BiliMemeNet: Using NLP to Learn Chinese Meme From Bullet Chat

Hongyu Tu hongyutu@umass.edu
Zihao Mao zmao@umass.edu
Chang Xue cxue@umass.edu
Yichao Zhang yichaozhang@umass.edu

1 Problem statement

Bilibili is a popular video website in China that provides user generated content (UGC), just like Youtube. A unique feature that Bilibili provides is the bullet chats, a creative form of real time comments that fly across the screen and then disappear. [1]

In traditional comments below the video, users usually post long and general reviews about the whole video, while in the bullet chats, the comments are short and specifically written for the current frame of the video. Therefore, there lies an abundance of inside jokes, memes and repartees in the bullet chats.

Bilibili sets up 27 categories, including Life, Animals, Technology etc. Therefore, each video is labeled with a category id. Based on our observation, each category has its own inside jokes. For example, the inside jokes from "animal" category are bound to be very different from inside jokes from "technology" category. On the other hand, there are some memes that are popular among all categories. Therefore, we are curious to learn how bullet chats are different among categories.

As the first task of our project, we would like to build a dataset out of the bullet chats. By crawling and preprocessing chat messages and certain properties from most played videos, we can get a fair amount of bullet chats in the form of text.

The second task of our project is to build a text classification model that given a chat message, predicts the corresponding category id. We will start with a baseline model to see if the task is valid, then we will use pretrained models such as Chinese BERT to exploit more complex correlations. We formally state our research problem as follows. For video v_i, its category is c_i, and the prediction is p_i. For each i, we want to maximize the accuracy of our prediction.

2 What you proposed vs. what you accomplished

In the proposal, we claim that we want to find and classify the memes in 127 subcategories. But we realize that annotating the memes in bullet chats requires a huge amount of human work. So we just trained our model to classify the bullet chats but not the memes. We also reduced the number of categories since its unpractical to classify 127 classes for a model.

- ~~Collect and preprocess bullet chats dataset~~
- ~~Build and train a baseline model (Linear~~

[1] In order to include Chinese characters in this paper, we had to use XeLatex to compile it, which results in larger line space. We will present with more than 8 pages to satisfy the page requirement.

~~classifer) on collected dataset and examine its performance~~

- ~~Build and finetune pretrained models on collected dataset and examine its performance~~

- ~~Implement discrete prompt and prompt tuning~~: not planned in proposal

- *Conduct hierarchical classification*: the accuracy for first-level category classification is low, so we do not aim for the harder task

3 Related work

3.1 Meme

The word of "meme" is first introduced by Richard Dawkins in his book The Selfish Gene many years ago and now become so popular in internet. (Dawkins, 1976) It's been observed and researched by many scholars like Susan Blackmore from Oxford. (Blackmore, 1999) She formally explained the relation between current meme and the science of Memetics in her book of The Meme Machine.

As many memes are constantly created by people online, the data has gained its attention by social media company like Facebook (now Meta). There are data sets of meme images built by Facebook that's for the task of hateful meme detection. (Kiela et al., 2021) Different from the bullet chat data, this dataset includes both image data and text data, and it's not mean for the measurement of popularity.

Many works have been done for the task, and one of the best ones currently is HateDetectron27 built by Riza Velioglu and Jewgeni Rose in 2020. (Velioglu and Rose, 2020)

3.2 Chinese Meme and bullet chat

Our data, Chinese meme, however, is slightly different from the western concept of meme, because it is text only. It's a very popular concept especially in the Chinese video sharing website, Bilibili. Therefore, we are using the data gathered from the bullet chat in Bilibili. Some studies has been made for the properties of bullet chats. (Mei, 2021) It's a new communication tool becoming popular nowadays.

3.3 Similar Data set

There are some similar data sets for text classification. Thanks to the work done by Zhang's research group from New York University, many large scalar data set have been collected and processed. (Zhang et al., 2016) Some smaller data sets contain less categories, like AG's news corpus[2] data set which only contains four classes with around 127k data samples. Some contain more classes like the DBPedia ontology data set with 14 ontology classes and 630k data samples. (Lehmann et al., 2014)

Like our bullet chat data set, Germeval 2021 data set is also collected from web users online. It's built to identify toxic comments in German. It contains four classes and over 4000 annotated Facebook user comments. (Risch et al., 2021)

Sogou News data set is in the same language with our data set, which is in Chinese. It contains 2,909,551 sample news and 5 major classes. Some reprocessing steps that they applied in the data set is very relatable to our bullet chat data set. They used pypinyin package to translate the data into Pinyin format,

[2] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

which can represent Chinese words with English letters. They also used Jieba Chinese segmentation system to tokenize the texts. (Zhang et al., 2016)

3.4 Text classification

Tex classification is for categorizing texts like sentences or documents, in our case, bullet chats. There have been many works done in the field, and many popular models made already. For example, XLNet is currently one of the best performing model for classifying the news articles from AG's News Corpus. (Yang et al., 2020) ERNIE-Doc is great for categorizing movie review from IMDb. (Ding et al., 2021) BERT that's fine tuned like BERT-ITPT-FiT is also effective on the type of tasks. (Sun et al., 2020)

One of the work that's more similar to ours is classifying the comments from websites like Facebook. The models of GBERT/GELECTRA Ensemble are the ones that performs great in the GermEval 2021 data set which label the comments by toxic, engaging and fact-clamming. (Bornheim et al., 2021)

3.5 Chinese NLP model

Due to the essential difference between Chinese and English, most of the models above need to be retrained in order to be used in our data set.

Currently, many popular models have been trained to understand Chinese already. For example, Universal Encoder Representations uses BERT for Chinese texts to create a serials of models called Chinese RoBERTa. (Zhao et al., 2019a) GPT-3 is also a great large model, but its built for English and it's hard to retrained for Chinese due to its huge size.

WuDao introduced the current largest Chinese pre-trained model, CPM, with 2.6 billion parameters. (Zhang et al., 2021) They also introduced a new model called GLM which outperforms BERT models. (Du et al., 2021) These two could potentially be the best Chinese models so far, but we are not sure if we could access them for our project. Further communication with WuDao is needed. In addition, due to the size of the model, we are not sure if we have enough computing resources for such big scale model.

UER-py provides an open resources of pretrained NLP models and general-domain corpus. (Zhao et al., 2019b) The model zoo that they provide inludes the pretrained Chinese RoBERTa, Chinese GPT-2, Chinese ALBERT, and Chinese T5, etc. This resource could be the major source of our selected pretrained model, and we could fine tune the models using our data.

4 Your dataset

Our data comes from video bullet chats from Bilibili website. Bullet chats text is made of comments from other viewers and it pops up while you are watching the video. A bullet chat usually is a short sentence that has only a few words or phrases. We build our dataset by using a web crawler to get the bullet chat from the hottest videos in different categories. Each video may include thousands of bullet chat and we collect over one million of them in total. Some examples of crawled bullet chats is in Figure 1.

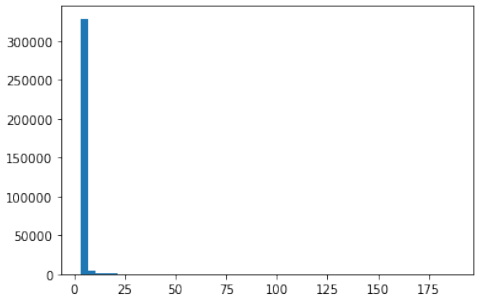
4.1 Data preprocessing

Jieba is a powerful Python module for Chinese text segmentation. We use Jieba li-

	danmu	Frequency	BVID	Source Video Title	Category ID
0	1人在观看	1	BV1M44y1n7Db	这个up又疯了他算出了海绵宝宝家里有多大附带详细户型图	27
661217	给内陆的朋友科普一下海边会随机刷这种水管大海里的水就是把水倒进水管里才有的	1	BV17i4y1273i	大庆赶海退潮后发现大蛏子会吐水的呼吸孔撒一点盐往外跑	214
661218	还是撒谎撒的少	1	BV17i4y1273i	大庆赶海退潮后发现大蛏子会吐水的呼吸孔撒一点盐往外跑	214
661219	鹌鹑它也想出来呀	1	BV17i4y1273i	大庆赶海退潮后发现大蛏子会吐水的呼吸孔撒一点盐往外跑	214
661221	无能狂怒	1	BV17i4y1273i	大庆赶海退潮后发现大蛏子会吐水的呼吸孔撒一点盐往外跑	214

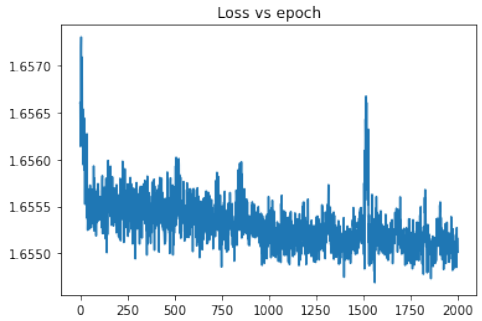
Figure 1: Dataset examples

brary to tokenize the bullet chat text by removing stopwords. For the tokenized bullet chat, we calculate the frequency of all bullet chats in each category. A bullet chat may appear across different categories and we decide to use the category with the most frequency as its label. After tokenizing all the sentences, we got a length distribution like this.



5 Baselines

We decided to go with a linear classifer as the baseline to handle our classification task, just to see how well a traditional model can perform without using any NLP trick. The model we used have 4 fully connected layer each with size set to 50 and leaky relu with value 0.1 at the end of each fully connected layer. A dropout was applied in the middle so we are less likely to overfit, and a softmax layer was added to the end of the model so it gives out a probability. The input to the model are the word tokens and the output are one-hot encoded category ids. With 2000 epochs and each time with a random mini batch, the model was able to achieve 24.7% accuracy when we restrict the data to only the five largest categories.



6 Your approach

6.1 Attempts on large-scale model

Unfortunately, in many of our approaches, we didn' t start small and build upon going, which costed us much time and resources. We first tried to access many large models like Pangu, CPM2 and GLM models from WuDao, those we later found out that are too large scaled for our available resources. For CPM2, it took around 8 GPUs each with 20G of memories only to prompt tune one of the small models in CPM2. However, we already took around multiple weeks to look at them and tried to set up the environments in Dockers, which was very expensive on time. The associated repository can be found here: CPM2, GLM, Pangu.

At the end, we only tested out the prompt tunning in Pangu, as explained in the following section.

6.2 Prompt Tuning

In recent years, the scale of language models (LM) become larger and larger, with GPT3() being the representative one. As we learned in lectures, with this huge scale of LM,

we can obtain a roughly same performance of zero-shot, one-shot and few-shot, without updating the gradients. Therefore, besides traditional scheme of pretrain and fine tuning, we also used prompt tuning to solve our classification problem. In this subsection, we will first introduce the pretrained model we chose, then we will show our methods for discrete prompts and prompt tuning.

6.2.1 Pretrained model

We used Pangu-Alpha-Evolution() as the pretrained model, which is an upgrade of Pangu-Alpha(). We chose Pangu for the following reasons. As we learned in class, in order to achieve better performance, we need larger scale of model size. Pangu is the first 200B-parameters Chinese language model, and it was trained on 80TB Chinese sentences, ranging from public dataset, encyclopedia, e-books etc. With such large scale, Pangu is able to achieve top performance on a series of downstream tasks, including reading comprehension, reasoning, text classification etc. Due to constraints of our computing resource, we could only use the smaller version of Pangu with 2.6B parameters to conduct our experiments. Another reason is that compared with another large-scale model CPM(), with same amount of parameters (2.6B), Pangu has greater ability of learning Chinese language, especially on few shot learning, which makes it a perfect match for our task.

6.2.2 Data preprocessing

As section 4 discussed, we built our dataset by crawling meme from 21 categories. However, the 21 categories are not strictly mutually exclusive. For example, "Movies and Television", "Movie" and "Television" are three categories, while the first one apprantly include the second and the third. Besides, after examinng the dataset, we noticed that given the size of our compiled dataset and the unqiueness of domain knowledge, it is hard for Pangu to predict the classification on 21 categories. Due to the above two reasons, for discrete prompt and prompt tuning, we merged 21 categories into 5 main categories, as shown in table 1. Note that the categories are originally in Chinese and we translated them into English.

As mentioned in section 4 for each piece of data, we have its probability distribution over 21 categories, then we calculate the distribution over 5 categories by summing the probabilities for the categories contained in the main category. Now we have the 5 scores for each data sample, corresponding to 5 main categories. The ground truth label for one sample is the main categories whose corresponding score is non-zero. Top-k ground truth would be the categories whose score is in the top-k out of 5.

6.2.3 Experimental Setup

We implemented prompt tuning based on the Pangu code repository. Pangu is implemented using PyTorch and Apex from NIVIDIA as the machine learning framework. For environment set up guide and notebook hacks, please see the "prompt tuning" section of our README.

As a language model, Pangu outputs a series of Chinese characters, and each ground truth label is also composed by Chinese characters. When we compare the model output with ground truth label, we select the token in model output that equals to a main category. In most cases, the model output is one of the main categories.

Main Category	Category List
Animation	Animation, Anime, Chinese Anime, Game, Remix
Music and Dance	Music, Dance
Knowledge	Knowledge, technology
Life	Sports, Cars, Life, Food, Animals, Fashion, News, Entertainment
Movie	Movies and Television, Movie, Television, Documentary

Table 1: 5 main categories

We use Acc@1 and Acc@3 as evaluation metrics. For one data sample, if the model output equals to one of the top-k ground truth label, Acc@k = 1; else, Acc@k = 0. For all data samples, we take the average of each sample's Acc@k. Obviously, we expect Acc@3 to be larger than Acc@1.

6.2.4 Discrete Prompts

We would like to investigate the performance of zero-shot using Pangu, and see whether we can leverage Pangu's large-scale knowledge and use it on our dataset.

We devised a few prompts, each follows the template of

[question]\n[choice]\n[answer]

as suggested in Pangu's original paper. "[question]" is a sentence that describes what kind of problem do we want Pangu to solve. In this field, we need to include the data sample (Chinese meme), but we can use different prompt to describe it, as shown in table 2. "[choice]" and "[answer]" are fixed for all template prompts. For each prompt template, we fill our input meme into the template and get the input for Pangu.

We run inference on Pangu and evaluate the output with metrics mentioned in section 6.2.3. The evaluation results are shown in table 2. In this table, we observed the some difference between each prompt, as shown in the following list. We also included their possible reasons.

- The first prompt is the worst. Pangu is more used to be instructed by a task name than a question.
- The third prompt is worse than the the second prompt. We expected that with a more specific word "danmu", we could give the model more hint. However, based on the result, it seems that Pangu does not recognize and fully understand this specific word.
- The last prompt is the best one. Follow the path of the thrid prompt, we replaced "danmu" with a general term, and it becomes better as we expected.

6.2.5 Prompt tuning

After some effort on discrete prompt designing, the accuracy is still pretty low, so we considered using prompt tuning and learn a continuous prompt. We expect this method yield to a better accuracy, because continouos prompt has the advantage of getting the best prompt embedding that is suitable for Pangu, which we cannot design using discrete prompt.

In order to achieve this, we apply a small set of parameters in the front of input embedding. These added parameters are specifically for the "[question]" part, since the "[choice]"

Prompt in Chinese	Prompt in English	Acc@1	Acc@3
[A] 属于哪个分区? 选项：动画，音乐舞蹈， 知识，生活，影视 答案:	To which category does [A] belong to? Choice: Animation, Music and Dance, Knowledge, Life, Movie Answer:	0.093	0.118
文本分类：[A] 选项：动画，音乐舞蹈， 知识，生活，影视 答案:	Text classification: [A] Choice: Animation, Music and Dance, Knowledge, Life, Movie Answer:	0.131	0.143
弹幕分类：[A] 选项：动画，音乐舞蹈， 知识，生活，影视 答案:	Danmu classification: [A] Choice: Animation, Music and Dance, Knowledge, Life, Movie Answer:	0.098	0.104
网络语言分类：[A] 选项：动画，音乐舞蹈， 知识，生活，影视 答案:	Internet meme classification [A] Choice: Animation, Music and Dance, Knowledge, Life, Movie Answer:	**0.145**	**0.158**

Table 2: Prompt template

and "[answer]" parts are fixed for all prompts. For model structure, we used 2560 hidden size, 32 attention heads, 0.1 dropout, batch size of 64, learning rate of 0.00005, and we set maxlen as 1024. We use the vocab file included in megatron. With these hyperparameters, we get a result of Acc@1 = 0.151 and Acc@3 = 0.156.

Although the accuracy for prompt learning is larger than the best one with discrete prompt, the gap is small and accuracy still unsatisfying. We think the reason might be Pangu does not have enough knowledge on such short meme sentences. If that is the case, no matter how good our prompt is, Pangu is still able to generate the right answer.

6.3 Fine tunning

We tried fine tuning several different Chinese pretrained models with our dataset, majorly our DanMu dataset. Most of the models we tested are from the model library of Hugging Face and UER.

6.3.1 ALBERT

Considering our limited memory and GPU resources, we turned to a new version of BERT model named ALBERT. An ALBERT model with much less parameters can achieve comparable or even better performance on different benchmarks compared to BERT. The significant changes of ALBERT is parameter sharing. BERT has 12 layers of encoder and each layer has its own parameters but ALBERT shares the parameters across all the layers.

Data Pre-Processing We preprocess the data by merging the subcategories to the main categories to reduce the number of classes. After merged, we have a total of 21 categories or

classes for this Muti-label classification task.

Implementation We implement the Albert model from Hugging Face. The pretrained model we use is "voidful/albert_chinese_base". A full implementation of model is in the link*. We train the model on our dataset for 40 epochs using AdamW optimizer with a learning rate of 5e-5. The train accuracy after 40 epochs achieves 95% but with a valuation accuracy 35%. The valuation accuracy would not increase after about 10 epochs. We think the model is overfitting because of the diversity of the categories. But the fine-turned model still learned from our dataset since a random guess of 21 classes would only have an accuracy less than 5%.

6.3.2 Chinese Bert

As our dataset is in Chinese, we decided to try out Chinese Bert to handle the classification task. We planned to fine-tune Chinese Bert with either tokenized dataset or the full dataset. We first planned to use the full dataset as it will give better results, however, due to compuational power restriction (It will take 37 days to run 10 epochs with a RTX 2080s), we switched to the tokenized dataset. After 5 epochs of training, the Chinese Bert was able to achieve 48% accuracy.

6.3.3 ByT5

The model we picked is from Hugging Face library: t5-base, which can be found in the link: link.This is one of the Pretrained model that we first tried but failed to learn from our data for this task. The GPU we used to train on this task is GTX 1080 with 8G memory.

Data Pre-Processing For this specific task, we divided the data into 21 major categories instead of the 126 sub categories to simplify the task. The script could be found in the link here: link.

Implementation We tried to build it from the process introduced in the link here: link. Then we spend much time on debugging the run_glue.py script from their github here: link. Eventually, we realized the script is not for encoder-decoder model like T5.

We then follow the script found in this link: link. We first tried the script with IMDB dataset which reached more than 70% accuracy.

However, when we tried to fine-tune the model with our own DanMu data, we soon found out the model couldn' t learn. Only 0.03% validation accuracy after one epoch, which spent us 19 hours to train. We then realized it' s not a valid option for our task.

6.3.4 Chinese Roberta

Our implementation of fine-tuning Chinese Roberta is from the open-source project UER (`https://github.com/dbiir/UER-py`). We used the finetune/run_classifier.py script to fine tuned the model with our DanMu dataset. Due to the limit of time and computing resources we had, we used the computing resource we used is g5.xlarge EC2 instance from AWS. The detailed implementation of running JupyterLab in EC2 instances we applied is from this link: (link). The model we are using is uer/chinese_roberta_L-12_H-768 from HuggingFace, which can be found in the link here: link.

Data Pre-Processing To use the script from UER correctly, we applied some pre-

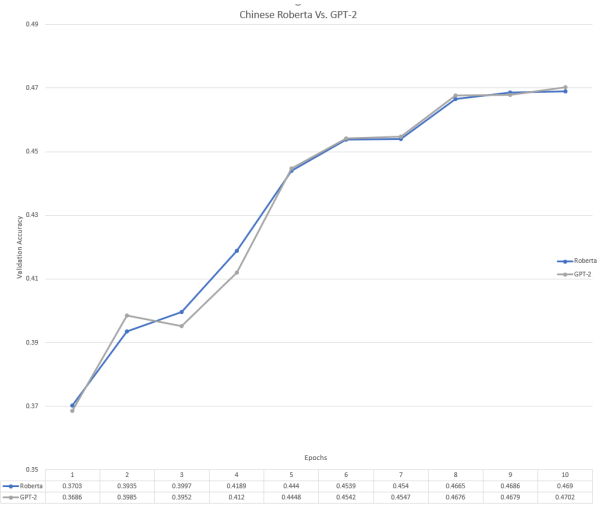
processing to the dataset we had. The detailed script could be found in the pre_process_6classes_UER.ipynb script in our github' s zhm_dev branch (link).

Specifically, we have the dataset first divided into 21 categories. (There are 27 major categories in total, but we didn' t get the data for the 6 categories of them) We then picked 5 major categories from them, which are ' 动画', ' 游戏', ' 科技', ' 生活' and ' 美食'. ('anime' , 'game' , 'technology' , 'lifestyle' , 'food') We also added a sixth category for the general appearing texts, that appeared in more than 3 major categories. Therefore, we had sixth categories in total for this task.

Implementation The implementation script that we used is test_UER.ipynb that could be found in the link: (link). The same script is also used for fine tunning for GPT2. The script could either be run in AWS EC2 instances with JupyterLab, or Google' s Colab. For AWS EC2, it requires installation of git-lfs function to be able to run. For this specific model, we fine-tuned it with 10 epochs with default parameters with our DanMu dataset, which gave us the result of 47.42% for the 6-category classification task, which is not very good. It went through 107985 steps but still couldn' t achieve more than 50% accuracy. We compared its performance with GPT-2, and then decided to use GPT-2 for further training.

6.3.5 GPT-2

We also fine tunned GPT-2 using the same library of Chinese Roberta, which is project UER (https://github.com/dbiir/UER-py). It' s also trained on g5.xlarge EC2 instance from AWS. Please refer to the previous section for detailed implementations. For



	1	2	3	4	5	6	7	8	9	10
Roberta	0.3703	0.3935	0.3997	0.4189	0.444	0.4539	0.454	0.4665	0.4686	0.469
GPT-2	0.3686	0.3985	0.3952	0.412	0.4448	0.4542	0.4547	0.4676	0.4679	0.4702

Figure 2: Comparing Chinese Roberta and GPT-2 in 10 Epochs

the limit of time and computing resources, the model we are using is a small GPT-2 distilled version pretrained on Chinese datasets. It can be downloaded from Hugging Face from the link here: link.

Data Pre-Processing For preprocessing the data for GPT-2 fine-tunning, we used the same approach from Chinese Roberta. The data is divided into six categories. The same pre_process_6classes_UER.ipynb script is used here which can be found from the link: link.

Implementation The script we are using for GPT-2 fine-tunning is the same test_UER.ipynb as the Chinese Roberta with changed parameters. The link to the script is here: be found from the link: link.ipynb. Same for it to run in EC2 instance of AWS, it requires installation of git-lfs function.

We first tried the model for 10 epochs, which it reaches the accuracy of 47.74%, which is very similar with Chinese Roberta. We then compared it performance with the Roberta, and decided to use it for further training of more epochs for its slightly higher accuracy.

At this point, we had to make a choice between them for further training due to the limit of time and computing resources.

We then further fine-tunned the model with 25 epochs. The detailed performance could be found in the Figure 2 below. Unfortunately, the accuracy couldn' t improve the accuracy further beyond 50% in 25 Epochs. We can obverse the slope starts flatten around 47 percent.

7 Error analysis

Many of our failed inputs are short general texts that are used very widely in different categories. This is especially obvious when we set the number of labels too high. When we were training with 126 labels, many of the inputs are shared among children' s categories with same parents in addition to the general texts that are shared among parent categories. In addition, due to the data pre-processing we did with Jieba module, we shorten the length of each input DanMu by first separate them into token words in Chinese. So theocratically, we did the word segmentation process twice. This could cause the model not having enough information to correctly learn and inference, which could potentially be why we couldn' t reach high accuracy.

Now we will investigate some randomly sampled failed input when inferencing using discrete prompt. Some examples are shown in table 3. Here we use the last prompt template in table 2 and we only include the bilibili meme in table 3. From this table, we can see the first two samplpes are labeled as Knowledge, while the prediction is Life. This indicates that Pangu has a problem differentiating Knowledge and Life. Especially when the second sample already contains the word "Knowledge". From the third and fourth samples, Animation is another category that we are having issues with. Since some comments about Animations are also valid for use in reality, the model cannot distinguish it. These hard categories might need more bilibili data to fine tune, because the input-target pair is not very common in Pangu's training dataset. For the last sample, the input meme is basically nonsense. We as human and bilibili users cannot interpret this, not to mention our model. For these kinds of input, we think it is better to filter them out from the dataset, in order to create confusion for the model.

8 Contributions of group members

- Hongyu Tu: Dataset collection, Data preprocessing, Baseline model implementation, Chinese-Bert Fine Tuning
- Zihao Mao: Fine Tunning ByT5, Chinese Roberta and GPT-2. Code implementation in AWS/sagemaker and EC2 instances.
- Chang Xue: For prompt-based learning, implemented data preprocessing and prompt tuning; conducted experiments; did error analysis
- Yichao Zhang: Fine Tunning GLM and ALBERT

9 Conclusion

For this project, one main takeaway that all group member agreed upon is that we should've keep the orignal sentences and use those to train our model. The word token we have gone with have low information density and when we are working with a large label set (126, even 21), the tokenzied words isn't representive enough for the model to learn to

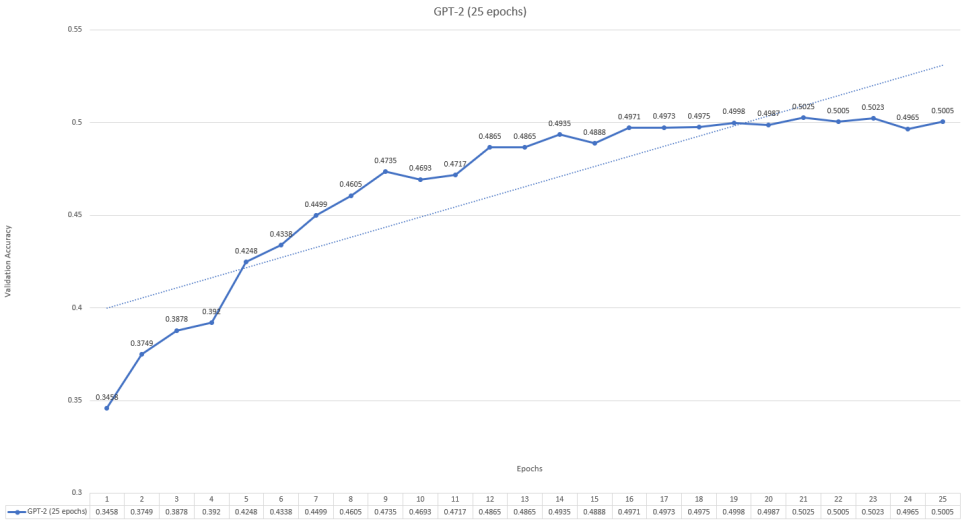


Figure 3: GPT-$2_2 5 epochs$

Meme in Chinese	Meme in English	Label	Prediction
设计好有意思	Desinging is so interesting	Knowledge	Life
知识就是力量	Knowledge is power	Knowledge	Life
人民的守护者	Peoples's guardian	Animation	Movie
确实美	Beautiful indeed	Animation	Life
啊吗啊吗	Ah ma Ah ma	Game	Life

Table 3: Failed examples of discrete prompt

differentiate them. We sort of found this problem when we are testing out the baselines but at the end of day we still went with the word dataset, and that is because the problem we faced with sentence dataset – compuational power.

Another one of the most important takeaways we had is to start small. We struggle so much at the beginning of the project to set up the environment for the huge fancy models that requires so much resources and time to train when we didn' t have our data purely sorted out. When we realized the huge model are not practical, it was relatively late. We had to rely on the computing resources we could found from online clouds to accelerates our progress.

One of the most difficult tasks is to work with AWS/EC2 and Sage maker to be able to use the cloud resources from Amazon. We found out late that their resources need to be applied for limit increase for instances like ml.g5.xlarge with good GPUs. In addition, it was very difficult to make the Sage maker work with Hugging face' s Transformers module. Although they had some of the functions built in, it' s hard to pass hyperparameters to the models due to the confusion of s3 paths. Especially when you like to pass the path of the downloaded model to the training script. It' s also hard to set up to EC2 instances to have the right path and permission to run Jupyter or JupyterLab for multiple users.

We aimed really high as our goals at the beginning was to learn not only to classify each meme, but give a distribution as well. How-

ever, now our model can at best be 50% accurate when giving out the category ids for only 6 cateogies. This improvement is to our surprise – both not as easy as we imagined, but still it's a lot better comparing to the baseline model which is almost same as a random guess (16%). Now with our hands on experience working on collecting a medium-scale dataset (over 900,000 sentences as well as their orign), we have appreciation toward the other famous models that is working really well and there's a lot more work we need to do. With that said, it's been a great experience and we definitely have learned a lot along the way.

References

Blackmore, S. (1999). *The Meme Machine*. Oxford University Press.

Bornheim, T., Grieger, N., and Bialonski, S. (2021). FHAC at germeval 2021: Identifying german toxic, engaging, and fact-claiming comments with ensemble learning. *CoRR*, abs/2109.03094.

Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

Ding, S., Shang, J., Wang, S., Sun, Y., Tian, H., Wu, H., and Wang, H. (2021). Ernie-doc: A retrospective long-document modeling transformer.

Du, Z., Qian, Y., Liu, X., Ding, M., Qiu, J., Yang, Z., and Tang, J. (2021). All NLP tasks are generation tasks: A general pretraining framework. *CoRR*, abs/2103.10360.

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., and Testuggine, D. (2021). The hateful memes challenge: Detecting hate speech in multimodal memes.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2014). *DBpedia - a large-scale multilingual knowledge base extracted from wikipedia*. Semantic Web Journal.

Mei, F. (2021). Bullet chats in china: Bilibili, language, and interaction.

Risch, J., Stoll, A., Wilms, L., and Wiegand, M. (2021). Overview of the GermEval 2021 shared task on the identification of toxic, engaging, and fact-claiming comments. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments*, pages 1–12. Association for Computational Linguistics.

Sun, C., Qiu, X., Xu, Y., and Huang, X. (2020). How to fine-tune bert for text classification?

Velioglu, R. and Rose, J. (2020). Detecting hate speech in memes using multimodal deep learning approaches: Prize-winning solution to hateful memes challenge.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2020). Xlnet: Generalized autoregressive pretraining for language understanding.

Zhang, X., Zhao, J., and LeCun, Y. (2016). Character-level convolutional networks for text classification.

Zhang, Z., Gu, Y., Han, X., Chen, S., Xiao, C., Sun, Z., Yao, Y., Qi, F., Guan, J., Ke, P., Cai, Y., Zeng, G., Tan, Z., Liu, Z., Huang, M., Han, W., Liu, Y., Zhu, X., and Sun, M. (2021). Cpm-2: Large-scale cost-efficient pre-trained language models.

Zhao, Z., Chen, H., Zhang, J., Zhao, X., Liu, T., Lu, W., Chen, X., Deng, H., Ju, Q., and Du, X. (2019a). UER: an open-source toolkit for pre-training models. *CoRR*, abs/1909.05658.

Zhao, Z., Chen, H., Zhang, J., Zhao, X., Liu, T., Lu, W., Chen, X., Deng, H., Ju, Q., and Du, X. (2019b). Uer: An open-source toolkit for pre-training models. *EMNLP-IJCNLP 2019*, page 241.