Multitask Bert with Task Embedded Attentions (TEA-BERT)

Stanford CS224N Default Project

Chunjiang Mou
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
`cmou2@stanford.edu`

Zifei Xu
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
`zifei98@stanford.edu`

Sally (Hanqing) Yao
Institute for Computational and Mathematical Engineering (ICME)
Stanford University
`yaohanqi@stanford.edu`

Abstract

Well known for its deep bidirectional Transformer architecture and capability of learning text-pair representations, pre-trained BERT Devlin et al. (2018) models have been widely adopted to get fine-tuned for different tasks. Given the large size of BERT models, it is infeasible to store separate parameters when adapting to each task. Multi-task learning is helpful in extracting meaningful features shared across tasks, which can provide a robust task representation to improve prediction accuracy and allow parameter-sharing among different tasks for training time and resource efficiency. For this project, we aim to improve the vanilla BERT model to a multi-task model which incorporates task embeddings so that this single model can flexibly be used to solve various NLP tasks with high accuracy while not increasing model size too much. In the first stage of the project, we implemented vanilla BERT and AdamW for the baseline, and experimented on directly adding task embeddings to input embeddings obtained from BERT to compare the modified model's performance with the baseline model. In the second stage of the project, we experimented with various methods to inject task embeddings into minBERT model.

1 Key Information to include

- Mentor: Hans Hanley

2 Introduction

BERT (Devlin et al., 2018) models enable bidirectional self-attention in the encoder by the use of masked-language models to fuse context from both directions and jointly pre-train with the "next sentence prediction" task to learn across sentence dependencies. Its deep bidirectional Transformer architecture and capability of learning text-pair representations allows it to tackle token-level and sentence-level NLP tasks really well. Because it is very difficult and resource-demanding to construct a well-performed multi-layer deep neural network model from scratch, people usually use pre-trained BERT models and adapt the models for different NLP tasks. However, similar to a lot of outstanding pre-trained models, BERT is very large in terms of a number of parameters. Fine-tuning BERT for various NLP tasks requires storing a large number of weight parameters for all of these models separately, which is too time and space-consuming. It is also a fact that many NLP tasks are similar in nature and many parameters in models aiming for different tasks can actually be shared. So, multitask learning can be applied to share a large proportion of weights among tasks to spare memory and space. However, each task still has its own distinct features, which gives the motivation to incorporate task information into the model.

In the project, we experimented with multiple ways to incorporate task information into the BERT's input embeddings layer and multi-head attention layer, allowing the task information to propagate throughout the neural network. We hope task embeddings can improve our neural network's adaptability to a wide range of NLP tasks and could hopefully result in state-of-the-art performance with a negligible increase in the base BERT network size.

3 Related Work

Transformers were first introduced by Google AI in 2017 (Vaswani et al., 2017) and soon became the top choices in many NLP tasks such as text classification, text summarization, etc. Unlike recurrent neural networks (RNNs), transformers process the entire input all at once (Vaswani et al., 2017) which alleviates the problems of long dependency issues and provides steady performance when long sentences dominate the text. Moreover, the mechanism of self-attention mechanism enables transformers to better encode (understand) the words and sentences based on their contexts.

Bidirectional Encoder Representations from Transformer (BERT) is a bidirectional transformer that was trained with masked language modeling (MLM) and next sentence prediction (NSP) objectives. The model was pretrained on BooksCorpus and English Wikipedia and considered both left and right contexts (Devlin et al., 2018) when making predictions. The large training corpus and full context conditioning provide BERT with a more thorough understanding of the words and phrases. Also, because the multi-head mechanism in the Transformer enables BERT to model various downstream tasks, fine-tuning BERT is relatively convenient and less cost-expensive, roughly 3 epochs for simple fine-tuning (Devlin et al., 2018). The experiments performed by Devlin et al. (2018) showed that both $BERT_{BASE}$ and $BERT_{LARGE}$ outperformed all the state-of-the-art models around that time on all tasks by a substantial margin.

However, BERT is a large language model, with $BERT_{BASE}$ having roughly 110 million parameters, and $BERT_{LARGE}$ having roughly 345 million parameters (Devlin et al., 2018). So, storing the weights of all the models trained for many different tasks became impractical and inefficient (Maziarka and Danel, 2021) especially on mobile devices where storage and battery life are constrained (Stickland and Murray, 2019).

Multitask learning shares parameters between related tasks to reduce the total number of parameters and thus comes into place (Stickland and Murray, 2019). Saving memory and training time, multitask learning is widely used in the NLP field. Stickland and Murray (2019) introduces the "Projected Attention Layer (PAL)", which is a low-rank multi-head attention layer added in parallel to normal BERT layers. PALs act as task-adaptation parameters and enable comparable performance to finetuned $BERT_{BASE}$ (Devlin et al., 2018) on many tasks with 7 times fewer parameters in total. Another noticeable contribution to multitask learning in NLP field is Pfeiffer et al. (2020)'s research on AdapterFusion, which combines the knowledge from multiple source tasks to boost the performance on a target task.

Our main reference paper is *Multitask Learning Using BERT with Task-Embedded Attention* (Rajpurkar et al., 2018). This paper introduces a task embedded attention BERT model (EmBERT) where information about the task is transformed into task-specific embeddings and inject to the multi-head attention layers. The EmBERT model only adds in 3 vectors at each self-attention layer per task, which largely reduces additional trainable parameters and allow task information to propagate throughout the entire network for performance improvement. The model was able to outperform the vanilla BERT and three BERT-based multitask models: with task-specific output layer, top task multi-head self-attention layer, and Projected Attention Layers (PAL) (Stickland and Murray, 2019) on 4 datasets (QNLI (Rajpurkar et al., 2016), SST-2 (Socher et al., 2013), CoLA (Warstadt et al., 2019), RTE) and shared first place on the QQP (Chen et al., 2017) dataset, with the least number of parameters.

However, there are still limitations and motivations pointed to our work in this paper. For example, the authors Rajpurkar et al. (2018) did not explore the effect of regularization and only considered injecting task embeddings at multi-head attention layers. Given the promising performance of the EmBERT model, we want to first replicate the idea from this model and apply onto our tasks, and then attempt various weight decay hyperparameters for regularization. Also, we would like to improve on the calculation of $\hat{Q}$, $\hat{K}$, $\hat{V}$ by trying different possibilities of activation functions to increase nonlinearity and experiment on directly adding task embedding information to input embeddings from BERT.

4 Approach

4.1 Main Approach

Inspired by the EmBERT model introduced by Łukasz et al. in the reference paper *Multitask Learning Using BERT with Task-Embedded Attention* Rajpurkar et al. (2018) we would like to improve BERT by incorporating task embeddings and adapt the model to different tasks by adding task-specific or task-shareable layers. Regarding task embeddings, we want to experiment on different ways to generate task embeddings and integrate them into the model, and find out which way can provide the best evaluation metric results as well as adding only limited amount of parameters to the basic BERT model. The followings are ways of incorporating task embeddings under our consideration 1

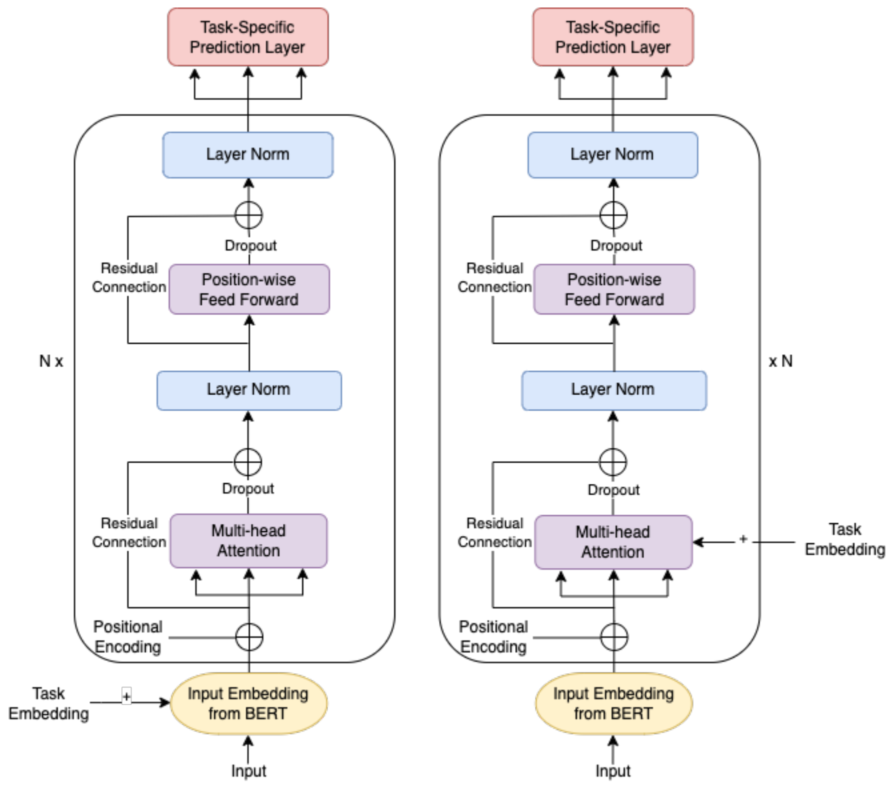

Figure 1: BERT Model with Task Embedding Injection

1. Task embeddings created at input embeddings layer: Generate task embeddings and integrate them to input embeddings before passing into the attention layers. For example, we could

use

$$\hat{E} = E + E_{task}$$

where E is the original input embedding and E_{task} is the task embedding.

2. Direct task attention embedding: Create task-embedded attention Q_{task}, K_{task}, V_{task} vectors directly from embeddings function and then add them to the original input matrices Q, K, V at the multi-head attention layers to get a new set of $\hat{Q}$, $\hat{K}$, $\hat{V}$ for scaled dot-product attention calculations. (Rajpurkar et al., 2018)

$$\begin{aligned}\hat{Q} &= Q + Q_{task}\\ \hat{K} &= K + K_{task}\\ \hat{V} &= V + V_{task}\end{aligned}$$

3. Projected task attention: Create task-embedded attention matrices Q_{task}, K_{task}, V_{task} by projecting task embeddings and then add them to the original input matrices Q, K, V at the multi-head attention layers to get a new set of $\hat{Q}$, $\hat{K}$, $\hat{V}$ for scaled dot-product attention.

$$\begin{aligned}\hat{Q} &= Q + W^{Q}_{task}E_{task}\\ \hat{K} &= K + W^{K}_{task}E_{task}\\ \hat{V} &= V + W^{V}_{task}E_{task}\end{aligned}$$

4. Task attention with activation: Create task-embedded attention matrices Q_{task}, K_{task}, V_{task} by projecting task embeddings and then add them to the original input matrices Q, K, V at the multi-head attention layers, then pass into an activation function followed by a projection layer to get a new set of$\hat{Q}$, $\hat{K}$, $\hat{V}$.

$$\begin{aligned}\hat{Q} &= W^{Q}_{proj} * activation(Q + W^{Q}_{task}E_{task})\\ \hat{K} &= W^{K}_{proj} * activation(K + W^{K}_{task}E_{task})\\ \hat{V} &= W^{V}_{proj} * activation(V + W^{V}_{task}E_{task})\end{aligned}$$

After passing embeddings throughout all the BERT layers, either task-specific layers or task-sharable layers are further applied to complete different tasks.

1. Task-specific layers: Distinct dropout and linear projection layers are applied to the sentiment analysis and paraphrase detection tasks to produce the logits. For the sentiment analysis task, each sentence receives a logit score corresponding to each sentiment category and the max logit label is assigned as the final predicted sentiment label. For the paraphrase detection task, sigmoid function and rounding are further applied to the logits to get the final predictions. For the semantic textual similarity task, we compute the cosine similarity of the results. The similarity is passed to a ReLU function and multiplied by 5.
2. Task-sharable layers: Distinct dropout and linear projection layers are applied to the sentiment analysis task while shared dropout and linear projection layers are applied to paraphrase detection task and semantic textual similarity task due to their sentence-pair input feature. For the sentiment analysis task and paraphrase detection task, same procedures as task-specific layers scenario are used to process logits. For the semantic texture similarity task, the logits are multiplied by 5 as the final similarity score.

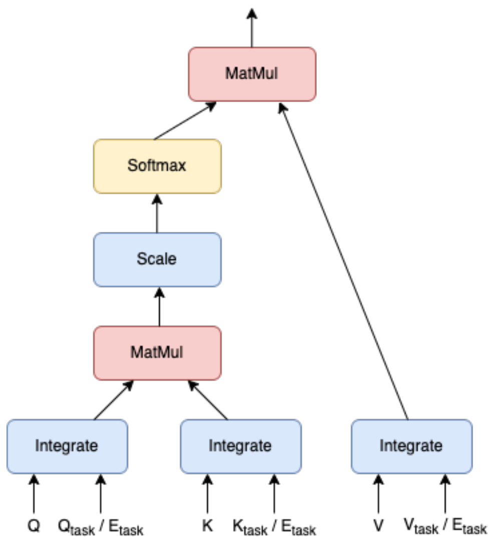

Figure 2: Modified Attention Layer with Q_{emb}, K_{emb}, V_{emb}

4.2 Baseline

We have two baselines, one is the minBERT model trained only on the sentiment analysis task data and the other is trained on all tasks (sentiment analysis, paraphrase detection and semantic textual similarity). Using learning rate of $1e^{-5}$, weight decay of 0.01, we get baseline paraphrase detection accuracy of 0.476, sentiment classification accuracy 0.506, and correlation 0.289 for training only on sentiment analysis task and paraphrase detection accuracy of 0.378, sentiment classification accuracy 0.692, and correlation 0.483 for training on all tasks.

5 Experiments

5.1 Data

1. Stanford Sentiment Treebank (SST) Richard Socher and Potts (2013), $11,855$ single sentences extracted from movie reviews for sentiment analysis task. Each piece of data consists of id, movie review text, and sentiment score. The sentiment scores have 5 different values: negative (1), somewhat negative (2), neutral (3), somewhat positive (4), or positive (5).
2. Quora dataset Chen et al. (2017), released by Quora contains labels of whether pairs of questions were paraphrases of each other for the paraphrase detection task. The inputs are two sentences and the output is whether they are duplicates of each other (1: yes, 0: no)
3. Semantic Textual Similarity (STS) Haoming Jiang (2013), a dataset that contains pairs of sentences with their semantic similarity score on a scale from 5 (same meaning) to 0 (not at all related)

5.2 Evaluation method

We used accuracy for sentiment analysis task and paraphrase detection task, and the correlation between the predicted and true similarity for semantic textual similarity task.

5.3 Experimental details

After running several trials, we decided to use the following configuration for our experiments. We trained on the full SST training dataset ($8,544$ examples), 6% of the Quora training dataset ($8,490$ examples) and 50% of the STS training dataset ($3,020$ examples). We used a batch size of 16, 10 epochs and a learning rate of e^{-5} 3. We used cross entropy loss for sentiment analysis task, binary cross entropy loss for paraphrase detection task, and mean-squared errors for semantic textual similarity task. The experiments were based on varying the method of integrating task information, as mentioned in the main approach section, and the weight decay rate for AdamW. Evaluation metrics are reported on the dev set containing $1,101$ SST task examples, $20,212$ Quora task examples, and 863 STS task examples.

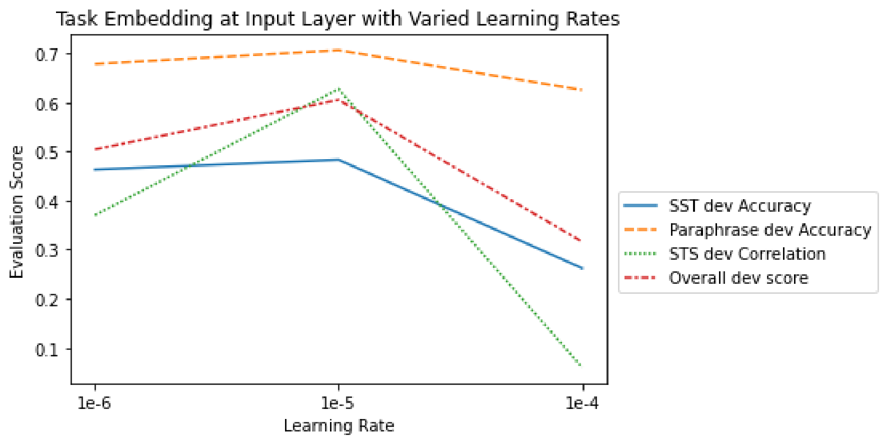

Figure 3: Task Embedding at Input Layer with Varying Learning Rates ($1e^{-5}$ provides the highest accuracies and is thus adopted for running all models)

5.4 Results

The results of our experiments using task-specific layers are summarized in Table 5.4:

	Weight Decay	SST dev Accuracy	Paraphrase dev Accuracy	STS dev Correlation	Overall dev score
Baseline (sst)	0.010	0.476	0.506	0.289	0.424
Baseline (all)	0.010	0.378	0.692	0.483	0.518
Task Embedding at Input Layer	0.000	0.483	**0.706**	**0.627**	**0.605**
	0.001	**0.484**	0.688	0.575	0.582
	0.010	0.444	0.647	0.568	0.553
Task Embedding at Attention Layer	0.000	0.469	0.699	**0.574**	0.581
	0.001	**0.510**	0.698	0.552	**0.587**
	0.010	0.445	**0.707**	0.515	0.556
Projected Task-embedded Attention	0.000	**0.480**	0.709	0.590	**0.593**
	0.001	0.442	**0.710**	**0.596**	0.583
	0.010	0.441	0.647	0.489	0.526
Task-embedded Attention with ELU	0.000	**0.392**	0.615	**0.511**	0.506
	0.001	0.374	**0.683**	0.489	**0.515**
	0.010	0.289	0.662	0.421	0.457
Task-embedded Attention with Tanh	0.000	**0.366**	0.603	0.411	0.460
	0.001	**0.366**	**0.692**	**0.444**	**0.501**
	0.010	0.318	0.632	0.367	0.439
Task-embedded Attention with SeLU	0.000	0.369	**0.694**	**0.543**	**0.535**
	0.001	**0.390**	0.667	0.449	0.502
	0.010	0.330	0.668	0.358	0.452
Test leaderboard	0.000	**0.486**	**0.705**	**0.623**	**0.605**

Barplot 4 compares task embeddings at input layer model accuracies obtained under task-specific layers structure against under task-sharable layers structure.

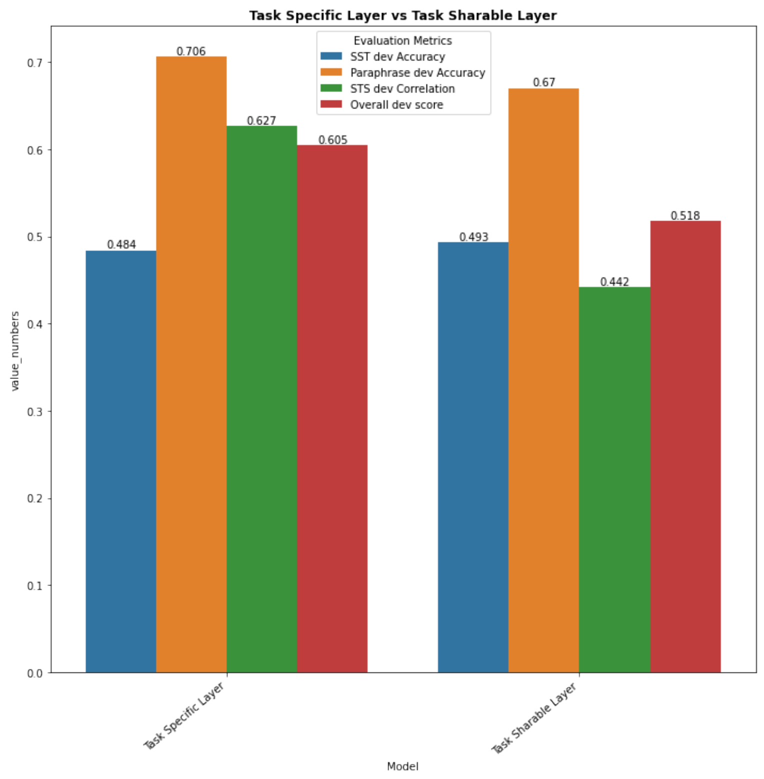

Figure 4: Task-Specific Layer VS Task-Sharable Layer under the Best Performance Model

6 Analysis

As shown in barplot 4, although reducing the amounts of added parameters, task-sharable layers structure failed to do as well as task-specific layers structure under the task embeddings at input layer model, especially for the semantic textual similarity task (0.442 correlation coefficient against 0.605). This is probably because the task-sharable layers concatenated the sentence-pair embeddings for projection instead of calculating the cosine similarity between sentence-pair embeddings, which is more suitable for semantic textual similarity task.

Comparing the baseline model against models with task information incorporated, we can see that the inclusion of task information is helpful for improving the performance. Task embeddings at input layer was found to be the approach that produced the highest accuracy, while the best accuracy scores for task embedding at attention layer and projected task embedded attention was slightly lower (<0.02). However, the accuracies for task embedded attention with activation are significantly lower (>0.05). This might be due to lacking reasoning on why activation layers are needed and how specific activation functions are chosen. It does not really make sense why negative values of $\hat{Q}$, $\hat{K}$ and $\hat{V}$ should be treated differently or why the input to scaled dot-product attention should be a value between -1 and 1. Moreover, task embeddings at input layer adds the least amounts of extra parameters (in total # Tasks $*$ Hidden Size many) compared with task embedding at attention layers adding three times more parameters and projected task embedded attention adding exponential more (in total Hidden Size^2 many).

We also found that using larger weight decays inhibits the performance of the model. This might be because we are using a large training data set and our model is not complex enough for overfitting to happen.

7 Conclusion

In this project, we tested different approaches to integrate task information to the vanilla BERT Model for the purpose of multitasking. We found that injecting task information at the input layer gives the best accuracy and adding activation functions impedes performance. We also found out that larger weight decay rates for AdamW optimizer lower the accuracies.

For future work, we would like to test our model on SuperGLUE dataset, which contains more diverse tasks such as coreference resolution and question answering tasks. We think that the tasks used for this project are too similar, thus the minBERT is already performing relatively well. With more diverse tasks, we are expecting a big improvement in accuracy after integrating task information.

References

Zihang Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2017. Quora question pairs.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Weizhu Chen Xiaodong Liu Jianfeng Gao Tuo Zhao Haoming Jiang, Pengcheng He. 2013. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization.

Łukasz Maziarka and Tomasz Danel. 2021. Multitask learning using bert with task-embedded attention. In *2021 International Joint Conference on Neural Networks (IJCNN)*.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *CoRR*, abs/2005.00247.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jean Wu Jason Chuang Christopher Manning Andrew Ng Richard Socher, Alex Perelygin and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.