NCF Movie Recommendation System

Khalid Nimri: 2140145
Aseel Suhail: 2140197

NCF Movie Recommendation System

- Introduction to Neural Collaborative Filtering
- Detailed Explanation of NCF
- Code Overview Part 1
- Code Overview Part 2
- Code Output Slide 1
- Code Output Slide 2
- Conclusion
- References

Introduction to Neural Collaborative Filtering (NCF)

- **Definition of NCF:** Neural Collaborative Filtering (NCF) integrates neural networks for personalized movie recommendations with AI-driven deep learning methods

- **Enhancements by NCF over Traditional Methods:** NCF surpasses traditional approaches by capturing intricate user-movie interactions through multi-layered neural networks, boosting recommendation accuracy

Code Overview Part 1

- **NCF Model Training Process:** Training the NCF model involves defining neural network layers, embedding movie input data, and optimizing using mean squared error for improved prediction
- **Similar Movie Recommendations with Word2Vec:** NCF utilizes Word2Vec to find similar movies based on descriptions, enhancing recommendation accuracy through semantic understanding
- **User Input Handling in the Recommendation System:** The system prompts users for genre, minimum rating, and similar movie inputs, enabling personalized and relevant movie suggestions

```python
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from gensim.models import Word2Vec
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dense
from tensorflow.keras.callbacks import EarlyStopping

movie_data = pd.read_csv('/kaggle/input/mooooovies/MovieCleanData.csv')

#Train Word2Vec model on movie titles and descriptions
def train_word2vec(data):
    sentences = [title.split() for title in data['title']] + [desc.split() for desc in data['description']]
    return Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)

word2vec_model = train_word2vec(movie_data)


#Neural collaborative filtering
def NCF_model(num_movies, embedding_size):
    movie_input = Input(shape=(1,))
    movie_embedding = Embedding(num_movies, embedding_size)(movie_input)
    movie_vec = Flatten()(movie_embedding)
    dense = Dense(128, activation='relu')(movie_vec)
    dense = Dense(64, activation='relu')(dense)
    output = Dense(1)(dense)
    model = Model(movie_input, output)
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

num_movies = movie_data['dataId'].nunique()
ncf_model = NCF_model(num_movies, 50)
ncf_model.fit(movie_data['dataId'].values, movie_data['rating'].values, epochs=32, batch_size=64, verbose=1)
```

Code Overview Part 2

- **Epoch Progress in Model Training:** Tracking the epochs while training the NCF model provides insights into loss reduction, optimizing recommendation accuracy
- **Output: Recommended Movies Based on User Input:** Displaying top movie recommendations prompts user engagement, enhances user experience and satisfaction

```
#Function to get similar movies based on description using Word2Vec
def get_similar_movies(model, movie_title, data):
    movie_vec = model.wv[movie_title.split()]
    movie_vec_mean = np.mean(movie_vec, axis=0)
    similarities = []
    indices = []
    for idx, row in data.iterrows():
        desc_words = row['description'].split()
        if desc_words:
            desc_vec = model.wv[desc_words]
            desc_vec_mean = np.mean(desc_vec, axis=0)
            sim_score = cosine_similarity([movie_vec_mean], [desc_vec_mean])[0][0]
            similarities.append(sim_score)
            indices.append(idx)
    return indices, similarities


#Recommendation function using similarity
def recommend_movies(genre, min_rating, similar_movie):
    filtered_data = movie_data[(movie_data['genre'].str.contains(genre, case=False, na=False)) & (movie_data['rating'] >= min_rating)]
    filtered_data = filtered_data.reset_index(drop=True)

    if similar_movie and similar_movie in word2vec_model.wv.key_to_index:
        indices, similarities = get_similar_movies(word2vec_model, similar_movie, filtered_data)
        if indices:
            similar_data = filtered_data.loc[indices]
            similar_data['similarity'] = similarities
            similar_data = similar_data.sort_values(by=['similarity', 'rating'], ascending=[False, False])
            return similar_data.head(5)
        else:
            return pd.DataFrame()
    return filtered_data.nlargest(5, 'rating')
```

```
#Function to ask the user for input
def ask_user():
    print("Please answer the following questions for movie recommendations:")
    preferred_genre = input("1. What genre do you prefer? ")
    minimum_rating = float(input("2. What is the minimum rating you would prefer? "))
    similar_movie = input("3. Do you have a similar movie in mind? (If not, leave blank) ")
    recommendations = recommend_movies(preferred_genre, minimum_rating, similar_movie)
    counter = 0

    if not recommendations.empty:
        print("\nRecommended Movies:\n")
        for index, movie in recommendations.iterrows():
            print(f"Movie {counter + 1}:")
            print(f"Title: {movie['title']}")
            print(f"Genre: {movie['genre']}")
            movie_length = movie.get('length', 'Not Available')
            movie_length = "Not Available" if movie_length == -1 else movie_length
            print(f"Movie Length: {movie_length} minutes")
            print(f"Release Year: {movie['releaseYear']}")
            print(f"Rating: {movie['rating']}")
            print(f"Description: {movie['description']}\n")
            print('-' * 50)
            counter += 1
    else:
        print("\nNo recommendations found based on the criteria.")
```

Code Output Slide 1

- **Visualization of Epoch Progress in Training NCF Model:** Visualizing epoch progress during model training provides insights into loss reduction over time, optimizing recommendation accuracy
- **User Engagement and Experience Enhancement through Movie Recommendations:** Providing top movie suggestions based on user input promotes engagement, tailors recommendations, and elevates overall user satisfaction

```
Epoch 20/32
824/824 10s 12ms/step - loss: 0.7220
Epoch 21/32
824/824 10s 12ms/step - loss: 0.7173
Epoch 22/32
824/824 10s 12ms/step - loss: 0.7004
Epoch 23/32
824/824 10s 12ms/step - loss: 0.7104
Epoch 24/32
824/824 10s 11ms/step - loss: 0.7063
Epoch 25/32
824/824 10s 12ms/step - loss: 0.7005
Epoch 26/32
824/824 10s 12ms/step - loss: 0.6993
Epoch 27/32
824/824 10s 12ms/step - loss: 0.6956
Epoch 28/32
824/824 10s 12ms/step - loss: 0.7044
Epoch 29/32
824/824 9s 11ms/step - loss: 0.6962
Epoch 30/32
824/824 10s 11ms/step - loss: 0.7018
Epoch 31/32
824/824 10s 12ms/step - loss: 0.6924
Epoch 32/32
824/824 10s 12ms/step - loss: 0.7087
```

Code Output Slide 2

- **Detailed Visualization of NCF Model Training Progress:** Visualizing epoch-by-epoch progress in NCF model training offers insights into loss reduction dynamics, optimizing recommendation accuracy
- **Interactive Outputs showcasing Recommended Movies for Users:** Displaying movie suggestions based on user input promotes engagement, tailors recommendations, and enhances overall user satisfaction

```
ask_user()
```

```
Please answer the following questions for movie recommendations:
1. What genre do you prefer?  comedy
2. What is the minimum rating you would prefer?  7
3. Do you have a similar movie in mind? (If not, leave blank)  friends

Recommended Movies:

Movie 1:
Title: Wayne's World
Genre: Comedy
Movie Length: 94 minutes
Release Year: 1992
Rating: 7.0
Description: Two slacker friends try to promote their public-access cable show.

--------------------------------------------------
Movie 2:
Title: Workin' Moms
Genre: Comedy
Movie Length: 30 minutes
Release Year: 2017
Rating: 7.8
Description: Four very different thirtysomething working-mother friends try to balance their jobs, family lives, and love lives in modern-day Toronto, Canada.

--------------------------------------------------
Movie 3:
Title: College Romance
Genre: Comedy
Movie Length: 30 minutes
Release Year: 2018
Rating: 8.4
Description: Three best friends look for love, laughs and some lifelong memories while attending college together.

--------------------------------------------------
```

Conclusion

Project Findings Summary: Innovative NCF architecture enhances movie recommendations leading to better user satisfaction

Implications of Enhanced Recommendation System: Accurate user preference prediction through NCF boosts recommendation accuracy and user engagement

Future Research Directions: Exploring dynamic user interaction insights in recommendation systems for continuous improvement

References

- **Academic Journal Utilization:** Referencing ACM journal article on NCF in recommendation systems adds credibility and showcases scholarly foundation
- **Dataset:** Utilizing IMDB dataset with 100k+ entries enhances model training, user interaction insights, and prediction accuracy