Solution of 2nd Place Logloss and 3rd Place Efficiency of Feedback Prize - Predicting Effective Arguments (Team SKT)

Shujun He, Tomohiro Takesako*, and Kossi Neroma†

August 29, 2022

1 Background of our team

We participated in the recent competition "Feedback Prize - Predicting Effective Arguments", which was hosted during the third quarter of 2022. Our team name is SKT. Our public leaderboard score is 0.55306 (2nd) and our private score is 0.55453 (2nd). We also rank 3rd on the efficiency leaderboard with an efficiency score of 1.055925.

Our team consistes of Shujun, who is a PhD student in chemical engineering at Texas A&M University in College Station, TX, US, Tom, who is a data scientist at Sprout.ai, and Kossi Neroma, who is a junior data scientist at Publicis Sapient.

2 Solution overview

Each of our team members has their own training pipeline for transformer models. On a high level, our transformer models look at the entirety of each essay and output predictions of effectiveness for each discourse either via pooling of discourse tokens or a classification token added to the front of the each discourse. Importantly, directly inputting essays results in a situation where the model is not sure about where it needs to make predictions, so to circumvent this, we use either a prompt (i.e. concat f'({discourse_type} start)' and f'({discourse_type} end)' to the beginning and end of each discourse to signal where predictions need to be made) or simply concat special tokens added to the tokenizer instead (i.e. f' [discourse_type]' and f'[discourse_type\]') (Figure 1). Following training of transformer models, we also use stacking models (e.g. xgboost) to further improve predictions. Further, we utilize unlabled data from the previous Feedback competition in a semi-supervised (i.e. pseudo-labeling) setting.

```
'(Lead start)Hi, i\'m Isaac, i\'m going to be writing about how this face on Mars is a natural landform or if there
is life on Mars that made it. The story is about how NASA took a picture of Mars and a face was seen on the planet.
NASA doesn\'t know if the landform was created by life on Mars, or if it is just a natural landform.(Lead end) (Pos
ition start)On my perspective, I think that the face is a natural landform because I dont think that there is any l
ife on Mars. In these next few paragraphs, I\'ll be talking about how I think that is is a natural landform(Positio
n end)\n\n(Claim start)I think that the face is a natural landform because there is no life on Mars that we have de
scovered yet(Claim end). (Evidence start)If life was on Mars, we would know by now. The reason why I think it is a
natural landform because, nobody live on Mars in order to create the figure. It says in paragraph 9, "It\'s not eas
y to target Cydonia," in which he is saying that its not easy to know if it is a natural landform at this point. In
all that they\'re saying, its probably a natural landform.(Evidence end)\n\n(Counterclaim start)People thought that
the face was formed by alieans because they thought that there was life on Mars.(Counterclaim end) (Rebuttal start)
though some say that life on Mars does exist, I think that there is no life on Mars.(Rebuttal end)\n\n(Evidence sta
rt)It says in paragraph 7, on April 5, 1998, Mars Global Surveyor flew over Cydonia for the first time. Michael Mal
in took a picture of Mars with his Orbiter Camera, that the face was a natural landform.(Evidence end) (Counterclai
m start)Everyone who thought it was made by alieans even though it wasn\'t, was not satisfied. I think they were no
t satisfied because they have thought since 1976 that it was really formed by alieans.(Counterclaim end)\n\n(Conclu
ding Statement start)Though people were not satified about how the landform was a natural landform, in all, we new
that alieans did not form the face. I would like to know how the landform was formed. we know now that life on Mars
doesn\'t exist.(Concluding Statement end)                '
```

Figure 1: Example of a preprocessed essay. The first discouse is segmented by prompts and highlighted.

*Refered to as Tom later on

†Authors are listed in alphabetical order

3 Transformer modeling

3.1 Encoders

Deberta [2, 1] worked the best since it supports unlimited input length and uses disentangled attention with relative positional embedding; in fact, our ensemble consists entirely of deberta variants. For Shujun, it was also helpful to add a GRU/LSTM after pooling on the pooled discourse representations. Tom used Shujun's SlidingWindowTransformerModel from the last Feedback competiion (https://www.kaggle.com/competitions/feedback-prize-2021/discussion/313235), which stablized training for him.

3.2 Pretraining

Kossi used pretrained weights from his solution in the last competiion (https://www.kaggle.com/competitions/feedback-prize-2021/discussion/313478) and Tom and I found pretrained tascj0 models to be good starting points. We used some of the weights that tascj0 released after the last Feedback competition and Tom also pretrained some new ones on his own. Please checkout out tascj0's solution post if you'd like to learn more (https://www.kaggle.com/competitions/feedback-prize-2021/discussion/313424). In addition, Tom used MLM for some of his models. Further, some of our models simply used huggingface weights.

3.3 Max sequence length

Shujun used a max sequence length of 1280 in both training and inference, since he found that 99.9% of discourses fall within that range, whereas other teammates used up to around 1800 during inference and as low as 648 in training.

4 Pseudo labeling

Pseudo labeling is an integral part of all our solution. We use essays from the training set of last Feedback competition that are also not present in the training set of this competition. Our procedure is as follows:

1. Train model with gt labels
2. Make predictions for old data (around 11000 essays) with each fold model
3. Retrain model with crossentropy on pseudo label probabilities (not discretized) generated by previous model trained on the same fold data: 3 epochs on pl labels only first and then 3 more epochs on gt labels only
4. Repeat from step 2

For Shujun's pipeline, I saw improvement until 5 rounds of the above procedure. For Tom, it was only helpful for one round and Kossi did not have enough time to try multi-round pl.

5 Stacking

Stacking provides significantly improvement in both cv/lb (around 0.004). Our stacking framework is primarily inspired by my team's solution (https://www.kaggle.com/competitions/feedback-prize-2021/discussion/313235) in the previous feedback competition. In addition to predicted probabilities outputted by the transformer models, we also utilized the token probabilities for each discourse, which we call prob_sequences. Compared to the previous Feedback competition, stacking is much faster since we don't have to deal with a huge amount of candidate sequences. Our features are as follows:

```
def get_xgb_features(train_df,prob_sequences):
    features2calculate=[f"instability_{i}" for i in range(4)]+\
    [f"begin_{i}" for i in range(3)]+\
    [f"end_{i}" for i in range(3)]#+\
    #["entropy"]

    calculated_features=[]
    for i,prob_seq in tqdm(enumerate(prob_sequences)):

        tmp=[]
        #quants = np.linspace(0,1,n_quan)
        prob_seq=np.array(prob_seq)
        instability = []
        #all_quants=[]
        tmp.append(np.diff(prob_seq[:,:],0).mean(0))
        tmp.append([(np.diff(prob_seq[:,[1,2]].sum(1))**2).mean()])

        tmp.append(prob_seq[:5,:].mean(0))
        tmp.append(prob_seq[-5:,:].mean(0))

        calculated_features.append(np.concatenate(tmp))


    train_df[features2calculate]=calculated_features
    train_df['len']=[len(s) for s in prob_sequences]

    calculated_features=np.array(calculated_features)
    calculated_features.shape

    p_features=[]
    n_features=[]
    neighbor_features=['Ineffective','Adequate','Effective','discourse_type']
    neighbor_features_values=train_df[neighbor_features].values
    for i in tqdm(range(len(train_df))):
        if i>1 and train_df['essay_id'].iloc[i]==train_df['essay_id'].iloc[i-1]:
            p_features.append(neighbor_features_values[i-1])
        else:
            p_features.append(neighbor_features_values[i])

        if i<(len(train_df)-1) and train_df['essay_id'].iloc[i]==train_df['essay_id'].iloc[i+1]:
            n_features.append(neighbor_features_values[i+1])
        else:
            n_features.append(neighbor_features_values[i])

    train_df[[f+"_previous" for f in neighbor_features]]=p_features
    train_df[[f+"_next" for f in neighbor_features]]=n_features

    train_df['mean_Ineffective']=train_df.groupby("essay_id")["Ineffective"].transform("mean")
    train_df['mean_Adequate']=train_df.groupby("essay_id")["Adequate"].transform("mean")
    train_df['mean_Effective']=train_df.groupby("essay_id")["Effective"].transform("mean")

    train_df['std_Ineffective']=train_df.groupby("essay_id")["Ineffective"].transform("std")
    train_df['std_Adequate']=train_df.groupby("essay_id")["Adequate"].transform("std")
    train_df['std_Effective']=train_df.groupby("essay_id")["Effective"].transform("std")

    train_df['discourse_count']=train_df.groupby("essay_id")['discourse_type'].transform("count")

    cnts=train_df.groupby('essay_id')['discourse_type'].apply(lambda x: x.value_counts())

    #new_df=[]
    discourse_types=['Claim','Evidence','Concluding Statement','Lead','Position','Counterclaim','Rebuttal']
    value_count_hash={}
    for t in discourse_types:
        value_count_hash[t]={}
    for key in cnts.keys():
```

```
        value_count_hash[key[1]][key[0]]=cnts[key]

    discourse_cnts=[]
    for essay_id in train_df['essay_id'].unique():
        row=[essay_id]
        for d in discourse_types:
            try:
                row.append(value_count_hash[d][essay_id])
            except:
                row.append(0)
        discourse_cnts.append(row)

    discourse_cnts=pd.DataFrame(discourse_cnts,columns=['essay_id']+[f'{d}_count' for d in discourse_types])
    #discourse_cnts

    train_df=train_df.merge(discourse_cnts,how='left',on='essay_id')
    train_df

    #train_df

    return train_df

```

Since stacking is fast, it works best when we use each fold predictions with xgb separately and then avg. For instance, because Shujun has 6 folds of neural network models and 6 folds of xgb models, this way he has 6x6=36 preds to avg for each single model.

6 Best single models

All our best single models were deberta-large/deberta-v3-large variants. For Tom and Kkiller, their best single models came from 1st round PL, whereas for Shujun it came from 4th round PL (Table 1).

	Shujun	Tom	Kkiller
Public LB	**0.560**	0.566	0.562
Private LB	**0.558**	0.571	0.562
Local CV	0.571	N/A	0.572

Table 1: Best single models of team SKT.

7 Ensemble

Our final submission is a weighted average of best ensembles from each individual member (Table 2). For Shujun, his best ensemble consists of 6 folds of (5th round pseudo-labeling models):

1. deberta-v3-large (tascj0 span detector pretrained)
2. deberta-v2-xlarge (tascj0 span detector pretrained)
3. deberta-v1-xlarge (tascj0 span detector pretrained)
4. deberta-v3-large (huggingface pretrained)
5. deberta-v2-xlarge (huggingface pretrained)

For Tom, his best ensemble consists of 5 folds of:

1. deberta-large (tascj0 span detector pretrained, 2nd round pseudo-labeling model, stacking not applied)
2. deberta-v3-large (tascj0 span detector pretrained, 2nd round pseudo-labeling model, stacking not applied)

3. deberta-large (tascj0 span detector pretrained, 1st round pseudo-labeling model)
4. deberta-v3-large (tascj0 span detector pretrained, 1st round pseudo-labeling model)

For Kkiller, hist best ensemble consists of 5 folds of:

1. deberta-large (kkiller pretrained + psl + shujun stacking)
2. deberta-v3-large (kkiller pretrained + psl + shujun stacking)
3. deberta-xlarge (kkiller pretrained + psl + shujun stacking)
4. deberta-v2-xlarge (kkiller pretrained + + psl + shujun stacking)

	Shujun	Tom	Kkiller	Ensemble
Public LB	**0.55727**	0.56190	0.55931	0.55306
Private LB	**0.55538**	0.56648	0.56245	0.55453
Weight in final ensemble	0.45	0.25	0.3	

Table 2: Best ensembles of team SKT.

8 Efficiency submission

Our efficiency model is the same as Kossi's deberta-v3-large mentioned above except that it was trained on all the data (psl + ground truth) for 3 epochs. For faster inference, Kossi also includes a [trunc...] token which allows to have a max length under 1024 whith no decrease in performance. Therefore, all discourses under the max length are taken into account and discourses appearing after max length are truncated. For all Kossi's models, discourse-pooling (when applied, otherwise a custom classification token) occurs before softmax.

9 Some more tips/tricks

For Tom AWP was useful, and he reported around 0.003 cv improvement with AWP (eps=1e-3 and lr=1e-2 for large models, 1e-4 for xlarge models).

It was also important to split the data with StratifiedGroupKFold instead of GroupKFold. For me I started out with GroupKFold but found better correlation between cv/lb after switching to StratifiedGroupKFold.

For ensembling models with the same cv split, we used GP_minimize (from sklearn) to find optimal weights and otherwise weights were determined arbitrarily.

Gradient accumulation was useful since we had to deal with very long sequences.

References

[1] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543, 2021.

[2] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654, 2020.