

Multilingual Customized Bert for Zero Shot Sequence Classification

Stanford CS224N Custom Project

Yuehao Wu

Department of Computer Science
Stanford University
yuehaowu@stanford.edu

Abstract

Training one language model to understand sequence in multiple languages has been proven feasible. One approach is Zero Shot learning described in Pires et al. [2019], which works well on low resource languages and does not depends on Machine Translation. Multilingual Bert(henceforth M-Bert) by Devlin et al. [2018] has reported solid result on XNLI data set(Conneau et al. [2018]) on 6 languages. In this paper I introduced a tailored approach by leveraging more hidden states in M-Bert, and a training strategy by dynamically freezing part of transformer architecture when fine-tuning. The result shows uplift on XNLI test set on all 6 languages.

1 Approach

1.1 Introduction

The common approaches of Multilingual Classification includes Translate Train, Translate Test and Zero Shot learning. **Translate Train** means that the training set was machine translated from English into the foreign language. So training and evaluation were both done in the foreign language. This approach requires a Machine Translation from English to foreign languages. **Translate Test** means that the test set was machine translated from the foreign language into English. So training and evaluation were both done on English. This approach requires Machine Translation from foreign languages to English. The result of both Translate Train and Translate Test heavily depends on the performance of the Machine Translation system. The **Zero Shot** means using English training set and foreign language test set directly. There is no Machine Translation used in the end to end process. In the report published by M-Bert, All 3 approaches achieved better result than original XNLI Baseline¹. In this paper, I will focus on Zero Shot approach since it's has less resources required and does not depend on NMT system's performance.

BERT has provided a classification layer which uses [CLS] token as the input to an dense layer plus tanh activation, followed by a softmax calculation. It leverages the hidden states on [CLS] token on the last layer, and uses one linear layer to make the classification. This may not use all the information captured by the model. Actually the semantics information which is important to multiple language understanding is spread across the entire model as mentioned in Tenney et al. [2019]. **I extracted all hidden states of transformer layers and applied pooling on them in order to capture more useful information for classification.**

Besides that, I have observed during the fine-tuning, after the first couple of epochs, continue to train with more epochs can bring up English score but non-English scores will start degrading. According to Hao et al. [2019] who reports the lower layers are invariant during fine-tuning, certain lower layers may be freezed when non-English scores achieved a good result. **I introduced an approach to**

¹<https://github.com/google-research/bert/blob/master/multilingual.md>

dynamically freeze certain layers during the training. In this way, we will be able to train more epochs in order to push English score higher and at the same time keep non-English scores stable.

In order to search for an ideal hidden states that has most meaning information for sequence classification I designed a phased approach. In phase one I use a smaller data set and run them through M-Bert with out fine-tune to get all the hidden states. I apply max pooing, avg pooling and simply select [CLS] token state respectively and save the output on disk. And then the problem has been simplified to an traditional classification problem with hundreds features. For the classification excises, I did grid search on 250 combination of 8 layers from 5th to 12th, 3 pooling strategies, and train 5 models for each configuration. This adds up to 3825 models in the search space and takes 10 hours to finish on 3 Nvidia V100 GPUs. From the search result I found the most informative layers are 6th, 7th, 8th and 11th, and the best pooling strategy is max pooling.

In phase two, I extended HuggingFace ² pytorch code to create a customized M-Bert Sequence Classification Model, which selects few well performing layers of hidden states according to the result of phase one. I apply Zero Shot fine-tuning on this model and freeze the lower layers on the go when non-English scores hitting a high record. This step takes 6 hours for each model to finish on a Nvidia V100 GPU.

The end to end process is shown as following. I name it Multilingual Customized Bert, henceforth MC-Bert.

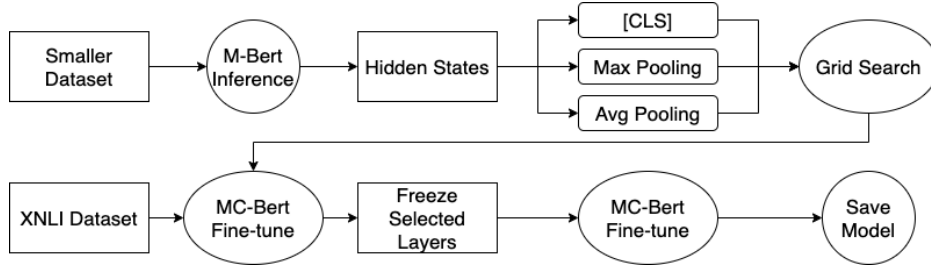


Figure 1: Approach of building Multilingual Customized Bert model

1.2 Baseline

I'm using the Zero Shot XNLI baseline published by M-Bert. In this baseline it provides evaluation result of accuracy on XNLI test set in English, Chinese, Spanish, German, Arabic, and Urdu. Together with this baseline data, It also compared with Translate Train and Translate Test on the same data. It shows Zero Shot score is slightly lower than the other two, and Translate Train Cased has the best overall scores. As a reference, I included this result in the comparison as well.

1.3 Dataset

The data set I used in phase 1 is an in-house data set which is much smaller than XNLI. The original data is a list of English utterances labeled by human. The labels are intents of the customer contact, i.e. what is the customer looking for. (for example, looking for refund status, want to speak to a real agent, etc.) It is collected from an in-house transcripts of customer conversation data on chat bot. I selected 4 top intents which has more data than the others, and split them into training set and validation set. From the validation set I applied google NMT to translate them into 4 different languages(de,es,fr,zh) and use them as test set. However, because the training data are imbalanced in terms of their class, which makes the classifier always return one intent that is majority in the data set, I augmented the data by duplicating data in other classes to achieve a balance.

Finally here is the distribution of my data

In phase 2, to evaluate this approach, I use the XNLI dataset, which is a version of MultiNLI created by Williams et al. [2017], where the dev and test sets have been translated (by humans)

²<https://github.com/huggingface>

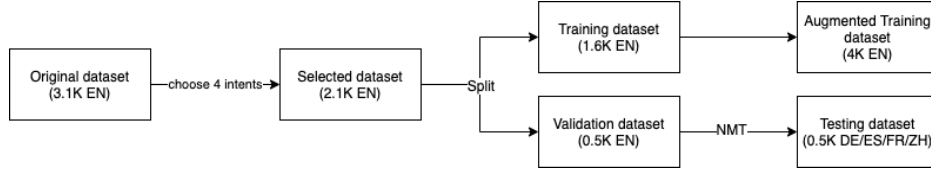


Figure 2: data preparation for smaller data set used in phase one

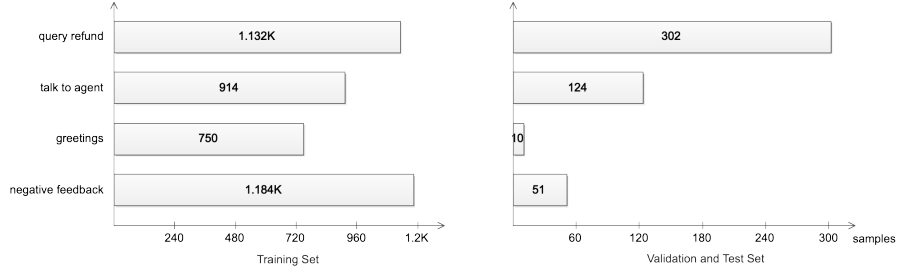


Figure 3: data distribution of smaller data set used in phase one

into 15 languages, including low-resource languages such as Swahili and Urdu. I use the original MultiNLI training set in English and test set provided by XNLI in other 6 languages for evaluation. Machine translation was not involved at all steps.

1.4 Architecture

The M-Bert has 12 transformer layers, each of which represents some semantic information of the input sentence. The states of one layer has a shape of (sequence length, hidden size), in order to use them for classification, I applied pooling strategy on top of each layer to convert them to an one dimension tensor, which later on will be feed into the input layer of classifier. The architecture is shown in figure below.

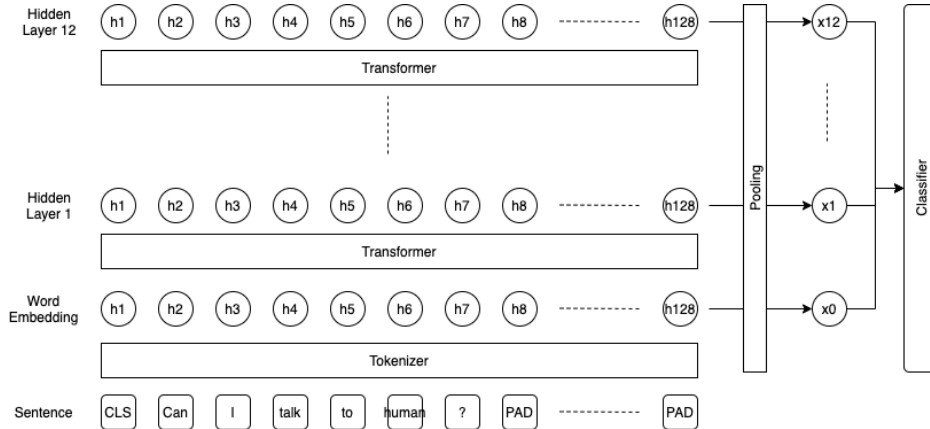


Figure 4: model architecture

1.5 Evaluation method

This is a multi-class classification problem, and it has test set in multiple languages. To be able to quickly select model architecture in phase one, I use the F1 score as the single metrics to direct my grid search. I generated confusion matrix for each language, on which I compute the Micro F1

score, which calculates metrics globally by counting the total true positives, false negatives and false positives as following,

$$Precision_t = \frac{TP_t}{TP_t + FP_t}$$

$$Recall_t = \frac{TP_t}{TP_t + FN_t}$$

$$F1_t = \frac{2 \times Precision_t \times Recall_t}{Precision_t + Recall_t}$$

and then I average the 4 F1 scores as the single metrics,

$$F1 = \frac{F1_{de} + F1_{es} + F1_{fr} + F1_{zh}}{4}$$

In phase two, I calculate the accuracy of 6 languages in order to do apple to apple compare with the baseline which only published accuracy data.

1.6 Experimental details

The best result I have found so far has following configurations: lr=2e-5, dropout=0.1, Adam optimizer, using max pooling of 6th, 8th and 11th layers concatenated, freezing 9th, 11th and 12th layers after 2 epochs.

Note that generally to push the English result higher the other languages' will drop. In my approach I tried to maintain the English result to be not lower than baseline meanwhile maximize the result of other languages. I monitored couple of experimental runs to find out the trending-down point of Non-English scores, and freeze selected layers of transformer architecture to keep the slope complanate.

1.7 Results

The first row is Translate Train result I use it as a reference. The second row is the actual baseline of Zero Shot. Both of them are published by M-Bert.³ The third row is MC-Bert without weights freeze. It shows better result on all languages except Urdu. The last row is MC-Bert with dynamic weights freeze. It is better than baseline on all languages tested. Note that the most uplift is 2.8 on Chinese test set. The English result has 0.7 above the baseline and is also better than Translate Train by 0.2.

Table 1: model result.

System	English	Chinese	Spanish	German	Arabic	Urdu
BERT - Translate Train Cased	81.9	76.6	77.8	75.9	70.7	61.6
BERT - Zero Shot Uncased(Baseline)	81.4	63.8	74.3	70.5	62.1	58.3
MC-BERT - Zero Shot Uncased	81.8	65.3	74.7	70.6	62.4	57.8
MC-BERT-WF - Zero Shot Uncased	82.1	66.6	74.4	71.8	64.3	59

1.8 Discussion

I captured the training progress and test set accuracy using tensoboard. The result has been uploaded to tensorboard.dev.⁴ The chart below shows test set accuracy of 6 languages(Arabic, German, English, Spanish, Urdu, Chinese) during fine-tune, the red line is the Zero Shot baseline I reproduced using the default configuration from M-Bert. The green line is MC-Bert without dynamic weights

³<https://github.com/google-research/bert/blob/master/multilingual.md>

⁴<https://tensorboard.dev/experiment/J50xkKHOQhGFA3nYslVYfA/>

freeze. The blue line is MC-Bert with dynamic weights freeze. Note that all of them start trending down after 5 epochs but slope of the MC-Bert with weights freeze is less steep than the other two. This is because some of the lower layers are frozen after 2 epochs. The blue line has a jump from 2nd epoch to 3rd epoch, which happened after weights freeze. It means after lower layers are frozen the model can still converge and make improvements. The last chart is Acc/Train, we can see from 2nd to 3rd epoch the blue line is going up faster than the other two, meaning after weights freeze it converges even faster. But after 3rd epoch it becomes much slower on learning from training set, this may help on generalization as the blue line shows less fluctuations than the other two after 3rd epoch.

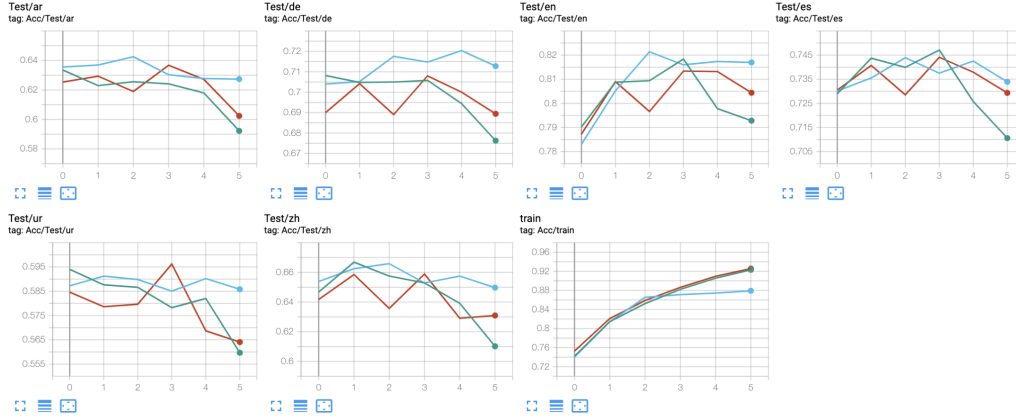


Figure 5: phase 2 training progress. Red is baseline. Green is MC-Bert without dynamic weights freeze. Blue is MC-Bert with Dynamic weights freeze.

1.9 Conclusion

1. Fine-tuning on pre-trained model can achieve a nice result, however more epochs may result in performance degradation. This is because fine tuning on one language may break the existing relationship of cross-language word embedding. In my excise I have seen drop after 5 epochs using XNLI data set.
2. Building a proper model using more information in hidden states can achieve better result than classification tokens provided in original model.
3. Max pooling generally performs better than Avg pooling and [CLS] tokens.
4. Not all layers are required. And it's not the lower the better. On my dataset the best F1 score achieved by a single layer is 8th layer. This validated conclusion in Liu et al. [2019] who reports transformers' middle layers are most transferable
5. Combining multiple layers may give an uplift. In my case the best combination I have found is 6th, 8th, and 11th.
6. freeze some of the weights at the lower layers on the go can give an uplift and it makes the model more stable. In my experiments the best option is freezing layer 9th, 11th and 12th after 2 epochs on XNLI data set.
7. Dropout is necessary to keep the model from overfitting on the training data. In this task it's more important because you only have training data in one language, overfitting on that language data will result in poor performance on other languages.
8. Deeper network does not help because the language information required for classification has been well extracted as features in hidden states.

2 Future work

1. replicate the approach on other NLP tasks such as QA, NER, NMT, etc.
2. replicate the approach on other transformer based multilingual models such as NLM.
3. excise on more low resource languages

References

- A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov. Xnli: Evaluating cross-lingual sentence representations, 2018.
- J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.
- Y. Hao, L. Dong, F. Wei, and K. Xu. Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4143–4152, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1424. URL <https://www.aclweb.org/anthology/D19-1424>.
- N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith. Linguistic knowledge and transferability of contextual representations, 2019.
- T. Pires, E. Schlinger, and D. Garrette. How multilingual is multilingual bert?, 2019.
- I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline, 2019.
- A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference, 2017.