MIND News Recommendation Technical Report

09.13.2020

Huige Cheng
Sogou

Overview:

In this competition, we will use 4 weeks data to predict user clicks in the 5th week(test dataset). With 87.5% new docs in the test set we can view this problem as a content based recommendation problem. Better representation of news and clicked news sequence is key for this problem.

I do the contest by myself, as training the model is very costly using P100 or T4, I did not try complex models, just relying on the familiar networks from my previous experience.

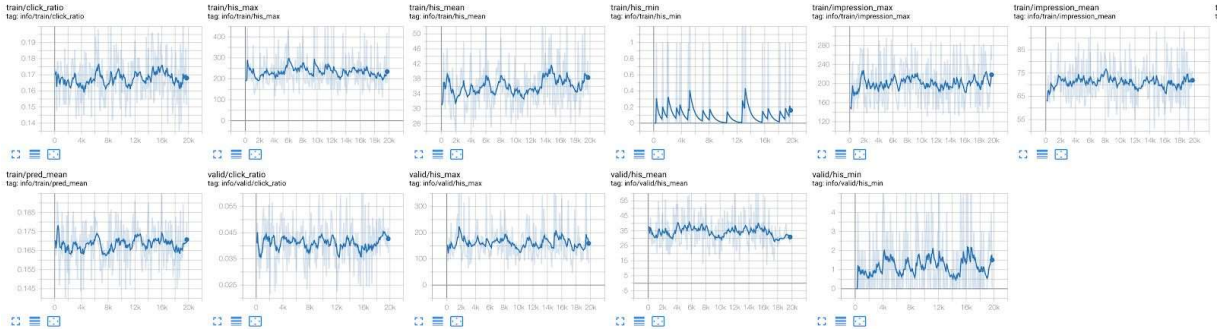
Open Source part of my model code:

https://github.com/chenghuige/mind

Dataset info:

User's max history length might be > 400 but average history length is about 35.

Only 4% positive ratio, we used downsampling to make the train set positive ratio about 16.5%.



How to evaluate?

1. There are huge difference between dev set(only 1day) and test set(1 week)

	dev	test
New user	15.3%	22.1%
New doc	32.3%	**87.5%**

2. New doc ratio is much higher in the test set then dev , that means algos highly depend on docid which got good results on dev set will not perform well on test set.

   But we found docid still helpful for the dev performance and we still use doc id as an input feature.

3. In order to make dev and test results consistent we **randomly masked 92% docids as UNK** in the dev set which do not exist in the train set for evaluation and we found using this strategy we got similar results on both dev and test.

4. However if we add context features like titles of the same impression we can still increase the dev result a lot but get worse results on the test set.

   Anyway just using id and nlp features of news we can get consistent results on both dev and test.

5. We have found **global auc of new docs** is a very important evaluation metric also. Higher new doc auc means better generalization on new docs.
6. We found evaluation of 100w instances is enough(not much difference with using all dev instances), so we just use the first 100w dev instances for evaluation. We also use **async validation** so we can get evaluation results in parallel with the training process.

   The dev result I will mention below will always be using the first 100w dev instances and randomly mask 92% docids.

Key points of model:

1. Pointwise training
2. **Shuffle between impressions**

   We found shuffle between impressions will get better per impression auc(which is the metric we are focusing on) while totally random shuffle all instances will get better global auc.

3. **Downsampling of negative instances**

   With only about 4% positive ratio. We found just using ⅕ negative instances can get nearly the same result while greatly reducing training time.

   We just randomly shuffle each impression to get ⅕ negative instances per impression. By doing this we got 5 datasets(all positive instances + ⅕ negative instances dataset 0, dataset 1... dataset 4).

   We did not do much experiments here, this might not be the best down sampling method ,for example we might leave all negative instances for impressions with less instances(<5). Also the split method we use will make dataset 1 with less negative instances then dataset 0.

   We just use **dataset 0** for most of our experiments, dataset 1-4 will be used for ensemble only if needed.

4. **Use all available infos**

   Class, entity, title, abstract and **body** all useful. (MIND paper has same conclusion)

   We just use the latest 10 abstracts and bodies in this competition for fast training.

After the contest we find that increasing the number of bodies can may further improve model performance.

5. Adds dev data for training
6. Ensemble

Just use 1 model training on 5 different datasets in parallel(different negative instances) and finally do mean ensemble of the 5 models(Our best test 10% result used 4 datasets 0,1,2,4, due to time limit dataset 3 not finishing inferencing).

7. Model structure

   We model each field like uid,docid,class, title, abstract,body separately and use **concat + dot pooling(inner product** for each field pair like **DLRM**) before feeding to mlp.

   For candidate news we use **attention pooling** for words of title, abstract and body.

   For history clicked news we use **attention pooling** for docids, cats,subcats and use **DIN attention pooling** for entities, titles, abstracts and bodies.

   Features:

   ['uid', 'docid', 'history_docids',

   'cat', 'sub_cat', 'history_cats','history_sub_cats',

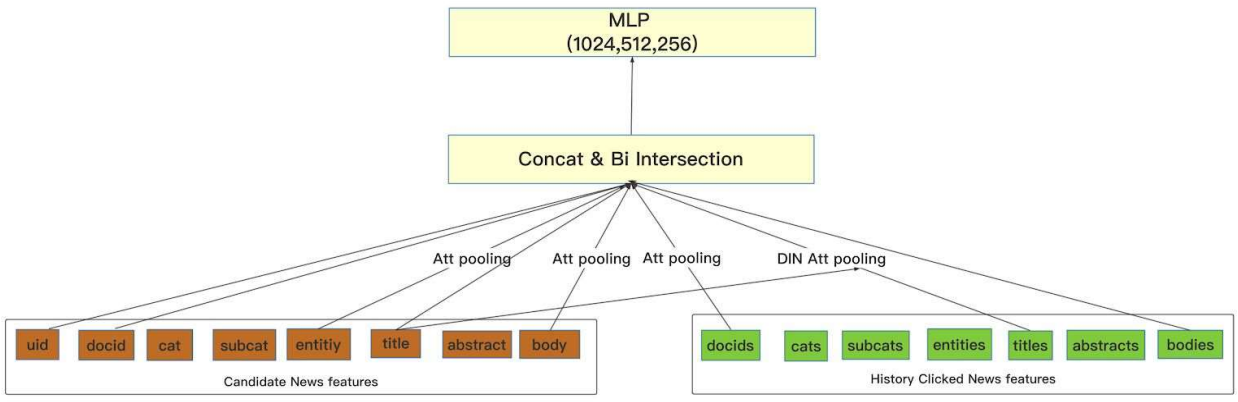
   'title_entities', 'abstract_entities',

   'history_title_merge_entities', 'history_abstract_merge_entities',

   'title', 'history_titles',

   'abstract', 'history_abstracts',

   'body', 'history_bodies']



8. Nlp part details:
    - Using bert base cased tokenizer and it's vocab(28996)
    - Word embedding init using glove pretrain(on train+dev+test title and abstract corpus), l2 normed.
    - For entity wiki embedding also do l2 norm before loading.
    - Using attention pooling for title encoding .
    - Using DIN attention pooling for history titles encoding.
9. Parameters:

    Adam optimizer, base lr 0.001, batch_size 1024, 1gpu training

    Lr schedule: 0.1 epoch warm up, triangle lr, 1epoch training.
    Uid: freq > 10 in train, total size 113534
    Docid : all in train + dev + test, total size 130381

    Title: max 30 words.

    Abstract: max 50 words.

    Body: max 100 words.

    Max history docids: 200

    Max history titles: 50

    Max history abstracts: 10

    Max history bodies: 10

    Entity: max 200 doc history with each doc at most 2 entities.

What do not work on the test:

1. Tring to optimize auc directly or other loss methods like adding pair loss.
2. Dense features like history length, doc freshness, and other features.
3. Context features like time, impression related features.

    Maybe the test set impression sequence is shuffled ?

    Adding surrounding recall titles in the same impression did improve on dev set +0.4% auc but got lower auc on the test.

4. Training day by day.

    Training day by day will pay more attention to latest instances, and be natural to add dev data at day 7. However training day by day will converge slower and the final result is not as good as random shuffle between impressions.

5. Training more than 1 epoch.

I spent a lot of time investigating this. Might due to overfit on docid and uid, removing them helps on this, but since it hurt the performance on dev set I did not try to submit results without docids on the test.

Milestones:

	Dev auc (first 100w with masked docid)	Test auc (10%)
uid + docid + history_docids	0.61 (without dev mask docid)	0.5272
+cat, entity		0.6763
+title, abstract	0.6987	0.6979
Adjust parameters (base model)	0.7004	0.7036
+body (body model)	**0.7042**	**0.707** **(best single model)**
+dev data		0.7104 (best single mode with dev data)
4 datasets of body model ensemble		0.7133

Additional Experiments:

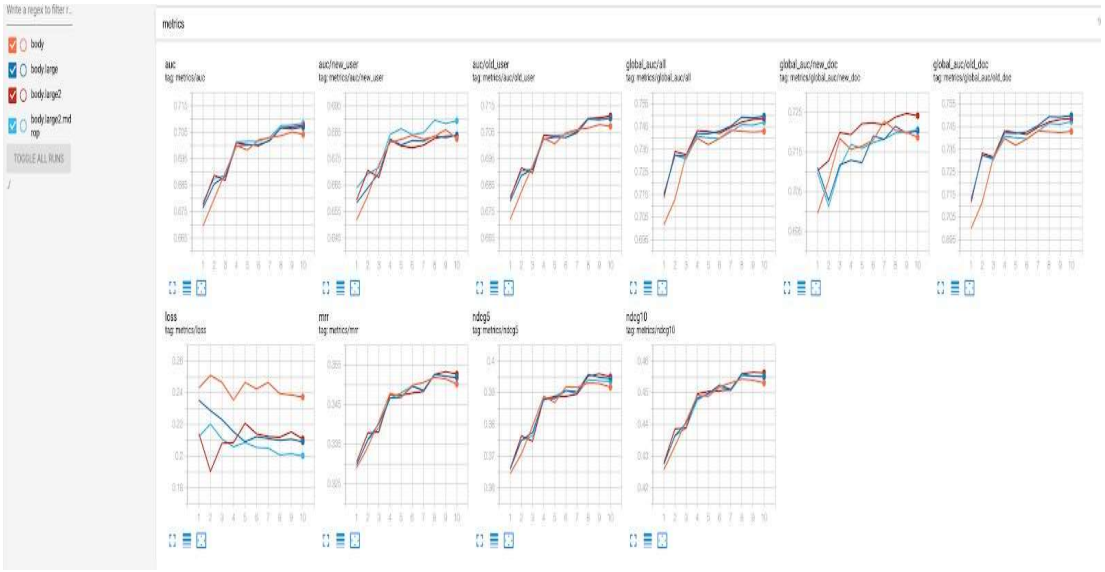
After the contest, we did some additional experiments using kaggle and colab tpu, it can train body.large in **45 minutes** while using a single P100 needs 5hours to finish(unfortunately for me using the same code to train on multiple gpus is very slow).

With tpu 8 cores we use batch size 256 * 8 = 2048, and base lr 0.002 seems better than 0.001.

1. **Use more abstract and body histories**.

   From 10 to **30**(Body.large) and **50**(Body.large2) the auc on dev set can increase **0.3%-0.4%** and all other metrics also improve so I believe this also works on the test.

Model	AUC (test 10%)	AUC (dev)	MRR (dev)	NDCG@5 (dev)	NDCG@10 (dev)	AUC (new doc)
base	0.7036	0.7004	0.3477	0.3888	0.4502	0.7125
body	**0.707**	**0.7042**	0.3502	0.3918	0.4531	0.7187
body.large		0.7069	0.3518	0.3944	0.4550	0.7204
body.large2		**0.7077**	0.3527	0.3951	0.4562	0.7242
body.large2 +multi-sample dropout		**0.7084**	0.3519	0.3936	0.4553	0.7207



Eval on each 0.1 epoch

2. Bert

   Thanks to tpu we can try bert encoder now. I choose tiny-bert(2 layers, 2heads, output 128) for speed concern.

   Let's first just compare using only uid,docid,history_ids and title, history_titles.

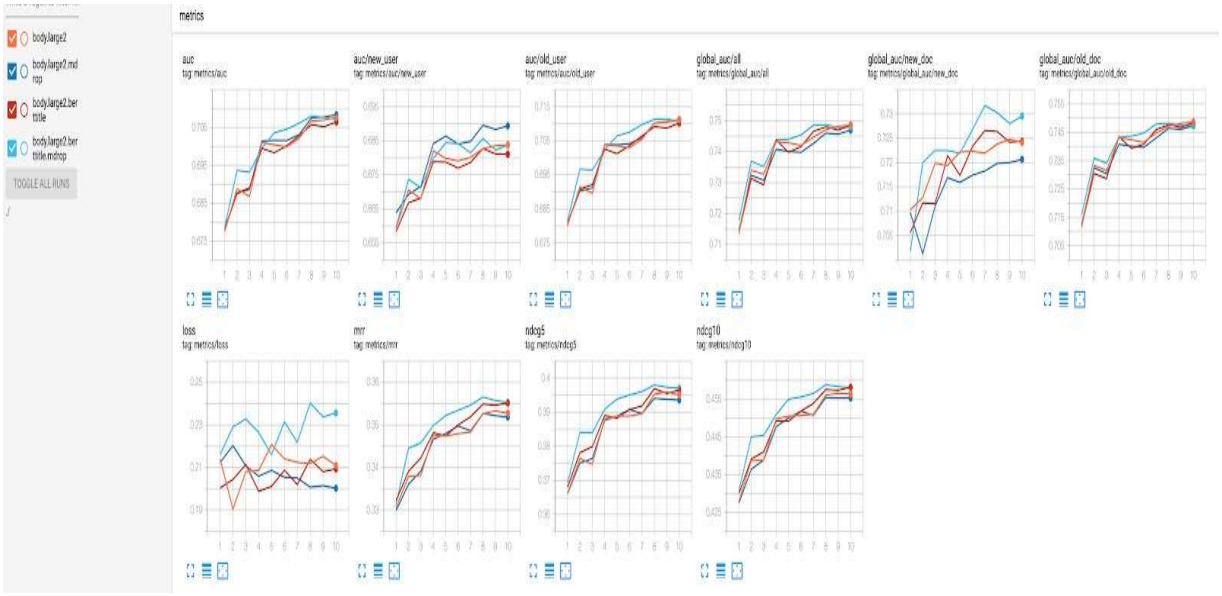
   Results show bert is a much better title encoder than using simple words attention pooling.

   Continue pretrain performs even better, I think the next sentence prediction also helps a lot here as we use clicked titles sequence as training corpus.

Model	AUC (dev)	MRR (dev)	NDCG@5 (dev)	NDCG@10 (dev)	AUC (new doc)	TrainTime
titleonly uid,docid,history_docids title,his_titles	0.6916	0.3445	0.3826	0.4443	0.6996	40m
bert.titleonly Use bert to encode title (uncased_L-2_H-128_A-2)	0.6987	0.3465	0.3862	0.4483	0.7123	1h55m
bert2.tiitleonly Continue pretrain on clicked titles (train + dev + test)	**0.7034**	0.3504	0.3906	0.4526	**0.7273**	

Then we add bert title to body.large2. But interestingly it did not improve auc compared to baseline model body.large2(though other metrics mrr, ndcg all improve). Seems adding bert title is overfit prone ? One possibility is batch size, as for using bert we have to use batch size 128 * 8 on tpu v3 and 64 * 8 on tpu v2, and seems the smaller batch got worse results. I believe we might get better results with more experiments and at least the bert version can help a lot on ensemble:)

Model	AUC (dev)	MRR (dev)	NDCG@5 (dev)	NDCG@10 (dev)	AUC (new doc)	TrainTime
body.large2	**0.7077**	0.3527	0.3951	0.4562	0.7242	1h
+bert title	**0.7065**	0.3551	0.3965	0.4581	0.7244	2h tpu v3 3h tpu v2
+multi sample dropout	**0.7072**	0.3553	0.397	0.4579	0.7296	
Replace title,abstract,body all to bert encoding						6h30m tpu v3



Thanks

Thanks to the mind organizer for holding such a great contest, the large mind dataset is very valuable for recommendation research.

The MIND paper helped me a lot.

https://msnews.github.io/assets/doc/ACL2020_MIND.pdf

Also learn a lot from the excellent open source code microsoft provides.

https://github.com/microsoft/recommenders

I'm very fortunate to have the chance to focus on the rank model of Sogou feed. Great thanks to all my feed colleagues.