Automatic Film Genere Prediction using NLP

Mrs Parimala Garnepudi, Dr S Sivanageswara Rao & Mr R Kalyan Chakravarthy

Assistant Professor, Dept. of CSE, VFSTR Deemed to be University, India
Associate Professor, Dept of Computer Science & Engineering, NEC, india
TGT , Dr B R Ambedkar, Gurukulam, Pamarru, India
garnepudi.parimala@gmail.com
profssnr@gmail.com
parimalachakri@gmail.com

Abstract:

Websites like Netflix and HBO Go, provide lists of movies based on genres, this makes it easier for user to select the movie that interests them. Tagging of movies is a complex process and usually involves a manual process where the movies are assigned to one or more Genres based on the suggestions sent by the users and consumers. Our task is to automate this process by building a model that can predict the genre of a movie using just the plot details (available in text form) using Natural Language Processing (NLP).

Keywords: *Natural Language Processing, Movies, Prediction.*

I. INTRODUCTION

Film Plot Synopses with Tags expected to foresee the class of the motion pictures dependent on their plot rundowns. Programmed labeling frameworks can help proposal motors to improve the recovery of comparable films just as help watchers to comprehend what's in store from a film ahead of time. The dataset contains all the IMDB id, title, plot summation, labels for the films. There are 14,828 motion pictures information altogether. The past sort forecast models use PC vision to characterize the pictures in the banner utilizing multi mark order to assemble a model. It utilized a pre prepared CNN (Convolution Neural Network) to foresee the likelihood of every kind given just the film banner.

DRAWBACKS

- All the data about a film may not be portrayed in a banner which makes the model to anticipate a bogus class of the film.

II. PROPOSED SYSTEM

In this we construct a model by utilizing multi name characterization in AI to foresee the film classification utilizing Natural Language Processing (NLP). Regular language handling (NLP) is a field worried about how to program PCs to measure and break down a lot of characteristic language information. Utilizing NLP we can have a profound comprehension of information and we can likewise take activities dependent on the results of information. In AI, multi-mark arrangement and the unequivocally related issue of multi-yield grouping are variations of the characterization issue where different names might be allotted to each case [1][2][3].

ADVANTAGES

• Tagging of films is a mind boggling measure and typically includes a manual cycle where the motion pictures are relegated to at least one types dependent on the recommendations sent by the clients and purchasers.

• If we can robotize this cycle of film labeling, not exclusively will it be quick, spare human exertion yet it will be more precise than an undeveloped human also.

The following are the different modules in this proposed method

Module 1: Information of films like their id, name, plot and sort are accumulated from various sources like IMDB site.

Module 2: Information assembled ought to be preprocessed and cleaned (stemming and evacuation of stop words).

Module 3: Highlights must be removed and a model ought to be constructed utilizing the highlights of preparing information utilizing calculated relapse.

Module 4: Model ought to be utilized to anticipate classifications of motion pictures in testing information. The below figure 1 represents the proposed method.

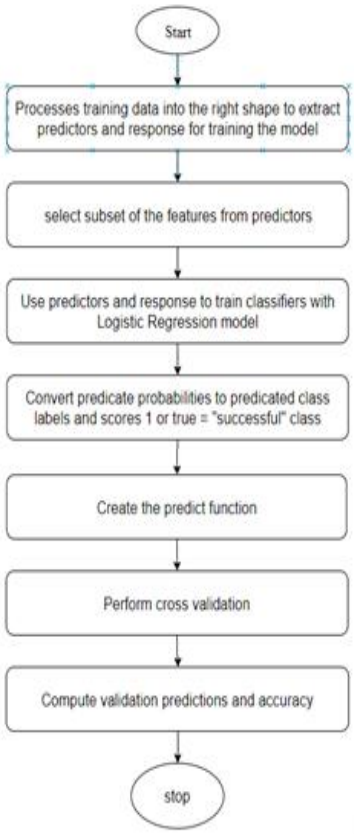

Fig.1 Proposed Method

Natural Language Processing (NLP)

Natural Language Processing (NLP) is a field worried about how to program PCs to measure and investigate a lot of regular language information. Utilizing NLP we can have a profound comprehension of information and we can likewise take activities dependent on the results of information. It has a wide scope of uses like Sentiment investigation, Text Classification, and Categorization, Automated Summarization and so on[7][8][9].

Multi-label Classification

In AI, multi-mark order and the emphatically related issue of multi-yield arrangement are variations of the characterization issue where numerous names might be doled out to each example. In multi-mark order, the preparation set is made out of examples each related with a lot of names, and the errand is to foresee the name sets of inconspicuous occasions through examining preparing cases with realized name sets. Distinction between multi-class characterization and multi-name arrangement is that in multi-class issues the classes are fundamentally unrelated, while for multi-mark issues each name speaks to an alternate order task, however the undertakings are by one way or another related[4][5][6].

Table 1

X	y
X_1	t_1
X_2	t_2
X_3	t_1
X_4	t_2
X_5	t_1

Binary Classification

Table 2

X	y
X_1	t_2
X_2	t_3
X_3	t_4
X_4	t_1
X_5	t_3

Multi-class Classification

Table 3

X	y
X_1	$[t_2, t_5]$
X_2	$[t_1, t_2, t_3, t_4]$
X_3	$[t_3]$
X_3	$[t_2, t_4]$
X_3	$[t_1, t_3, t_4]$

Multi-label Classification

There are 5 interesting labels in the information. Next, we have to supplant the current objective variable with numerous objective factors, each having a place with the special names of the dataset. Since there are 5 special names, there will be 5 new objective factors with values 0 and 1 as demonstrated as follows:

X	y
X_1	$[t_2, t_5]$
X_2	$[t_1, t_3, t_2]$
X_3	$[t_4]$

X	t_1	t_2	t_3	t_4	t_5
X_1	0	1	0	0	1
X_2	1	1	1	0	0
X_3	0	0	0	1	0

Reading Input from the CSV file:

```
#matplotlib inline
pd.set_option('display.max_colwidth', 300)

#reading data into meta variable

meta = pd.read_csv("movie.metadata.tsv", sep = '\t', header = None)
print(meta.head())


# rename columns
meta.columns = ["movie_id",1,"movie_name",3,4,5,6,7,"genre"]

# loading data into memory of plot and movie id
plots = []
with open("plot_summaries.txt", 'r',encoding="utf8") as f:
    reader = csv.reader(f, dialect='excel-tab')
    for row in tqdm(reader):
        plots.append(row)
```

Output of reading CSV file:

```
                                                                  8
0    975900  ... {"/m/01jfsb": "Thriller", "/m/06n90": "Science Fiction", "/m/03npn": "Horror", "/m/03k9fj": "Adventure
", "/m/0fdjb": "Supernatural", "/m/02kdv5l": "Action", "/m/09zvmj": "Space western"}
1   3196793  ...                                                              {"/m/02n4kr": "Mystery", "
/m/03bxz7": "Biographical film", "/m/07s9rl0": "Drama", "/m/0hj3n01": "Crime Drama"}
2  28463795  ...
                        {"/m/0lsxr": "Crime Fiction", "/m/07s9rl0": "Drama"}
3   9363483  ...                                                                          {"/m/01jfsb"
: "Thriller", "/m/0glj9q": "Erotic thriller", "/m/09blyk": "Psychological thriller"}
4    261236  ...
                                                   {"/m/07s9rl0": "Drama"}

[5 rows x 9 columns]
42303it [00:01, 29287.05it/s]
100%|██████████████████████████████| 42303/42303 [00:00<00:00, 1365404.95it/s]
```

Merging data from plot summaries and movie data:

```
# extract movie Ids and plot summaries
for i in tqdm(plots):
    movie_id.append(i[0])
    plot.append(i[1])

# create dataframe
movies = pd.DataFrame({'movie_id': movie_id, 'plot': plot})


print(movies.head())

# change datatype of 'movie_id' into string
meta['movie_id'] = meta['movie_id'].astype(str)

# merge meta with movies
movies = pd.merge(movies, meta[['movie_id', 'movie_name', 'genre']], on = 'movie_id')

print(movies.head())
```

Output of merging data from plot summaries and movie data:

```
   movie_id  ...
                              genre
0  23890098  ...                                                          {"/m/07s9rl0":
 "Drama", "/m/03q4nz": "World cinema"}
1  31186339  ...                  {"/m/03btsm8": "Action/Adventure", "/m/06n90": "Science Fiction", "/m/02kd
v5l": "Action", "/m/07s9rl0": "Drama"}
2  20663735  ...                            {"/m/04t36": "Musical", "/m/02kdv5l": "Action", "/m/07s9r
l0": "Drama", "/m/01chg": "Bollywood"}
3   2231378  ...                                                          {"/m/06qm3": "Sc
rewball comedy", "/m/01z4y": "Comedy"}
4    595909  ... {"/m/0lsxr": "Crime Fiction", "/m/07s9rl0": "Drama", "/m/01f9r0": "Docudrama", "/m/03q4nz": "World cin
ema", "/m/05bh16v": "Courtroom Drama"}

[5 rows x 4 columns]
(41793, 5) (42204, 5)
```

Building the model:

```
# building model
multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit(movies_new['genre_new'])

# transform target variable
y = multilabel_binarizer.transform(movies_new['genre_new'])


tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=10000)


# split dataset into training and validation set
xtrain, xval, ytrain, yval = train_test_split(movies_new['clean_plot'], y, test_size=0.2, random_state=9)


# create TF-IDF features
xtrain_tfidf = tfidf_vectorizer.fit_transform(xtrain)
xval_tfidf = tfidf_vectorizer.transform(xval)

lr = LogisticRegression()
clf = OneVsRestClassifier(lr)

# fit model on train data
clf.fit(xtrain_tfidf, ytrain)
```

Predicting the f1 score of the model:

```
print(y_pred[3])



print(multilabel_binarizer.inverse_transform(y_pred)[3])

# evaluate performance
print(f1_score(yval, y_pred, average="micro"))

# predict probabilities
y_pred_prob = clf.predict_proba(xval_tfidf)


t = 0.3 # threshold value
y_pred_new = (y_pred_prob >= t).astype(int)


# evaluate performance
print(f1_score(yval, y_pred_new, average="micro"))
```

Output of f1 score

```
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
('Action', 'Drama')
0.3156683206302113
0.437668381092334
```

Predicting genres of the testing data:

```python
for i in range(100,150):
    k = xval.sample(1).index[0]
    print("Movie: ", movies_new['movie_name'][k], "\nPredicted genre: ",
        infer_tags(xval[k])), print("Actual genre: ",movies_new['genre_new'][k], "\n")
```

Output of predicting genres of the testing data:

```
Movie:  When a Stranger Calls
Predicted genre:  [('Thriller',)]
Actual genre:  ['Thriller', 'Horror', 'Psychological thriller', 'Crime Thriller', 'Slasher', 'Teen']

Movie:  HMS Defiant
Predicted genre:  [('Action', 'Drama')]
Actual genre:  ['Action/Adventure', 'Costume drama', 'Drama', 'Adventure']

Movie:  Twitches Too!
Predicted genre:  [('Comedy', 'Romance Film')]
Actual genre:  ['Family Film', 'Comedy', 'Teen', 'Television movie']

Movie:  Tunnel Vision
Predicted genre:  [('Comedy',)]
Actual genre:  ['Parody', 'Comedy']
```

III. SCOPE FOR FUTURE DEVELOPMENT

We manufactured a model which uses plot rundowns to foresee the film class utilizing the NLP and Logistic Regression of SkLearn it likewise utilizes One Vs Rest Classifier for double importance. The future advancement may utilize film sort grouping from plot rundowns of motion pictures utilizing bidirectional LSTM (Bi-LSTM). We first partition each plot outline of a film into sentences and relegate the class of relating film to each sentence. Next, utilizing the word portrayals of sentences, we can prepare Bi-LSTM organizations. We can gauge the class for each sentence independently. Since plot synopses by and large contain different sentences, we can utilize greater part deciding in favor of a ultimate choice by considering the back probabilities of sorts allocated to sentences. Additionally, utilizing Bi-LSTM performs better contrasted with fundamental Recurrent Neural Networks (RNNs) and Logistic Regression (LR) as a benchmark.

IV. CONCLUSION

Utilizing plot synopses to anticipate the film kind is superior to utilizing the film banner to portray the film type. This model generally predicts the film classes that can be valuable to label motion pictures of comparable sort which thusly helps the client or the client to get motion pictures of the class he enjoys.

REFERENCES

[1] https://www.analyticsvidhya.com/blog/2019/04/build-first-multi-label-image-classification-model python/?utm_source=blog&utm_medium=predicting-movie-genres-nlp-multi-label-classification
[2] https://github.com/ishmeetkohli/imdbGenreClassification
[3] https://www.semanticscholar.org/paper/Movie-genre-classification%3A-A-multi-label-approach-Wehrmann-Barros/d0cb74af061e3c1e94b18ed8062746cae7ccb30c
[4] https://www.semanticscholar.org/paper/Movie-Genre-Classification-from-Plot-Summaries-LSTM-Ertugrul-Senkul/98e9a2e86b37aa0d010520a207d1f1ae1c62fe4a
[5] https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede
[6] https://www.geeksforgeeks.org/python-programming-language/
[7] https://www.geeksforgeeks.org/processing-text-using-nlp-basics/
[8] https://docs.anaconda.com/
[9] https://scikit-learn.org/stable/