04_ncf_dev

April 21, 2025

[1]:
```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from sklearn.model_selection import train_test_split # Can use this for a quick␣
 ↪validation set
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import sys
from pathlib import Path
import math

# Add project root to sys.path
project_root = Path.cwd().parent # Should be RECSYS_FINAL
sys.path.append(str(project_root))

# Import project modules
from src import config
from src.data.dataset import CFDataset, create_mappings_and_unique_ids # Import␣
 ↪dataset class and helper
from src.models.ncf import NCF # Import the NCF model

# Set device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

# Set display options
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_rows', 100)
```

```
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Using device: cpu
```

[2]:
```
# Load the final aggregated interactions data
interactions_path = config.PROCESSED_DATA_DIR / "interactions_final.parquet"
try:
    interactions_df = pd.read_parquet(interactions_path)
    print(f"Loaded interactions data shape: {interactions_df.shape}")
    print(interactions_df.head())
except FileNotFoundError:
    print(f"Error: {interactions_path} not found.")
    print("Please ensure the preprocessing pipeline (run_preprocessing.py) has␣
 ↪run successfully.")
    raise
```

```
Loaded interactions data shape: (28466, 7)
   id_student presentation_id  total_clicks  interaction_days  \
0        6516       AAA_2014J          2791               159
1        8462       DDD_2013J           646                56
2        8462       DDD_2014J            10                 1
3       11391       AAA_2013J           934                40
4       23629       BBB_2013B           161                16

   first_interaction_date  last_interaction_date  implicit_feedback
0                     -23                    269           7.934513
1                      -6                    118           6.472346
2                      10                     10           2.397895
3                      -5                    253           6.840547
4                      -6                     87           5.087596
```

[3]:
```
# Define column names
USER_COL = 'id_student'
ITEM_COL = 'presentation_id'

# Create mappings from original IDs to contiguous indices
user_id_map, item_id_map, unique_users, unique_items =␣
 ↪create_mappings_and_unique_ids(
    interactions_df, USER_COL, ITEM_COL
)
n_users = len(unique_users)
n_items = len(unique_items)

print(f"Number of unique users: {n_users}")
print(f"Number of unique items: {n_items}")

# Split interactions data (optional, for quick validation during training)
# Using a simple random split here just for dev purposes.
# The final evaluation will use the proper time-based split test_df.
train_interactions, val_interactions = train_test_split(
    interactions_df, test_size=0.1, random_state=config.RANDOM_SEED
```

```
)
print(f"Train interactions shape: {train_interactions.shape}")
print(f"Validation interactions shape: {val_interactions.shape}")

# Create Datasets (using the full mappings created from the whole␣
 ↪interactions_df)
# Training dataset WITH negative sampling
train_dataset = CFDataset(
    interactions_df=train_interactions,
    all_item_ids=unique_items.tolist(), # Pass all unique items
    user_id_map=user_id_map,
    item_id_map=item_id_map,
    user_col=USER_COL,
    item_col=ITEM_COL,
    num_negatives=4 # Example: 4 negative samples per positive
)

# Validation dataset WITHOUT negative sampling (only positive interactions)
# We will predict scores for these and compare against a threshold or use␣
 ↪ranking metrics
val_dataset = CFDataset(
    interactions_df=val_interactions,
    all_item_ids=unique_items.tolist(),
    user_id_map=user_id_map,
    item_id_map=item_id_map,
    user_col=USER_COL,
    item_col=ITEM_COL,
    num_negatives=0 # No negative sampling for validation of positives
)

# Create DataLoaders
BATCH_SIZE = 1024 # Adjust based on memory
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True,␣
 ↪num_workers=4, pin_memory=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE * 2, shuffle=False,␣
 ↪num_workers=4, pin_memory=True) # Usually larger batch size for validation

print(f"\nDataLoaders created with batch size: {BATCH_SIZE} (train),␣
 ↪{BATCH_SIZE*2} (val)")
```

```
Number of unique users: 25364
Number of unique items: 22
Train interactions shape: (25619, 7)
Validation interactions shape: (2847, 7)
Preparing CFDataset…
Dataset contains 25619 positive interactions.
Generating 4 negative samples per positive interaction.
```

```
CFDataset preparation complete.
Preparing CFDataset…
Dataset contains 2847 positive interactions.
CFDataset preparation complete.

DataLoaders created with batch size: 1024 (train), 2048 (val)
```

[4]:
```python
# === New Cell: Instantiate and Train NCFRecommender ===
from src.models.ncf import NCFRecommender # Import the wrapper

# Define hyperparameters for the wrapper
MF_DIM_WRAP = 32
MLP_EMBEDDING_DIM_WRAP = 32
MLP_LAYERS_WRAP = [64, 32, 16, 8]
DROPOUT_WRAP = 0.2
LEARNING_RATE_WRAP = 0.001
EPOCHS_WRAP = 2 # Train for only 2 epochs as before
WEIGHT_DECAY_WRAP = 1e-5
BATCH_SIZE_WRAP = 1024 # Match DataLoader batch size used before
NUM_NEGATIVES_WRAP = 4 # Match negative samples used before

print("\n--- Initializing NCFRecommender ---")
ncf_recommender = NCFRecommender(
    user_col=USER_COL, # Defined earlier in notebook
    item_col=ITEM_COL, # Defined earlier in notebook
    mf_dim=MF_DIM_WRAP,
    mlp_layers=MLP_LAYERS_WRAP,
    mlp_embedding_dim=MLP_EMBEDDING_DIM_WRAP,
    dropout=DROPOUT_WRAP,
    learning_rate=LEARNING_RATE_WRAP,
    epochs=EPOCHS_WRAP,
    batch_size=BATCH_SIZE_WRAP,
    num_negatives=NUM_NEGATIVES_WRAP,
    weight_decay=WEIGHT_DECAY_WRAP,
    device='auto' # Or specify 'cuda'/'cpu'
)

# Train the model using the 'fit' method of the wrapper
# Pass the full interactions_df used to create mappings/dataset originally
print("\n--- Training NCFRecommender ---")
# Make sure interactions_df, USER_COL, ITEM_COL are defined from earlier cells
# Ensure 'interactions_df' is the full dataset intended for training this␣
 ↪instance
# For dev, you might use 'train_interactions' if you only want to fit on the␣
 ↪dev split
# ncf_recommender.fit(train_interactions) # Option 1: Fit on dev split
ncf_recommender.fit(interactions_df)      # Option 2: Fit on full data
```

```
print("\n--- NCFRecommender Training Complete ---")
```

```
--- Initializing NCFRecommender ---
Initialized NCFRecommender
Using device: cpu

--- Training NCFRecommender ---
Fitting NCFRecommender…
 Mapped 25364 users and 22 items.
Initializing NCF Network…
NCF Network Initialized.
Preparing CFDataset…
Dataset contains 28466 positive interactions.
Generating 4 negative samples per positive interaction.
CFDataset preparation complete.

--- Starting NCF Training (2 Epochs) ---
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Epoch 1/2:   0%|          | 0/139 [00:03<?, ?it/s]
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Epoch 1/2 - Training Loss: 0.5268
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
```

```
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
```

```
Epoch 2/2:   0%|          | 0/139 [00:03<?, ?it/s]
```

```
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Loading .env from: /Users/mohit/Desktop/everything/ATLAS/Semester
4/Pinnacle/recsys_final/.env
Database URI configured: Yes
Epoch 2/2 - Training Loss: 0.4908
--- NCF Training Finished ---

--- NCFRecommender Training Complete ---
```

[5]:
```python
# Cell [7] - Evaluate NCF Model (Corrected WITH Wrapper)

import pandas as pd
import numpy as np
import torch
from pathlib import Path
import sys

# --- Ensure project root is in sys.path ---
project_root = Path.cwd().parent
if str(project_root) not in sys.path:
    sys.path.append(str(project_root))
# ----------------------------------------

# --- Import necessary functions/classes ---
from src import config
from src.data import preprocess # For time_based_split
from src.evaluation.evaluator import RecEvaluator
```

```
#----------------------------------------

# --- Ensure necessary variables are defined (Check before proceeding) ---
# --- MODIFIED CHECK: Check for the wrapper instance ---
if 'ncf_recommender' not in locals():
    raise NameError("NCFRecommender instance 'ncf_recommender' not defined. Run␣
 ↪the training cell first.")
# -----------------------------------------------------
if 'user_id_map' not in locals(): raise NameError("'user_id_map' not defined.␣
 ↪Run cell [3] first.") # Keep these checks
if 'item_id_map' not in locals(): raise NameError("'item_id_map' not defined.␣
 ↪Run cell [3] first.")
#-----------------------------------------------

# --- Load or Recreate the CORRECT Time-Based Train/Test Split ---
# (This section remains the same)
print("Recreating time-based split for evaluation...")
interactions_path_eval = config.PROCESSED_DATA_DIR / "interactions_final.
 ↪parquet"
# Use interactions_df if already loaded, otherwise load it
if 'interactions_df' not in locals() or not isinstance(interactions_df, pd.
 ↪DataFrame):
    if not interactions_path_eval.exists():
         raise FileNotFoundError(f"Cannot find {interactions_path_eval}. Run␣
 ↪preprocessing first.")
    interactions_df_eval = pd.read_parquet(interactions_path_eval) # Use a␣
 ↪different name to avoid confusion if needed
else:
    interactions_df_eval = interactions_df # Use the one already loaded

TIME_THRESHOLD = config.TIME_SPLIT_THRESHOLD # <<< USE CONFIG VALUE
train_df_eval, test_df_eval = preprocess.time_based_split(
    interactions_df=interactions_df_eval, # Use the potentially reloaded DF
    user_col='id_student',
    item_col='presentation_id',
    time_col='last_interaction_date',
    time_unit_threshold=TIME_THRESHOLD
)
print(f"Time-based split recreated. Train: {train_df_eval.shape}, Test:␣
 ↪{test_df_eval.shape}")
#-----------------------------------------------------------------

# --- Load Item Features ---
# (This section remains the same)
items_df_path = config.PROCESSED_DATA_DIR / "items_final.parquet"
# Use items_df if already loaded and correctly indexed, otherwise load it
```

```
if 'items_df' not in locals() or not isinstance(items_df, pd.DataFrame) or␣
 ↪items_df.index.name != 'presentation_id':
    print("Loading items_df...")
    items_df_eval = pd.read_parquet(items_df_path) # Use a different name
    if 'presentation_id' in items_df_eval.columns:
       items_df_eval = items_df_eval.set_index('presentation_id')
    elif items_df_eval.index.name == 'presentation_id':
        pass # Already indexed correctly
    else:
       raise ValueError("Items DataFrame must have 'presentation_id' column or␣
 ↪index.")
else:
    items_df_eval = items_df # Use the one already loaded

print("Items DataFrame ready for evaluator.")
#----------------------------------------------------------------

# --- NO WRAPPER NEEDED HERE - Model is already wrapped ---
# --- (Delete the old NCFEvaluatorWrapper class definition if it's still here)␣
 ↪---
# --- (Delete the old ncf_eval_wrapper = ... line if it's still here) ---
# ----------------------------------------------------------------
print("Using the trained 'ncf_recommender' instance directly.")

# --- Initialize Evaluator and Evaluate ---
if test_df_eval.empty:
    print("\nCannot evaluate NCF model: Test data (time-split) is empty.")
elif items_df_eval.index.name != 'presentation_id': # Check the correct␣
 ↪items_df variable
    print("\nError: items_df_eval must have 'presentation_id' set as index for␣
 ↪evaluator.")
else:
    print(f"\nInitializing evaluator with Train: {train_df_eval.shape}, Test:␣
 ↪{test_df_eval.shape}")
    ncf_evaluator = RecEvaluator(
        train_df=train_df_eval,
        test_df=test_df_eval,
        item_features_df=items_df_eval, # Pass the correctly loaded/indexed␣
 ↪items_df
        user_col='id_student',
        item_col='presentation_id',
        k=config.TOP_K
    )

    # --- MODIFIED EVALUATION CALL: Use the wrapper instance ---
    print("\n--- Starting Evaluation of NCFRecommender ---")
```

```
    # Use the 'ncf_recommender' variable from the training cell
    ncf_results = ncf_evaluator.evaluate_model(ncf_recommender,␣
 ↪n_neg_samples=100)
    # ----------------------------------------------------------

    print("\nNCF Model Evaluation Results:")
    print(ncf_results)
#-----------------------------------------------------------------
```

```
Recreating time-based split for evaluation…
Performing time-based split…
Original interactions shape: (28466, 7)
Splitting based on time threshold: last_interaction_date <= 250
 Initial train size: 22892, Initial test size: 5574
Filtered 4836 interactions from test set (users/items not in train).
Final Training set shape: (22892, 7)
Final Test set shape: (738, 7)
Users in Train: 20701, Users in Test: 731
Items in Train: 22, Items in Test: 13
Time-based split recreated. Train: (22892, 7), Test: (738, 7)
Loading items_df…
Items DataFrame ready for evaluator.
Using the trained 'ncf_recommender' instance directly.

Initializing evaluator with Train: (22892, 7), Test: (738, 7)
Evaluator initialized with 22 unique candidate items.
Stored 20701 training interactions for filtering.
Prepared test data for 731 users.

--- Starting Evaluation of NCFRecommender ---

--- Evaluating Model: NCFRecommender ---
Total test users: 731. Evaluating 731 users known by the model.
Evaluating users:   0%|          | 0/731 [00:00<?, ?it/s]


--- Evaluation Results (K=10) ---
Precision@10: 0.0689
Recall@10: 0.6840
NDCG@10: 0.5749
n_users_evaluated: 731.0000
n_users_skipped: 0.0000
------------------------------

NCF Model Evaluation Results:
{'Precision@10': 0.06894664842681257, 'Recall@10': 0.6839945280437757,
'NDCG@10': 0.5749139349294585, 'n_users_evaluated': 731}
```