Transformer based Movie Recommendation

source env/bin/activate
pip3 install -r requirements.txt
python training.py

Overall Statement

- Build a recommendation engine to recommend movies to users
- For this project, the target is predicting the rating score a user will give to a potential movie
- This scoring system can help ranking all movies for a give user, which can act as a standard ranking system (but will not be implemented in this project)

File Structure:

- **models_and_helpers.py**: main body of model and other helper functions
- **training.py**: main body of training process
- **data_cleaning.py**: clean the raw data and save to ./data_clean
- requirements.txt
- Readme.md
- train_data.csv test_data.csv movie_meta.csv: cleaned data
- Dockerfile

Data Source

- ratings.dat: main user-item interaction data
- users.dat: user metadata
- movies.dat: item metadata
- movies_metadata: item metadata, in this project only 'overview' column is used

Data Preprocessing:

users

- categorize user metadata

ratings

- create movie sequence (length=4, step=2):

  > > > [1,2,3,4,5,6,7]-> [1,2,3,4],[3,4,5,6],[4,5,6,7]

- the former 3 movies_id and movie_rating , align with the last movie_id in the sequence are treated as sequential features
- the rating for the last movie is treated as label

movies & movie_meta

- one hot encode genre
- merge **movie_meta** with **movies** on 'title'
  - titles in **movies** is in format '{title} ({release year})'
  - title in **movie_meta** is in format '{title}'
  1. extract release year from **movie_meta** (some in mm/dd/yy format , others in yyyy-mm-dd format)
  2. concat title with release year, join with **movies** dataset
  3. however only 1/2 gets successfully joined
  4. mark joined dataset as **movies_meta_clean**, unmatched ones as **movies_mismatch**
  5. clean titles on both side
     1. remove punctuations, lower the title
     2. remove all contents in (), except (release year)
     3. remove common words (the a an)
     4. tokenize, sort, and rejoin
     5. in the end one duplicate join is detected, remove it manually
     6. In the end match rate is 85%, other titles are really hard to match (wrong release year, etc.).

Model Structure

The forward process of the model is processed in 3 ways

Other features:

- all user metadata: age,occupation,sex,user_id
- features are forwarded to embedding layers separately, then concat as one

Transformer features:

include features which will be included in transformer layer

- former 3 movie id and movie ratings in the sequence
- target movie id
- movie genre

1. For encoding genre information in movie sequence: movie ids are firstly embedded, then concat with their genre embedding, and finally forwarded to a linear layer, output shape will be the movie_id embedding dimension. The output from the linear layer is treated as movie embeddings containing genre information.
2. The position of a movie in a sequence is defined as t_reco-t_movie, so in the sequence, the position encoder is simply defined as [3,2,1]
3. For encoding positional information and rating information: movie embedding firstly add positional vector, then multiply its rating.

> > > new_movie_embedding = (movie_embedding+position_code)*rating

4. Target movies also follow step 1. , it will then be concatenated to the vector from step 3.
5. The concatenated vector will then be forwarded to transformer layer
6. The original embedding will be concatenated with the transformed one as a short cut loop (follow res net)

Bert Features

For movie overviews

- BERT is too heavy for my computer, even the forward process might be able to raise memory errors. So I decided to use fixed pre-trained BERT vectors.
- Since it's pre-trained and will not be fine-tuned, I didn't put a BERT layer. Instead, I calculated embeddings for all overview in advance, and then feed into the model.
- In other word, I used **transformers** to pre-calculate overview embeddings for each movie and then feed into the model.
- Every movie overview will be encoded as a vector with shape [1,768]
- For one sequence [movie1,movie2,movie3,movie4], it has [overview1, overview2,overview3,overview4]. [overview1,overview2,overview3] will be concatenated together, and then added positional encoding and rating weight by

    > > > (overview embedding+position_code)*rating

- overview4 will be embedded and concatenated with other overviews
- the concatenated vector will be forwarded to linear layers for dimension reduction.

In the end **other_features**, **transformer_features** and **bert_features** will be concatenated together and then be forwarded to 3 learky relu layers.

Future Work:

1. Finish requirements.txt, polish Readme.md
2. Dockerize
3. Add tuning process