
TensorFlow Recommenders



TensorFlow Recommenders



TensorFlow Recommenders is a library for building recommender system models using TensorFlow.

It helps with the full workflow of building a recommender system: data preparation, model formulation, training, evaluation, and deployment.

It's built on Keras and aims to have a gentle learning curve while still giving you the flexibility to build complex models.

Installation

Make sure you have TensorFlow 2.x installed, and install from [pip](#):

```
1 pip install tensorflow-recommenders
```

Documentation

Have a look at our tutorials and API reference.

Quick start

Building a factorization model for the Movielens 100K dataset is very simple (Colab):

```
1 from typing import Dict, Text
2
3 import tensorflow as tf
4 import tensorflow_datasets as tfds
5 import tensorflow_recommenders as tfrs
6
7 # Ratings data.
8 ratings = tfds.load('movielens/100k-ratings', split="train")
9 # Features of all the available movies.
10 movies = tfds.load('movielens/100k-movies', split="train")
11
```

```

12 # Select the basic features.
13 ratings = ratings.map(lambda x: {
14     "movie_id": tf.strings.to_number(x["movie_id"]),
15     "user_id": tf.strings.to_number(x["user_id"])
16 })
17 movies = movies.map(lambda x: tf.strings.to_number(x["movie_id"]))
18
19 # Build a model.
20 class Model(tfrs.Model):
21
22     def __init__(self):
23         super().__init__()
24
25         # Set up user representation.
26         self.user_model = tf.keras.layers.Embedding(
27             input_dim=2000, output_dim=64)
28         # Set up movie representation.
29         self.item_model = tf.keras.layers.Embedding(
30             input_dim=2000, output_dim=64)
31         # Set up a retrieval task and evaluation metrics over the
32         # entire dataset of candidates.
33         self.task = tfrs.tasks.Retrieval(
34             metrics=tfrs.metrics.FactorizedTopK(
35                 candidates=movies.batch(128).map(self.item_model)
36             )
37         )
38
39     def compute_loss(self, features: Dict[Text, tf.Tensor], training=
40         False) -> tf.Tensor:
41
42         user_embeddings = self.user_model(features["user_id"])
43         movie_embeddings = self.item_model(features["movie_id"])
44
45         return self.task(user_embeddings, movie_embeddings)
46
47 model = Model()
48 model.compile(optimizer=tf.keras.optimizers.Adagrad(0.5))
49
50 # Randomly shuffle data and split between train and test.
51 tf.random.set_seed(42)
52 shuffled = ratings.shuffle(100_000, seed=42, reshuffle_each_iteration=
53     False)
54
55 train = shuffled.take(80_000)
56 test = shuffled.skip(80_000).take(20_000)
57
58 # Train.
59 model.fit(train.batch(4096), epochs=5)
60
61 # Evaluate.
```

```
61 model.evaluate(test.batch(4096), return_dict=True)
```