:
57                 res=temp
58             else:
59                 res.extend(lst[i])
60     return res
61
```

2.1.2.3 search_story()

```
def db_movie.search_story (
              rooted_word )
```

Search in story line for given token
and find best match movies possible.

Parameters

rooted_word	list of root word

Returns

story movies based on story

Definition at line 34 of file db_movie.py.

```
34 def search_story(rooted_word):
35     story=[]
36     for w in rooted_word:
37         tmp=cur.execute(''' select title, year from MoviesData where storyline like '%'||?||'%';''',(w+' ',
      )).fetchall()
```

```
38         if story:
39             tmp2=[v for v in story if v in tmp]
40             if tmp2:
41                 story=tmp2
42             else:
43                 story.extend(tmp)
44         else:
45             story=tmp
46     return story
47
```

2.1.2.4 search_title_actor_char()

```
def db_movie.search_title_actor_char (
               word )
```

Searche in title actor and character seperately
and return their search reslut.

Parameters

word	word to be searched in title actor and character

Returns

title list of movies found from title
actor list of movies found from actor
char list of movies found from character

Definition at line 68 of file db_movie.py.

```
68 def search_title_actor_char(word):
69     title,actor,char=[],[],[]
70     title=cur.execute(''' select title, year from MoviesData where title like '%'||?||'%';''',(word,)).
       fetchall()
71     actor=cur.execute(''' select title, year from MoviesData where actor like '%'||?||'%';''',(word+' ',)).
       fetchall()
72     char=cur.execute(''' select title, year from MoviesData where character like '%'||?||'%';''',(word+' ',
       )).fetchall()
73     return title,actor,char
74
```

2.1.3 Variable Documentation

2.1.3.1 infile

```
db_movie.infile = open('data3.csv','r')
```

creating table MoviesData

reading csv file containing data about movies

Definition at line 83 of file db_movie.py.

2.1.3.2 title

```
db_movie.title = cur.execute(,(sys.argv[1],)).fetchall()
```

inserting values into table MoviesData

Assuming movie name has been entered.Searching for entire movie

Definition at line 90 of file db_movie.py.

Index