Investigate multitask Performance of minBERT Ensemble

Stanford CS224N Default Project

Cheng Chang
Institute for Computational and Mathematical Engineering
Stanford University
chc012@stanford.edu

Abstract

The pre-trained Bidirectional Encoder Representations from Transformers (BERT) is used extensively to construct models with impressive performance for various downstream language tasks. Since these models exhibit promising results, it would be interesting to investigate whether a BERT-based model can be adapted to perform well in a multitask setting. We conduct experiments on whether ensemble learning can be applied to train a BERT-based model on 3 classification tasks simultaneously: sentiment classification, paraphrase detection, and semantic similarity evaluation. We compare different ways of utilizing the sub-model outputs within the ensemble, including majority voting, linear regression of prediction output (stacking), and combining task-specific and generalized BERT embeddings (4-BERT), among others. We find that ensemble with voting and 4-BERT increase overall model performance. Specifically, ensemble with voting provides superior classification accuracy for the sentiment classification task, while 4-BERT triumphs in paraphrase detection and semantic evaluation. 4-BERT test results suggest that combining domain-specific and generalized embeddings may increase model robustness over unseen datasets.

1 Introduction

1.1 BERT

The Bidirectional Encoder Representations from Transformers (BERT) is a language model trained over a large corpus of texts to acquire general knowledge about language and the world (Devlin et al., 2018). By utilizing the BERT sentence embeddings the encode this understanding, BERT-based models fine-tuned over specific objectives show promising results over a multitude of different downstream language tasks. For example, the original BERT paper demonstrates that we can construct models with state-of-the-art performance on evaluation systems such as GLUE and SQuAD 1.1 by simply adding tasks-specific output layers and fine-tuning over their respective datasets (Devlin et al., 2018). Inspired by these results, later papers such as Alberti et al. (2019) use BERT to raise the baseline of many language tasks, including question answering, sentiment classification, and Winograd sentence inference, to name a few.

The main reason for the domain-agnostic success of BERT appears to be the general knowledge it encodes during pre-training. Hypothetically, this means that BERT-based models may also perform well in multitask settings. However, the optimal parameters for different tasks, especially tasks with no obvious correlations with each other, may be very different. Therefore, it would be interesting to investigate whether we can use some common model augmentation techniques to construct a BERT-based model that, after fine-tuning over a diverse set of language tasks, increases the performance of all of the tasks.

1.2 Ensemble Learning

One technique that may be helpful is ensemble learning, which uses the prediction results of a group of weaker sub-models to generate better predictions. There are several established methods of creating ensembles; here we list two approaches that are relevant to this paper.

- Hard voting: gather the label predictions of each model and select the majority result as our final prediction.
- Soft voting: the average of the predicted values is used as the final output
- Stacking: gather the label predictions of the sub-models for the training dataset and use them as data to train a "meta-model" for the final output.

Previous papers that discuss BERT ensembles either fine-tune over one specific task (Dang et al., 2020) or tasks that are correlated with each (Kim et al., 2019) to boost domain-specific model performance. Though none of them attempts to create BERT ensembles for multitask learning, the great results that many of these models achieve give rise to hopes that similar performance improvement is possible in the multitask setting.

1.3 Multitask Experiments

In this paper, we investigate whether an ensemble of BERT can perform well over competing prediction objectives. Due to the limit of computational resources, we use the minBERT[1] variant of BERT in our implementation. The 3 tasks we want to optimize simultaneously over are

- Sentiment classification: given a movie comment, predict whether it is positive, negative, or neutral, on a scale of 1 to 5;
- Paraphrase detection: given two sentences, predict if they are paraphrases of each other;
- semantic similarity prediction; given two sentences, provide the degree of equivalence of their meaning on a scale of 0 to 5.

There is no obvious correlation between the 3 tasks. Therefore, when fine-tuning minBERT parameters over the 3 tasks, the update direction is likely conflicting. Conversely, the generalized nature of BERT suggests that the minBERT components within the ensemble may benefit from further training over ex-domain data, so it may also be beneficial to allow data from the other two tasks to influence the parameter update of each task in some way.

To find the right balance between these two ideas, we train and compare several BERT ensembles with distinct internal structures and methods of combining sub-component outputs. The main approaches we attempt are voting, stacking, and direct use of specialized and generalized minBERT ensembles (the 4-BERT model). We also compared these models with the output of individual minBERT-based models fine-tuned over each task (the 3-BERT model). Our results suggest that the ensemble with voting and 4-BERT achieve better results than the baseline, which is a simple minBERT-based model fine-tuned on all available data. HV achieves the best result for sentiment classification, while 4-BERT has the highest accuracy for paraphrase detection and semantic similarity prediction.

2 Related Work

Here we discuss several papers relevant to our experiment.

- Dang et al. (2020) uses embeddings from BERT-large and Bio+Clinical BERT, a BERT model fine-tuned on biomedical text data, to create a 10-fold ensemble to detect medication-mentioning tweets. It compares hard voting and soft voting for its final prediction. This suggests that task-specific performance can be improved by combining the output of domain-dependent and domain-independent BERT embeddings.
- Kim et al. (2019) uses word-level translation quality data to fine-tune BERT on sentence-level quality classification. It then combines 5 BERT-based models with a performance-based

[1] https://github.com/neubig/minbert-assignment

aggregation scheme to create the final output. This shows that training over data that are highly related but not specific to the language task we are fine-tuning over is helpful as an augmentation.

- Mnassri et al. (2022) integrates 3 models with distinct architectures with BERT embeddings and compares many methods of taking the vote on the final output, including hard voting, soft voting, and stacking. We draw the idea of stacking from this paper.

While we are interested in some aspects of all of the papers mentioned, it is unknown whether their findings related to BERT-based ensembles can be generalized to multitask learning with minBERT. We try to investigate this potential generalization in this paper.

3 Approach

Again, the 3 language tasks that we optimize our model performance over are sentiment classification, paraphrase detection, and semantic similarity prediction. We first construct the transformer within minBERT, the AdamW optimizer for training our model, and the baseline multitask classifiers using the pre-trained minBERT. We implement these components mainly with PyTorch (Paszke et al., 2019). The transformer architecture that we implement follows the one described in Devlin et al. (2018). Our AdamW Optimizer follows the algorithm detailed in Kingma and Ba (2014). Since these systems are architectures are well-known, we refer our readers to these papers instead of describing them here. We then constructed several ensembles of the baseline models with different internal structures to boost the performance of the combined multitask model.

3.1 Baseline

For the baseline multitask classifier, we use 3 prediction functions, each for a specific task.

- For sentiment classification, we stack a dropout and then a linear layer over the minBERT embedding output layer. The linear layers output five logistic values, which are used to calculate the possibility that the sentiment of an input sentence belongs to one of the five categories.
- For paraphrase detection, we use two Dropout and two linear layers with shared parameters to convert the minBERT embeddings of two sentences into two new "paraphrase" embeddings. We then use the cosine similarity between the paraphrase embeddings as the output.
- For semantic similarity prediction, we use the same approach as paraphrase detection (with two separate linear layers), except that the cosine similarity output is scaled up by five times to output a score between 0 and 5.

The **baseline classifier** is trained over each task-specific dataset. Figure 1 shows this simple multi-head architecture.

3.2 Ensemble

Now we describe our BERT ensemble architecture.

Weak classifiers We train 3 multitask classifiers specialized in each task with the same architecture as the baseline model. In total, this would yield nine different minBERT-based classifiers. We call these the weak (sentiment, paraphrase, or semantic similarity) classifiers. For each weak classifier, we generate 3 training datasets for it, each for a task. We test two strategies for generating this customized training data:

1. Random samples without replacement of 90% of the training dataset pertaining to the specialized task of the weak classifiers; 2000 random samples without replacement from the other two training datasets.
2. Bootstrapping (random sampling with replacement) with a ratio of 1 from the training dataset of the specialized task; 1000 random samples without replacement from the other two training datasets.

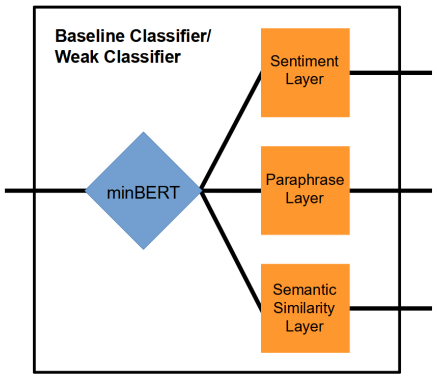

Figure 1: Architecture of the baseline/weak classifier.

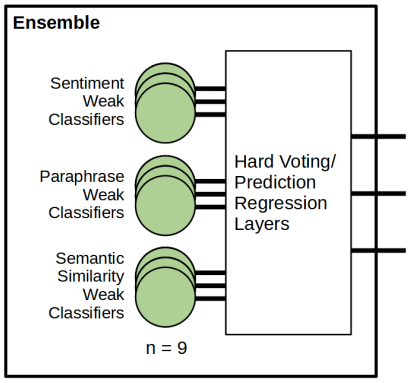

Figure 2: Architecture of the minBERT ensemble.

Strategy	Sentiment	Paraphrase	Semantic Similarity
1	**0.507**	0.486	**0.738**
2	0.493	**0.591**	0.710

Table 1: Average prediction accuracy of weak classifiers trained using the two strategies.

The intuition behind the two strategies is that they represent two levels of the amount of ex-domain data the specialized weak classifiers are allowed to see during training.

After we attain a total of 18 different weak classifiers, we evaluate them in a group of 3 using their average accuracy score over the development datasets of their specialized tasks. The performance of the batches of the weak sentiment and semantic similarity classifiers is better with training strategy 1, while the paraphrase batch that was trained using strategy 2 performs significantly better. We believe this is because our paraphrase training dataset is the largest among the three, so it can still perform well with bootstrapped data. This result also suggests that the update direction during fine-tuning for paraphrase detection may be more dissimilar from those of sentiment classification and semantic similarity prediction. We show the results of this batch evaluation of weak classifiers in Table 1. Based on these evaluation results, we select 3 batches, each for a task, to construct our ensemble.

Voting Our ensemble with voting performs a majority vote on the predictions of the specialized weak classifiers to decide the final predictions. If there is a tie, the ensemble selects the first prediction it sees. In the case of semantic similarity where the output is continuous, the soft voting method described in Section 1.2 is used.

Stacking We try the stacking approach over the same selected weak classifiers. For each task, we collect the output of all classifiers regardless of their specialties, concatenate them into a 1-dimensional tensor, and feed it into a linear layer. We scale the output of the weak sentiment classifiers by passing it through a sigmoid function before concatenation to avoid drastic changes in the logistic values. The ensemble architecture is visualized in Figure 2.

4-BERT Lastly, after experimenting with the ensembles, we hypothesize that by providing the output layers of each task with both domain-specific and domain-independent embeddings, we can achieve better prediction accuracy. We create a new model that directly utilizes 4 fine-tuned minBERT embeddings. Three of these minBERT sub-components are only fine-tuned over one specific task, while the last minBERT is fine-tuned over all of the training data. For each task, the embeddings of the task-specific minBERT and the generalized minBERT are concatenated, which are then used directly for prediction. Furthermore, while the output layer of sentiment and semantic similarity remains the same, we experiment with a new linear regression approach to the paraphrase output layer. We subtract one embedding from the other and take this difference as the input to a linear layer, which outputs in the paraphrase detection prediction. The resulting model, called 4-BERT, is shown in Figure 3.

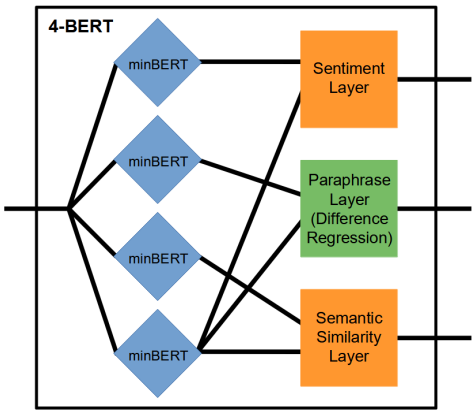

Figure 3: Architecture of the 4-BERT classifier.

4 Experiments

4.1 Data

The dataset we used for fine-tuning the BERT-based models are:

- **Stanford Sentiment Treebank (SST)** dataset for sentiment classification. It contains 11,855 single sentences extracted from movie reviews categorized into five sentiment classes: negative, somewhat negative, neutral, somewhat positive, and positive. we use a split of 8,544 training examples, 1,101 development examples, and 2,210 test examples (Socher et al., 2013).
- **Quora** dataset for paraphrase detection. Each sample is a pair of sentences with a binary label of whether they are paraphrases of each other. we use 141,506 examples for training, 20,215 for development, and 40,431 for testing[2].
- **SemEval STS Benchmark (STS)** dataset for semantic similarity. It consists of sentence pairs of varied similarities on a scale of 0 (unrelated) to 5 (equivalent). we use 6,041 examples for training, 864 for development, and 1,726 for testing (Agirre et al., 2013).

4.2 Evaluation methods

For sentiment classification outputs, we find the category with the highest logistic value and use it as the predicted label. For paraphrase detection, we convert the outputs to logistic values and evaluate values over 0.5 as 1, and value less than or equal to 0.5 as 0. The main evaluation metric is prediction accuracy for both of the tasks. For semantic similarity, we calculate the Pearson correlation coefficient of outputs and ground truth and use it as the evaluation metric. This is done because the label is continuous for the STS data set.

4.3 Experimental details

We train or attempt to train the models with specifications in Table 2. All training is done on an Nvidia A10G GPU for 10 epochs using the AdamW optimizer described in Section 3. All models contain dropout layers with a dropout rate of 0.1 in the minBERT architecture. Unless otherwise specified, all models are allowed to see the full training data of all tasks in the sequence {SST, Quora, STS} during each epoch. To offer a higher-level baseline, we train a pseudo-multitask classifier that has separate substructures for each task. We call this model 3-BERT and it is constructed by simply removing the shared BERT embedding of within 4-BERT. Details of model architectures can be found in Section 3.

[2]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model name	Description	Learning rate	Batch size	time
baseline-pretrained	Baseline model that only trains the output layers	0.001	8	about 2 hours
baseline-finetune	Baseline model fine-tuned over all training data	0.00001	16	about 3 hours
weak-classifier	Each trained with custom data described in Section 3.2; data of specialized tasks are seen first	0.00001	32	2 to 3 hours each, for a total of about 40 hours
ensemble-stack	Ensemble model with stacking	0.00001/ 0.0001/ 0.001/ 0.1	16	about 4 hours
3-BERT	A pseudo-multitask model with three independent minBERT ensemble underneath; this is used as a high standard baseline	0.00001	16	about 4 hours
4-BERT	Ensemble model that directly uses task-specific and generalized minBERT embeddings, Quora data are seen first during each epoch	0.00001	16	about 4 hours

Table 2: Training specifications.

Note that, since the ensemble model with voting (ensemble-voting) uses the output of weak classifiers directly, it does not need to be trained in addition to the weak classifiers.

4.4 Results

We use the project leaderboard of Stanford CS224N, Natural Language Processing with Deep Learning, Winter 2023, to check model performance. The accuracy of each model on the development and test datasets is listed in Table 3 and 4. The absolute best performance is marked in blue; the best performance from multitask models is bolded. Due to the limitation in submission count for the test dataset leaderboard, no test results are generated for models that perform consistently worse than baseline-finetune.

As expected, fine-tuning minBERT ensembles over the training data improves overall performance. Ensemble with voting also boosts the performance of weak/baseline models. One surprising finding is that, even for the best ensemble-stack model we can find through learning rate tuning, it performs much worse than almost any other model, including baseline-finetune. For the multitask models we test, ensemble-voting consistently achieves the best sentiment classification results over all other models, including the separate training baseline (3-BERT). While 3-BERT has slightly better results than 4-BERT for all tasks evaluated on the development datasets, 4-BERT slightly surpasses 3-BERT for all tasks on the test sets. The overall best model according to the test data set is 4-BERT.

We now draw some conclusions from these results.

5 Analysis

There are three points we would like to comment on.

Unexpected performance of ensemble-stacking We believe that the low accuracy of the model output suggests that stacking is not a good approach for constructing minBERT ensembles in the multitask setting. This may be due to the setup of the model; while the non-specialized sub-models have not seen much data from the task-specific domain, they may nevertheless generate strong

Model name	Average	sentiment (SST)	paraphrase (Quora)	Semantic (STS)
baseline-pretrained	0.370	0.384	0.396	0.330
baseline-finetune	0.577	0.485	0.474	0.772
ensemble-stack	0.418	0.291	0.445	0.518
ensemble-voting	0.627	**0.520**	0.590	0.771
3-BERT	0.642	0.508	0.603	0.815
4-BERT	**0.638**	0.502	**0.601**	**0.812**

Table 3: Development dataset accuracy.

Model name	Average	sentiment (SST)	paraphrase (Quora)	Semantic (STS)
baseline-finetune	0.587	0.515	0.471	0.775
ensemble-voting	0.626	**0.530**	0.589	0.758
3-BERT	0.640	0.526	0.602	0.792
4-BERT	**0.643**	0.527	**0.607**	**0.797**

Table 4: Test dataset accuracy.

"opinions" over samples based on their parameters. Therefore, it may be hard for the ensemble to understand which model is more confident about their output and hence cannot reach a good linear approximation from these outputs for the final output. Furthermore, the prediction outputs used to train the final meta-learner can be considered as word embeddings themselves. With the extremely low dimensionality of these "embeddings", a lot of information is lost. This result prompts us to create 4-BERT, which combines minBERT embeddings directly for classifications.

Comparison between ensemble-voting and 4-BERT While ensemble-voting is consistent in its sentiment classification performance across datasets, 4-BERT achieves good results for all tasks and has the higher average performance across the three tasks. Furthermore, training ensemble-voting is very time-consuming and computation-heavy (about 20 hours on 1 GPU), while it takes about 4 hours to train a 4-BERT model in 10 epochs. We think this means that 4-BERT is the overall better model. Based on the results of ensemble-stacking, we think this is because 4-BERT better utilizes the combined result of minBERT models. By accessing a combination of minBERT embeddings directly, 4-BERT gains more consistent and a larger amount of information from its sub-components. Its output layers can therefore be better trained.

Comparison between 3-BERT and 4-BERT As expected, without interference from the other two task-specific update directions, 3-BERT achieves better results than most of the other models. However, 4-BERT appears to exhibit competitive results in both the development and test datasets. It also consistently surpasses the performance of 3-BERT for the test data, albeit only slightly. We think this suggests that at the very least, proper construction of the minBERT-based model would allow a multitask model to achieve comparative results as independent models. 4-BERT's performance on the test dataset also suggests that, by using a shared generalized minBERT embedding to train the output layers of the ensemble, the model is more robust towards unseen data.

6 Conclusion

We develop several minBERT ensembles for multitask learning. We compare the performance of ensembles with voting, stacking, and direct usage of combined minBERT embeddings. While we think the 4-BERT model achieves overall better results, the performance improvement from 3-BERT is too small to draw a conclusive understanding of their differences. Therefore, for future work, we would like to further compare the performance of 4-BERT and 3-BERT for unseen data. We can also try new methods of constructing and training the weak classifiers to improve the overall performance of the ensembles.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions.

Huong Dang, Kahyun Lee, Sam Henry, and Özlem Uzuner. 2020. Ensemble BERT for classifying medication-mentioning tweets. In *Proceedings of the Fifth Social Media Mining for Health Applications Workshop & Shared Task*, pages 37–41, Barcelona, Spain (Online). Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Hyun Kim, Joon-Ho Lim, Hyun-Ki Kim, and Seung-Hoon Na. 2019. QE BERT: Bilingual BERT using multi-task learning for neural quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 85–89, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Khouloud Mnassri, Praboda Rajapaksha, Reza Farahbakhsh, and Noel Crespi. 2022. Bert-based ensemble approaches for hate speech detection.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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.