Fine-Tuning BERT For Monolingual Intent Classification

Ali Haidar[*], **François Bertholom**[†]

March 17, 2023

Abstract

Intent classification is a crucial task in Natural Language Understanding, with numerous applications in chatbots, virtual assistants, and other conversational Artificial Intelligence (AI) systems. Recently, deep learning models, particularly pre-trained language models such as BERT, have achieved state-of-the-art results in various Natural Language Processing tasks, including intent classification. In this study, we explore the effectiveness of fine-tuning BERT using three different architectures for both single- and multi-target intent classification tasks: BertMLPLayer1, BertMLPLayer2, and BertGRU. We conduct experiments on the SILICONE datasets and achieve excellent results on single-target intent classification, with BertGRU outperforming the other two methods and previous benchmarks on the same datasets. However, our experiments on multi-target intent classification tasks did not yield satisfactory results.

1. Introduction

Natural Language Processing (NLP) is a field that aims to enable machines to understand and interpret human language, and has received considerable attention in recent years thanks to rapid advances and spectacular results. One of the most important tasks in NLP is intent classification, which involves identifying the intent behind a user's input. It is a problem of Natural Language Understanding (NLU), and constitutes a crucial step in many NLP applications such as virtual assistants and customer support systems. With the increasing demand for personalized and efficient customer experiences, accurate intent classification has become more important than ever.

In recent years, pre-trained language models such as Google's Bidirectional Encoder Representations from Transformers (BERT, [Devlin et al., 2018]) have shown great success in various NLP tasks. Specifically, BERT was built and trained with the intent to be fine-tuned on task-specific datasets by simply adding a single output layer. While it has proven to be a high-performance language representation model, it is still unclear which fine-tuning techniques perform and generalize best.

In this paper, we aim to compare and evaluate various fine-tuning techniques for monolingual intention classification using a BERT backbone. In particular, we will investigate the usefulness of adding certain modules on top of BERT. We will analyze the results to determine which fine-tuning technique performs best for monolingual intent classification, and has the highest generalization ability across different datasets.

The article is organized as follows. Section 2 provides a background on intention classification and BERT. Section 3 describes the various fine-tuning methods used in this study. Section 4 presents the experimental setup and datasets used. Section 5 reports the results and analysis of the performance of the different fine-tuning techniques. Finally, Section 6 concludes the paper and provides insights for future research. The code used for experiments is available at https://github.com/AliHaidar97/NLP-ENSAE-Project.

[*]ali.haidar@polytechnique.edu
[†]francois.bertholom@ensae.fr

2. Background

2.1. Problem statement

The monolingual intent classification problem boilds down to multi-label one target classifier. Our goal is to evaluate and compare the effectiveness of various deep learning models, taking an utterance as input and predicting a label.

We denote $\boldsymbol{x} = (x_1, \ldots, x_T)$ an input sentence of length T. For single-target classification, each sentence is labeled by some $y \in \mathbb{N}$. For multi-target classification, it is labeled by $\boldsymbol{y} = (y_1, \ldots, y_m)$, with m the number of correct classes. The total number of classes appearing in the dataset is denoted K. Our goal is to retrieve the correct $\boldsymbol{y}$ from a given input $\boldsymbol{x}$.

2.2. Pretraining

[Radford et al., 2018b] show that when a Large Language Model (LLM) is trained on a sufficiently large and diverse dataset, it is able to generalize to many domains and datasets. Their experiments suggest that training high capacity models to learn representations of a sufficiently diverse text corpus grants them the ability to perform suprisingly well on various NLP tasks.

[Radford et al., 2018a] further show that generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning for a given task yields significantly better results than training a task-specific model from scratch. Precisely, learning complex language representation enables the creation of more flexible and performing models, with a higher generalization potential, and opens the door to more complex tasks like few-shot or even zero-shot (intent) classification.

2.3. BERT

BERT's architecture [Devlin et al., 2018] is based on a multi-layer bidirectional Transformer encoder, similar to the vanilla Transformer model [Vaswani et al., 2017] which has proven to be particularly effective at modeling sequential data and learning long-range dependancies. The input is represented using a concatenation of WordPiece embeddings [Wu et al., 2016], positional embeddings - to take temporal ordering into account -, and segment embedding. The latter being used to distinguish sentences, it does not discriminate single sentences. A special classification embedding ([CLS]) is added as the first token and a special token ([SEP]) marks the end of the input sentence. Given an input token sequence $\boldsymbol{x} = (x_1, ..., x_T)$ of length T, we denote BERT's output $\boldsymbol{h} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_T)$.

The pretrained BERT model is a powerful tool to produce context-based sentence representation and can be fine-tuned for various tasks, including intention classification.

3. Methodology

We build three models based on BERT [Devlin et al., 2018]. In this section we describe the models' architectures and explain design choices. We also give more detail on the loss functions and performance assessment methods.

The most simple way to use BERT's powerful embedding is to simply add a dense layer on top of the pretrained network. The intent is then predicted as:

$$\tilde{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}\boldsymbol{h}_1 + \boldsymbol{b})$$

We use more sophisticated approaches to make a better use of BERT's outputs.

3.1. BertMLP1Layer

This first model uses the embedding layer of a pre-trained BERT model. We then add a chaining layer of GlobalMaxPooling1D and GlobalAveragePooling1D after the embedding. The resulting output was then passed through a single layer of the neural network.

GlobalMaxPooling1D and GlobalAveragePooling1D are commonly used pooling operations in deep learning models, especially for natural language processing (NLP) tasks.

In many NLP problems, the inputs are sequences of varying lengths. The pooling operations allow us to combine the information from these sequences into a fixed-length vector that can be passed to subsequent layers of the neural network. GlobalMaxPooling1D computes the maximum value from each feature dimension over the entire input sequence. This can be useful to capture the most salient information contained in the input sequence. GlobalAveragePooling1D, on the other hand, calculates the average value from each feature dimension across the entire input sequence. This can be useful to capture the overall distribution of information in the input sequence.

By using both GlobalMaxPooling1D and GlobalAveragePooling1D in a concatenated layer, we can capture both the most salient information and the overall distribution of information in the input sequence, which leads to a more ro-

bust representation of the input and can lead to improved performance.

The model uses sparse categorical cross-entropy as a loss function in the training phase. It is commonly used for multi-class classification problems with integer labels, i.e. each class is assigned to a specific integer value. We must treat differently the single- and multi-class cases. In single-target classification, if we have n examples we denote $\boldsymbol{y}^* = (y_1^*, \dots, y_n^*)$ the true labels and $\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{y}}_1, \dots, \tilde{\boldsymbol{y}}_n)$ the model's output. The loss function is then defined as:

$$\ell_H^1(\boldsymbol{y}^*, \tilde{\boldsymbol{y}}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_i^* \log(\tilde{y}_{ik})$$

For multi-target classification, since examples may not have the same number of correct classes, we need to use masks. Specifically, we denote $\boldsymbol{y}^* = (\boldsymbol{y}_1^*, \dots, \boldsymbol{y}_n^*) \in \mathbb{N}^{n\times m}$ the true labels and $\tilde{\boldsymbol{y}} = (\tilde{\boldsymbol{y}}_1, \dots, \tilde{\boldsymbol{y}}_n) \in \mathbb{R}^{n\times m\times K}$ the model's output. To deal with the differing number of classes, we use a mask $M \in \{-1, 1\}^{n\times m}$. Sparse categorical cross-entropy is defined as:

$$\ell_H^m(\boldsymbol{y}^*, \tilde{\boldsymbol{y}}) = -\frac{1}{S}\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{K} y_{ij}^* \log(\tilde{y}_{ijk}) M_{ij}$$

where $S = \sum_{i=1}^{n}\sum_{j=1}^{m} M_{ij}$.

We evaluate the performance of the model using the $0-1$ accuracy metric. This metric measures the percentage of times the model correctly predicts the label for the input utterance. In other words, it calculates the ratio between the number of correctly predicted labels and the total number of labels and then expresses this as a percentage. We use it for both single-target and multi-target classification. In the single-target case, the accuracy is given by:

$$\mathrm{Acc}(\boldsymbol{y}^*, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i^* = \arg\max_{1\le k\le K} \tilde{y}_{ik}\}$$

In the multi-target case, we apply the following formula:

$$\mathrm{Acc}(\boldsymbol{y}^*, \tilde{\boldsymbol{y}}) = \frac{1}{n\times K}\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbb{1}_{Y_i^*}\left(\arg\max_{1\le k\le K} \tilde{y}_{ijk}\right)$$

where $Y_i^* = \{y_{ij'}^*, j' \in [\![1, m]\!]\}$.

3.2. BertMLP2Layer

The main difference between this model and BertMLP1Layer is that we added an additional dense layer before the output layer.

[1] https://huggingface.co/datasets/silicone.

3.3. BertGRU

We choose to use Gated Recurrent Units (GRU) over other common types of recurrent neural networks (e.g. Long-Short Term Memory) for it is a reasonable compromise between model size and performance [Chung et al., 2014]. The output of BERT's embedding is passed through two bidirectional GRUs. We then use GlobalMaxPooling1D and GlobalAveragePooling1D like in the previous models. The outputs are concatenated on the last dimension and passed through a dense layer and a softmax. The architecture is summarized in Appendix A.

4. Experimental setting

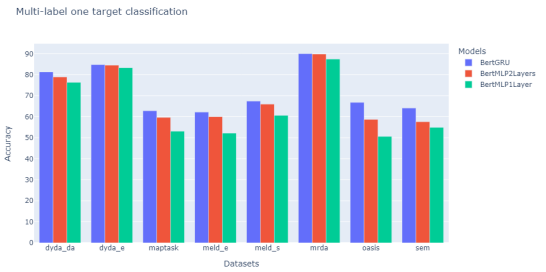
To assess the performance of these models, we used the SILICONE datasets [Chapuis et al., 2020] provided by Hugging Face[1]. The datasets are in the English language and contain between 4264 and 190709 training examples.

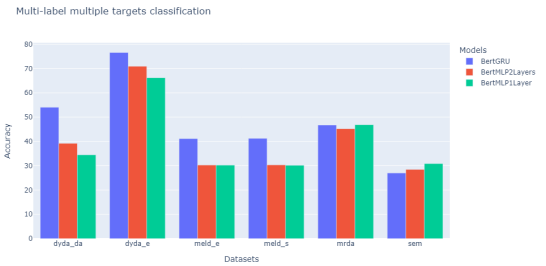
Each model is trained for 100 epochs with a batch size of 32. We use the Adam optimizer [Kingma and Ba, 2014] with a starting learning rate of 5×10^{-5}. The learning rate has a polynomial decay such that it is null after 100 epochs. We also clip the gradient norm to 1 to avoid large swings and perturbations.

5. Results

The results we obtain for single-target and multi-target classification are summarized in the figures below. For single-target classification, we get excellent results for all three methods, with BertGRU consistently outperforming BertMLP2Layers, which itself outperforms BertMLP1Layer. For multi-target classification, though, we obtain unsatisfactory results across all datasets, except maybe for `dyda_e`. Still, BertGRU seems to be better-suited to the task, either outperforming or being on a tie with the other methods.

The full results tables are given in Appendix B.





6. Conclusion

In summary, this study has shown that fine-tuning BERT for intent classification is a promising approach for single-target classification. The results obtained using BertGRU show that the inclusion of bidirectional GRU modules on top of BERT can improve the performance compared to using more elemntary techniques. However, the study also highlights the difficulty of accurately classifying multiple intentions with BERT, suggesting that further research is needed to address this challenge. Multi-intent classification is a difficult task for the model needs to capture subtle nuances in the input text.

Future studies could explore different architectures to tackle this problem, such as hierarchical or ensemble models, which may be more effective than the explored methods. Additionally, incorporating external knowledge sources or domain-specific features could help improve the accuracy of intent classifiers. Finally, examining the influence of preprocessing techniques and the amount and quality of training data could provide valuable insights to improve the performance of intent classification models. This study provides a starting point for further research in the area of NLU, which has a variety of applications in several fields.

References

[Chapuis et al., 2020] Chapuis, E., Colombo, P., Manica, M., Labeau, M., and Clavel, C. (2020). Hierarchical pre-training for sequence labelling in spoken dialog.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

[Radford et al., 2018a] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018a). Improving language understanding by generative pre-training.

[Radford et al., 2018b] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018b). Language models are unsupervised multitask learners.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

A. BertGRU architecture

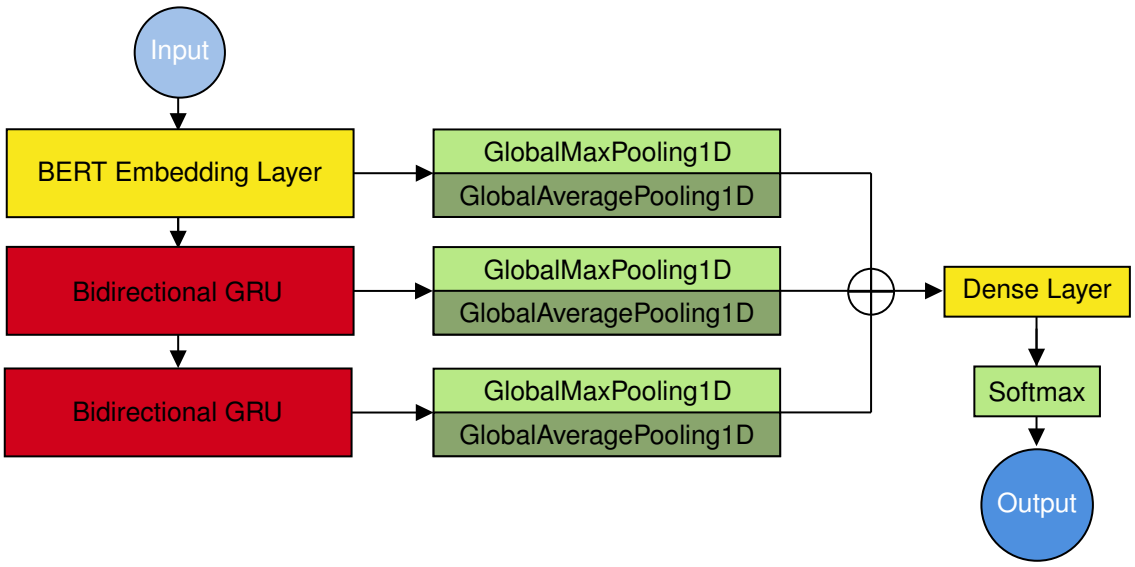


Figure 1: Bidirectional GRU fine-tuning on BERT ($\bigoplus$ denotes concatenation on dimension -1).

B. Full results tables

Table 1: Results table for multi-label single-target classification.

Model	dyda_da	dyda_e	maptask	meld_e	meld_s	mrda	oasis	sem
GRU	81.32	84.78	62.85	62.18	67.43	90.02	66.78	64.12
MLP2Layers	78.91	84.53	59.64	60.00	65.98	89.83	58.73	57.63
MLP1Layer	76.33	83.32	53.11	52.18	60.65	87.43	50.61	54.90

Table 2: Results table for multi-label multi-target classification.

Model	dyda_da	dyda_e	meld_e	meld_s	mrda	sem
GRU	54.03	76.59	41.15	41.23	46.73	26.99
MLP2Layers	39.20	70.92	30.27	30.34	45.22	28.47
MLP1Layer	34.46	66.21	30.23	30.15	46.86	30.87