multi-tasking NLP model

Shuyan Lin
slin353@gatech.edu

Jiarui Xu
jxu605@gatech.edu

1 Introduction

The idea of multi-tasking learning is inspired by human learning where knowledge from one task can be applied to the other task. This project is inspired by the powerful Bert model as an entry point to extract features or fine tunes for the downstream tasks and its ability to have good performance across all different tasks. We aim to perform a multi-task model to train multiple downstream tasks all at once. The advantage of doing so is a more generalized representation and embed more than contexts but also intentions, sentiments ect. It's also helpful for tasks that has small data-sets. By training multitask, it can augment the data-sets. Training multitasks also is a method to prevent over-fitting. Our approach is to train multi-task of Name Entity recognition(NER), intent detection, fragment detection 3 tasks all together using bert pretrained model and plugged in downstream tasks using a linear layer and drop-out layer. The multi-task is parameter sharing, meaning the weight of bert model is shared, while the downstream task's parameters are separate. The result of using multitask can increase the performance to 1 percent of 2 tasks out of 3 tasks compared with a baseline that is trained alone. We were able to make conclusion that the improved in perfomance is due to the effect of multi-task training not because of merely increase in the size of the dataset. We experimented with a decrease in a size of all tasks' dataset by 30 percent and compared it with a decrease in size of the dataset of only NER. We observed that even though the later have more dataset, because NER has more relevancy, the drop in NER decrease the performance of intent detection.

1.1 problem domain

The definition of multi-task learning is given m learning task where all the tasks are related, multi-task aim to learn m tasks together at the same time to improve each task using the knowledge learned for all tasks. Our problem domain is focusing on multiple NLP related tasks such as NER(Name Entity Recognition), sentence intent classification, fragment classification which is homogeneous Multi-tasking. All the tasks are supervised providing a label/labels. The model is parameter based learning where the lower level parameters are shared and downstream parameters are trained for specific tasks.

1.2 inputs and outputs

The input of the model is 1 single data-set containing both NER labels and the intent of the sentence. Fragments are generated from the complete sentence as counter-examples. All of the inputs are single sentences and they are feed into Bert pretrained Tokenizer to get input id(for word embedding), token id(to separate sentences) and attention mask(to indicate paddings). output is separate for 3 tasks. Currently including:

1. Intent Detection: given a sentence, predict the intent label, for example output can be bookflight.

2. NER: Name entity recognition task, the output is a sequence of NER label.

3. Fragment Detection: Given a single sentence predict if it's a whole sentence or not. Fragment sentences are generated from complete sentences. The output is a binary 1 or 0 to indicate if it's fragment.

1.3 Importance

This task is important because we can envision by training multiple related sub-task together, it can be plugged in to train a downstream task. For example, the training language models can be used as word representations or sentence representations. Side-effects of training well-designed tasks together can

encode a better and more comprehensive representation of languages. It may be able to capture the more complex nature of languages such as sentiments,co-reference, and objectives. From the related work of research, since multi-task learning can share a representation, in the future by experimenting with sets of tasks trained together, a better word representation can be used. An example of such word embedding being effective is that For example, Elmo and Bert embedding works better because it also represented the context of the word.

1.4 Goal to accomplish

Our final goal is come up with a set of tasks to be learned together so that they can provide a richer representation for embedding to aid the training of each downstream task. We also want to examine if the increase in performance is due to increase in dataset sizes by combining tasks or it's because of different tasks aid each other to provide a richer context.

1.5 a survey for multitasking deep learning

Multitask learning is a domain not only in NLP, but a general learning method like transfer learning. There are 2 types of MTL(multitasking learning) heterogeneous and homogeneous. Heterogeneous can include tasks with different types such as supervised or reinforcement learning. Important decisions to make are when to share: making a choice between single-task and multi-task model. The best way to through experimentation.

What to share, including 3 different types of sharing: feature, instance and parameter sharing. Feature based learns common features among different task. When applied to NLP, this can be sharing the same word representations. Because of multiple task is being learned all together, a more powerful and expressive representation can be learned. The earliest Multi-task learning model is a FNN(feedforward neural network), where the output unit is m instead of 1 where m corresponds to m tasks.Therefore the last layer of MLP can be considered as feature representation.

Bakker and Heskes proposed a way to share inputs to hidden layers's parameters and have task specific parameters from hidden layer to output using multi-task Bayesian Neural Network, where they share the same Gaussian distribution. This is very similar to our model where the weights from bert is shared and the downstream tasks have task-specific weights. (Zhang and Yang, 2021)

1.6 Related work

One multi-tasking DNN paper for NLP included 4 tasks: single and pair sentence classication, text similarity,relevance ranking. The model architecture is first extract word, position,sequence embedings from sentences, and feed into transformers, which learns a shared representations using multi-task objective. The training stage is pre-training and multi-task traning. Pre-training is the bert pre-training of embeddings. The algorithm for multi-task training is: Pack t tasks datasets into multiple Minibatch D_t. Merge all of the mini-batch and shuffle. Each of the minibatch is updated based on their objective function. We will be implemented the same method for the multi-task training. It shows increased performance in all tasks in the multi-task DNN. The results are on par or sometimes pass human performance. (Liu et al., 2019)

Another paper is training on Multi-task learning to learn a richer representation. It's a hierarchical model on carefully selected tasks. The model trains a lower level task at the bottom layer and a higher layer task and was able to reach state of art result. (Sanh et al., 2019)

In 1993, Caruana(Caruana, 1993) proposed the hard parameter sharing neural network as the prototype of MTL model. This model shares the hidden layer between tasks and preserves the output layer for each task. The advantage of this model is that overfitting of the model is avoided greatly. In 1997, Baxter(Baxter and Eyles, 1997) proposed that when the model shared parameters for N tasks, the overfitting risk was of order N. This risk is less than the risk of overfitting the output layer parameters for a particular task. The intuitive understanding is as follows: when the model learns multiple tasks at the same time, it has to learn the similarities between these tasks to find the compromise parameters, so the overfitting probability on the original task will be greatly reduced.

In 2015, Duong(Duong et al., 2015) proposed another multi-task learning model for soft parameter sharing. He configured each task with its own model, and then regularized the parameters to form a similarity between the models. In 2016, Yang and Hospedales(Yang and Hospedales, 2016) used trace norms for the same purpose.

Our approach is similar to the above approach because we are also using shared weights. There are previous work that tried to train multiple NLP task all at once. The difference is that we tried to exper-

iment with if the dataset can actually learn a richer context by examing the relationship of the tasks when trained together. Some experimentation we did is decrease the size of dataset of 3 tasks to see if it makes a difference. Becuase by training 3 tasks together we are trippling the dataset, therefore it's obvious to see an increase even though the weight update is separately. We want to see a stronger evidence of the effects of the tasks. We decrease the batch number of around 4000 down to 1000 to see if the effect is still obvious. Next to examine the effects of task on each other and based on the result of of our preliminary. We observe that by training the baseline alone, fragmentation detection is not impacted a lot when training together versus training by itself. However, the accuracy of NER and intent detection is strong even though only trained 4 epochs. Therefore, we have a theory that this is due to making the correct prediction of intent detection and NER labeling does not help with the task of fragment detection. However, since NER and intent detection are more closely related, increasing or decreasing the size of the dataset of one out of the two will influence the improvement.

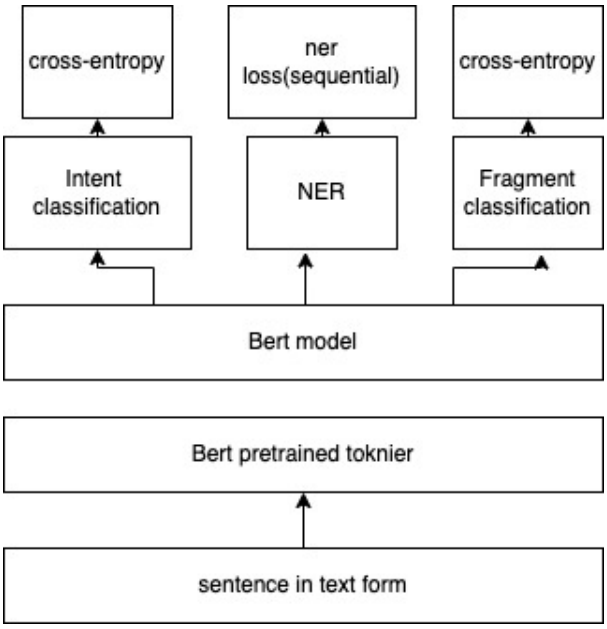
2 Dataset:

SNIPS(Coucke et al., 2018) data set is generated based on user data of snips voice platform. This platform provides very advanced automatic speech recognition and natural language processing technology. Technicians trained small neural networks and ASR and NLU language models to generate SLU data and test its accuracy. This approach leverages a combination of crowdsourcing and machine learning to generate high-quality data sets without harming the user.

3 Technical Approach:

3.1 Model Architecture

The lower layer of our model is shared among all tasks and we are using BERT tokenizer as a embedding encoding to feed into the pre-trained Bert model. The output of the Bert is shared among all the tasks specific tasks.



3.2 Implementation

We reimplemented dataset class and batchsampler class. Dataset class now contains all data from the 3 tasks with the label of which task the dataset belongs to. Batchsampler makes batches of the merged single dataset and shuffle the batches. For model, We used the pretrained Bert and Bert tokenizer, we get the last hidden layer and the Pooler output. Last hidden layer is used for NER task and the Pooler output is for classification. After this applied dropout and Linear weights to each specific task(not shared parameters). The training batch size is 16, dropout layer is 0.2-0.3 and used the learning rate 2e-5. The optimizer is AdamW using bert's optimizer and scheduler.
The difference between multi-task learning and single-task learning is the data loader and batch sampler. For the data loader, it needs to merge all 3 tasks' datasets into all task datasets. For a batch sampler, it needs to shuffle the all task dataset and make an iterable such that one batch is a random selection of one of the 3 tasks with a batch size of 16. For training the data, an accumulative gradient update is used. The reason to use accumulative gradient update is because of the shuffling, it can update gradient on a larger batch. The weights are updated to improve the specific task of the batch not the total of 3 goals. Training the 3 tasks together can be achieved by storing a meta data of each batch to contain information about which task the data batch corresponds to. Therefore, it can allow for flexibility of using different loss, different metrics to evaluate accuracy.

Method	Accuracy
$Training_{baseline}$	98.42
$Tesing_{baseline}$	98.33
$Traning_{multi-task}$	99.44
$Testing_{multi-task}$	99.16
$Testing_{30\%dataset}$	99.63
$Testing_{30\%NER}$	98.77

Table 1: Intent detection baseline

Method	Accuracy
$Training_{baseline}$	99.44
$Training_{baseline}$	99.3382
$Traning_{multi-task}$	99.44367176634215
$Testing_{multi-task}$	99.16405433646813
$Testing_{30\%dataset}$	99.44317894203999
$Testing_{30\%onlyNER}$	99.00140486767665

Table 2: Fragment detection baseline

Method	F1	Precision	Recall
$Traning_{baseline}$	60.86	55.48	67.39
$Testing_{baseline}$	57.80	52.36	64.50
$Traning_{multi}$	65.82	60.25	72.51
$Testing_{multi}$	62.94	56.89	70.43
$Testing_{30\%dataset}$	63.83	57.69	71.43
$Testing_{30\%onlyNER}$	70.11	58.96	71.69

Table 3: NER evaluation baseline

3.3 baseline

The baseline is training the same number of epochs on only a single task.

3.4 evaluation

We evalute the performance of each task separately.

1. Intent Detection: evaluate using accuracy.
2. NER: Evaluate using Precision, recall and F1 score.
3. Fragment Detection: evaluate using accuracy.

4 observation

It takes different tasks different speed to converge. We note that by training only on 1 epoch the accuracy of intent detection and fragment detection can reach 99 percent, however, F1 score of NER is only 61 percent. Therefore, it's important to decide the number of epoch to stop by considering all 3 tasks's performance since the weights are updated separately.
Second, we are able to achieved our second goal that we theorized that the increase of the performance is not due to the increase of the data-set by tripling the dataset samples. We experiment it with using only 30 percent of the dataset of the each tasks' dataset. The number of batch is reduced from 4000 batches to 900 batches. However, the accuracy is not effected. Therefore, some conclusion can be made that the increase in performance is not due to the increase in dataset size. Next, because we thought NER helps to train Intent detection, we greatly reduced the number of training set of NER to only 30 percent and keep the rest of the dataset sample the same size. Even though we only reduced the batches from 4000 to 3000, we see a significant 1 percent drop of performance, which further reinforced the idea that it's not able the size of the dataset, but about the relevancy of the tasks.

5 future works

For future works, it can be used to aid a more complex task by trainig a complex task with multiple subtask to aid the complex task. For example, for translation, NER can help it better translate. Another idea can be design well related tasks to be trained together and extract word embed-dings to see if it can be a richer and more generalize embedding than the current SOTA embedding.

References

Jamie Baxter and John Eyles. 1997. Evaluating qualitative research in social geography: establishing 'rigour'in interview analysis. *Transactions of the Institute of British geographers*, 22(4):505–525.

R Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias1. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48. Citeseer.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, pages 12–16.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956.

Yongxin Yang and Timothy M Hospedales. 2016. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*.

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*.