MOVIE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

18CSE484T-E -DEEP LEARNING

Submitted by

PAIDIPATI CHARAN RAM [RA2011027010186]
PAIDIPATI LAKSHMAN [RA2011027010202]

Under the guidance of

Dr. Mercy Theresa

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

with specializtion in Big Data Analyics

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"MOVIE RECOMMENDATION SYSTEM"** is the bona fide workof PAIDIPATI CHARAN RAM [RA2011027010186], PAIDIPATI LAKSHMAN [RA2011027010202] who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Mercy Theresa
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

TABLE OF CONTENTS

CHAPTERS CONTENTS PAGE NO.

MOVIE RECOMMENDATION SYSTEM

Abstract

Recommender System is a tool helping users find content and overcome information overload. It predicts interests of users and makes recommendation according to the interest model of users.

The original content-based recommender system is the continuation and development of collaborative filtering, which doesn't need the user's evaluation for items. Instead, the similarity is calculated based on the information of items that are chose by users, and then make the recommendation accordingly. With the improvement of machine learning, current content-based recommender system can build profile for users and products respectively. Building or updating the profile according to the analysis of items that are bought or visited by users. The system can compare the user and the profile of items and then recommend the most similar products. So this recommender method that compare user and product directly cannot be brought into collaborative filtering model. The foundation of content-based algorithm is acquisition and quantitative analysis of the content. As the research of acquisition and filtering of text information are mature, many current content-based recommender systems make recommendation according to the analysis of text information.

This paper introduces content-based recommender system for the movie website of VionLabs. There are a lot of features extracted from the movie, they are diversity and unique, which is also the difference from other recommender systems. We use these features to construct movie model and calculate similarity. We introduce a new approach for setting weight of features, which improves the representative of movies. Finally we evaluate the approach to illustrate the improvement.

2. Introduction

As the development of information technology and Internet, people enter the era of information overload from that of information deficiency gradually[17]. The report is the result of the Master Thesis in Information and Communication Technology School at Royal Institute of Technology in Stockholm, Sweden. The project is provided in VionLabs AB that is a media-tech company providing media content such as movie information for customers. The recommender system I implemented is for the movie website Vionel.com.

2.1 Background

In the era of information overload, it is very difficult for users to get information that they are really interested in. And for the content provider, it is also very hard for them to make their content stand out from the crowd. That is why many researchers and companies develop Recommender System to solve the contradiction. The mission of Recommender System is to connect users and information, which in one way helps users to find information valuable to them and in another way push the information to specific users. This is the win-win situation for both customers and content providers.

VionLabs is a media-tech startup company. The company provides a new way on how consumers are given access to good and suitable content. The mission of VionLabs is to increase needs of its digital user base. Vionel is the movie website developed by VionLabs, which is a place for people who love movies can gather all the information about films in one place[5]. This thesis report will present a more practical recommendation method that can be used on a movie website that does not have enough users.

2.2 Problem Statement

For building a recommender system from scratch, we face several different problems. Currently there are a lot of recommender systems based on the user information, so what should we do if the website has not gotten enough users. After that, we will solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies. But for each feature of the movie, there should be different weight for them and each of them plays a different role for recommendation. So we get these questions:

- How to recommend movies when there are no user information.
- What kind of movie features can be used for the recommender system.

3. Literature survey

Recommender system is a very hot research topic in recent years. Many researchers raised a lot of different recommendation approaches. The most famous category of these approaches is:

- Content-based Recommendation.
- Collaborative-filtering Recommendation.
- Hybrid Recommendation.

3.1 Content-based Recommendation

Content-based recommendation is an important approach in recommender systems. The basic idea is to recommend items that are similar with what user liked before[21]. The core mission of content-based recommender system is to calculate the similarity between items. There are a lot of methods to model item and the most famous one is Vector Space Model[2]. The model extracts keywords of the item and calculate the weight by TF-IDF. For example, set k_i as the ith keyword of item d_j, w_{ij} is the weight of k_i for d_j, then the content of d_j can be defined as:

$$Content(d_j) = \{w_{1j}, w_{2j}, ...\}$$

As we talked before, content-based recommender system recommends items that are similar with what user liked before. So the tastes of a user can be modeled according to the history of what the user liked. Consider *ContentBasedProfile*(u) as the preference vector of user u, the definition is:

$$ContentBasedProfile(u) = \frac{1}{|N(u)|} sum_{d \in N(u)} Content(d)$$

$N(u)$ is what the user u liked before. After calculating content vector *Content*(.) and content preference vector *ContentBasedProfile*(.) of all users, given any user u and an item d, how the user like the item is defined as the similarity between *ContentBasedProfile*(u) and *Content*(d):

$$p(u,d) = sim(ContentBasedProfile(u), Content(d))$$

Using keywords to model item is an important step for many recommender systems. But extracting keywords of an item is also a difficult problem, especially in media field, because it is very hard to extract text keywords from a video. For solving this kind of

problem, there are two main ways. One is letting experts tag the items and another one is letting users tag them. The representative of experttagged systems are Pandora for music and Jinni for movies. Let's take Jinni as an example, the researchers of Jinni defined more than 900 tags as movie gene, and they let movie experts to make tags for them. These tags belong to different categories, including movie genre, plot, time, location and cast. Figure 2.1 is from *Jinni*, which are the tags for movie Kung Fu Panda. As we can see from the figure, the tags of Kung Fu Panda are divided into ten categories totally, *Mood*, *Plot*, *Genres*, *Time*, *Place*, *Audience*, *Praise*, *Style*, *Attitudes* and *Look*. These tags contain all aspects of movie information, which can describe a movie very accurately.

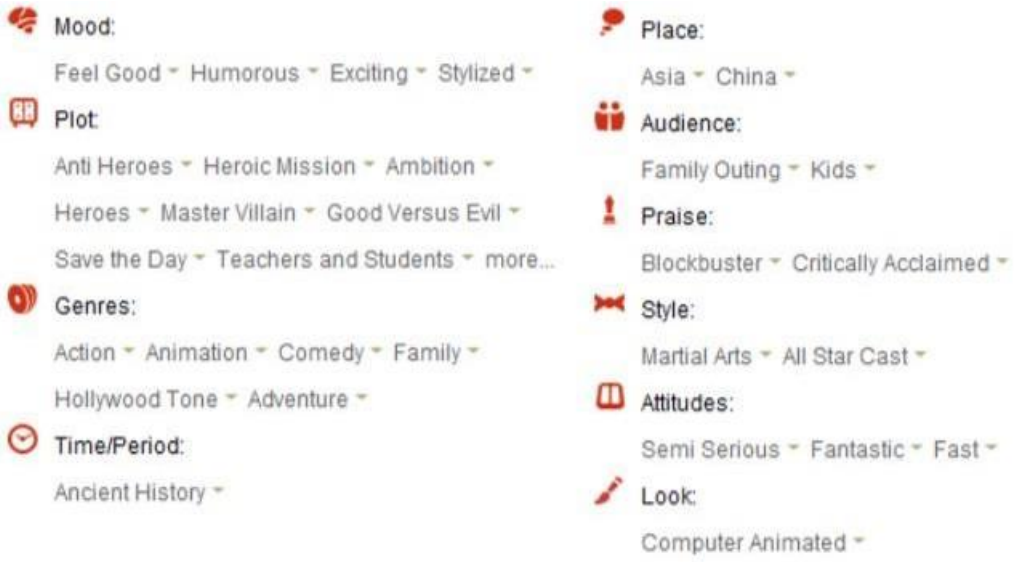


Figure 2.1. Tags for Kung Fu Panda

Compared with expert-tagged system, user-tagged system is applied more widely. The representative websites are *Delicious* and *Flickr*. The feature of user-tagged system is the tags are more diversity than that of expert-tagged system. But the weakness is that the tags are of lower quality, even there are a lot of wrong tags. So in the user-tagged systems, there are two main problems, one is tag recommendation[34], which means when a user tags an item, the system can recommend some relative

tags for him to choose. The purpose is first to be convenience for users and second it can increase the quality of tags. Another question is how to recommend items based on tags(tag-based recommendation[33]). After items are tagged, the simplest recommendation approach is to use tags as keywords of the item, and recommend by the content-based algorithm.

3.2 Collaborative-filtering Recommendation

Collaborative-filtering recommendation is the most famous algorithm in recommender systems. This algorithm models user's taste according to the history of user behavior. GroupLens published the first paper[31] about collaborative filtering and the paper raised user-based collaborative filtering. In 2000, Amazon came up with item-based collaborative filtering in their paper[20]. These two algorithms are very famous in business recommender systems.

3.2.1 User-based collaborative-filtering

In user-based collaborative filtering, it is considered that a user will like the items that are liked by users with whom have similar taste. So the first step of user-based collaborative-filtering is to find users with similar taste. In collaborative filtering, the users are considered similar when they like similar items. Simply speaking, given user u and v, $N(u)$ and $N(v)$ are items set liked by u and v respectively. So the similarity of u and v can be simply defined as:

$$s_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

There are a lot of similarity algorithm, Equation 2.4 is one of them. User u's likeability for item i can be calculated by:

$$p_{ui} = \sum_{v \in S(u,k) \cap N(i)} s_{uv} p_{vi}$$

User/Item	Item A	Item B	Item C	Item D
User A	!		!	recommend
User B		!		
User C	!		!	!

Table 3.1. User-based CF

Table 2.1 is an example of User-based CF recommendation. According to the interest history of User A, only User C can be the neighbor of him, so Item D will be recommended to User A.

3.2.2 Item-based collaborative-filtering

Item-based collaborative-filtering is different, it assumes users will like items that are similar with items that the user liked before. So the first step of item-based collaborative-filtering is to find out items that are similar with what the user liked before. The core

point of item-based collaborative-filtering is to calculate the similarity of two items. Item CF considers that items that are liked by more same users, the more similar they are. Assume $N(i)$ and $N(j)$ are user sets who like i and j respectively. So the similarity of i and j can be defined as: $|N(i) \cap N(j)|$

$$s_{ij} = \frac{}{|N(i) \cup N(j)|}$$

User u's likeability of item i can be calculated by:

$$p_{ui} = \sum_{j \in S(i,k) \cap N(u)} s_{ij} p_{uj}$$

Table 2.2 is an example of Item-based CF recommendation. According to the interest history of all the users for Item A, people who like Item A like Item C as well, so we can conclude that Item A is similar with Item C. While User C likes Item A, so we can deduce that perhaps User C likes Item C as well.

User/Item	Item A	Item B	Item C
User A	!		!
User B	!	!	!
User C	!		recommend

Table 3.2. Item-based CF

User-based and Item-based collaborative-filtering algorithms are all neighborhoodbased algorithm, there are also a lot of other collaborative-filtering algorithms. Hoffman raised Latent Class Model in this paper[13], the model connects user and item by latent class, which considers that a user will not become interested in items directly. Instead, a user is interested in several categories that contain items, so the model will learn to create the categories according to user's behavior. On top of Latent Class Model, researchers came up with Matrix Decomposition Model, which is called Latent Factor Model[4] as well. There are a lot of models based on matrix decomposition and they mostly came from Netflix Prize Competition, such as RSVD[28], SVD++[18] and so on.

Besides Matrix Decomposition Model, Graph Model is widely applied in collaborativefiltering. Baluja introduced graph model of co-view behind the recommender algorithm of YouTube in [3] and also raised a broadcast algorithm on graph to measure how much a user like an item. This literature[27] research how to increase serendipity of recommendation result by means of the analysis of the path between nodes

in the graph. Mirza[26] systematically studied recommendation problems based on graph model and point out the essence of the recommendation is to connect user and item. The graph is the natural method for that. [10] studies similarity algorithms between the nodes of the graph and compares the recommendation precision of different algorithms.

Figure 3.2. Personalized recommendation of Amazon

Figure 3.3. Relevant Recommendation, Customers Who Bought This Item Also Bought

4. Requirement Analysis

The datasets in this project are from Vionel database and public place such as Wikipedia. The data is in JSON format and each JSON object represents one movie information. We store the whole data in MongoDB, which is a cross-platform document-oriented database.

Each item of the data has a IMDb id as a unique identifier, which in the future will be replaced by our vionel id. There are now 8 features that are used in the project. They are *director*, *actor*, *genre*, *keyword*, *theme*, *scene* and *location*. Note that not all movies have complete features, some of the movies may miss some feature information, and this fact has been considered in the algorithm.

4.1 Feature Extraction

There are a lot of benefits to represent document as vector space model, for example we can measure cosine similarity and even extent to Clustering and Classification. However, vector space model does not have the ability to solve these two typical problems: a word has multiple meaning and a meaning has multiple words[22].

We ever consider to build a thesaurus, but it is too much time consuming. There are two mainstream approach, one is lexical co-occurrence and another is using shallow parsing to analyze the grammatical relation or syntax dependence between vocabulary[22]. Generally speaking, lexical co-occurrence is more robust and grammatical relation is more accurate.

In the previous vector space, we only pay attention on the frequency of a single word. But the word co-occurrence is a very important information as well, which based on a fact that the appearance of two or more co-occurrence words in the document is not occasional. Latent Semantic Indexing is an approach to explore

the inner semantic relations[9][23]. The latent semantic indexing is to map the co-occurrence words to the same dimensional space. The co-occurrence words are considered semantically related.

Compared with our previous vector space, latent semantic space has less dimensions. That is because many words are mapped to the same dimension. Therefore, latent semantic indexing is a kind of dimensionality reduction method. The process of dimensionality reduction is mapping objects in high-dimensional space to lowdimensional space.

We can transform previous vector to term-document matrix, which is a $M \times N$ matrix C consists of M terms and N documents. Each row of the matrix represents a term and each column of the matrix represents a document.

Singular Value Decomposition

Singular Value Decomposition is an important matrix decomposition in linear algebra. Let us see a theorem first.

Assume r is the *rank* of $M \times N$ matrix C, so the SVD(Singular Value Decomposition) of matrix C is:

$$C = USV^T$$

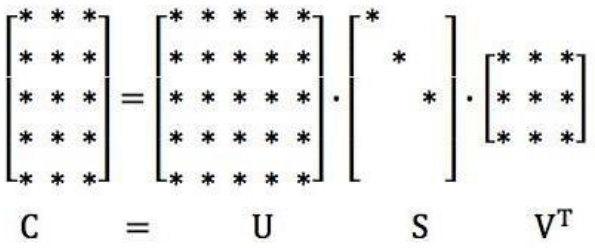


Figure 4.1. Singular Value Decomposition

Thereinto, the columns of U are orthonormal and so are columns of V . S is a diagonal matrix and elements on diagonal are singular value of C.

Matrix Low Rank Approximation

This is the process to solve matrix low rank approximation problem:

1. Given C, construct SVD by Equation 3.1 and decompose it: $C = USV^T$
2. Set $r - k$ minimum singular value on the diagonal of S zero, then get S_k
3. Calculate $C_k = US_kV^T$ as the approximation of C

Application of LSI

Matrix $S_k V^T$ is $k \times N$, k is the rank after using LSI, N is the total number of documents. Each column of matrix is the new coordinate of corresponding document on dimension-reduced space.

SVD is first raised by Deerwester[9] in Information Retrieval. LSI represent the documents on a new dimension-reduced space, which is actually the linear combination of each dimension on old space. It can be considered as a soft clustering.

Feature extraction is not the main task of the thesis, so I just introduced the basic principle simply, which is with the help of other research team in VionLabs.

Feature Representation

Vector Space Model

Before being deep into feature representation, it is very important to get the idea of document representation. Assume there are several documents and each document consists of one single sentence. Then we can represent the document as a model in Figure 3.2, which is called Vector Space Model. Consider each feature of a movie as a term, then a feature can represented by this model. It is obviously that some features are more important than others and the importance is the weight in the similarity calculation.

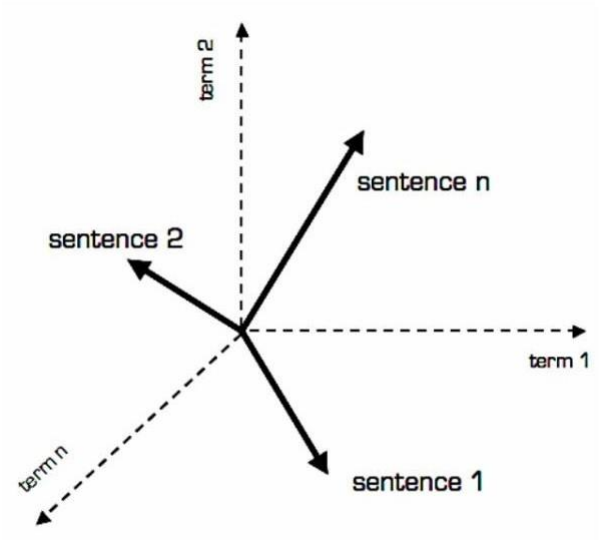


Figure 4.2. Vector Space Model of documents

4.2 TF-IDF

TF-IDF is short for term frequency-inverse document frequency that is a common weighting technique for information retrieval and text mining, which reflects how important a word is for a document[29]. The importance of a word increases proportionally to the times the word appears in the document, but it also decreases inversely proportional to the frequency the word appears in the whole corpus.

TF-IDF consists of TF and IDF, which are Term Frequency and Inverse Document Frequency respectively. TF represents the frequency a word appears in the document. The main idea of IDF is: if a term appears more in other documents, the term will be less important.

4.2.1 Term Frequency

Term frequency is about how many times a term t_i appears in document d_j, which can be represented by $TF(t_{ij})$. In the condition of removing stop-words, the more t_i appears in the document, the more important term t_i is for the document. It can be defined as:

$$TF(t_{ij}) = \frac{N(t_i, d_j)}{N(d_j)}$$

$N(t_i, d_j)$ is the number of times t_i appears in d_j and $N(d_j)$ is the total number of terms in document d_j

4.2.2 Inverse Document Frequency

In order to understand inverse document frequency, let us see what document frequency is. Document frequency is how many times term t_i appears in all documents C, which is represented by $N(t_i, C)$. The more term t_i appears in all documents C, the weaker term t_i can represent document d_j.

Inverse document frequency means that the represent ability of term t_i for document d_j and its amount in all documents $N(t_i, C)$ is inverse proportion, which is represented by $IDF(t_i)$:

$$IDF(t_i) = \log \frac{N(C)}{N(t_i, C)}$$

$N(C)$ is the total amount of documents, $IDF(t_i)$ decrease with the increase of $N(t_i,C)$. The less $N(t_i,C)$ is, the more representative t_i is for d_j.

3.5. WEAKNESS OF TF-IDF

4.2.3 Normalization

In order to reduce the inhibition of stop words, we will normalize each variable. After normalization, the calculation of $TF - IDF$ is:

$$weight_{TF-IDF}(t_{ij}) = \frac{TF(t_{ij}) \times IDF(t_i)}{\sqrt{\sum_{j=1}^{n} [TF(t_{ij}) \times IDF(t_i)]^2}}$$

Equation 3.4 base on the principle: The term that is more representative for a document is the word that appears in the document more often and less often in other documents.

4.2.4 Distribution Coefficient

If the term t_i is distributed evenly in every document of a category, it is obvious that t_i is representative for the category. We define a distributed coefficient:

$$DC = 1 - \frac{\sqrt{\frac{1}{N(C_l)-1} \sum_{j=1}^{N(C_l)} [TF(t_{ij}) - \overline{TF(t_i,C_l)}]^2}}{N(C_l) \times TF(t_i,C_l)^2}$$

$\overline{TF(t_i,C_l)}$ is the average number of term t_i appearing in documents of category C_l.

$$\overline{TF(t_i,C_l)} = \frac{1}{N(C_l)} \sum_{j=1}^{N(C_l)} TF(t_{ij})$$

When t_i appears in every document of C_l, it is obvious that $TF(t_{ij}) = \overline{TF(t_i,C_l)}$. Then DC

5. Dataset Description

In this chapter, we will introduce how to implement the content-based recommender system based on the principle mentioned. After that, we will test the system and give out the result to prove the improvement of our system.

Dataset

All the movie data we used is from IMDb, Wikipedia and our own database in VionLabs. In the end we get 178356 movies and related information. For the perspective of recommender system, a movie can be described by a collection of features, which can be genres, actors, directors and so on.

- *Director*: The director is from IMDb, most of movies only have one director, but some of them have two or more.
- *Actor*: A movie normally has a lot of actors, but most of them is useless for recommender system and bring disadvantageous effects. So we only get three main actors for one movie. They are from IMDb as well.
- *Keyword*: We use LSI to extract keywords from Wikipedia plot, which is under the help of the colleagues in VionLabs.
- *Release Year*: This is when the movie is released and data is from IMDb.
- *Vionel Theme*: Theme is a kind of keyword that describes movies in a different perspective, such as *Time Travel* and *Comic Book*. They are defined by VionLabs.
- *Language*: Language is from IMDb, which is the language that occurs in the movie.
- *Location*: Location is from IMDb, which is where the movie happens.
- *Vionel Scene*: Scene of the movie is analyzed by other research in our team. We will recognize the background of every frame in the movie by machine learning. For example, bar, hall room, store are what we recognized.

In the scenario of movie, we will divide the movies into 23 categories by the normal genres. Table 4.1 shows the categories we used. Each movie in the case is a document, which is represented by the eight features described in Section 4.1. As we said before, the movie is represented by Vector Space Model, each feature for the movie is a term in the document.

Sci-Fi	Crime	Romance	Animation	Music
Comedy	War	Horror	Adventure	News
Biography	Thriller	Western	Mystery	Short
Drama	Action	Documentary	Musical	History
Family	Fantasy	Sport		

Table 5.1. Categories

In many other content-based recommender systems, the genre is used as vector to calculate similarity. But this is only one aspect of the movie and there are a lot of other features of movie such as background, actor, etc. So we add more features and some of them are very unique because they are extracted by our own research.

But we didn't simply add features together to calculate TF-IDF, we have discussed the reasons in Section 3.5. The genre is the natural feature that we can use as category. Figure 4.1 shows the distribution of genres in the movie database. Compared with the principle mentioned before, each genre is a category for documents. The number of document in each category is shown in the figure. Each document contains many terms which are features in our case, they are described in Section

As we discussed before, the document in our case is the movie which contains many features. The movie will be represented by vector space model in the experiment. In Section 4.1, we introduced the features that are used to model the movie. The vector space model is like this kind of format:

$$MovieModel = [Directors, Actors, Keywords, ReleaseYear, VionelThemes, Languages, Locations, VionelScenes]$$

(

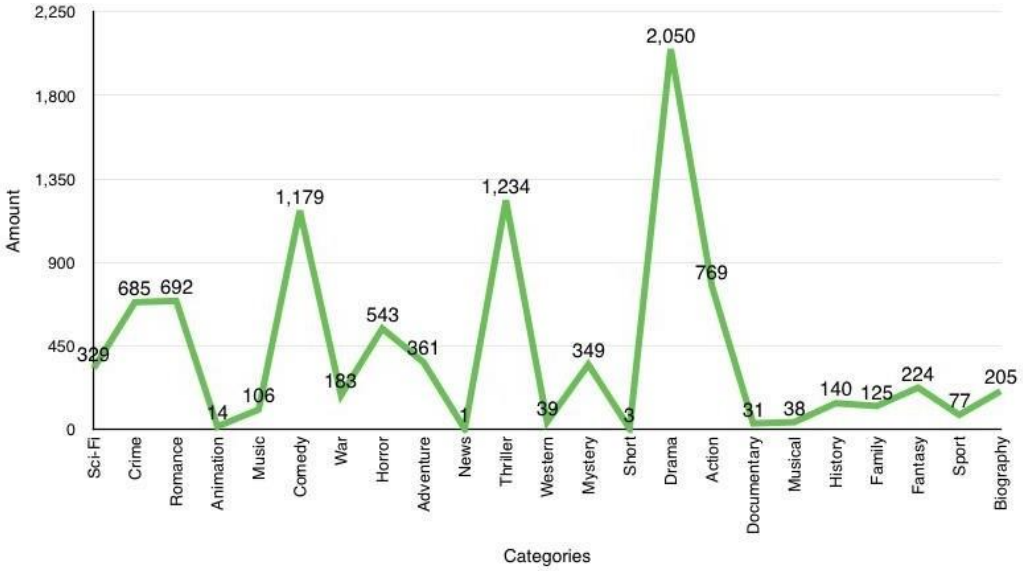


Figure 5.1. Category Distribution

A movie can have multiple directors and actors, so the vector is pretty long generally. Here we use movie *The Dark Knight* to illustrate the model.

Directors	Christopher Nolan
Actors	Christian Bale Heath Ledger Aaron Eckhart
Keywords	Keywords
Release Year	2008
Vionel Themes	Business and finance Comic book Bromance Cars and racing Mafia & Gangsters Revenge
Language	English, Mandarin
Location	USA, UK
Vionel Scene	Basement, Bar

Table 5.2. Information of The Dark Knight

Table 5.2 is the basic information of movie *The Dark Knight*. I don't list keywords in the table because there are 63 keywords for the movie, which is difficult

to show them in the table.

We can get a very long vector after the calculation according to Equation 3.14, which is the model for the movie. Each weight represents the importance of a feature for the movie. In order to show it intuitively, I present it in Figure 4.2.

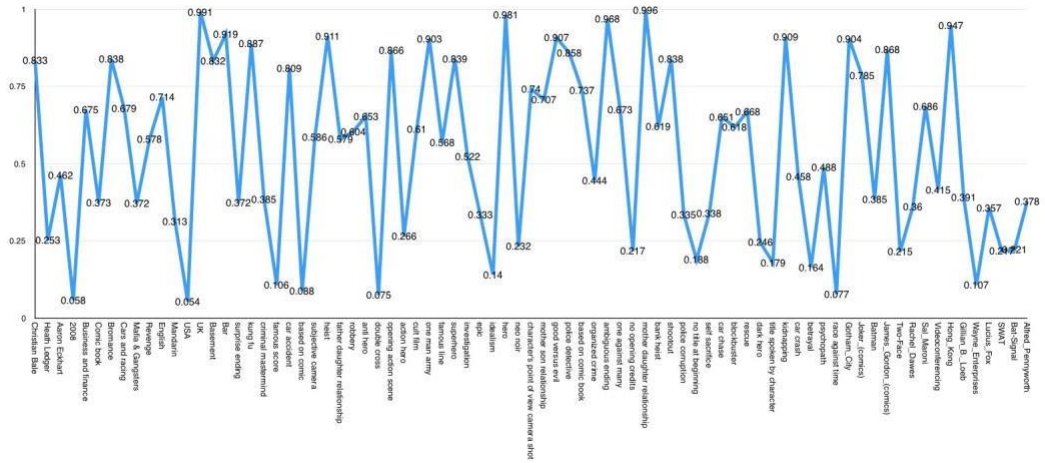
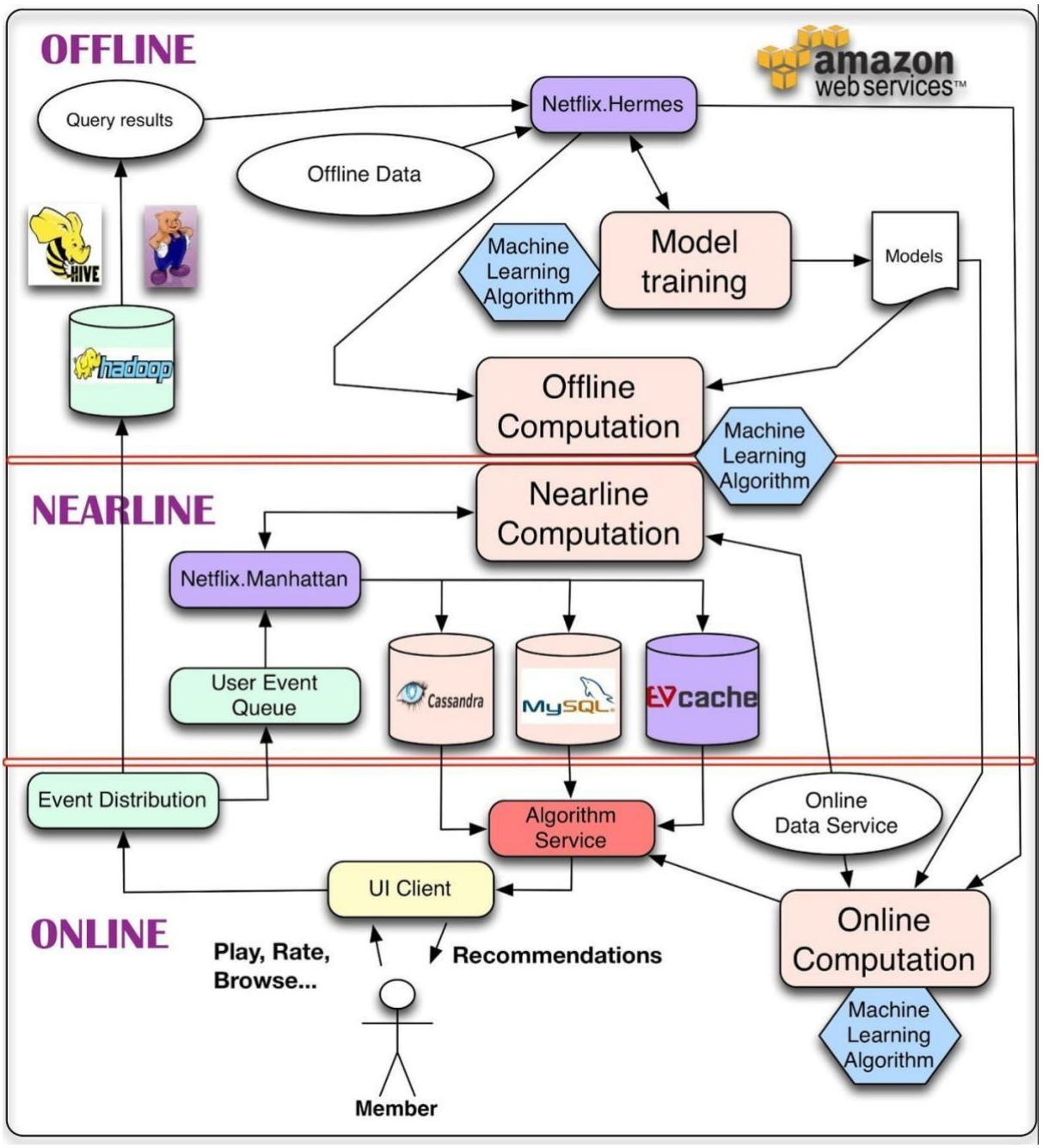


Figure 5.3. Feature Weight of The Dark Knight

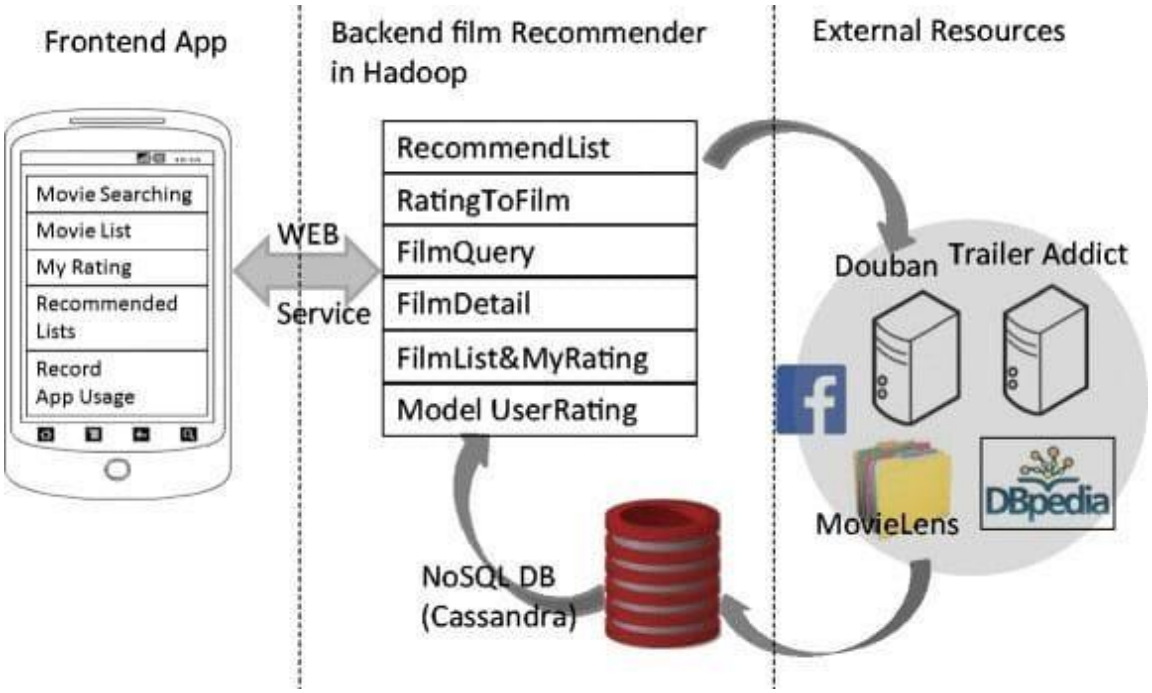
There are 80 features totally for movie *The Dark Knight*, the number is calculated by our TF-IIDF-DC, which shows the importance of each feature. From this perspective, we can see that if features of a movie have similar distribution, it means that the two movies are similar.

Actually, one feature is one dimension for the model, I list all features in one dimension in the figure just because multiple-dimension is hard to show by figure.

6. Architecture & Design



OVERLOOK :


Frontend App
Backend film Recommender in Hadoop
External Resources
Movie Searching
Movie List
My Rating
Recommended Lists
Record App Usage
WEB
Service
RecommendList
RatingToFilm
FilmQuery
FilmDetail
FilmList&MyRating
Model UserRating
Douban
Trailer Addict
MovieLens
DBpedia
NoSQL DB
(Cassandra)

7.Implementation

In order to illustrate the improvement of our recommender system, we will firstly prove our improved TF-IDF can generate better weight for features of the movie. In this chapter, we introduce a classification approach to validate the effect of movie model that is generated by different weight.

7.1 k-NN

k-Nearest Neighbors is a classical algorithm in Machine Learning. *k* represents its nearest *k* items.

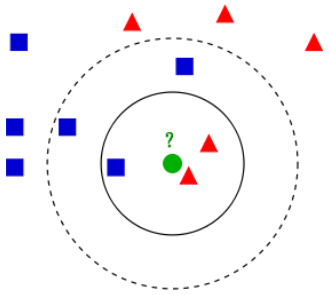


Figure 7.1. k-NN

Figure 5.1 is from Wikipedia of k-NN, we can see there are two types of items, one is blue rectangle and another is red triangle. The green cycle is the item to be classified.

In this case, if *k* = 3, the nearest items to the green cycle are two red triangles and one blue rectangle. This three items vote and the green cycle belongs to red triangle category. If *k* = 5, the nearest items to the green cycle are two red triangles and three blue rectangles. Then the five items vote and the green cycle should belong to the blue rectangle category.

Algorithm that is used to implement classification, is called as Classifier. The term "classifier" is referred as the mathematical function, which is implemented by classification algorithm, which plots the given data into a category. K-NN is a algorithm that stores all the available cases and also classifies new cases on the similarity measures (For Example., distance functions).

8. Coding ,Testing and Screenshots

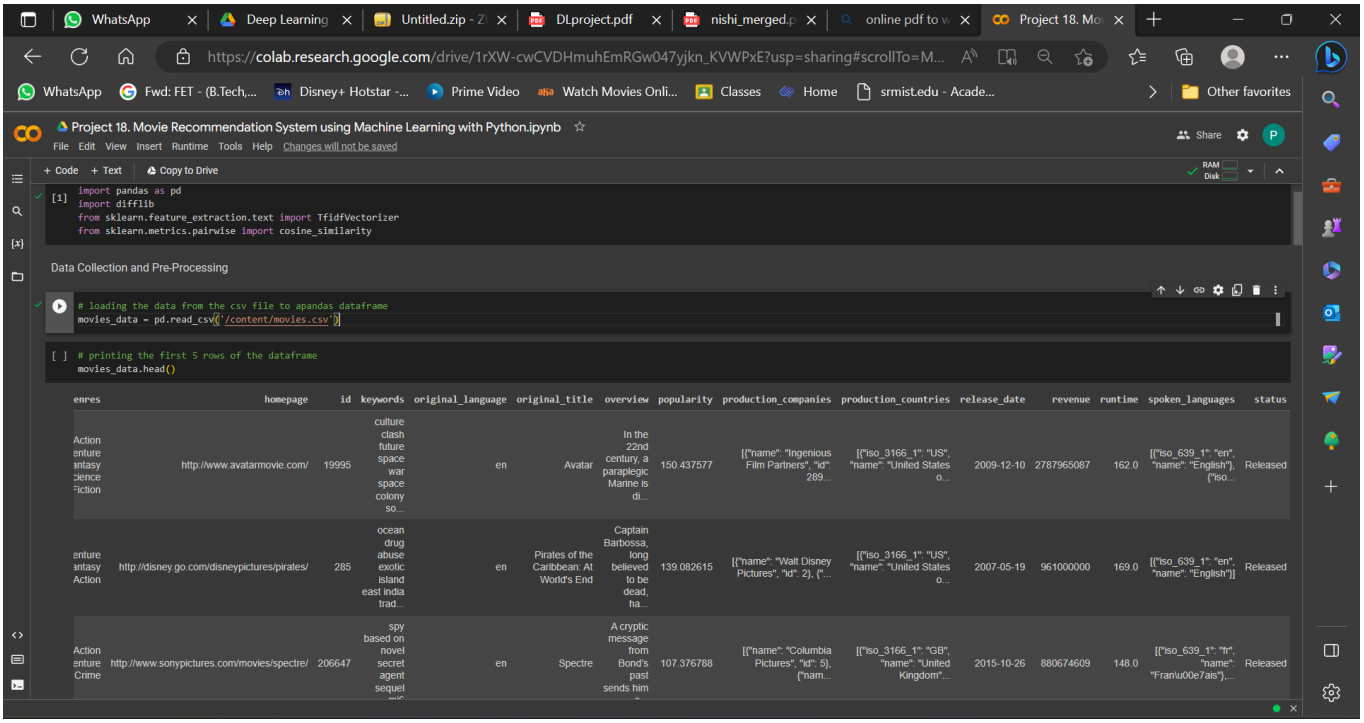
8.1 Coding and testing

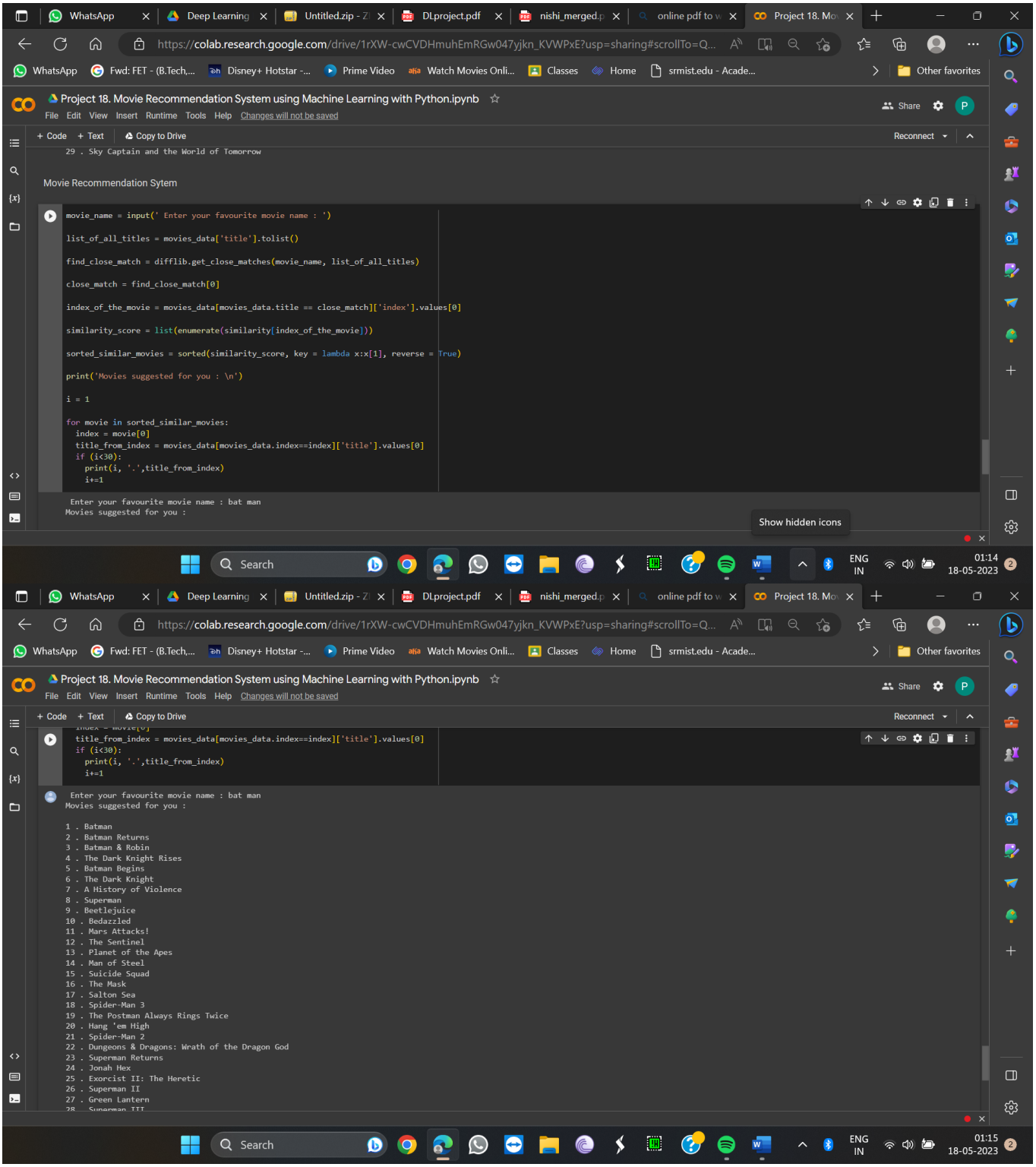
```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
# loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('/content/movies.csv')
# printing the first 5 rows of the dataframe
movies_data.head()
# number of rows and columns in the data frame

movies_data.shape
# selecting the relevant features for recommendation

selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
# replacing the null valuess with null string

for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
# combining all the 5 selected features

combined_features = movies_data['genres']+' '+movies_data['keywords']+'
'+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['director']
print(combined_features)
# converting the text data to feature vectors

vectorizer = TfidfVectorizer()
feature_vectors = vectorizer.fit_transform(combined_features)
print(feature_vectors)
# getting the similarity scores using cosine similarity

similarity = cosine_similarity(feature_vectors)
print(similarity)
print(similarity.shape)
```

```
# getting the movie name from the user

movie_name = input(' Enter your favourite movie name : ')
# creating a list with all the movie names given in the dataset

list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)
# finding the close match for the movie name given by the user

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
close_match = find_close_match[0]
print(close_match)
# finding the index of the movie with title

index_of_the_movie = movies_data[movies_data.title ==
close_match]['index'].values[0]
print(index_of_the_movie)
# getting a list of similar movies

similarity_score = list(enumerate(similarity[index_of_the_movie]))
print(similarity_score)
len(similarity_score)
# sorting the movies based on their similarity score

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =
True)
print(sorted_similar_movies)
# print the name of similar movies based on the index

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
movie_name = input(' Enter your favourite movie name : ')

list_of_all_titles = movies_data['title'].tolist()

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
```

```
close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title ==
close_match]['index'].values[0]

similarity_score = list(enumerate(similarity[index_of_the_movie]))

sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =
True)

print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

8.2 Screenshots




Project 18. Movie Recommendation System using Machine Learning with Python.ipynb
Movie Recommendation Sytem
movie_name = input(' Enter your favourite movie name : ')
list_of_all_titles = movies_data['title'].tolist()
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
close_match = find_close_match[0]
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
similarity_score = list(enumerate(similarity[index_of_the_movie]))
sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse = True)
print('Movies suggested for you : \n')
i = 1
for movie in sorted_similar_movies:
index = movie[0]
title_from_index = movies_data[movies_data.index==index]['title'].values[0]
if (i<30):
print(i, '.',title_from_index)
i+=1
Enter your favourite movie name : bat man
Movies suggested for you :
1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
10 . Bedazzled
11 . Mars Attacks!
12 . The Sentinel
13 . Planet of the Apes
14 . Man of Steel
15 . Suicide Squad
16 . The Mask
17 . Salton Sea
18 . Spider-Man 3
19 . The Postman Always Rings Twice
20 . Hang 'em High
21 . Spider-Man 2
22 . Dungeons & Dragons: Wrath of the Dragon God
23 . Superman Returns
24 . Jonah Hex
25 . Exorcist II: The Heretic
26 . Superman II
27 . Green Lantern

9.Experiment Results & Analysis

9.1Result

For our movie case, we split all the movie data into two sets. One is used as unclassified set which is test set and another is classified set which is validation set.

For each movie in test set, we will calculate the similarity between it with all the movies in validation set. Then we can find the most *k* similar movies for each movie in the test set. So the movie to be classified should belong to the category that appears most in the *k* movies.

The *k* is set to 15 in our project which is decided by a lot of experience. We cannot make sure 15 is the best for our project but it is enough for the validation.

After that, every movie in test set is classified, which means every movie in test set is classified with a genre in our case. We have known the right genre of the movie, so we will find out the right and wrong classifications and measure the improvement by metrics in following section.

9.2 Evaluation Metrics

Precision and Recall are two measurements for statistics, which are used to evaluate the quality of statistic result. Precision is used to calculate the ratio of related documents with selected documents. Recall is used to calculate the ratio of related documents with all related documents in selected documents. Below is the definition of the precision and recall under the context of our movie case.

Assume TP_i represents the number of test documents belonging to C_i and they are classified to C_i as well. FP_i represents the number of test documents that do not belong to C_i are classified to C_i. FN_i represents the number of test documents belonging to C_i are classified to other categories. So the Precision and Recall in category C_i is defined by:

$$P = \frac{TP_i}{TP_i + FP_i}$$

$$R = \frac{TP_i}{TP_i + FN_i}$$

Generally we should comprehensively consider precision and recall, then we introduce F-Measure, which is defined in Equation 5.3.

$$F = \frac{2 \times P \times R}{P + R}$$

9.3 Analysis

In order to show the improvement of our TF-IIDF-DC, we split the data into 80% as training data and 20% as testing data. Both TF-IDF and TF-IIDF-DC are calculated and evaluated by Equations above. The result is in Table 5.1.

	TF-IDF			**TF-IIDF-DC**		
Category	**P**	**R**	**F**	**P**	**R**	**F**
Sci-Fi	86.64	83.56	85.07	89.32	85.51	87.37
Crime	78.57	75.98	77.25	85.54	81.19	83.31
Romance	70.65	67.87	69.23	76.76	69.34	72.86
Animation	78.78	77.32	78.04	85.34	81.49	83.37
Music	70.45	67.29	68.83	77.45	72.83	75.07
Comedy	80.67	75.19	77.83	88.76	82.73	85.64
War	77.87	72.87	75.29	85.91	77.34	81.40
Horror	82.34	80.17	81.24	89.21	84.65	86.87
Adventure	86.32	83.76	85.02	88.75	84.35	86.49
News	70.44	67.93	69.16	73.79	69.72	71.70
Biography	69.98	62.48	66.02	74.39	68.93	71.56
Thriller	79.89	78.28	79.08	84.18	80.38	82.24
Western	75.45	70.32	72.79	77.65	73.76	75.66
Mystery	71.22	68.95	70.07	76.85	71.45	74.05
Short	70.39	67.87	69.11	74.28	69.89	72.02
Drama	89.87	85.69	87.73	92.48	88.41	90.40
Action	80.43	76.21	78.26	84.56	81.44	82.97
Documentary	73.93	70.48	72.16	78.58	75.65	77.09
Musical	70.22	69.28	69.75	73.23	71.28	72.24
History	76.89	73.47	75.14	81.74	79.34	80.52
Family	73.49	70.34	71.88	76.43	72.67	74.50
Fantasy	73.68	71.94	72.80	77.98	74.87	76.39
Sport	77.23	73.45	75.29	80.64	75.34	77.90

Table 8.1. Comparison of TF-IDF and TF-IIDF-DC(%)

As we can see in Table 8.1 and Figure 8.2, the Precision, Recall and F-Measure by TF-IIDF-DC are all higher than that of TF-IDF. From the experiment, TFIIDF-DC strengthens the

weight of representative terms and weaken terms of no use in the category, which is so-called noise.

The most important factor for content-based recommender systems is feature. How to describe a movie is the most important task because the more accurate a movie is described, the better results recommender system generates. So from this perspective, we have proved the improvement of our approach in content-based

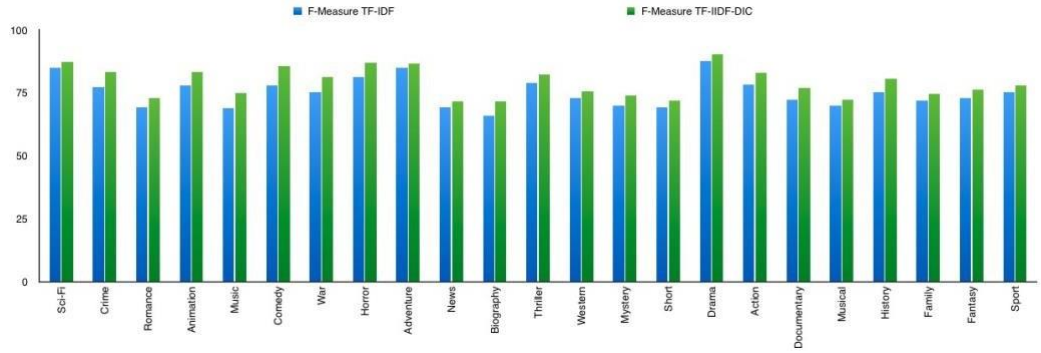


Figure 8.2. Comparision of F-Measure

Methodology

In order to achieve the goal of the project, the first process is to do enough background study, so the literature study will be conducted. The whole project is based on a big amount of movie data so that we choose quantitative research method. For philosophical assumption, positivism is selected because the project is experimental and testing character. The research approach is deductive approach as the improvement of our research will be tested by deducing and testing a theory. Ex post facto research is our research strategy, the movie data is already collected and we don't change the independent variables. We use experiments to collect movie data. Computational mathematics is used data analysis because the result is based on improvement of algorithm. For the quality assurance, we have a detail explanation of algorithm to ensure test validity. The similar results will be generated when we run the same multiptimeswhich

10Conclusion

Recommender system has become more and more important because of the information overload. For content-based recommender system specifically, we attempt to find a new way to improve the accuracy of the representative of the movie.

For the problems we mentioned at beginning, firstly, we use content-based recommender algorithm which means there is no cold start problem. In Section 4.1, we list all the features in our recommender system. Some of them are from other research team in the company, so the features are diversity and more accurate than others. Then we introduced the cosine similarity which is commonly used in industry. For the weight of features, we introduced TF-IIDF-DC which improve the representative of the movie.

This master thesis introduces a content-based recommender system for the movie website of VionLabs. The features used in the system are extracted from various aspects of the movie, which are diversity and unique. We introduce a new approach for setting weight for these features, the movie can be represented more accurately by TF-IIDF-DC which is the key point of our research.

In the end of the project, we use k-NN and various metrics to evaluate the improvement of the new approach. It is illustrated that the new approach contributes positively according to the evaluat

Future Work

Recommender system has developed for many years, which ever entered a low point. In the past few years, the development of machine learning, large-scale network and high performance computing is promoting new development in this field. We will consider the following aspects in future work.

- Use collaborative filtering recommendation.
  After getting enough user data, collaborative filtering recommendation will be introduced. As we discussed in Section 2.2, collaborative filtering is based on the social information of users, which will be analyzed in the future research.[37]
- Introduce more precise and proper features of movie.[1]

Typical collaborative filtering recommendation use the rating instead of object features. In the future we should extract features such as color and subtitle from movie which can provide a more accurate description for movie.

- Introduce user dislike movie list.
  The user data is always useful in recommender systems. In the future we will collect more user data and add user dislike movie list. We will input dislike movie list into the recommender system as well and generate scores that will be added to previous result. By this way we can improve the result of recommender system.

- Introduce machine learning.
  For future study, dynamic parameters will be introduced into recommender system, we will use machine learning to adjust the weight of each feature automatically and find the most suitable weights.

- Make the recommender system as an internal service.
  In the future, the recommender system is no longer a external website that will be just for testing. We will make it as an internal APIs for developers to invoke. Some movie lists in the website will be sorted by recommendation

Reference

[1] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems:

A survey of the state-of-the-art and possible extensions,” IEEE Trans. Knowl. Data Eng.

[2] G. Linden, B. Smith, and J. York, “Amazon recommendations: Itemto-item collaborative filtering,” IEEE Internet Comput., Feb. 2003.

[3] Michael Hashler, “Recommender Lab: A Framework for Developing and Testing Recommendation Algorithms” Nov. 2011.

[4] R. Bell, Y. Koren, and C. Volinsky, “Modeling relationships at multiple scales to improve accuracy of large recommender systems” KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA,