BERT-based and XLNET-based Empathetic Dialogues Multiclass Sentiment Analysis

Chih-Chao, Hsu 310554004
Hai-Yin, Huang 310581013
Pei-Hsiu, Hsu 310551163

1 Introduction

Nowadays, with most kinds of markets become more and more competitive, it is crucial that a company understand their customers' sentiments, which including what they are saying, what they mean and how they are conveying. Attempting to seize the true emotions behind the customers' reviews, tweets, or comments on social media platforms can help a company knows not only the opinions regarding their brands, but also what are the things that their users genuinely concern (Westerski, 2007). Apart from the ordinary reviews, the amounts of commentary in the form of dialogues have increased rapidly in recent years, thanked to how wide-spread and convenient the social media platforms are. Thus, sentiment classification in dialogues quickly becomes a popular and challenging task. Moreover, humans tend to have multiple emotions while expressing their thoughts and feelings. Therefore, choosing the correct sentiment behind utterances of dialogue among several candidate emotions is particularly complex and interesting (Firdaus et al., 2020).

In this paper, we would like to utilize the well-known BERT model (Devlin et al., 2018) and an improvised model called XLNet model (Yang et al., 2019) to conduct multiclass sentiment analysis on a public empathetic dialogue dataset (Rashkin et al., 2018). Besides from examining the performances of these 2 models, we would like to test the effectiveness of some data pre-processing trick, such as removing the punctuations and stop words, lemmatization, and data augmentation via back translation. Our results will be presented in the fashion of macro-F1 score trend on the testing dataset with several worth-mentioning milestone.

2 Related Work

In this section, we have reviewed some papers regarding some data pre-processing tricks and models to be used. Based on the outline of our research, we would like to split these parts into 3 little sections: **data pre-processing, BERT model,** and **XLNet model.**

2.1 Data pre-processing

It is a common practice to remove stop words and punctuations in lots of text classification related tasks. Many previous researchers generally removed the punctuation marks in their studies (Akba et al., 2014; Cetin et al., 2013; Demirtas et al., 2013; Kaya et al., 2012), as these punctuations often interfere the word tokenizing processes. However, as the results Kaya et al. (2012) experimented, sometimes the impacts of keeping punctuation marks as a textual feature in some languages such as Turkish are worth discussing.

As for removing the stop words, Ladani and Desai (2020) mentioned that the benefits of removing stop words include decrease in size of corpus, improvement of efficiency and accuracy of the text mining applications thus helping in reduction of time and space complexity of overall application. Other reasons suggested by Ladani and Desai are that stop words have low discrimination power, and they have no meaning, no predictive ability but high frequency of occurrence in the text, which generally make them noises when doing the text classification tasks.

We also considered applying lemmatization on the datasets, as there are researchers (Toman et al., 2006) stated that lemmatization in some cases can be beneficial, since it produces the basic word form which is required in many application areas.

Last but not least, Ma and Li (2020) have argued that when the amount of data is insufficient, or the distribution of samples is unbalanced, the accuracy of text classification will be greatly affected. Data augmentation is a necessary action in these scenarios. As for why back-translation might be a good data augmentation technique, it is because that back-translation can generate diverse paraphrases while preserving the semantics of the original sentences (Ma and Li, 2020).

2.2 BERT

A language representation model called BERT, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

2.3 XLNet

Relying on corrupting the input with masks, BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy. In light of these pros and cons, we apply XLNet(Yang et al., 2019), a generalized autoregressive pretraining method that (1) enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and (2) overcomes the limitation of BERT thanks to its autoregressive formulation. Furthermore, XLNet integrates ideas from Transformer-XL, the state-of-the-art autoregressive model, into pretraining. Empirically, under comparable experiment settings, XLNet outperforms BERT on 20 tasks, often by a large margin, including question answering, natural language inference, sentiment analysis, and document ranking. So, we apply XLNet in our final project according to its outperformance on sentiment analysis.

3 Method

Same as the order described in the previous "Related Works" section, based on the outline of our research, we would like to split these parts into 3 little sections: **data pre-processing, BERT model,** and **XLNet model.**

3.1 Data pre-processing

The empathetic dialogue datasets include a prompt and several utterances for each conversation. Besides those, there is a numeric label indicating the ground truth sentiment label for each dialogue. There are totally 32 candidate emotions in the datasets. Both training dataset and validation dataset, but not testing dataset, have these labels.

The first step of data pre-processing is to remove punctuation marks. We removed all period dots ".", "_comma_" symbols which are assumed to be comma originally, exclamation marks "!" and question marks "?". Upon checking the first few samples in the training dataset, we also found out that there exist some emojis, such as ":(". So, we cleaned those, too. After that, we got rid of the leading spaces and trailing spaces, as well as the new line character "\n" at the end of each sentences. Finally, since we would like to try the uncased pretrained BERT model later on, we lowered cases of all English letters.

The next step is to delete stop words and to execute lemmatization on all the remaining words. These 2 pre-processing are done by the aid of NLTK package (Bird et al., 2009), which is a prominent natural language toolkit. According to the official documentation, there are 40 stop words in the stop words list. We filtered them out. After that, we did the lemmatization for each word remaining, based on their part-of-speech tagging, respectively. Lemmatization removes the suffix of a word completely to get the basic word form or replaces the suffix of a word with a different one. Since in NLTK package, the WordNetLemmatizer uses some built-in function from the WordNet (Fellbaum, 1998), a large and publicly available lexical database in English, this lemmatization step would only be executed in "advanced" task. Furthermore, note that we basically conducted these NLTK pre-processing on utterances, not on prompt. We would, however, also try these 2 pre-processing techniques on prompts later in our implementation as well, for showing how prompts play an important role in empathetic classification.

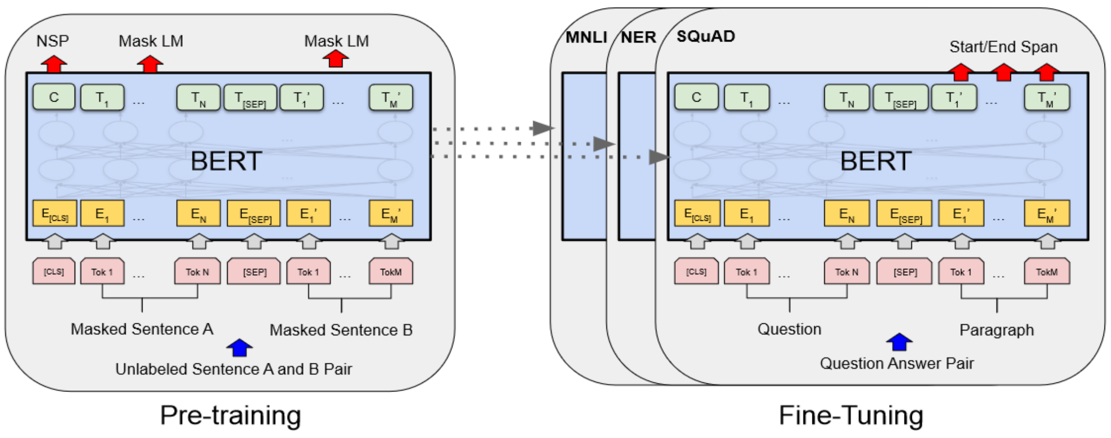

Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

All these pre-processing tricks are wrapped in a function. We called the function to clean the datasets before loading them in data loaders.

We have also tried data augmentation in the "advanced" task, by back-translation. About the implementation details, we call the Google translate API (Han, 2015) to translate prompts. In our practice, we randomly choose 5 samples from each of the 32 emotion labels. And then, randomly translate them to 1 of the 10 target languages we chose, and then translate them back. The 10 target languages are Chinese (both simplified and traditional), Japanese, Korean, French, Germen, Spanish, Indonesian, Arabic and Russian. The reason we chose these 10 languages is that they are high-resource languages with many users.

Note that we only tried to back-translate prompts, not utterances in this case. This is owing to that utterances have lost grammatical structure after deleting stop words and doing lemmatization. It is inappropriate to back-translate these obviously broken sentences.

3.2 BERT

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section. A distinctive feature of BERT is its unified architecture across different tasks. There is mini mal difference between the pre-trained architecture and the final downstream architecture.

(1) Model Architecture: BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described by Vaswani et al. (2017) and released in the `tensor2tensor` library. Because the use of Transformers has become common and our implementation is almost identical to the original, we will omit an exhaustive background description of the model architecture and refer readers to Vaswani et al. (2017) as well as excellent guides such as “The Annotated Transformer.” In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. We primarily report results on two model sizes: **BERT$_{\text{BASE}}$** (L=12, H=768, A=12, Total Parameters=110M) and **BERT$_{\text{LARGE}}$** (L=24, H=1024, A=16, Total Parameters=340M). BERT$_{\text{BASE}}$ was chosen to have the same model size as OpenAI GPT for comparison purposes. Critically, however, the BERT Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.

(2) Input/Output Representations: To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., h Question, Answering) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. We use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embedding to every token indicating whether it belongs to sentence A or sentence B. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special [CLS] token as $C \in R^H$, and the final hidden vector for the ith input token as $T_i \in R^H$. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings.

3.3 XLNet

In this section, we review and compare the conventional AR language modeling and BERT for language pretraining. Given a text sequence $\mathbf{x} = (\chi_1, \cdots, \chi_T)$, AR language modeling performs pretraining by maximizing the likelihood under the forward autoregressive factorization:

$$\max_{\theta} \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t})$$

$$= \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^T e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^T e(x')\right)} \qquad (a)$$

Where $h_\theta(\mathbf{x}_{1:t-1})$ is a context representation produced by neural models, such as RNNs or Transformers, and $e(x_t)$ denotes the embedding of x. In comparison, BERT is based on denoising auto-encoding. Specifically, for a text sequence $\mathbf{x}$, BERT first constructs a corrupted version $\hat{\mathbf{x}}$ by randomly setting a portion (e.g. 15%) of tokens in $\mathbf{x}$ to a special symbol [MASK]. Let the masked tokens be $\bar{\mathbf{x}}$. The training objective is to reconstruct $\bar{\mathbf{x}}$ from $\hat{\mathbf{x}}$:

$$\max_{\theta} \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}})$$

$$= \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t{}^T e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t{}^T e(x')\right)} \qquad (b)$$

Where $m_t = 1$ indicates x_t is masked, and H_θ is a Transformer that maps a length-T text sequence $\mathbf{x}$ into a sequence of hidden vectors $H_\theta(\mathbf{x}) = [H_\theta(\mathbf{x})_1, H_\theta(\mathbf{x})_2, \ldots, H_\theta(\mathbf{x})_T]$. The pros and cons of the two pretraining objective are compared in the following aspects:

(1) **Independence Assumption**: As emphasized by the ≈ sign in Eq. (b), BERT factorizes the joint conditional probability $p(\bar{\mathbf{x}} \mid \hat{\mathbf{x}})$ based on an independence assumption that all masked tokens $\bar{\mathbf{x}}$ are separately reconstructed. In comparison, the AR language modeling objective (a) factorized $p_\theta(\mathbf{x})$ using the product rule that holds universally without such an independence assumption.

(2) **Input noise**: The input to BERT contains artificial symbols like [MASK] that never occur in downstream tasks, which create a pretrain-finetune discrepancy. Replacing [MASK] with original tokens as in does not solve the problem because original tokens can be only used with a small probability — otherwise Eq. (b) will be trivial to optimize. In comparison, AR language modeling does not rely on any input corruption and does not suffer from this issue.

(3) **Context dependency**: The AR representation $h_\theta(\mathbf{x}_{1:t-1})$ is only conditioned on the tokens up to position t (i.e. tokens to the left), while the BERT representation $H_\theta(\mathbf{x})_t$ has access to the contextual information on both sides. As a result, the BERT objective allows the model to be pretrained to better capture bidirectional context.

To bring the advantages of both BERT and AR language modeling while avoiding their weaknesses, a permutation language modeling objective is applied in XLNet. Specifically, for a sequence $\mathbf{x}$ of length T, there are $T!$ different orders to perform a valid autoregressive factorization. Intuitively, if model parameters are shared across all factorization orders, in expectation, the model will learn to gather information from all positions on both sides.

To formalize the idea, let Z_T be the set of all possible permutations of the length-T index sequence [1, 2, …, T]. We use z_t and $\mathbf{z}_{<t}$ to denote the t-th element and the first t-1 elements of a permutation $\mathbf{z} \in Z_T$. Then, the proposed permutation language modeling objective can be expressed as follows

$$\max_{\theta} \; E_{\mathbf{z} \sim Z_T}\left[\sum_{t=1}^{T} \log p_{\theta}\left(x_{z_t} \middle| \mathbf{x}_{\mathbf{z}_{<t}}\right)\right]. \qquad (c)$$

As this objective fit into the AR framework, it naturally avoids the independence assumption and the pretrain-finetune discrepancy discussed above.

4 Experiment

We divided our experiment into 3 durations. We use “utterance only” as our input data in duration 1. Then, we add prompt into our input data in duration 2. Last, we changed our model from BERT to XLNET in duration 3.

4.1 Duration 1

At first, we designed an initial model. The input data we used in initial model is “utterance only” and trained with 4-layer BERT.

Then, we tried some operations in this duration, like fine-tuning hyperparameters, adding token type and deleting stop words, reducing nn layer from 4 to 1. We found that the first and second operations will get better score on Kaggle, and reducing nn layer to 1 still remains the score at the same level, so we kept these adjustments in duration 2 and 3.

We also found something special in duration 1, when we trained BERT with 10 epochs will cause overfitting. We guessed that since BERT is a pretrained model, maybe it’s not suitable for such long epoch.

4.2 Duration 2

In duration 2, we changed our input data from “utterance only” to “prompt only”, we found that the result it’s pretty good. Then, we changed input data again, from “prompt only” to “prompt and utterance”, and remained all the adjustments in duration 1, got a much better score.

We also did some modifications in duration 2, like deleting some punctuations, deleting the dialogue of person B, deleting stop words of prompt.

We guessed maybe the dialogue of person B is a noise of the whole dialogue relatively, but deleting the dialogue of person B didn’t make noticeable change, so we kept the dialogue of person B. And deleting stop words of prompt makes the score even worser, so we remained the original prompt.

4.3 Duration 3

In duration 3, we attempted to increase the score. We changed the model from “BERT” to “XLNET”, and remained the same input data and nn layer.

We tried the same operation with duration 2, deleting the stop word of prompt, and found out it still make the score worser, so we will remain the original prompt.

Then, we fine-tuned the hyperparameters, with max_length=400, batch_size=20, epochs=4, learning_rate=0.00003, random_seed=1337, got the best score 0.63428 in this competition.

4.4 Advanced

We considered that not making any adjustment on prompt is the best choice, so we only did lemmatization on utterance, but the result is not better than duration 3.

Then, we did data augmentation with back translation on prompt, the result also is not better than duration 3.

Milestone	score
Utterance only with BERT	0.53312
Prompt only with BERT	0.57413
Prompt and utterance with BERT	0.61040
Prompt and utterance with XLNET	0.61587
Best fine-tune with XLNET	0.63428

Table 1: Each experiment milestone and score

5 Conclusion

As sentiment classification in dialogues quickly becomes a popular and challenging task. In this paper, we utilize the well-known BERT model (Devlin et al., 2018) and an improvised model, XLNet (Yang et al., 2019) to conduct multiclass sentiment analysis on a public empathetic dialogue dataset (Rashkin et al., 2018). In our experiment, we found that “Prompt” is quite important in the dataset, and retaining it as origin perform better. Further, with the data preprocessing tricks of removing the punctuations and stop words seems slightly improve the training. Eventually, due to a permutation language modeling objective is applied in XLNet, bringing the advantages of both BERT and AR language modeling while avoiding their weakness, it performs better in our experiment with applying XLNet than with BERT. However, it should be noticed that the training epochs should not be too large for the classifying NN layers, due to XLNet and BERT are both pretrained, the large training epoch may lead to overfitting problems.

References

Adam Westerski. 2007. *Sentiment Analysis: Introduction and the State of the Art overview.* Universidad Politecnica de Madrid, Spain westerski@dit.upm.es

Mauajama Firdaus∗ , Hardik Chauhan, Asif Ekbal and Pushpak Bhattacharyya, 2020. *MEISD: A Multimodal Multi-Label Emotion, Intensity and Sentiment Dialogue Dataset for Emotion Recognition and Sentiment Analysis in Conversations.* Department of Computer Science and Engineering Indian Institute of Technology Patna, India (mauajama.pcs16,hardik,asif,pb)@iitp.ac.in

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova Google AI Language {jacobdevlin,mingweichang,kentonl,kristout}@google.com

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le, 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding.* Carnegie Mellon University, Google AI Brain Team {zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

Hannah Rashkin, Eric Michael Smith, Margaret Li, Y-Lan Boureau, 2018. *Towards Empathetic Open-domain Conversation Models: a New Benchmark and Dataset.* Paul G. Allen School of Computer Science & Engineering, University of Washington Facebook AI Research hrashkin@cs.washington.edu, {ems,margaretli,ylan}@fb.com

Fırat Akba, Alaettin Uçan, Ebru Akcapinar Sezer and Hayri Sever, 2014. *ASSESSMENT OF FEATURE SELECTION METRICS FOR SENTIMENT ANALYSES: TURKISH MOVIE REVIEWS.* Hacettepe University, Computer Engineering Department, Ankara, Turkey

Mahmut Cetin and M. Fatih Amasyah, 2013. *Egiticili ve Geleneksel Terim Aglrllklandlrma Yöntemleriyle Duygu Analizi Supervised and Traditiünal Term Weighting Methüds für Sentiment Analysis.* Bilgisayar Mühendisligi Bölümü YIldlZ Teknik Üniversitesi istanbul, Türkiye

Erkin Demirtas and Mykola Pechenizkiy, 2013. *Cross-lingual Polarity Detection with Machine Translation.* Department of Computer Science Eindhoven University of Technology the Netherlands

Mesut KAYA, Güven FİDAN, Ismail H. Torosl, 2012. *Sentiment Analysis of Turkish Political News.* Department of Computer Engineering Middle East Technical University, Ankara, Turkey

Dhara J. Ladani and Nikita P. Desai, 2020. *Stopword Identification and Removal Techniques on TC and IR applications: A Survey.* M.Tech (IT) Student Department of Information Technology Dharmsinh Desai Unive

Michal Tomana , Roman Tesara and Karel Jezek, 2006. *Influence of Word Normalization on Text Classification.* University of West Bohemia, Faculty of Applied Sciences, Plzen, Czech Republic.

Jun Ma, Langlang Li, 2020. *Data Augmentation For Chinese Text Classification Using Back-Translation.* School of Information Science & Engineering, Lanzhou University, Lanzhou, Gansu Province, China. junma@lzu.edu.cn, lill19@lzu.edu.cn

Matthew E. Peters , Mark Neumann , Mohit Iyyer , Matt Gardner, 2018. *Deep contextualized word representations.* Allen Institute for Artificial Intelligence Paul G. Allen School of Computer Science & Engineering, University of Washington

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, 2018. *Language Models are Unsupervised Multitask Learners.* Equal contribution 1OpenAI, San Francisco, California, United States. Correspondence to: Alec Radford .

Steven Bird, Ewan Klein, and Edward Loper, 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'REILLY.

Christiane Fellbaum, 1998. *WordNet An Electronic Lexical Database.*

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez and Łukasz Kaiser, 2017. *Attention Is All You Need.* Google Brain.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, 2016. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean.

A Work Division

Chih-Chao, Hsu: team leader, main coding, fine tuning, QA, report preparation
Hai-Yin, Huang: auxiliary coding, fine-tuning, project presentation, report preparation
Pei-Hsiu, Hsu: auxiliary coding, fine-tuning, slides preparation, report preparation

B Question and Answer

Question 1 (Professor Gu): In terms of model architecture, why do you think the XLNet would have better performance than BERT did?
Answer 1: Although similar to BERT model, XLNet does outperform BERT in roughly 20 NLP tasks, and sometimes with quite substantial margins (Jain, 2020). Also, as Adoma et al. (2020) pointed out in their study, XLNet does have better performance than BERT does in text-based emotion recognition, which is similar to our task. The major differences between these 2 models come in the approaches to pre-training.

Basically, BERT is an autoencoding based model, while XLNet is an autoregressive based model. This difference materializes in the masked language model (MLM) tasks, where randomly masked language tokens are to be predicted by the model. By the help of a technique called permutation language modeling (PLM), XLNet can learn more contextual information without assuming the independence between masked tokens as the BERT model does. Thus, XLNet is able to capture the dependency between the masked tokens.

Question 2 (student 0810749): Why do you think deleting punctuation marks would result in better performance? Is there any intuitional explanation?
Answer 2: Because we would tokenize the sentences before feeding them to the models, we would like to remove these punctuation marks. We found out that no matter the BERT tokenizer or the XLNet tokenizer, these tokenizers deem these punctuations as tokens as well. These punctuation tokens are just meaningless noises when performing contextual recognitions. The experimental results indeed indicated better performance when removing punctuations.

Question 3 (student 0716050): Why do you set the layer number of your downstream classifier as 4 layers initially? And why do you reduce to 1 layer later on?
Answer 3: Actually, the layer number of a classifier should be considered relative to the difficulty of the classification tasks. As a multiclass classification problem with up to 32 kinds of labels, we initially set up 4 layers as suggested in a paper with similar classification problem (Sahoo et al., 2020). However, later on we found out that as simple as it should be, only 1 layer of downstream classifier is enough for generating good classification results, referred to the study conducted by original BERT team (Devlin et al., 2018) regarding sentence pair classification. We've tried using only 1 layer, and found out that the performance holds roughly the same. Thus, for simplicity principle, we kept this setting in our later experiments.