Multilabel Classification for Movie Genres

Samantha Nadler
CSC 149: Introduction to Text Mining
Prof. Simona Doboli

Introduction

Streaming services classify movies into different genres to help organize their libraries and recommend new movies for users based on their watching habits and history.

Introduction

Streaming services classify movies into different genres to help organize their libraries and recommend new movies for users based on their watching habits and history.

However, movie genres are not rigid. One movie could be classified as multiple genres!

Musical
Fantasy
Romance

Data

Kaggle Competition: “Predict Movie Genres from Plot”

- Contains labeled train data and unlabeled test data
- Labels are numerical representations of 19 different genres
- Most movies have 2-3 genres

kaggle

Data

Genres Represented (with numerical IDs):

28 - Action	18 - Drama	9648 - Mystery
12 - Adventure	10751 - Family	10749 - Romance
16 - Animation	14 - Fantasy	878 - Science Fiction
35 - Comedy	36 - History	10770 - TV Movie
80 - Crime	27 - Horror	53 - Thriller
99 - Documentary	10402 - Music	10752 - War
		37 - Western

Methodology

A brief overview:

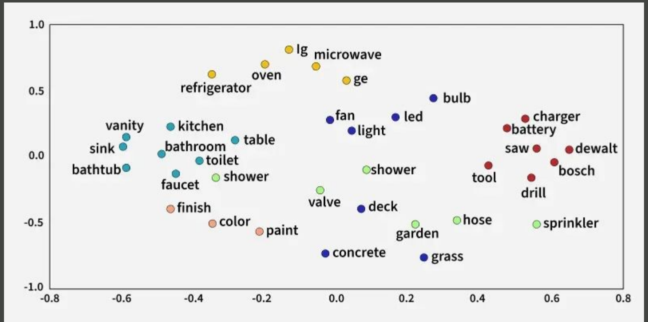
1. Craft and evaluate the performance of a Naive Bayes model from scratch.
2. Train a BERT-like model on tokenized text and evaluate its performance during each epoch.

Naive Bayes Model

Data Preparation:

- Simple preprocessing with gensim
- Use gensim to perform Word2Vec

Gensim is a Python library for topic modeling and efficient natural language processing.



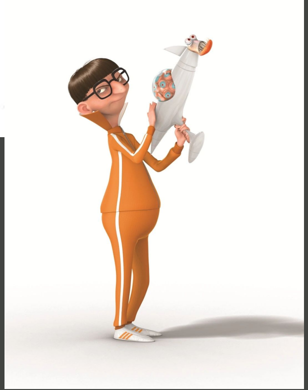
Naive Bayes Model

Data Preparation:

- Simple preprocessing with gensim
- Use gensim to perform Word2Vec

Gensim is a Python library for topic modeling and efficient natural language processing.

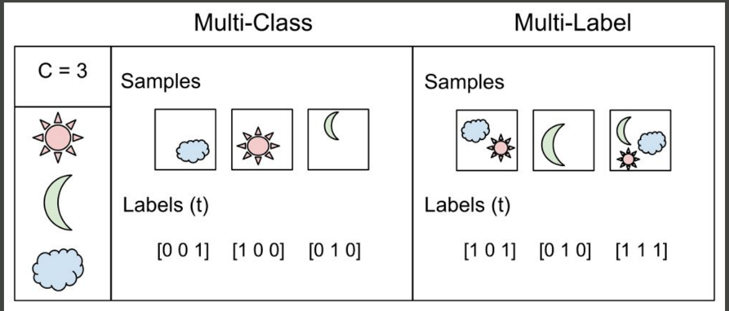
Text was then vectorized with NumPy.



Naive Bayes Model

Multi-Label Binarizer (MLB):

- Used to create unified labels for multi-label classification problems
- Since we have 19 movie genres, this step is important for Naive Bayes classification!



Naive Bayes Model

The following functions were implemented from scratch:

- fit
- _gaussian_log_likelihood
- predict_logproba
- predict_proba
- predict

The model was then fit to a OneVsRestClassifier for multilabel classification.

Naive Bayes Model

```
# Fit the data

base_model = CustomGaussianNB()
model = OneVsRestClassifier(base_model)
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

model.fit(X_train_scaled, y_train_binarized)

probability_predictions = model.predict_proba(X_val_scaled)
class_label_predictions = model.predict(X_val_scaled)
f1 = f1_score(y_val_binarized, class_label_predictions, average = 'weighted')
f1
```

```
0.3596989059402609
```

BERT Model

Data Preparation (different from Naive Bayes, accounting for different format):

- Use HuggingFace tokenizer to convert text into torch format
- Create dictionary to map tokenized text to binarized labels

BERT Model

Data Preparation (different from Naive Bayes, accounting for different format):

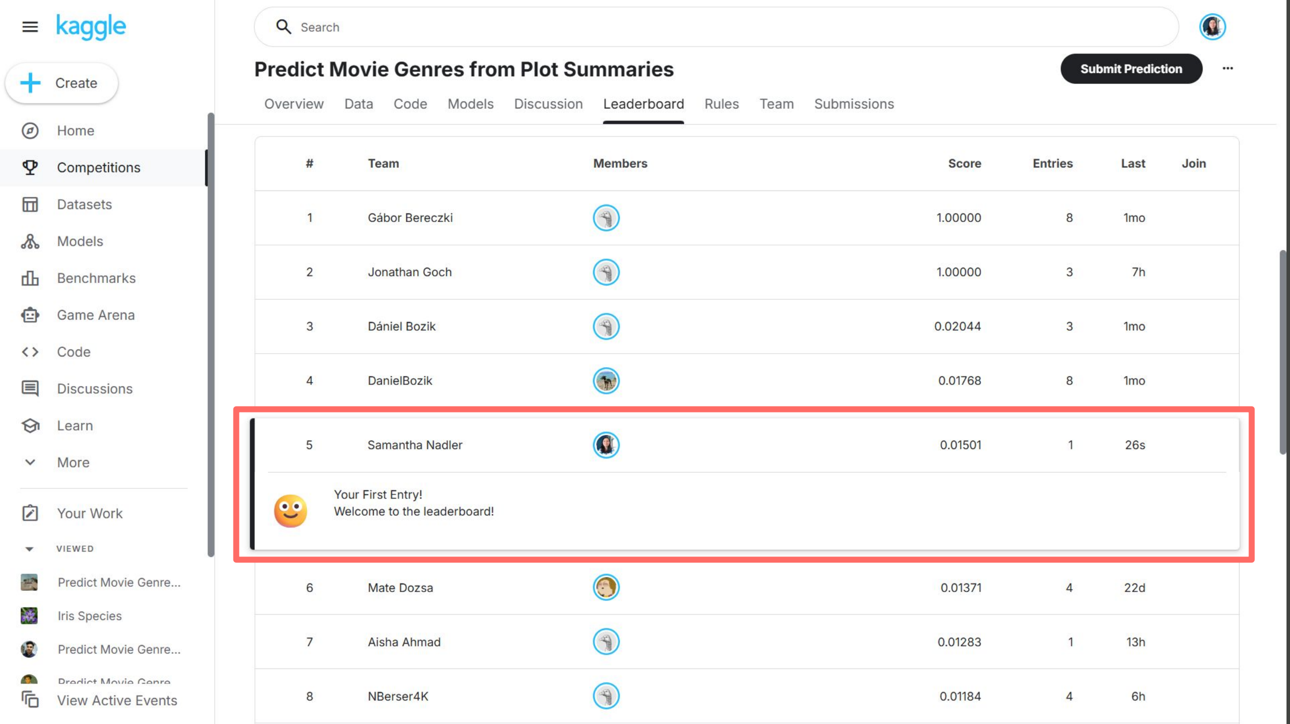
- Use HuggingFace tokenizer to convert text into torch format
- Create dictionary to map tokenized text to binarized labels

F1 score and relevant metrics were calculated for each epoch.

BERT Model

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.272700	0.259437	0.890903	0.581402	0.682728	0.506265
2	0.233600	0.246682	0.898750	0.626537	0.699257	0.567517
3	0.199800	0.247668	0.902986	0.657010	0.697602	0.620882
4	0.175900	0.249107	0.904097	0.658422	0.704979	0.617633
5	0.149000	0.267564	0.901944	0.654091	0.692787	0.619490
6	0.130200	0.281794	0.901319	0.655181	0.686673	0.626450
7	0.113500	0.297132	0.901944	0.656448	0.690026	0.625986
8	0.100700	0.306373	0.905000	0.671154	0.696259	0.647796
9	0.089300	0.319647	0.902639	0.661353	0.689673	0.635267
10	0.080000	0.320597	0.903472	0.664575	0.692308	0.638979

Some awesome news!



Thank you for your attention!