TRADE Domain State Tracking with BERT Encoder

Joshua Lemmon
University of Waterloo
jmlemmon@uwaterloo.ca

Abstract

Dialogue state tracking is a difficult problem that has become too complex for the traditional solution of domain ontologies and classification models. This is due to the lack of knowledge transferring across domains causing issues when encountering unknown domains. (Wu et al., 2019) have proposed the TRADE architecture, which achieves state-of-the-art performance on the multi-domain MultiWOZ dataset through the use of various knowledge transferring techniques. The TRADE model utilizes the relatively outdated GRU model for its encoding step. I explore the feasibility of replacing the GRU encoder with the more recent BERT encoder, which has been widespread in the NLP field since its inception. Due to some incompatibilities with the pre-trained BERT model size and the TRADE architecture, I use a shallow BERT model trained in parallel with the TRADE model. The original GRU encoder outperforms this shallow BERT model, however results imply that BERT is better at preventing catastrophic memory loss when performing transfer learning on an unseen domain. This implies that if TRADE was overhauled to be compatible with the size of pre-trained BERT, then BERT would likely outperform the GRU.

1 Introduction

Task-oriented dialogue systems, such as Siri or Alexa, have become commonplace in modern society. These are systems that must be able to understand what a user is saying, and then execute commands based off that understanding. These commands cover a wide range of domains, from ordering a taxi to reserving tickets to a movie. As such, these dialogue systems must understand a variety of domains that may or may not be related. An example of a potential dialogue a system must interpret can be seen in figure 1.

The system must use previously seen state-

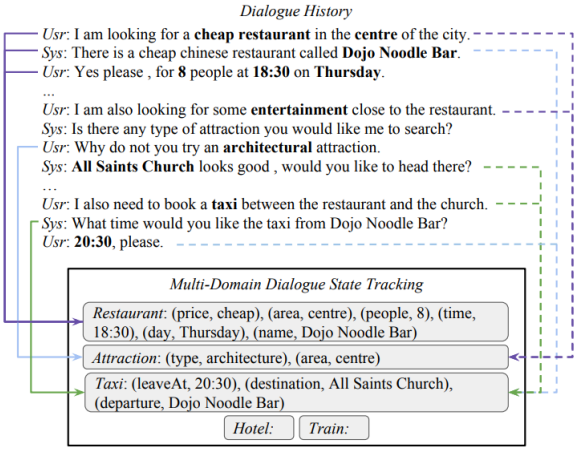

Figure 1: An example of a dialogue history from (Wu et al., 2019). Shows the various domain slots that must be tracked across multiple conversation statements.

ments from the user as context for future statements in order to properly discern the intention of the user. In the past, dialogue state tracking (DST) techniques relied on the use of a predefined ontology in order to reduce the state tracking problem into a classification problem (Henderson et al., 2014; Mrkšić et al., 2017; Zhong et al., 2018). As the use cases of task-oriented dialogue systems increase, they must encapsulate an increasing number of domains which is difficult to sustain in the long term with the traditional methods. This is because defining a full ontology is difficult (Xu and Hu, 2018) and a full ontology might contain too many possible values to be effective in real world scenarios (Wu et al., 2019).

In order to help facilitate the training of DST models, the Multi-domain Wizard-of-Oz, or MultiWOZ, (Budzianowski et al., 2018) data set was created, which contains human-to-human conversations that span multiple domains. Using this MultiWOZ data set, (Wu et al., 2019) created the Transferable Multi-Domain State Generator (TRADE) architecture. TRADE utilizes knowledge transferring techniques to share domain knowledge, and achieves state-of-the-art performance on multi-domain conversations. In this project, I replace the GRU (Chung et al., 2014) encoder used in the original TRADE model with a BERT (Devlin et al., 2018) encoder and evaluate the new TRADE model on the MultiWOZ data set.

2 TRADE

The TRADE model, seen in figure 2, is composed of three main components:

- The Utterance Encoder
- The State Generator
- The Slot Gate

The model is trained end-to-end, with each component being trained in parallel.

2.1 Utterance Encoder

The Utterance Encoder accepts a dialogue history (i.e. the concatenation of all words found in the dialogue) as input. Each word of the history is initialized using a concatenation of its Glove embedding (Pennington et al., 2014) and character embedding (Hashimoto et al., 2017). The Utterance Encoder encodes each of the input words into a fixed length vector, and also encodes the entire history by concatenating each of its hidden states. TRADE uses a GRU model as its encoding model.

2.2 The State Generator

The state generator accepts the encoded output from the Utterance Encoder as well as a set of domain-slot pairs as input. The state generator then uses a decoder model, a GRU, to predict the correct domain and slot value for each of the inputted pairs. It does this by taking the word embedding from the Utterance Encoder and creating a hidden state for that embedding. A vocabulary distribution is constructed using the hidden state of the decoder and the dialogue history. This distribution is then weighted using a trainable weight and the correct slot value is predicted by choosing the most likely word from the weighted distribution.

2.3 Slot Gate

The Slot Gate uses the context given by the dialogue history in order to predict the importance of the word generated by the State Generator. It is a simple classifier with three output classes: *ptr, none* and *dontcare*. If the classifier predicts *none* or *dontcare* then the generated word is ignored. If the classifier predicts *ptr*, then the generated word is taken as the final value for the domain-slot pair.

2.4 Loss Functions

(Wu et al., 2019) have defined custom loss functions for use in training the TRADE model.

Slot Gate Loss The Slot Gate loss, L_g, is defined as follows:

$$L_g = \sum_{j=1}^{J} -log(G_j \cdot (y_j^{gate})^\top)$$

Where J is the number of domain-slot pairs, G_j is the output of the slot gate for the j^{th} pair and y_j^{gate} is the true slot gate value.

State Generator Loss The loss for the state generator, L_v, is defined as follows:

$$L_v = \sum_{j=1}^{J} \sum_{k=1}^{|Y_j|} -log(P_{jk}^{final} \cdot (y_{jk}^{value})^\top)$$

Where again J is the number of domain-slot pairs, $|Y_j|$ is the number of decoding time steps for the j^{th} pair, P_{jk}^{final} is the final state output of the generator and y_{jk}^{value} is the true state.

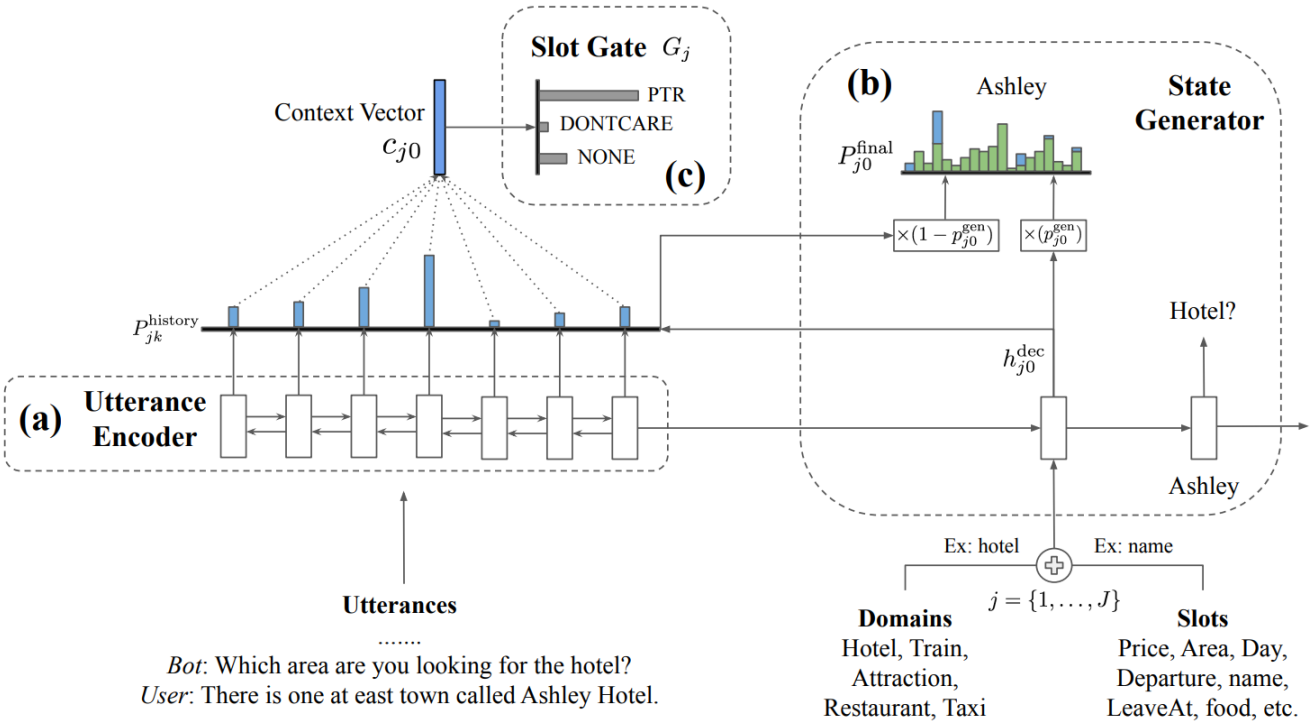

Figure 2: Overview of the TRADE model (Wu et al., 2019).

Both losses are combine using hyperparameters α and β to form the total loss L:

$$L = \alpha L_g + \beta L_v$$

2.5 Zero-shot DST

One of the tasks TRADE uses to measure performance is how effective it is at zero-shot transferring. Zero-shot transferring occurs when a domain is completely removed from the training set, but is retained in the testing set. If the model can perform well on the test set despite never seeing any examples of the removed domain in training, then the model has successfully performed zero-shot transferring.

2.6 Few-Shot DST

The other task TRADE evaluates itself on is few-shot learning. Few-shot transferring is when a model is trained with a left out domain, and then retrained with a small number of samples from the unseen domain. There are three different techniques used by TRADE.

Fine Tuning The first is fine tuning, which is simply retraining the pre-trained model on the unseen domain with no parameter modifications.

EWC The second is elastic weight consolidation (EWC) which uses a Fisher information matrix as a regularization term for the unseen domain (Kirkpatrick et al., 2017). EWC modifies the original loss function L using the original source domain parameters Θ_S, resulting in:

$$L_{ewc}(\Theta) = L(\Theta) + \sum_i \frac{\lambda}{2} F_i(\Theta_i - \Theta_{S,i})^2$$

Where λ is a hyperparameter, F is the Fisher information matrix and Θ is the new parameters trained on the unseen domain.

GEM The third technique is gradient episodic memory (GEM) which uses a small number of samples from the training set in order to prevent the model from forgetting previously seen domains while training on the new unseen domain (Lopez-Paz and Ranzato, 2017). GEM also modifies the original loss

function using the following process:

$$\text{Minimize}_{\Theta} L(\Theta)$$
$$\text{Subject to } L(\Theta, K) \leq L(\Theta_S, K)$$

Which aims to optimize the new loss while preventing the K samples from the source domains from modifying the loss. This has a freezing effect on the loss when training on one of the K samples, which preserves the knowledge of the old domains while minimally effecting the learning of the new domain.

2.7 Evaluation Metrics

The TRADE model utilizes two evaluation metrics when testing performance. These metrics are joint goal accuracy and slot accuracy. The joint goal accuracy is calculated by comparing the ground truth dialogue state B_t to the predicted value t. If all predicted values match the ground truth, then the prediction is considered correct. The slot accuracy is calculated by comparing each predicted (domain, slot, value) to the ground truth.

3 BERT

The Bidirectional Encoder Representations from Transformers (BERT) model is an encoding model that utilizes the idea of pre-training a language model to produce better encoded representations of language models. It does this by using deep bidirectional transformer connections in order to obtain language context in both directions. An overview of the architecture can be seen in figure 3.

3.1 Transformers

The bidirectional transformer architecture used by BERT allows the attention mechanisms in the transformer layers to extract features that are affected contextually in both (left-right and right-left) directions. The transformers used in BERT are identical to the architecture proposed by (Vaswani et al., 2017).

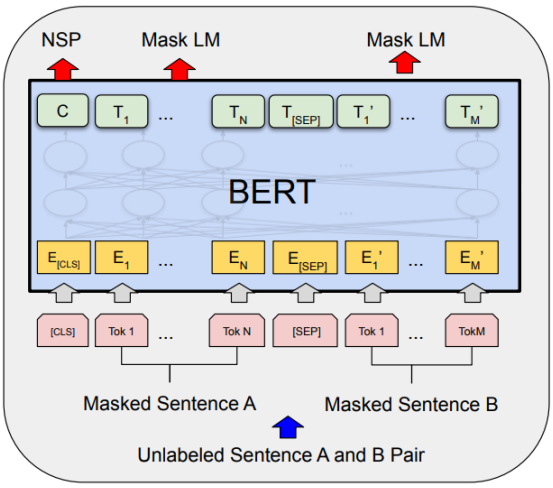

Figure 3: Architecture of a BERT model being pre-trained on the NSP task (Devlin et al., 2018).

3.2 Tasks

The BERT architecture is extremely versatile. Despite only being pre-trained on two unsupervised tasks, it is shown that minor fine tuning can lead to state-of-the-art performance in a variety of NLP tasks. BERT is pre-trained on the following tasks:

Masked LM Traditionally, training a bidirectional language model is trivial, as the model would easily be able to predict any word by simply looking in both directions during training. To circumvent this, (Devlin et al., 2018) randomly mask input tokens in a process they call Masked LM, and then only predict the masked tokens.

Next Sentence Prediction The second task seeks to teach BERT how to determine relationships between two sentences. Next Sentence Prediction (NSP) is a binary classification task where the model is given two sentences A and B, and must determine if B follows A or not in the original text.

(Devlin et al., 2018) show that in doing the above pre-training tasks, BERT is capable of achieving state-of-the-art performance in tasks such as Question Answering, Natural Language Inference and Sentence Classification with minimal fine tuning required.

4 Modifications Made

I had to make a number of modifications to the original TRADE system in order to integrate the BERT model. I first started by cloning their repository from their Github page[1]. My modified TRADE system can be found on my Github page[2].

4.1 BERT Encoder Class

The first modification I had to do was adding a BERT encoder class into the TRADE model super class so that the BERT model could be initialized on runtime. The original model is implemented using PyTorch and as such is relatively modular, so this involved creating a BERT class mimicking the required parameters the original *EncoderGRU* class used. I utilized the *transformers* library (Wolf et al., 2019) to create and initialize the BERT model.

EncoderBERT Initially I was only planning on using the pre-trained BERT in the project. However there seems to be an issue with the loss function of the TRADE model and the configuration of the base pre-trained BERT. While training, any batch used beyond the first batch of the training epoch will result in NaN loss values for both L_g and L_v. As a result, the model cannot perform back-propagation and the weights remain untrained, leading to training accuracy values of 0. I attempted to modify the training sequence of the TRADE model to disable the built-in training of the encoder, however even if I set the encoder training to false the NaN values still occur.

UntrainedEncoderBERT In light of the issues when attempting to use the pre-trained BERT model, I decided to pivot and try training a BERT model from scratch in parallel with the TRADE model. This method does not lead to any issues with the loss values, so the TRADE model succeeds in training. This confirms my belief that there is some incompatibility between the pre-trained BERT's configuration and the TRADE model parameters. Due to time constraints in the project, I leave discussion of this to section 6.1.

Additionally, in both BERT classes I had to modify the hidden state output shapes as well as the word embedding output shape in order to match the output sizes expected by the TRADE model.

4.2 Configuration Changes

There were a few configuration flags and checks that I had to implement in order to integrate BERT into the TRADE model. They are as follows:

- Encoder flag: An argument that allows the user to specify the encoder model
- Train Encoder Flag: A flag that enables or disables the training parameter for the encoder model
- Untrained BERT Flag: A flag that specifies if the encoder should be a pre-trained or untrained BERT model
- Layer Number Flag: A flag that denotes the number of hidden layers for the encoder model.

After adding these I also had to modify each of the training and testing scripts to properly utilize the added flags. The layer number flag was included in the original code, but was commented out and not used in any part of the code base. In addition, since the BERT model requires a vocabulary file to initialize the BERT Tokenizer object, I also had to generate and link said file to the implemented BERT classes.

5 Experiments

All experiments were performed on a Linux server equipped with two Nvidia Tesla P40 GPUs as well as 192GB on RAM. Since the original paper does not specify training details, as a baseline I trained two TRADE models with a GRU encoder for 10 epochs; one

[1] https://github.com/jasonwu0731/trade-dst
[2] https://github.com/joshualemmon/trade-dst-bert

	Restaurant	Taxi	Hotel	Attraction	Train
Slots	food, price, area, name, time, day, people	destination, departure, arrive by, leave by	price, type, parking, stay, day, people, area, stars, internet, name	area, name, type	destination, departure, day, arrive by, leave at, people
Train	3813	1654	3381	2717	3103
Valid	438	207	416	401	484
Test	437	195	394	395	494

Table 1: Overview of the domains used in MultiWOZ, including slot types and number of samples.

on all domains and one on all domains except taxi. Each epoch took 3.5 hours on average for a total of 35 hours a model, therefore due to time constraints I could only test one unseen domain when originally I planned to test all unseen domains. I chose the unseen domain to be taxi, as doing transfer learning to the taxi domain showed the best results in (Wu et al., 2019).

5.1 Dataset

As mentioned before, the dataset used is the MultiWOZ human-human conversation corpus. While the original MultiWOZ dataset has 7 domains, I only use the same five domains that were used in the original TRADE paper. These domains are: restaurant, taxi, hotel, attraction and train. These domains encompass 30 domain-slot pairs with over 4500 possible values (Wu et al., 2019). Each domain has a number of training, testing and validation samples which are outlined in table 1

5.2 BERT Configuration

I experimented with a number of BERT configurations to see how the configuration effects performance.

Pre-Trained BERT The base pre-trained BERT model is quite large with 12 layers, a hidden size of 768 and 12 attention heads. I quickly found that using an encoder of this size caused issues in the training process, resulting in NaN loss values after only a few batches were run in the first epoch. I go into more detail on this is section 6.1. Since the pre-trained model was unusable, all future configurations discussed use an untrained BERT configuration.

Untrained BERT with Pre-Trained Configuration To test if the NaN issue was due to faulty back-propagation on the pre-trained model, I tried using a pre-trained model with the same parameters as the pre-trained model. This untrained BERT resulted in NaN loss as well, which lends evidence toward the size of the encoder being the issue.

Deep Untrained BERT I tried using an untrained BERT model that is more shallow than the pre-trained BERT. I used the default TRADE hidden state size of 400, 16 attention heads and the following number of layers: 2, 4, and 6. In every case the TRADE model was capable of training, however the results were quite poor: 2% for joint accuracy and 80% for slot accuracy. As such, I do not report on these models in the results.

Shallow Untrained BERT I also tried to create an untrained BERT model that is similar to the base GRU encoder used by TRADE. This BERT model has only a single layer, a hidden state size of 400 and 16 attention heads. Since this was the best performing BERT model, I use its results in the following analysis. I also tested a single layer BERT model with a hidden state size of 768, however its performance was similar to the deeper models described above, so I have omitted it from the results as well.

5.3 Full Domain Comparison

The first comparison performed was on the full domain training set. I trained the BERT model for 10 epochs in a similar time frame as the GRU baseline before testing both on the test set. The results can be seen in table 2.
Clearly the GRU encoder outperforms the BERT encoder in all metrics. While the slot accuracy is comparable, the joint accuracy has a huge disparity.

Encoder	Joint Acc.	Slot Acc.	Joint F1
GRU	45.91%	96.53%	88.09%
BERT	16.72%	91.48%	66.21%

Table 2: Joint, slot and F1 scores evaluated on all domains.

5.4 Zero-shot Comparison

The next comparison is performing zero-shot transferring. I trained a base model (BM) on all domains except taxi for both the GRU and BERT encoders. I then evaluate the BM on the full test set including the taxi samples. The results of the zero shot transferring can be seen in table 3.

Encoder	Joint Acc.	Slot Acc.	Joint F1
GRU	44.67%	96.35%	87.69%
BERT	17.01%	91.81%	68.65%

Table 3: Joint, slot and F1 scores when performing zero-shot transferring on the taxi domain.

Once again, the GRU encoder outperforms BERT in both joint and slot accuracy. Interestingly enough however, the joint accuracy of the BERT model actually improved (+0.29%) compared to the full domain BERT, while the joint accuracy of the GRU encoder decreased (-1.24%). Similarly, the BERT slot accuracy increased by +0.33%, and the GRU slot accuracy decreased by -0.18%.

5.5 Few-shot Comparison

The next comparisons are on the various few-shot techniques utilized in the original paper. I use the same BM trained for the zero-shot transferring in all of the following few-shot transfer tasks. Each few-shot task was run using separate training scripts implemented in the original code-base, with modifications made to allow for the use of the new BERT encoder classes.

Fine Tuning The first few-shot technique I evaluated was the fine-tuning task. The zero-shot BMs were naively retrained using a small amount (1%) of the taxi training samples. Once retrained the new models were evaluated on the full test set including the taxi samples. The results of the fine tunign can be seen in 4.

Encoder	Joint Acc.	Slot Acc.	Joint F1
GRU	41.53%	96.01%	86.13%
BERT	15.31%	91.20%	64.81%

Table 4: Joint, slot and F1 scores when performing few-shot transferring on the taxi domain with fine tuning.

Similarly to the previous results, the GRU outperforms the BERT overall. The GRU encoder has an overall joint accuracy decrease of -3.14%, and BERT encoder has a decrease of -1.70%. This seems to imply that while the GRU encoder can achieve better accuracy as a baseline, the BERT encoder is more robust to knowledge transferring and can retain information of the previous domains better than the GRU encoder can. The slot accuracy of the GRU, -0.34%, fairs a bit better than that of the BERT, -0.61%, however.

Elastic Weight Consolidation The next few-shot technique tested was the EWC task. Again only 1% of the taxi training data was used to retrain the BMs. The original paper does not specify what value they use for the hyperparameter λ, instead stating that they set different values for each domain that is found by testing with the validation set. Since I did not have enough time to test multiple λ values, I simply opted to set $\lambda = 1$. Likewise, the original paper does not state the size of the Fisher matrix, so I used the default value of 10000. The results of the EWC training can be seen in 5.

Encoder	Joint Acc.	Slot Acc.	Joint F1
GRU	41.68%	96.03%	86.22%
BERT	15.93%	91.43%	65.69%

Table 5: Joint, slot and F1 scores when performing few-shot transferring on the taxi domain with EWC.

The GRU joint accuracy drop off for EWC is slightly improved compared to the fine tuning method, with a decrease of -2.99%.

The BERT joint accuracy also improves, becoming -1.08%. Likewise the slot accuracies see an improvement of -0.32% and -0.38% for GRU and BERT respectively. Again the joint accuracy values seem to imply that the BERT model is more robust to memory loss compared to the GRU.

Gradient Episodic Memory The final few-shot task I evaluated is GEM. Following the original paper, I set the K value to 1%, so the BM is retrained on 1% of the original training data and 1% of the taxi training data. The results of the GEM training can be seen in 6.

Encoder	Joint Acc.	Slot Acc.	Joint F1
GRU	41.56%	96.02%	85.99%
BERT	14.43%	91.06%	64.06%

Table 6: Joint, slot and F1 scores when performing few-shot transferring on the taxi domain with GEM.

The GEM training results in slightly worse accuracy decreases for the GRU; being -3.11% for joint accuracy and -0.33% for slot accuracy. Comparatively, GEM causes much worse results for the BERT encoder, with values of -2.58% and -0.75% for joint accuracy and slot accuracy respectively. This shows that the BERT encoder is not very compatible with GEM training relative to the other techniques, as it results in the overall worst performance.

5.6 Analysis of Results

I have compiled all the accuracy results of the various technique in table 7, where I have shown the absolute difference in accuracy between each technique and the full-domain models. It can be seen that outside of the GEM technique, the relative change in accuracy for the BERT models are all better than their GRU counterpart. This is a very promising result, as the original TRADE paper states that a large issue with these knowledge transfer techniques is that of catastrophic memory loss (Wu et al., 2019). This means that when training on an new unseen domain, the model will forget much of what it has learned about the previously seen domains. While the GRU encoder outperforms the BERT encoder as a whole, the fact that BERT is more robust to this memory loss issue is encouraging.

Encoder	FD	ZS	FT	EWC	GEM
GRU-J	45.91	44.67 (-1.24)	41.53 (-4.38)	41.68 (-4.23)	41.56 (-0.35)
GRU-S	96.53	96.35 (-0.18)	96.01 (-0.52)	96.03 (-0.50)	96.02 (-0.51)
BERT-J	16.72	17.01 (+0.29)	15.31 (-1.41)	15.93 (-0.79)	14.43 (-2.29)
BERT-S	91.48	91.81 (+0.33)	91.20 (-0.28)	91.43 (-0.05)	91.06 (-0.42)

Table 7: Comparison and relative accuracy increase/decrease across all training techniques. -J denotes the joint accuracy while -S denotes the slot accuracy.

The main advantages of using BERT as an encoder is its pre-training and deep network architecture. Due to the fact that the TRADE systems seems incompatible with large encoders nullifies the advantages of BERT that caused it to supplant GRUs in NLP research. It is not unreasonable to think that the memory loss resistance displayed by the shallow BERT model could be attributed to a full pre-trained BERT model. This idea is strengthened by the fact that BERT has been shown to be extremely versatile to a number of tasks with small amounts of fine-tuning, which has parallels to the idea of knowledge transferring. If the TRADE model was overhauled to be compatible with larger encoders, it is likely that a pre-trained BERT encoder would outperform the GRU encoder by a significant amount.

Another interesting result is that the BERT model with zero-shot transferring actually performed better than the full-domain BERT model. While (Wu et al., 2019) shows that, of all the domains, taxi zero-shot performed the best, there was no case where the zero-shot training outperformed the full training. I believe the reason for this is because the BERT encoder is better than the GRU encoder at encoding general context from input sequences. The original paper states that there is a lot of overlap between the *train* and *taxi* domains. Then when both encoders receive a sequence they have never seen before (e.g. the taxi data) the BERT is better at generalizing the *train* domain information for use in the *taxi*

domain. Perhaps a combination of the generalization and some overfitting to the full-domain data is what caused the performance discrepancy.

6 Future Work

I have identified a number of potential future avenues to explore in relation to the TRADE architecture.

6.1 Integrating Pre-Trained BERT

As mentioned before, I found that there were some incompatibilities between the base pre-trained BERT configuration and the TRADE model parameters that resulted in issues computing the loss while training. Another potential source of the issue is an exploding gradient. Since both the size of the BERT model is far deeper than the base TRADE GRU, and the hidden state size of BERT, 768 vs 400, is larger, this could be causing the gradients during trading to explode. This seems like the most probable cause since, as mentioned in section 5.2, if I increase the depth of the untrained BERT the performance drops, and if I increase the hidden state size to 768 the untrained BERT results in NaN loss values as well.

A potential solution to this would be to modify the TRADE architecture slightly to use different activation functions that are less prone to vanishing gradients. Currently TRADE uses a sigmoid activation function, so potentially changing that to ReLU or a similar rectifier could alleviate the issue (Maas et al., 2013). Another potential solution could be to introduce batch normalization into the training process, which has been show to reduce model susceptibility to vanishing gradients (Klambauer et al., 2017). However if either of these options were to be pursued, it must be studied how such changes affect the original GRU-based TRADE model's performance.

Furthermore, since a shallow BERT can be integrated successfully into the TRADE architecture, there is potential that pre-training this shallow BERT in a similar manner to the base BERT would result in much better performance.

6.2 Replacing TRADE Decoder

The TRADE model uses a GRU for both its encoder and decoder models. One possible future avenue of research is to replace this GRU decoder with a transformer decoder as outlined in (Vaswani et al., 2017). I believe that this could result in a performance boost for the BERT encoder, as the transformer decoder will also be utilizing an attention mechanism. The transformer decoder will also likely be able to handle the larger output the BERT generates, which the base GRU encoder does not seem to be able to handle. This will likely allow the decoder to more sufficiently decode the BERT embeddings and capture more context from the input sequence.

6.3 Other Encoding Options

While BERT is an extremely popular encoding model, there are a number of other encoding models that could be explored. The BERT model itself has spawned many variants that have improved upon the original architecture. Given that the loss issues are solved using one of the above potential solutions, here are some other language models that I think would be promising prospects to use as encoders:

- ALBERT - A smaller BERT model show to have improved performance on downstream tasks (Lan et al., 2019).
- ERNIE - A language model that incorporates knowledge graphs into its training process and shows improved performance on knowledge-based tasks (Zhang et al., 2019).
- TransBERT - A BERT model that uses transfer learning to transfer both general and specific domain knowledge for use in a target task (Li et al., 2019).
- GPT-2 - A powerful generative language model trained on a large internet corpus (Radford et al., 2019).

- BART - A generalized BERT/GPT model that excels in text generation tasks (Lewis et al., 2019).

7 Conclusion

In conclusion, I show that it is possible to integrate a BERT encoder model into the TRADE DST model architecture. Unfortunately due to some limitations with the TRADE decoder model, the full pre-trained base BERT model is incompatible, and results in errors while computing training loss.

In lieu of the pre-trained BERT I use an untrained shallow BERT model that is trained in parallel with the TRADE model. After experimental evaluation, it is seen that the GRU encoder outperforms the shallow BERT, likely because the shallow BERT is unable to utilize the main strengths of the BERT model. However, despite being outperformed accuracy-wise the BERT model seems to be more robust to catastrophic memory loss than the GRU encoder is.

The BERT model also achieves better absolute accuracy increases/decreases relative to its full-domain results than the GRU model does to its own full-domain results. This seems to imply that if a fully pre-trained BERT model was capable of being integrated into the TRADE system then it would likely outperform the state-of-the-art result achieved in the original paper.

Furthermore, I have also proposed a number of possible future avenues of research that could potentially rectify the issues with integrating the full BERT model. These include modifying the activation function of the GRU decoder, introducing batch normalization into the training process and replacing the GRU decoder with a transformer encoder.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018). https://doi.org/10.18653/v1/d18-1547

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv:cs.NE/1412.3555

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:cs.CL/1810.04805

Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017). https://doi.org/10.18653/v1/d17-1206

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-Based Dialog State Tracking with Recurrent Neural Networks. *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)* (2014). https://doi.org/10.3115/v1/w14-4340

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (Mar 2017), 3521–3526. https://doi.org/10.1073/pnas.1611835114

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. arXiv:cs.LG/1706.02515

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:cs.CL/1909.11942

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. arXiv:cs.CL/1910.13461

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story Ending Prediction by Transferable BERT. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (Aug 2019). https://doi.org/10.24963/ijcai.2019/249

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. arXiv:cs.LG/1706.08840

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural Belief Tracker: Data-Driven Dialogue State Tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1777–1788. https://doi.org/10.18653/v1/P17-1163

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014). https://doi.org/10.3115/v1/d14-1162

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:cs.CL/1706.03762

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019). https://doi.org/10.18653/v1/p19-1078

Puyang Xu and Qi Hu. 2018. An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 1448–1457. https://doi.org/10.18653/v1/P18-1134

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced Language Representation with Informative Entities. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019). https://doi.org/10.18653/v1/p19-1139

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-Locally Self-Attentive Encoder for Dialogue State Tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 1458–1467. https://doi.org/10.18653/v1/P18-1135