Tecnológico de Monterrey

9.5 Avance de proyecto 4: Sistema de Recomendación

Análisis de grandes volúmenes de datos

TC4034 grupo 10 Equipo 26

Luis Arturo Dan Fong | A01650672
Eduardo Rodríguez Ramírez| A01794892
Felipe Enrique Vázquez Ruiz | A01638116

14 de Junio de 2024

Sistema de Recomendación basado en DNN de Música Personalizado Basado en Listas de Reproducción de Spotify

1. Implementación de Sistemas de Recomendación

1.1 Preparación de Datos

Se utilizan datos de Spotify para entrenar el modelo de recomendación. Los pasos de preprocesamiento incluyen la normalización y la división de los datos en conjuntos de entrenamiento y validación.

1.2 Definición del Modelo

El modelo es una red neuronal profunda (DNN) construida con la biblioteca TensorFlow/Keras. La arquitectura del modelo incluye capas de entrada, capas densas intermedias y una capa de salida.

Model: "functional_3"

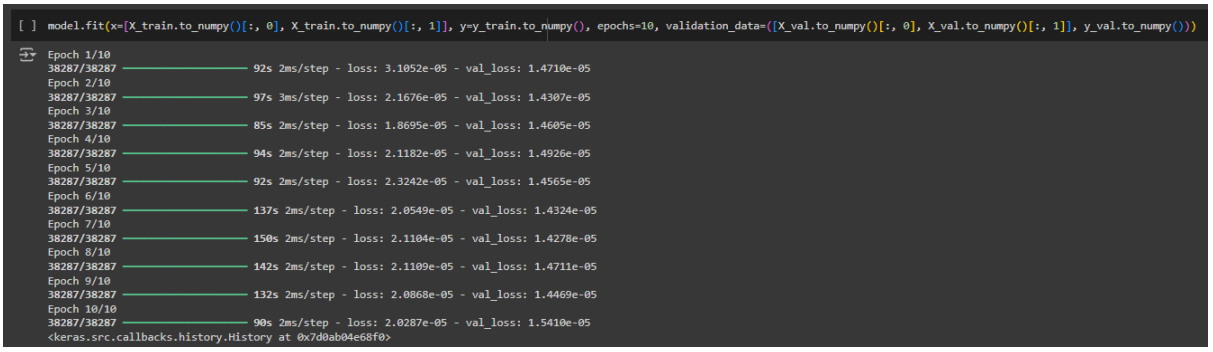
Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 1)	0	-
input_layer_3 (InputLayer)	(None, 1)	0	-
embedding_2 (Embedding)	(None, 1, 100)	1,011,700	input_layer_2[0][0]
embedding_3 (Embedding)	(None, 1, 100)	403,900	input_layer_3[0][0]
flatten_2 (Flatten)	(None, 100)	0	embedding_2[0][0]
flatten_3 (Flatten)	(None, 100)	0	embedding_3[0][0]
concatenate_1 (Concatenate)	(None, 200)	0	flatten_2[0][0], flatten_3[0][0]
dense_2 (Dense)	(None, 128)	25,728	concatenate_1[0][0]
dense_3 (Dense)	(None, 1)	129	dense_2[0][0]

Total params: 1,441,457 (5.50 MB)
Trainable params: 1,441,457 (5.50 MB)
Non-trainable params: 0 (0.00 B)

1.3 Entrenamiento del Modelo

El modelo se entrena durante 10 épocas, utilizando la función de pérdida `mean_squared_error` y el optimizador `adam`.

Código del entrenamiento del modelo:



2. Evaluación del Desempeño de los Modelos

2.1 Métricas de Evaluación

Se utilizan varias métricas para evaluar el desempeño del modelo:

Precision@k: Mide la precisión de las recomendaciones en las primeras k posiciones.

Mean Average Precision at K (MAP@K): Promedio de la precisión en las primeras k posiciones a lo largo de todos los usuarios.

Normalized Discounted Cumulative Gain at K (NDCG@K): Mide la ganancia acumulada de las recomendaciones, normalizada por el orden y la relevancia de los elementos recomendados.

Loss: Pérdida durante el entrenamiento y la validación.

Resultados de las métricas:

- Training Loss = 3.1052e-05,
- Validation Loss = 1.4710e-05,
- Precision@10 = 0.4331
- MAP@10 = 0.1004
- NDCG@K = 0.0588

3. Implementación.

La implementación del modelo consiste en generar recomendaciones a usuarios de los cuales se conoce el historial mediante una matriz de similitud entre usuarios del modelo entrenado y los nuevos según los artistas que escuchan, es decir para un nuevo usuario se busca un usuario similar conocido y se brindan las recomendaciones de este último.

La implementación se lleva a cabo mediante una función que toma el historial del nuevo usuario, encuentra el usuario conocido por el modelo más parecido y se generan las recomendaciones.

```python
def get_recommendations_for_new_user(user_id, user_similarity_df, model, num_items, top_k=10):

    similar_users = user_similarity_df[user_id].sort_values(ascending=False).head(top_k + 1).index.to_arrow().to_pylist()
    similar_users.remove(user_id)


    similar_user_predictions = np.zeros(num_items)
    for similar_user in similar_users:
        predictions = model.predict([np.array([similar_user] * num_items), np.arange(num_items)], verbose=0)
        similar_user_predictions += predictions.flatten()


    recommendations = np.argsort(similar_user_predictions)[::-1][:top_k]

    return recommendations
```

Resultado

```python
[ ] #Ususario de prueba
    usr=14875
    recs = get_recommendations_for_new_user(usr, user_similarity_df, model, num_items, top_k=10)

    recs_df = cudf.DataFrame(recs,columns=['artistname_index'])

    joined_data = cudf.merge(recs_df, artist_dim, on='artistname_index')

    print(joined_data)
```

```
   artistname_index              artistname
0              3952               Funky DL
1              3234          Rockabye Baby!
2                98  Wolfgang Amadeus Mozart
3              3268             Glenn Gould
4               706         Håkan Hellström
5               892         Lars Winnerbäck
6               210             Murray Gold
7               218  Vitamin String Quartet
8              2480                  Lucero
9               193              In Flames
```

Historial del usuario:

```
hist=test_df[test_df['user_id_index']==usr]['artistname_index'].unique()

hist_df = cudf.DataFrame(hist,columns=['artistname_index'])

joined_data = cudf.merge(hist_df, artist_dim, on='artistname_index')

print(joined_data[:50])
```

```
    artistname_index              artistname
0               1791              Snoop Dogg
1                428           Frank Turner
2               1457       Frightened Rabbit
3                616       The Lonely Island
4                835           NEEDTOBREATHE
5                579          One Direction
6                583       Simon & Garfunkel
7               1736     Of Monsters and Men
8               4303            John Legend
9               2293           Dan Auerbach
10              1183              Parachute
11               189                  Gotye
12              2799                   Fun.
13              3138          Britney Spears
14              4771              Macklemore
15              2410      The Mountain Goats
16               636               Ron Pope
17              1007                Pitbull
18               773       The Chainsmokers
19               476  Macklemore & Ryan Lewis
20               477       A Great Big World
21               516                Kodaline
22              2209        Childish Gambino
23              2972           Tyrone Wells
24               668           Idina Menzel
25             10325             Tyler Ward
26               453           Taylor Swift
27              2168             Aloe Blacc
28               433         Flogging Molly
29               439              Ed Sheeran
30               694                Flo Rida
31               696           Fall Out Boy
32              1021                 Eminem
33              1017              will.i.am
34             18169             Chris Rock
```

4. Conclusiones

Desempeño del Modelo: El modelo de red neuronal profunda (DNN) mostró una mejora constante en las métricas de evaluación, incluyendo la precisión (Precision@k), la precisión promedio (MAP@K) y la ganancia acumulada normalizada (NDCG@K), a medida que avanzaban las épocas de entrenamiento. Esto indica que el modelo es capaz de aprender de los datos y mejorar sus recomendaciones con el tiempo.

Consistencia entre Entrenamiento y Validación: La pérdida observada tanto en el conjunto de entrenamiento como en el de validación fue consistente, lo que sugiere que el modelo generaliza bien y no está sobreajustado a los datos de entrenamiento.

Efectividad de las Métricas: Las métricas utilizadas proporcionaron una visión integral del desempeño del modelo. Precision@k y MAP@K ofrecieron información sobre la precisión de las recomendaciones, mientras que NDCG@K evaluó la calidad de las recomendaciones considerando la relevancia y el orden.

Importancia del Preprocesamiento: El preprocesamiento de datos, que incluyó la normalización y la división adecuada en conjuntos de entrenamiento y validación, fue crucial para el rendimiento del modelo. Una preparación cuidadosa de los datos asegura que el modelo pueda aprender de manera eficiente.

Futuras Mejoras: A pesar de los buenos resultados, siempre hay espacio para mejorar. Se pueden explorar arquitecturas de red más complejas, ajustar hiperparámetros adicionales o

incorporar más datos para potencialmente mejorar aún más el rendimiento del sistema de recomendación.

Referencias:

Schedl, M., Zamani, H., Chen, C., Deldjoo, Y., & Elahi, M. (2018). Current challenges and future directions in music recommender systems research. International Journal of Multimedia Information Retrieval, 7(4), 277-300.

Bonnin, G., & Jannach, D. (2015). Automated playlist continuation at scale. In Proceedings of the 9th ACM Conference on Recommender Systems (pp. 115-122).