tags: `ADL`

Wrap-up Report

Abstract

The goal in the project is to use DL to predict the courses which the users will buy in the future. We haved tried using the Seq2Seq model and the top-K sampling method on the training process. Experiments show that the top-K sampling with course similarity have the highest performance.

Introduction

Nowadays, the Internet has become the most important part of our life. The Internet leads to a convenient life for people, and more and more people buy goods online, moreover, take some courses online. Just like the online shopping site recommand "the goods you may like" for customers, a online courses platform recommands "the courses you may like" for its user. Thus, here we introduce a deep learning model which predicts some courses that a user may be interested in by learning from the courses he or she already bought.

Related work

https://huggingface.co/google/mt5-small
https://huggingface.co/shibing624/text2vec-base-chinese
https://ithelp.ithome.com.tw/articles/10219511

Approach

Data Augmentation

In the light of CNN image augmentation, we randomly masked some (fewer than 10%) interest from a user to produce a "human clone" which enlarge the size of training dataset and thus may help to improve the performance. According to our experiments, we found that masking approximately 10% interest leads to the minimum noise and may produce lots of data(use 56K given user data to produce 240K augmented user data).

Data Preprocessing

Courses Data Preprocessing

In the courses data preprocessing, we picked some important features of courses, like target group, groups, subgroups and topics. Then, we concatenated these features with Chinese special token inspired from featrue engineering, representing the context after this special token. For example, when we concatenated target group and groups, after concatenation, the processed data looks like [目標客群][...some description of target group...][課程類別][...some description of groups...].

Users Data Preprocessing

In the users data preprocessing, we picked important features of each users, like gender, occupation, interest and recreation. We used the same strategy as we did in courses data preprocessing to concatenate these features of users. Furthermore, we observated that the order of interest column is out of order. We expected clustered interests by sorting them would help model comprehense these data, we clustered interests when we concatenated these features.

Methods for "Unseen" Courses in Training Set

We found that some courses of 728 courses aren't in training set, that is, nobody bought these courses or the course didn't upload to the course platform before 2021/08/31. When we train our models, these "unseen" courses may lower the performance of our models. Thus, we came up with 2 methods to solve this problem.

kNN (k-nearest neighbor) for Unseen Courses

During normal inference for regression models, we treat model outputs as recommendation scores given inputs (user embeddings) and simply rank the top-50 courses using the recommendation scores. However, we noticed that some well-selling courses in validation set is never or seldom seen in the training set (Possibly because of their release date). To deal with this problem, we tried to re-calculate the recommendation scores for the courses that are never or seldom seen in training set. For each of these candidates whose score needs to be re-calculated, we first find the k-nearest neighbors according to cosine similarity of text2vec course embeddings. And then we replace the recommendation score with the mean of these k neighbors' recommendation scores.

Course Similarity by CoSENT (Cosine Sentence)

We use the CoSENT model to perform semantic search in order to find the similarity between two courses. During the training session, we add some unseen courses that are similar to the courses previously purchased by users to the labels."

Common Base Architecture for Models

Embedding Layer

There are two embedding for our models, user embedding and course embedding. For building embedding, we used two encoding method to convert data into embeddings, mT5-small encoder and text2vec SentenceModel encoding.

Multihead Attention Layer

This multihead attention layer focused on two embedding, and we used user embedding as query and course embedding as key and value. In here we used 4-head attention layer.

Model Architecture

Model 1 (Seq2seq model)

- Embedding Layer:
  For this seq2seq model, we used the embedding encoded by mT5-small encoder.
- Multihead Attention Layer:
  In here we used the multihead attention layer mentiond above.
- Decoder:
  As for decoder, we tried two kinds of decoders, one is mT5-small decoder and the other is LSTM decoder.

The output of the decoder is our prediction for the recommended courses.

Model 2 (Top-k sampling with kNN)

In this model, user embeddings are as input and 728 course embedding vectors stored in the model as model parameter. In each forward call, the 728 course embeddings are used as query and the input (user embedding) is used as keys and values. We will get 728 tokens as output and each of these tokens will mapped to 0~1 scalar value using a fully-connected prediction head.

- Embedding Layer:
  Same as Model 1, pretrained mt5-small encoder for user embedding
  As for course embedding, we experimented a few setups as below:
  - From scratch
  - text2vec embedding
  - mt5 embedding(mean-pooling)
    We found that the choice of pretrained course embedding did not matter much, so we trained course embedding from scratch in our final experiment for simplicity and training speed.
- Multihead Attention Layer (Note that we use course embeddings as queries in this case):
  Here we use torch.nn.MultiheadAttention. The I/O shapes are shown as below
  query: (bs, 728, embed_dim) (Copy course embedding bs times)
  key, value: (bs, seq_len, embed_dim) (User embedding from pretrained mt5)
  output: (bs, 728, embed_dim)
- Fully connected prediction head:
  nn.Linear(embed_dim, 1) + nn.Sigmoid()
  This maps the each of the output tokens from MHA layer to a scalar.
  output shape: (bs, 728, 1)
  We then calculate BCELoss between output logits and labels(bought courses by the user)

Model 3 (Top-k sampling with CoSENT)

- Embedding Layer:
  For this seq2seq model, we used the embedding encoded by text2vec SentenceModel encoding.
- Multihead Attention Layer:
  In here we used the multihead attention layer mentiond above.
- Convolutional Layer:
  The input of this layer is the matching score of courses and users from the multihead attention layer. And we let the number or attention heads become the input channel of this 1D convolutional layer.

Then, the output of the convolutional layer will be mapped to a value in [0,1] by softmax function as score of 728 courses. Last, we picked top-50 with highest score as our prediction.

Experiments

The following results are the performance on the course challenge in unseen domain.

	Model1(Seq2Seq)	**Model2(Top-K Sampling)**	**Model2 (with k-nearest neighbor)**
Training Score	0.08	0.21	0.21
Validation Score	0.01	0.11	0.11
Test Score	N/A	0.085	0.089

	Model3(Top-K Sampling)	**Model3(Course Similarity)**
Training Score	0.16	0.25
Validation Score	0.10	0.133
Test Score	0.08	0.114

In model 1 (Seq2Seq), after training for 10 epochs, we got 0.08 on the training set and 0.01 on the validation set. Since we got 0.01 on the validation set, we didn't use this model to predict on the test set.

In model 2 (Top-k sampling), after training for 10 epochs, we got 0.21 on the training set, 0.11 on the validation set and 0.085 on the test set. Then, we applied kNN method for "unseen" courses to this model. By using kNN, we got an improvement on the test set from 0.085 to 0.089.

Discussion

- Our methods all focus on "mapping user text embeddings to courses"
  - Strength: Such method is general and can be applied to all 4 domains without changing model architecture.
  - Weakness: We weren't able to provide personalized recommendation based on user's actual behavior (bought courses)
- Seq2Seq
  We thought that seq2seq model should be able to model user's sequential buying behavior. However, our experiment has shown that strong overfit happened on the model. During evaluation, we can often see the same pattern being decoded regardless of users. We think there's two possible reasons.
  1. seq2seq models usually require a large amuont sequential data for training in order to get good results. However, our data may not be large enough.
  2. Training an encoder-decoder seq2seq model with insufficient data, we have to assume that a user's buying decision is strongly correlated to previous bought courses, however this assumption might not always be true.

- About data
  Besides model architecture, training data can also have a large impact on learned models. We explored two directions:
    - Data Augmentation: We increased the amount of data by removing some of interests from users, while expecting that they will still buy the same course. We found that this didn't help.
    - CoSENT Pseudo-Labeling/kNN score re-labeling: We also tried adding some courses that is not seen in the training set back to the training set. The assumption is that the user did not buy this course just because it did not come out before 8/31. Our result shows that doing this can boost performance significantly. Besides this, another attempt of ours using kNN to decide recommendation scores for unseen courses also boosts performance, which means it's very important to keep the course candidates for recommendation up to date.

Conclusion

In this challenge, we tried to build a recommendation system for seen/unseen users predicting course/subgroup that a user will buy. We proposed a attention-based architecture to mix informations from user and courses. For output strategies we tried seq2seq and regression and found that regression performs better. We also tried working on the dataset, and we found that it is very important to consider the courses that is rarely seen or unseen in the training set since our trained model will not recommend those courses but these courses can become popular in the future.

Work Distribution

b08902149 custom MultiheadAttention, Model 1, Model 3 + CoSENT Pseudo-Labeling, course predictions.
b09902040 preprocessing(Data augmentation, Cluster interest...), video edit, wrap-up codes and reproduce scripts, submission handling.
b09505014 preprocessing, video recording
b09901073 preprocessing, Model 1, Model 2 + kNN experiments, subgroup predictions
b09203001 Slide making