Building Question Classifiers with Transfer Learning NLP models

Zibai Wang
Department of Computer Science
University of Waterloo
Waterloo ON Canada
zibai.wang@uwaterloo.ca

ABSTRACT

Traditional deep learning NLP models have achieved great results on my many tasks, but these models are generally trained from scratch which requires a large amount of training dataset and takes days to converge. Inductive transferring learning solved this problem and has greatly impacted natural language processing since 2018 [3]. In this paper, I give a brief introduction to a few recently developed state-of-the-art transferring learning methods and assess their performance using a traditional question classification task. My obtained results demonstrate that these transferring learning models are very efficient for the selected downstream task.

CCS CONCEPTS

•Information systems → Information retrieval → Document representation

KEYWORDS

Transfer Learning, NLP, ULMFiT, BERT, BERT-AS-Service, Universal Sentence Encoder, Question Classifiers

1 Introduction

Natural language processing with deep learning models has been a very powerful tool for a lot of real-world applications, however, there are tasks which don't have enough training data and result in poor model generalization. Transfer learning is the technique to solve this problem by allowing researchers to build a pre-trained model of one task and use it for others. [8]

The concept of transfer learning has been around in the machine learning field for many years. For example, most of the applied computer vision(CV) models are fine-tuned from models that have been pre-trained on ImageNET, MS-COCO, and other datasets [3, 9, 10, 11, 12]. However, inductive transfer learning for NLP models has not been very successful prior to 2018 [3,13]. In January 2018, Howard and Ruder proposed Universal Language Model Fine-tuning (ULMFiT), which was the first method (to my knowledge) that can be used to achieve CV-like transfer learning for any NLP tasks and it outperformed the state-of-the-art results at that time significantly [3]. After Howard and Ruder's work, there were a few other successful models that specifically target transfer learning to other NLP tasks invented, for example, Universal-Sentence-Encoder (USE) and BERT. Now you can see they show state-of-the-art performance and are used as a benchmark in many NLP tasks [14]. In the below sections, I will start to introduce you to three widely used transfer learning models.

1.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a method of pre-training language representations developed and published by Googlein October 2018. Google trained a general-purpose model on a large text corpus for language understanding purposes and the model can be fine-tuned on downstream NLP tasks. It is the first **unsupervised and deeply bidirectional contextual** representation system for pre-training NLP.

The meaning of **unsupervised** here refers to the model being trained on plain text corpus with no label. **Contextual** representation here means the same word would have different representations in different contexts, versus **context-free** representation, always having the same representation for the same word, so "bank" would have the same representation in "bank deposit" and "river bank".

BERT was built upon other pre-training contextual representations, for example, ULMFit, which we will discuss soon. One of the major differences between BERT and other models is that BERT uses bidirectional representation, but other models are either unidirectional or shallowly bidirectional. [2] For example, the unidirectional representation of "bank" in the sentence "I made a bank deposit" is only based on "I made a", in contrast, the bidirectional representation is based on "I made a … deposit".

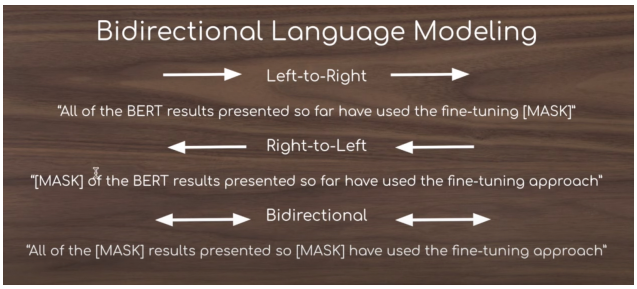

Figure 1: the difference between unidirectional and bidirectional [23]

To build this representation, BERT masks out 15% of the words from the text corpus, runs the entire sequence through a deep bidi-

rectional Transformer encoder, and then predicts the masked words. Additionally, BERT was also trained to learn relationships between sentences.

There are two steps to use BERT. First is the Pre-training, which is an expensive task (4 days on Cloud TPUs). However, most NLP researchers can just download the provided pre-trained model from google. The second step is the Fine-tuning, the BERT model is first initialized with the pre-trained parameters, and **all of the parameters** are fine-tuned using labeled data from the downstream tasks. [1] After fine-tuning the BERT model, it can create state-of-the-art models with just one additional output layer, such as a text classifier.

1.2 BERT-AS-SERVICE

Bert-as-service uses BERT as a sentence encoder and hosts it as a service via ZeroMQ, allowing you to map sentences into fixed-length representations in just two lines of code. [5] It's very convenient to use bert-as-service to get BERT encoding, especially when you want to test the basic performance of the pre-trained model during the initial model selections stage. Bert-as-service takes care of a lot of details for you, for example, you don't need to worry about tokenizing your sentence, breaking words into word pieces, etc.

Figure 2: an example of transforming sentences into BERT encoding using BERT-AS-SERVICE

1.3 ULMFit

ULMFit, which stands for Universal Language Model Fine-tuning for Text Classification, was released earlier than BERT by fast.ai in January 2018. It was the first method that can be used to achieve Computer-Vision-like transfer learning for any task for NLP [3]. They proposed discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing which are novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning. Similar to BERT, ULMFit uses context-aware presentation and also requires pre-training and fine-tuning on the downstream tasks. The release pre-trained model was trained on Wikipedia data. The fine-tuning process includes two parts: target task language-model fine-tuning and target task classifier fine-tuning.

1.4 Universal Sentence Encoder

Universal Sentence Encoder was published by Google in March 2018. They released two versions of sentence encoder, one was trained using Transformers, the other one was trained on Deep Averaging Network. Two variants of the encoding models allow for trade-offs between accuracy and compute resources. [4] The pre-trained model was trained on various unsupervised web data (Wikipedia, web news, etc) and supervised data from Stanford Natural Language Inference corpus.

Similar to BERT and ULMFit, Universal sentence encoder pre-trained model can be fine-tuned for specific tasks using gradient-based updates. However, I have noticed some differences. In the USE paper, the authors specifically mentioned: "When included **within larger models**, the sentence encoding models can be fine-tuned ..." It seems to indicate that it is only recommended to fine-tune the pre-trained model when your downstream task has a large amount of data. In contrast, in the BERT and ULMFit paper, the authors give strong recommendations to fine-tune the pre-trained model on almost any downstream tasks and also provide technical details on how to fine-tune the pre-trained model, but universal sentence encoder paper has no information related to fine-tuning. It looks to me that in a lot of use cases the universal sentence encoder pre-trained model can be used as-is without fine-tuning.

Alias	Context	Model	Input	Tasks
BERT	Bidirectional	Transformer	WordPiece	Mask LM + Sentence predictions
ULMFit	Unidirectional	LSTM	Word	Casual LM
USE	Unidirectional	Transformer/DAN	Word	SkipThought task + conversational input-response task + classification

Table 1: A comparison of three models

2 Experiment

In this section, I am going to build a few text classifiers using the above-discussed techniques and evaluate them on a Question Classification task.

2.1 Question Classification (QC) task

The definition of QC here is the task that, given a question, maps it to one of k classes, which provide a semantic constraint on the sought-after answer [15]. QC is an important element of Question Answering (QA) tasks, and interest in QA has grown dramatically over the past couple of years [17].

2.2 Dataset

The dataset I use is the same data Li and Roth used in their original research [15]. Data were collected from four sources, about 4500 English questions published by USC [16], 894 TREC 8 and TREC 9 questions, and 500 questions from TREC 10 which serves as the test set. All questions were manually labeled by human annotators as part of Li and Roth's research. The questions

are all fact-based, divided into semantic categories and there are a total of 50 different fine classes.

Question	QType
How did serfdom develop in and then leave Russ...	DESC:manner
What films featured the character Popeye Doyle...	ENTY:cremat
How can I find a list of celebrities ' real na...	DESC:manner
What fowl grabs the spotlight after the Chines...	ENTY:animal

Figure 2: sample questions and labels

2.3 Baseline

Previous work on QC can be divided into three categories: a) machine learning model b) purely rule-based technique c) hybrid of the two [17].

The current state-of-the-art machine learning model has achieved 91.6% accuracy in fine-grained classification by Van-Tu and Anh-Cuong (2016), whose work was based on using semantic features in a linear SVM [18].

The current state-of-the-art rule-based model has achieved 97.2% accuracy in the same dataset by Madabushi and Lee (2016), whose work was based on a rule-based system [17].

2.4 My implementations

To avoid bias, all hyperparameters were determined using cross-validation within training data. Test data was not involved in any training process.

2.4.1 Rule-based model

I attempted to reproduce Madabushi and Lee's paper [17] result by leveraging the API [19] provided by the authors. The API takes a question as an input and returns the predicted label. The API provided is a modified version of what they used to calculate accuracy in the paper. I reverted all the changes based on their documented modification but the accuracy I got in my implementation was only 88.2%, as opposed to the 97.2% result in the paper. I suspect they did not document all of their modifications on their website.

2.4.2 Pre-trained BERT model:

I start with building a simple classification model using below two steps. First, transform each question sentence to a fixed-length vector using a pre-trained BERT model. In this step, I leveraged Bert-As-Service to do the encoding job for me. Second, create a TensorFlow DNNClassifier with one layer 512 hidden units using these fixed length vectors as input features and the question class as labels.

I experimented with two pre-trained BERT models, BERT-Base-Uncased and BERT-Large-Uncased. The best accuracy I achieved using base and large models were 86.2% and 86.4%, respectively.

2.4.3 Fine-tuned BERT model

Since the pre-trained BERT model gave reasonable results, I tried to boost the accuracy by fine-tuning the pre-trained BERT model on my task data using the below steps. Most of the code I used was following these two examples [21, 22] online. I found it's easier to use the BERT library code directly from google repository instead of using Bert-as-service when you need to fine-tune the model.

First, preprocess input sentences and labels so that it matches the data BERT was originally trained on. This step contains a few sub-steps, including lowercase the text (since we are using a lowercase model), tokenization, break words into WordPieces, map words to indexes using a vocabulary file BERT provides, adding "CLS" and "SEP" tokens and appending "index" and "segment" tokens to each input. Now I have converted my input example to features that BERT understands.

Second, build a model by first loading the BERT pre-trained model and set the trainable = True so the weights in the pre-trained model can be tuned using our new data and add a single output layer that will be trained to classify our questions.

I ran into Out-Of-Memory error when fine-tuning using a BERT large model on Google Colab with GPU enabled. It seems like I need to leverage TPU to fine-tune BERT large model. Fine-tuning BERT base model boosted the accuracy to 92.6%.

2.4.4 Universal Sentence Encoder model:

Similar to the pre-trained BERT model, the universal sentence encoder model was built using the following two steps. First, transform each question sentence to a fixed-length vector using a pre-trained universal sentence encoder model from TF HUB. (I used DAN version) Second, create a TensorFlow DNNClassifier with one layer 512 hidden units using these fixed-length vectors as input features and the question class as labels.

The accuracy of this pre-trained model was 84.8%. I also tried to fine-tune the pre-trained model by setting trainable = True and unfroze all layers, however, the accuracy dropped to 83%. I think the reason might be the training data is not big enough, which caused the embedding weights no longer represented the language model trained on diverse data, instead, they converged to the ideal representation of the training dataset, and the model overfitted on the training set. There is no clear documentation about how to fine-tune the pre-trained model, it also could be my way of finetuning is just incorrect. Further research on how to fine-tune universal sentence encoder is needed.

2.4.5 ULMFit:

Fast.ai provides clear instructions about how to fine-tune ULMFit model on task data. First, I fine-tuned the pre-trained language model (Wikitext-103 version) on the question data and saved the fine-tuned language encoder as QC-language-model. Second, I leveraged this QC-language-model to build the final QC-Classifier.

The best accuracy I got for this model was 87.4%. Adjusting learning-rates and epoch number involved a lot of manual efforts in the fine-tuning process, I think it's possible to improve the accuracy further with more experiments.

3 Result

Study	Classifier	Accuracy
This Work	Fine-tuned BERT	**92.6%**
	Pre-trained BERT Large	86.4%
	Pre-trained BERT Base	86.2%
	Universal Sentence Encoder	84.8%
	ULMFit	87.4%
	Rule Based Model	88.2%
Van-Tu and Anh-Cuong (2016)	Linear SVM	91.6%
Madabushi and Lee (2016)	Rule Based Model	**97.2%**
Li and Roth (2002)	SNoW	84.2%

Table 2: Results from this Work and previous work that use the same Dataset.

As you see from this table, using a fine-tuned BERT model we achieved 92.6% accuracy which outperformed the previous state-of-the-art machine learning model (91.6% Van-Tu and Anh-Cuong 2016). Although it's still not as good as the previous work rule-based model 97.2%, as mentioned by Madabushi and Lee, their method has focused on a particular type of questions [17], their method may not generalize well when the dataset changes or when the type of questions are not known beforehand. I believe machine learning models are better at generalization and more practical in real-world dataset. I also think the result can be further improved with the more advanced tuning of hyperparameters.

4 Conclusion

In this paper, I discussed three state-of-the-art language models for NLP: BERT, ULMFit, and Universal Sentence Encoder. Recent researches have shown that transfer learning is an efficient and effective approach in many NLP tasks. Our empirical study on the Question Classification dataset has also shown great results. I believe these transfer learning techniques are undoubtedly a breakthrough for Natural Language Processing in a wide range of real-world applications because they don't require a large amount of training data and allow fast tuning on the target data.

There are many directions for future work. One way to extend the Question Classification work is to build a full Question Answering system (for example, TREC question answering track). I have also attempted to leverage BERT encoding to do TREC 2016 Total Recall Track which is to find all the relevant documents given a query. I started with an extremely simple approach that simply compares the vector similarity between the encoded given query and each encoded sentence in the document (the whole document is too long for BERT takes as a single input). The approach was too simple to deliver a good result. I intend to modify this approach to use DocBERT [24] which can take the whole document as input for the classification.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

[2] Google-Research. 2020. google-research/bert. Retrieved from https://github.com/google-research/bert

[3] Sebastian Ruder and Jeremy Howard. 2018. Universal Language Model Fine-tuning for Text Classification. *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia. Volume 1: Long Papers. 328–339.*

[4] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N.,Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. 2018. Universal Sentence Encoder. arXiv: 1803.11175. Retrieved from https://arxiv.org/abs/1803.11175

[5] Han Xiao. 2018. Bert-as-service. Retrieved from https://github.com/hanxiao/bert-as-service

[6] E. M. Voorhees and D. M. Tice. 1999. The TREC-8question answering track evaluation. *TextRetrieval Conference TREC-8.*

[7] Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf.2019. Transfer Learning in Natural Language Processing*. In Proceedings of the 2019Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials.* 15–18.

[8] Pratik Bhavsar. 2019. Transfer Learning In NLP. Retrieved from https://medium.com/modern-nlp/transfer-learning-in-nlp-f5035cc3f62f

[9] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition.*806–813.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015a. Fully convolutional networks for semantic segmentation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3431–3440.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[12] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. *In Proceedings of CVPR 2017.*

[13] Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution*. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*

[14] Sebastian Ruder. 2019. NLP Progress - Text classification. Retrieve from https://github.com/sebastianruder/NLP-progress/blob/master/english/text_classification.md

[15] Xin Li and Dan Roth. Learning Question Classifiers. 2002. *In Proceedings of the 19th International Conference on Computational Linguistics (COLING'02).*

[16] E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. 2001. Toward semantics-based answer pinpointing. *In Proceedings of the DARPA Human Language Technology conference (HLT). San Diego, CA.*

[17] H. Tayyar Madabushi and M. Lee. High accuracy rule-based question classification using question syntax and semantics. 2016. *In proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* 1220–1230.

[18] Nguyen Van-Tu and Le Anh-Cuong. 2016. Improving question classification by feature extraction and selection. *Indian Journal of Science and Technology* 9, 17 (2016).

[19] Harish Tayyar Madabushi. 2016. QUESTION CLASSIFICATION AT COLING. Retrieved from http://www.harishmadabushi.com/research/questionclassification/

[20] Harish Tayyar Madabushi. 2016. QUESTION CLASSIFICATION API DOCUMENTATION. Retrieved from http://www.harishmadabushi.com/research/questionclassification/question-classification-api-documentation/

[21] Google. 2019. Predicting Movie Review Sentiment with BERT on TF Hub Retrieved from https://colab.research.google.com/github/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb

[22] Aman. 2019. Question-Classification-using-BERT. Retrieved from https://github.com/amankedia/Question-Classification-using-BERT/blob/master/QuestionClassificationUsingBERT(Fine).ipynb

[23] Hendry AI Labs. 2020. Retrieved from https://www.youtube.com/watch?v=OR0wfP2FD3c

[24] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: BERT for document classification. arXiv: 1904.08398. Retrieved from https://arxiv.org/abs/1904.08398