MODEL SUMMARY

A1. Background on My team

- Competition Name: Learning Equality - Curriculum Recommendations
- Team Name: X+Y+Z^3
- Private Leaderboard Score: 0.75120
- Private Leaderboard Place: 3rd

Team member #1:

- Name: ZONG SHENGYA (@syzong)
- Location: Shanghai, Shanghai, China
- Email: 1095055590@qq.com

Team member #2:

- Name: XIA MAOJIN (@xia)
- Location: Hefei, Anhui province, China
- Email: 879989535@qq.com

Team member #3:

- Name: ZHOU YANG (@sayoulala)
- Location: Hangzhou, Zhejiang province, China
- Email: 100972458@qq.com

Team member #4:

- Name: ZHENG YUE (@yzheng21)
- Location: Shanghai, Shanghai, China
- Email: y.zheng211@gmail.com

Team member #5:

- Name: ZHENG HENG (@hengzheng)
- Location: Guangzhou, Guangdong province, China
- Email: zhengheng2008@163.com

A2. Background on My team

What is your academic/professional background?

> - ZONG SHENGYA: NLP algorithm engineer, Kaggle competition master.
> - XIA MAOJIN: NLP algorithm engineer, Kaggle competition master.
> - ZHOU YANG: NLP algorithm engineer.
> - ZHENG YUE: Recommend system algorithm engineer, Kaggle competition master.
> - ZHENG HENG: SRE engineer, Kaggle competition & notebook master.

Did you have any prior experience that helped you succeed in this competition?

> All members of the team have participated in and won awards in previous NLP competitions, including Kaggle and domestic competitions in China.

What made you decide to enter this competition?

> 1. We all have relevant experience in NLP competitions.
> 2. The quality of the data provided in this competition is very high.
> 3. The local CV and LB scores are relatively consistent.
>
> So we decided to invest time in this competition.

How much time did you spend on the competition?

> We spend an average of about 3 hours per day per person on this competition.

If part of a team, how did you decide to team up?

> we decide to team up because the GPU resources, and on the other hand, trying to fusion of retriever and reranker models to get higher rank on leaderboard.

If you competed as part of a team, who did what?

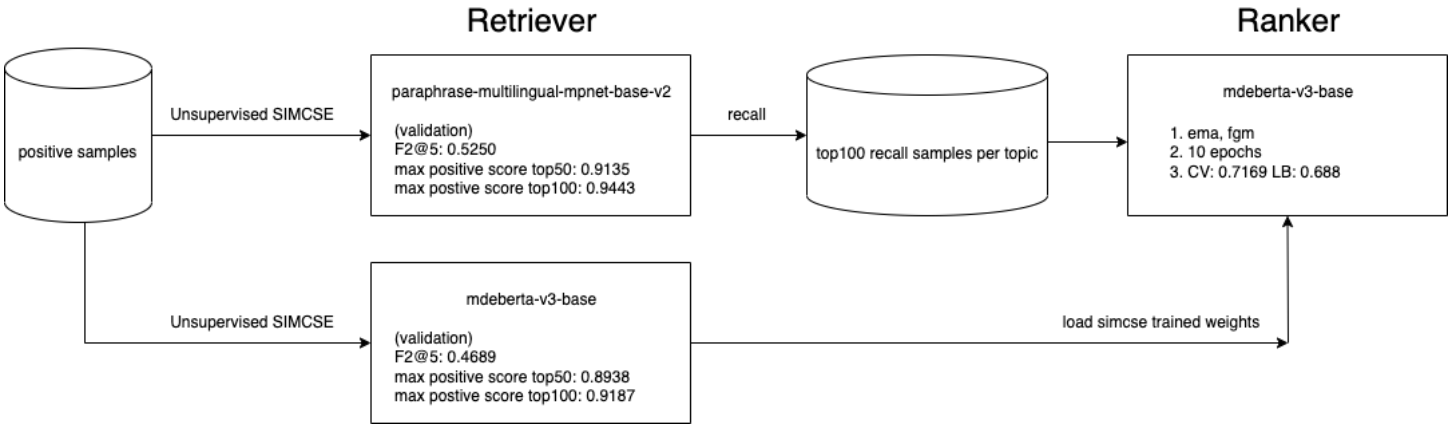
> - ZONG SHENGYA: Retriever model and Reranker model
> - XIA MAOJIN: Retriever model, Reranker model and ensemble
> - ZHOU YANG: Retriever model and Reranker model
> - ZHENG YUE: Retriever model and Reranker model
> - ZHENG HENG: CV strategy, Reranker model and submit pipeline

A3. Summary

- Random sampling non-source topics as holdout set for validation.
- Training divided into two stages: stage1 for Retriever, stage2 for Reranker.
- SimCSE method for training retriever models of stage1
- Text binary classification model for training reranker models of stage2
- Weighted blending reranker models for ensembling
- finding best threshold and post-processing

A4. Training Methods

training pipeline



CV strategy

We only used 4,000 random topics which **category != 'source'** as hold out data. Those topics were as the validation data and never used in any training process. This simple CV strategy was unexpectedly stable.

In the last month of the competition, we changed 4,000 topics to 1,000, which was still relatively consistent until we started to ensemble.

Retriever

We used unsupervised SIMCSE (Simple Contrastive Learning of Sentence Embeddings: https://github.com/princeton-nlp/SimCSE) method for training retriever models.

Training retriever

- only used positive samples from correlations.csv for unsupervised simcse training
- random choice 100 negative samples per validation topic from same language, for validation set
- content text format: `title [SEP] kind [SEP] description [SED] text` , maxlen = 256 (string level)
- topic text format:
  `title [SEP] channel [SEP] category [SEP] level [SEP] language [SEP] description [SEP] context [SEP] parent_description [SEP] children`
  maxlen = 256 (string level)
- `simcse_unsup_loss`

```
def simcse_unsup_loss(feature_topic, feature_content) -> 'tensor':
    y_true = torch.arange(0, feature_topic.size(0), device=device)
    sim = F.cosine_similarity(feature_topic.unsqueeze(1), feature_content.unsqueeze(0), dim=2)
    sim = sim / 0.05
    loss = F.cross_entropy(sim, y_true)
    loss = torch.mean(loss)
    return loss
```

- training code like:

```
for step, (inputs_topic, inputs_content, labels) in enumerate(train_loader):
        inputs_topic = collate(inputs_topic)
        for k, v in inputs_topic.items():
            inputs_topic[k] = v.to(device)
        inputs_content = collate(inputs_content)
        for k, v in inputs_content.items():
            inputs_content[k] = v.to(device)
        batch_size = labels.size(0)
        with torch.cuda.amp.autocast(enabled=CFG.apex):
            feature_topic = model(inputs_topic)
            feature_content = model(inputs_content)
            loss = simcse_unsup_loss(feature_topic, feature_content)
```

Choosing models

We tested on 3 models:

- paraphrase-multilingual-mpnet-base-v2
- all-MiniLM-L6-v2
- mdeberta-v3-base

Performance on 1,000 topics validation data:

model	F2@5	max positive score top50	max positive score top100
paraphrase-multilingual-mpnet-base-v2	0.5250	0.9135	0.9443
all-MiniLM-L6-v2	0.4879	0.9045	0.9353
mdeberta-v3-base	0.4689	0.8938	0.9187

making rerank dataset

We used our best simcse finetuned model (paraphrase-multilingual-mpnet-base-v2) to infer on train set topics, calculate cosine similarity for each topic and all content samples of the topic's language and then choose top100 samples. We also added all positive samples from correlations.csv.

Texts were prepared as same as stage1. Then concatenate to text pairs: `topic [SEP] content`.

Reranker

The reranker in stage2 basically is a binary classification model.

Choosing models

We tested on 4 models:

- mdeberta-v3-base
- xlm-roberta-large
- xlm-roberta-base
- paraphrase-multilingual-mpnet-base-v2

We have used two model initialization methods. One is to directly load the model weight from huggingface, and the other is to load the model weight after simse finetuning. The performance of the two methods is basically the same, while the latter is slightly higher and can converge faster.

model	validation (1,000 topics)	LB score	PB score
mdeberta-v3-base (loading simcse weights)	0.7149	0.688	0.727
mdeberta-v3-base	0.6378	0.669	0.693
xlm-roberta-large (loading simcse weights)	0.6987	-	-
xlm-roberta-base (loading simcse weights)	0.6780	-	-
paraphrase-multilingual-mpnet-base-v2 (loading simcse weights)	0.6299	-	-

Training tricks

We used FGM (Fast Gradient Method), EMA (Exponential Moving Average) on training.

FGM+EMA can impove score by 0.01.

Finding threshold

We set the threshold in loop to calculate the f2 metric on the 1,000 topics validation data, codes like:

```
best_thres = 0.
best_score = 0.
best_n_rec = 10
for thres in tqdm(np.arange(0.01, 0.2, 0.005)):
    for n_rec in range(30, 50):
        test_sub = test_data[test_data['score'] >= thres].reset_index(drop=True)
        sub_df = test_sub.groupby('topic_id').apply(lambda g: g.head(n_rec)).reset_index(drop=True)
        score = calc_f2(sub_df, label_df)
        if score > best_score:
            best_score = score
            best_thres = thres
            best_n_rec = n_rec
```

When submitting a single model, this method basically CV-LB consistency (CV is about 0.02-0.03 higher than LB).

But When it came to the last two weeks of the competition, when started to ensemble, we lost the CV-LB consistency, and I think the reason may be that 1,000 topics validation data is not big enough.

Post-Processing

When dividing the threshold, we will have a small number of topics that do not match any contents. We just simply using top4 contents ranked by the original scores.

We tried recalling more contents for this part of topics, but LB score didn't improve.

We also tried different languages using different threshold, both CV and LB score dropped a little.

Ensemble

We had trained 20+ ranker models, trained on different number of recall samples per topic, like 50, 70, 100.

- mdeberta (simcse weights, 4,000 validate topics)
- mdeberta (simcse weights, 4,000 validate topics, with FGM,EMA)
- mdeberta (simcse weights, 1,000 validate topics)
- mdeberta (simcse weights, 1,000 validate topics, with FGM,EMA)
- mdeberta (1,000 validate topics, with FGM,EMA)
- xlm-roberta-large (simcse weights, 1,000 validate topics, with FGM,EMA)
- xlm-roberta-base (simcse weights, 1,000 validate topics, with FGM,EMA)

We used LinearRegression to fit on 1,000 topics validation model output score to get coef_ array, and then used as blending weights:

```
pcols = [c for c in valid_data.columns if c.startswith('score')]
for cols in tqdm([i for i in combinations(pcols, 10)]):
    cols = list(cols)
    X = valid_data[cols].values
    y = valid_data['label'].values
    lr = LinearRegression().fit(X, y)
    coef = lr.coef_
    print(get_score(valid_data, df_target_metric, cols, coef))
```

We started with 100 recall samples per topic, but due to time limits, we can only use up to 6 models. So we tried 70 and 50 recall samples in the later stage of the competition.

number of recall samples per topic	models	validation (1,000 topics)	LB score	PB score
100	6	0.725	0.705	0.738
70	10	0.738	0.714	0.749
50	12	0.743	0.715	0.751

A5. Interesting findings

- What was the most important trick you used?

> The most important trick we used is training Retriever model with the SimCSE method.
>
> The hard negative samples from retriever model can greatly improve the performance of ranker models.

retriever model (max positive score top100)	ranker f2 score (LB)
0.80	0.585
0.94	0.688

- What do you think set you apart from others in the competition?

> First, we have a solid CV strategy which not divided by channel like others talking in disscussion. On the other hand, we think our advantage over others in the competition is that we can use relatively more GPUs to train a large number of fine-tuned ranking models for ensemble.

A6. Simple Methods

Our solution looks relatively heavy compared to others, so we provide a simplier but less effective solution here:

- retriever model training epochs reduce to 50.
- rerank model training epochs reduce to 5.

In this solution, we should get a PB score of about 0.67.

A7. Model Execution Time

- How long does it take to train your model?

> - Retriever model / paraphrase-multilingual-mpnet-base-v2: training for 120 epochs, 9min/epoch on A100, total 1080 minutes.
> - Reranker model / mdeberta-v3-base: training for 10 epochs, 440min/epoch on A100, total 4400 minutes
> - Reranker model / xlm-roberta-large: training for 10 epochs, 730min/epoch on A100, total 7300 minutes
> - Reranker model / xlm-roberta-base: training for 10 epochs, 360min/epoch on A100, total 3600 minutes

- How long does it take to generate predictions using your model?

> - Single retriever model with single rerank mdeberta-v3-base model took 80 minutes to generate predictions in the Kaggle GPU notebook.
> - Single retriever model with 12 ensembling rerank models took 7.5 hours to generate predictions in the Kaggle GPU notebook.

- How long does it take to train the simplified model (referenced in section A6)?

> - training retriever model took 7.5 hours (50 epochs) on A100.
> - training reranker models took 36 hours on A100.

- How long does it take to generate predictions from the simplified model?

> 1 hour on Kaggle GPU notebook.

A8. References

- tips from hosts: https://www.kaggle.com/code/jamiealexandre/tips-and-recommendations-from-hosts
- text pre-processing: https://www.kaggle.com/competitions/learning-equality-curriculum-recommendations/discussion/376873
- stage1 and stage2 train and submit pipeline: https://www.kaggle.com/competitions/learning-equality-curriculum-recommendations/discussion/373640
- stage1 and stage2 modeling: https://www.kaggle.com/competitions/learning-equality-curriculum-recommendations/discussion/381509, https://www.kaggle.com/code/ragnar123/lecr-xlm-roberta-base-baseline
- SimCSE: https://github.com/princeton-nlp/SimCSE, https://github.com/yangjianxin1/SimCSE