Online Movie Recommendation System using Sentiment Analysis

Artificial Intelligence Project Report

Submitted by
17BCB0016 – Akancha Agarwal
17BCB0038 – Saideepika Kandula
17BCB0055 – Dishi Jain

Semester: Fall 2019-2020

Preparation of J Component (Artificial Intelligence)

Akancha Agarwal:17BCB0016, Saideepika Kandula:17BCB0038, Dishi Jain:17BCB0055

***Abstract*— The Online Movie Recommendation System provides reviews and ratings to released movie and suggest movie to the user. This system generates a common review related to a movie by using Latent-Semantic Analysis (LSA) algorithm. LSA algorithm analyses the relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. Here analysis of comments given by various users will be done and a common review will be generated. This generated review will be a simple English statement and will help user to take a correct decision while selecting any movie. Also, the system is providing ratings out of five. With the help of this, user will be able to search his/her favourite movie with in no time. There is a section in our website which also provides a brief information about the movie and genres. If user wants to recommend his/her favourite movie to any other user/friend then he/she can.**

***Index Terms*—Movie review, Latent Semantic Analysis, Sentiment analysis, Recommendation system, Opinion mining.**

I. Introduction

People's opinion has become one of the extremely important sources for various services in ever-growing popular social networks. In particular, online opinions have turned into a virtual currency for businesses looking to market their products, identify new opportunities, and manage their reputations [6]. In general, recommender systems are defined as the supporting systems which help users to find information, products, or services (such as books, movies, music, digital products, websites, and TV programs) by aggregating and analysing suggestions from other users, which means reviews from various authorities, and user attributes. After viewing such reviews, they take their decisions. So, such reviews must be correct and proper [3].

Generally, the reviews are generated in graphical format that is in star ratings. Users just have to see the ratings which are generated by analysing the ratings given by other users to that product and have to take his/her decisions. Such ratings are easily understandable by any user. But they don't give clear idea of how the product is. They are helpful only in the scenario where if any product is excellent or very poor. The scenario where product is average, star ratings prove bit confuse for any user. They don't get clear views of what the other users think of that product. If the reviews are in simple English statement it would be easy for any user to understand the feelings of the other users too, about the product. Also, star ratings will be there for his/her help. So, the review about any product will give clear idea to any user so that he can easily take his/her decisions in such confusing scenario too.

Our system is a movie recommendation system which will generate reviews related to the movies which are released. Unlike other systems, we are going to generate a common review by analysing only the comments of the people (no heavy feedback). This will reduce the overhead of any user who is commenting on any movie and will make the system more user-friendly. So, the system will generate better review which will be a simple English statement. Also, our system will provide star ratings.

II. Literature Survey

Sentiment Analysis is an application of Natural Language Processing (NLP) which is used to find the sentiments of users' reviews, comments etc. on the internet. Nowadays, social websites like Facebook, Twitter are widely used for posting the users reviews about different things such as movies, news, food, fashion, politics and much more. Reviews and opinions play a major role in identifying the level of satisfaction of users regarding a particular entity. These are then used to find the polarity i.e. positive, negative and neutral. In this paper an approach to Sentiment Analysis on movie reviews in Hindi language is discussed [2].

There are various papers which have used machine learning based approach for sentiment analysis on product reviews [1][4] and it showed better results than lexical based approach [6]. In [12], this paper focuses around looking at the effectiveness of three AI strategies (Support Vector Machines (SVM), Naive Bayes (NB) and Maximum Entropy (ME)) for classification of online surveys utilizing a web model utilizing regulated learning techniques.

The main problem of this sentiment analysis of customer reviews is that classification of opinions of customers as negative, positive or neutral is a very tedious and hard task to be accomplished. The foremost challenge is the identification of fake reviews [7]. The outcomes have shown that AI calculations function admirably on weighted unigrams and SVM has come about greatest exactness. This isn't useful for buyers those need to look through the audits of items preceding buy yet additionally for organizations those need to watch the open's response to their items. In web time, advance in use of online life destinations, for example, twitter, Facebook, and survey webpage delivers an enormous measure of printed data. The printed data fills an imperative source to recognize open/client's assumption towards ideological group, items or an occasion. The slants communicated by open/clients are in the structure of positive, negative or unbiased extremity. The printed data in online life destinations assumes a significant job in choice emotionally supportive networks and individual choice makers. The procedure of mechanizing recognizable proof of assessment in a content is known as Sentiment Analysis. The framework can screen and assess continuously online perspectives, to exhibit how the entire world is responding to an idea/belief system/occasion. Growing such a framework which appoints extremity to a tweet is a hard assignment. In this paper [14], a scoring system to discover the slant extremity of the Twitter messages is proposed.

Recommender system is used to recommend items and services to the users and provide recommendations based on prediction. The prediction performance plays vital role in the quality of recommendation. To improve the prediction performance, this paper proposed a new hybrid method [5] based on naïve Bayesian classifier with Gaussian correction and feature engineering. The proposed method is experimented on the well known movie lens 100k data set. The results show better results when compared with existing methods [15].

III. Methodology

Online Movie Review system is designed to overcome the drawbacks of existing system. It also provides fast searching for any movie and also viewing reviews of that movie and recommending movie to any of the friend by sending him/her a mail. Main Aspects:

- Review Generation
- Recommendation

This system will provide fast searching as we implement Alignment algorithm. This area introduces the proposed system to expand the feature extraction which is helpful in sentiment analysis. The proposed system uses a combination of NLTK (Natural Language Tool Kit) systems for preparing datasets for positive and negative reviews (using python) and supervised learning.

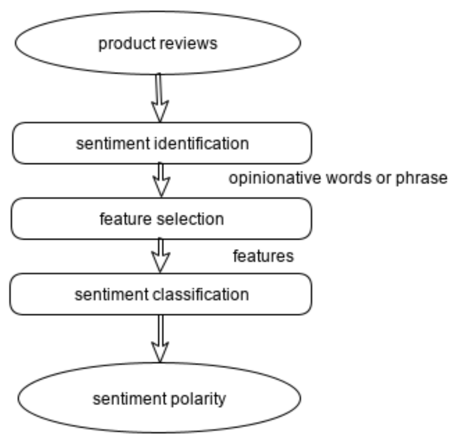


Figure 1. Flow chart for sentiment analysis on product reviews

The above figure shows the architecture of proposed system. In this movie review dataset is used to perform the operations.

Data Preprocessing: The preprocessing is done to expel the noise information from the dataset. In other word, dataset is advanced according to prerequisite. The strategy utilized as a part of preprocessing is feature extraction. The following pre-processing steps are followed to scoring the sentence.

- *Tokenization* is the process of breaking a sequence of string into pieces. It may be words, keyword, symbols, keyword, sentences and other elements called tokens. Tokens can be a words, phrases or they may be whole sentences. In this process some characters like punctuation mark are removed and these tokens become the input to the other task such as parsing or text mining.
- *POS (Part of speech)* tagging is the preprocessing technique. In this techniques words are tagged to specific part of speech such as a nouns, adjective, adverbs, verbs etc. based on their association with resulting words. E.g. “This movie is so wasteful of talent, it is truly disgusting” this sentence will be tagged as “This (Determiner) movie (Noun) is (Verb) so (Adverb) wasteful (Adjective) of (Preposition) talent (Noun), it (Pronoun) is (Verb) truly (Adverb) disgusting (Adjective)”. As per relationship of words we will give score for sentence.

After the Data Pre-processing, the bag of words for each positive and negative reviews is extracted from all the reviews received. Now, comes the part of movie review classification into either positive or negative review. For this, an algorithm called LSA (Latent Semantic Analysis) is used. It is also compared with the results of direct correlation of movie ratings results achieved.

For performing LSA, the following methodology is used:

- A matrix ‘A’ is made with rows as bag of words and 2 columns, one of the positive review and the other of negative review.
- A query column vector is formed for the review to be classified based on the presence of query review words in bag of words. (If present, 1 is added in the corresponding row, else 0).

- Decomposition of matrix ‘A’ into U, S and V matrices where $A=USV^T$. This is done by finding the eigen values of the matrix formed by multiplication of A and transpose of A.
- Rank approximation is done.
- A new query vector is formed using $q=q^TUS^{-1}$.
- Using cosine similarity, the maximum score corresponds to the positive or negative review achieved accordingly.

IV. Experimentations and Results

- Dataset: In this paper, to perform sentiment analysis on movie review by using some approach for feature extraction using opinion lexicon English is done. For the analysis of sentiment large dataset is used. The dataset contain total 25000 number of movie review. In which 12500 are the positive review and 12500 are the negative review.
- Technology Used: For performing all this models python is used. Python is widely used high level programming language. Pre-processing is done by using NLTK. It is most popular library for natural language processing.

A website for movies is created which is linked with our code to rate, review and recommend movies. Here, the users can rate and review the movies. Also, they get the ratings in the form of stars and in the form of numerical score out of 5 (representing the number of stars). The users can view various movie reviews and accordingly get the movie recommendation. In the website, they can also choose from different genres to make our search field narrower. Some of the famous genres are: horror, comedy, drama, romcom, scifi, tech, fictional, musical, etc.

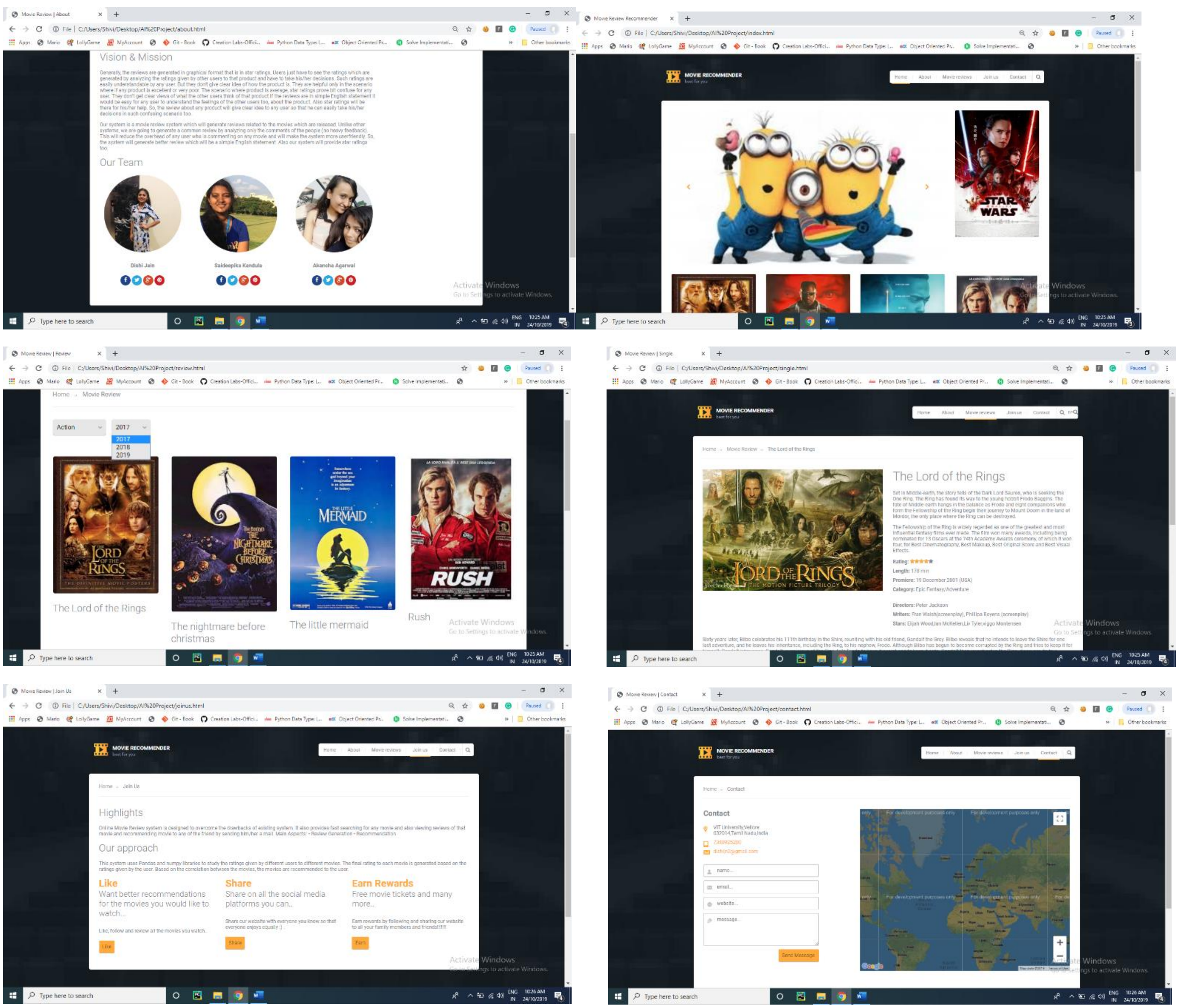


Figure 2-7: Screenshots of our website

In case of rating generation, this is done by correlating variables of the table with movie name rating and number of users

In [1]:
```
from string import punctuation
from os import listdir
from collections import Counter
from nltk.corpus import stopwords
```

In [2]:
```
# read the doc
def load_doc(filename):
    # open the file as read only
    file = open(filename, 'r')
    # read all text
    text = file.read()
    # close the file
    file.close()
    return text
```

In [3]:
```
# document tokenization
def clean_doc(doc):
    # split into tokens by white space
    tokens = doc.split()
    # remove punctuation from each token
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    # remove remaining tokens that are not alphabetic
    tokens = [word for word in tokens if word.isalpha()]
    # filter out stop words
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]
    # filter out short tokens
    tokens = [word for word in tokens if len(word) > 1]
    return tokens
```

In [4]:
```
# load all docs in a directory
def process_docs(directory, vocab):
    lines = list()
    # walk through all files in the folder
    for filename in listdir(directory):
        # skip files that do not have the right extension
        if not filename.endswith(".txt"):
            continue
        # create the full path of the file to open
        path = directory + '\\' + filename
        # load and clean the doc
        line = doc_to_line(path, vocab)
        # add to list
        lines.append(line)
    return lines
```

In [5]:
```
# load doc, clean and return line of tokens
def doc_to_line(filename, vocab):
    # load the doc
    doc = load_doc(filename)
    # clean doc
    tokens = clean_doc(doc)
    # filter by vocab
    tokens = [w for w in tokens if w in vocab]
    return ' '.join(tokens)
```

In [6]:
```
# save list to file
def save_list(lines, filename):
    data = '\n'.join(lines)
    file = open(filename, 'w')
    file.write(data)
    file.close()
```

In [10]:
```
# load doc and add to vocab
def add_doc_to_vocab(filename, vocab):
    # load doc
    doc = load_doc(filename)
    # clean doc
    tokens = clean_doc(doc)
    # update counts
    vocab.update(tokens)

# load all docs in a directory
def process_docs2(directory, vocab):
    # walk through all files in the folder
    for filename in listdir(directory):
        # skip files that do not have the right extension
        if not filename.endswith(".txt"):
            continue
        # create the full path of the file to open
        path = directory + '\\' + filename
        # add doc to vocab
        add_doc_to_vocab(path, vocab)

# define vocab
vocab = Counter()
# add all docs to vocab
process_docs2('C:\\Users\\HPW\\Desktop\\txt_sentoken\\neg', vocab)
process_docs2('C:\\Users\\HPW\\Desktop\\txt_sentoken\\pos', vocab)
# print the size of the vocab
print(len(vocab))
# print the top words in the vocab
print(vocab.most_common(50))
# keep tokens with > 5 occurrence
min_occurane = 5
tokens = [k for k,c in vocab.items() if c >= min_occurane]
print(len(tokens))
# save tokens to a vocabulary file
save_list(tokens, 'C:\\Users\\HPW\\Desktop\\txt_sentoken\\vocab.txt')
```

```
46557
[('film', 8860), ('one', 5521), ('movie', 5440), ('like', 3553), ('even', 2555), ('good', 2320), ('time', 22
83), ('story', 2118), ('films', 2102), ('would', 2042), ('much', 2024), ('also', 1965), ('characters', 194
7), ('get', 1921), ('character', 1906), ('two', 1825), ('first', 1768), ('see', 1730), ('well', 1694), ('wa
y', 1668), ('make', 1590), ('really', 1563), ('little', 1491), ('life', 1472), ('plot', 1451), ('people', 14
```

```
20), ('movies', 1416), ('could', 1395), ('bad', 1374), ('scene', 1373), ('never', 1364), ('best', 1301), ('n
ew', 1277), ('many', 1268), ('doesnt', 1267), ('man', 1266), ('scenes', 1265), ('dont', 1210), ('know', 120
7), ('hes', 1150), ('great', 1141), ('another', 1111), ('love', 1089), ('action', 1078), ('go', 1075), ('u
s', 1065), ('director', 1056), ('something', 1048), ('end', 1047), ('still', 1038)]
14803
```

In [12]:

```
# load vocabulary
vocab_filename = 'C:\\Users\\HPW\\Desktop\\txt_sentoken\\vocab.txt'
vocab = load_doc(vocab_filename)
vocab = vocab.split()
vocab = set(vocab)
# prepare negative reviews
negative_lines = process_docs('C:\\Users\\HPW\\Desktop\\txt_sentoken\\neg', vocab)
save_list(negative_lines, 'C:\\Users\\HPW\\Desktop\\txt_sentoken\\negative.txt')
# prepare positive reviews
positive_lines = process_docs('C:\\Users\\HPW\\Desktop\\txt_sentoken\\pos', vocab)
save_list(positive_lines, 'C:\\Users\\HPW\\Desktop\\txt_sentoken\\positive.txt')
```

In [ ]:

In [24]:
```
import numpy as np
import pandas as pd
```

In [25]:
```
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('C:\\Users\\HPW\\Desktop\\Movie-Recommender-in-python\\u.data', sep='\t', names=column_names
)
```

In [26]:
```
df.head()
```

Out[26]:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

In [27]:
```
movie_titles = pd.read_csv("C:\\Users\\HPW\\Desktop\\Movie-Recommender-in-python\\Movie_Id_Titles")
movie_titles.head()
```

Out[27]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

In [28]:
```
df = pd.merge(df,movie_titles,on='item_id')
df.head()
```

Out[28]:

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

In [29]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
%matplotlib inline
```

In [30]:
```
ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
ratings.head()
```

Out[30]:

	rating
title	
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

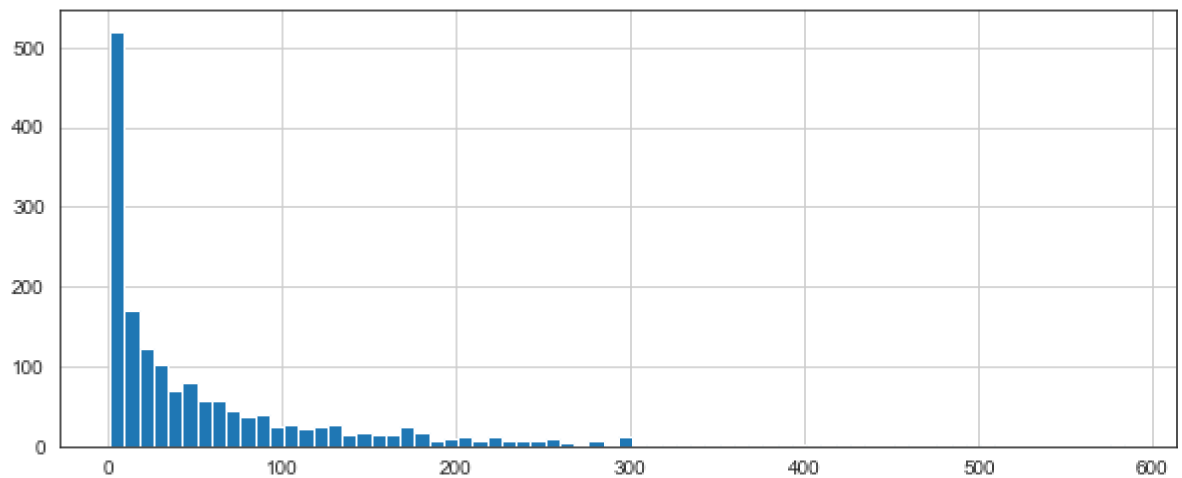
In [31]:
```
ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
ratings.head()
```

Out[31]:

title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

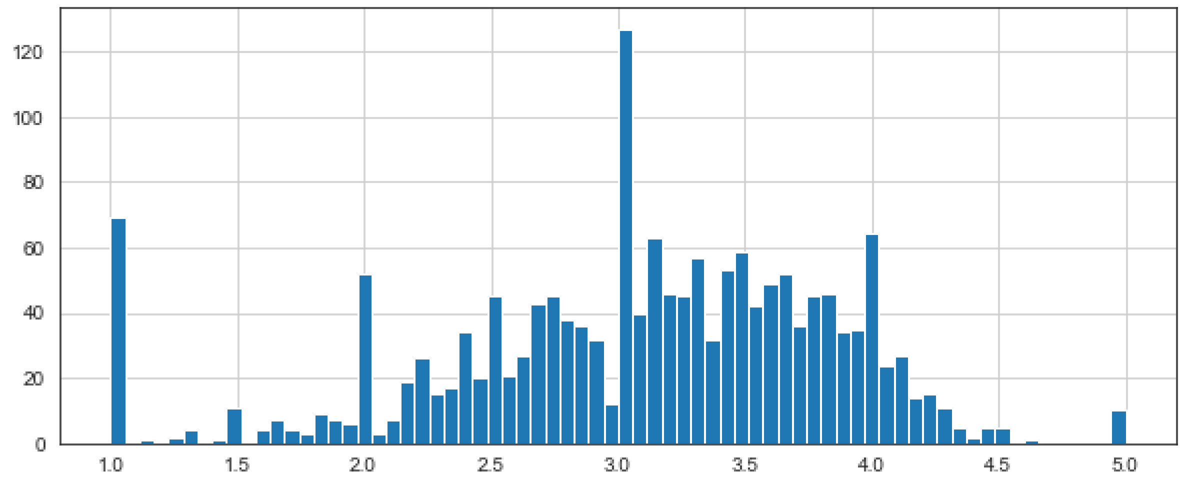
In [32]:
```
plt.figure(figsize=(10,4))
ratings['num of ratings'].hist(bins=70)
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1ebf7e84ac8>



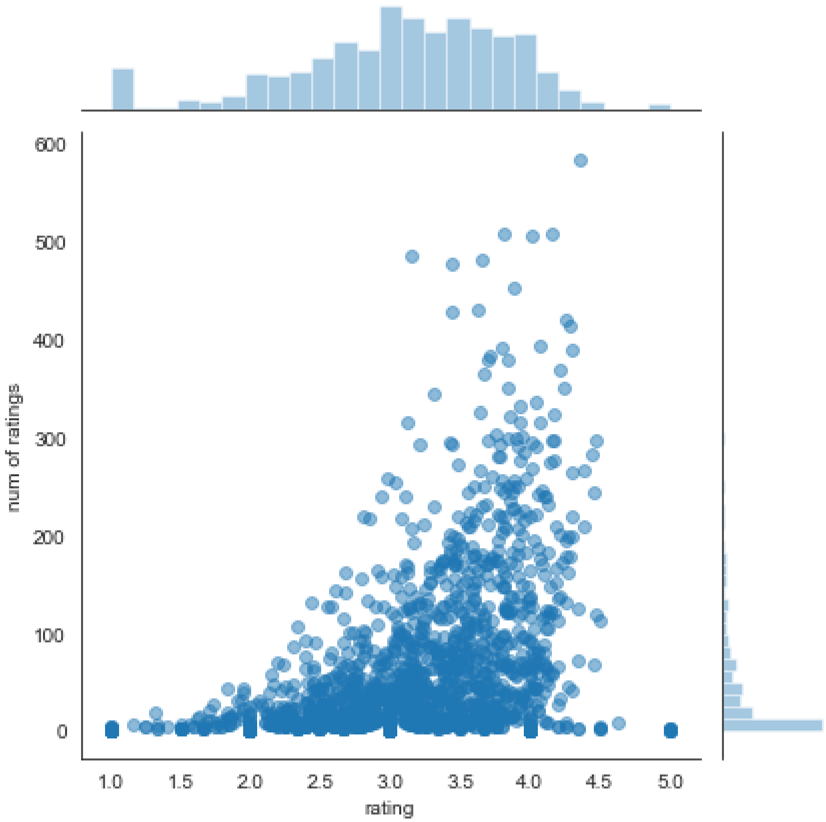
In [33]:
```
plt.figure(figsize=(10,4))
ratings['rating'].hist(bins=70)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1ebf80257b8>



In [34]:
```
sns.jointplot(x='rating',y='num of ratings',data=ratings,alpha=0.5)
```

Out[34]: <seaborn.axisgrid.JointGrid at 0x1ebfa023048>



In [35]:
```
moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
moviemat.head()
```

Out[35]:

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	You Frankenste (197
user_id															
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	Na
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Na

5 rows × 1664 columns

In [36]:
```
#see the most rated movie
ratings.sort_values('num of ratings',ascending=False).head(10)
```

Out[36]:

title	rating	num of ratings
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

In [37]:
```
ratings.head()
```

Out[37]:

title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

In [38]:
```
starwars_user_ratings = moviemat['Star Wars (1977)']
liarliar_user_ratings = moviemat['Liar Liar (1997)']
starwars_user_ratings.head()
```

Out[38]:
```
user_id
0    5.0
1    5.0
2    5.0
3    NaN
4    5.0
Name: Star Wars (1977), dtype: float64
```

In [39]:
```
similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)
```

In [40]:
```
corr_starwars = pd.DataFrame(similar_to_starwars,columns=['Correlation'])
corr_starwars.dropna(inplace=True)
corr_starwars.head()
```

Out[40]:

	Correlation
title	
'Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

In [41]:
```
corr_starwars.sort_values('Correlation',ascending=False).head(10)
```

Out[41]:

	Correlation
title	
Commandments (1997)	1.0
Cosi (1996)	1.0
No Escape (1994)	1.0
Stripes (1981)	1.0
Man of the Year (1995)	1.0
Hollow Reed (1996)	1.0
Beans of Egypt, Maine, The (1994)	1.0
Good Man in Africa, A (1994)	1.0
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991)	1.0
Outlaw, The (1943)	1.0

In [42]:
```
corr_starwars = corr_starwars.join(ratings['num of ratings'])
corr_starwars.head()
```

Out[42]:

	Correlation	num of ratings
title		
'Til There Was You (1997)	0.872872	9
1-900 (1994)	-0.645497	5
101 Dalmatians (1996)	0.211132	109
12 Angry Men (1957)	0.184289	125
187 (1997)	0.027398	41

In [43]:
```
corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[43]:

title	Correlation	num of ratings
Star Wars (1977)	1.000000	584
Empire Strikes Back, The (1980)	0.748353	368
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

In [44]:
```
corr_liarliar = pd.DataFrame(similar_to_liarliar,columns=['Correlation'])
corr_liarliar.dropna(inplace=True)
corr_liarliar = corr_liarliar.join(ratings['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[44]:

title	Correlation	num of ratings
Liar Liar (1997)	1.000000	485
Batman Forever (1995)	0.516968	114
Mask, The (1994)	0.484650	129
Down Periscope (1996)	0.472681	101
Con Air (1997)	0.469828	137

In [ ]:

In [ ]:

V. Conclusions

The proposed methodology finds the pattern of sentences by applying POS and dependency parser to locate the best feature from the dataset. Use of Naïve Bayes Algorithm decides the polarity of the comments with the help of which expert comments are provided. This work focuses on extracting the new feature from the movie review that have an extremely great effect on deciding the opinion of movie reviews. After this analysis of sentiment is done by applying the extracted features to the supervised learning i.e. Naive Bayes Classifier, it gives the accurate accuracy of 66%. We can conclude from our project model that various keywords can be used to recommend and rate movies using the LSA algorithm. We use correlation of machine learning to rate a particular movie and according to that user's ratings, we can recommend similar movies. In the future, we can also allow a user to recommend movies to his/her friends or family by sending a mail related to that movie through our website. This mail will include many different factors of the shared movie review like name of the movie, actor of the movie, producers and directors of the movie, and of course ratings of the movie, etc. This mail will be sent with the help of JAVA MAIL API.

REFERENCES

[1] M, Mamtesh, and Seema Mehla. "Sentiment Analysis of Movie Reviews Using Machine Learning Classifiers." International Journal of Computer Applications, vol. 182, no. 50, Nov. 2019, pp. 25–28., doi:10.5120/ijca2019918756.

[2] Nanda, Charu, et al. "Sentiment Analysis of Movie Reviews in Hindi Language Using Machine Learning." 2018 International Conference on Communication and Signal Processing (ICCSP), 2018, doi:10.1109/iccsp.2018.8524223.

[3] Wang, G., Sun, J., Ma, J., Xu, K., & Gu, J. (2014). Sentiment classification: The contribution of ensemble learning. Decision Support Systems,57, 77-93. doi:10.1016/j.dss.2013.08.002

[4] Tripathy, A., Agrawal, A., & Rath, S. K. (2016). Classification of sentiment reviews using n-gram machine learning approach. Expert Systems with Applications,57, 117-126. doi:10.1016/j.eswa.2016.03.028

[5] Kaur, B., & Kumari, N. (2016). A Hybrid Approach to Sentiment Analysis of Technical Article Reviews. International Journal of Education and Management Engineering,6(6), 1-11.doi:10.5815/ijeme.2016.06.01

[6] Melville, Prem, et al. "Sentiment Analysis of Blogs by Combining Lexical Knowledge with Text Classification." Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 09, 2009, doi:10.1145/1557019.1557156.

[7] Mukherjee, Arjun, et al. "Spotting Fake Reviewer Groups in Consumer Reviews." Proceedings of the 21st International Conference on World Wide Web - WWW 12, 2012, doi:10.1145/2187836.2187863.

[8] Liu, Bing, and Lei Zhang. "A Survey of Opinion Mining and Sentiment Analysis." Mining Text Data, 2012, pp. 415–463., doi:10.1007/978-1-4614-3223-4_13.

[9] Khan, Khairullah, et al. "Mining Opinion Components from Unstructured Reviews: A Review." Journal of King Saud University - Computer and Information Sciences, vol. 26, no. 3, 2014, pp. 258–275., doi:10.1016/j.jksuci.2014.03.009.

[10] Araque, Oscar, et al. "A Semantic Similarity-Based Perspective of Affect Lexicons for Sentiment Analysis." Knowledge-Based Systems, vol. 165, 2019, pp. 346–359., doi:10.1016/j.knosys.2018.12.005.

[11] Araque, Oscar, et al. "Enhancing Deep Learning Sentiment Analysis with Ensemble Techniques in Social Applications." Expert Systems with Applications, vol. 77, 2017, pp. 236–246., doi:10.1016/j.eswa.2017.02.002.

[12] Rathor, A. S., Agarwal, A., & Dimri, P. (2018). Comparative Study of Machine Learning Approaches for Amazon Reviews. Procedia Computer Science,132, 1552-1561. doi:10.1016/j.procs.2018.05.119

[13] Cernian, A., Sgarciu, V., & Martin, B. (2015). Sentiment analysis from product reviews using SentiWordNet as lexical resource. 2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). doi:10.1109/ecai.2015.7301224

[14] J, A., & G, J. (2018). Sentiment Classification of Tweets with Non-Language Features. Procedia Computer Science,143, 426-433. doi:10.1016/j.procs.2018.10.414

[15] Devi, S. Sathiya, and G. Parthasarathy. "A Hybrid Approach for Movie Recommendation System Using Feature Engineering." 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, doi:10.1109/icicct.2018.8473335.