Volume: 11 Issue: VI Month of publication: June 2023

DOI: https://doi.org/10.22214/ijraset.2023.53887

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Universal Language Model Fine-Tuning for Text Classification

R. Neha Reddy[1], R. Akshaya[2], R. Hemanth[3], R.Venkatesh[4], Prof. Sujit Das[5], Dr. Thayyaba Khatoon[6]
[1, 2, 3, 4]UG Students, [5]Assistant Professor, [6]Professor & HoD, Department of Artificial Intelligence and Machine Learning, School of Engineering, Malla Reddy University, Maisammaguda, Dulapally, Hyderabad, Telangana 500100.

Abstract: We describe approaches that are essential for fine-tuning a language model further utilized for text categorization and propose Universal Language Model Fine-tuning (ULMFiT), an efficient transfer learning method that can be used for any NLP activity. Transfer learning methods have greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications.
Universal Language Model Fine-tuning, or ULMFiT, is an architecture and transfer learning method that can be applied to NLP tasks. It involves a 3-layer architecture. Three steps make up the training process: pre-training for the general language model on a text taken from Wikipedia; fine-tuning the language model on a target task; and fine-tuning the classifier on the target task.
Deep learning techniques enable computers to learn and comprehend natural language, facilitating human-machine interaction. Deep learning models are often employed in medical research, from medication candidate identification to picture analysis.
On text classification tasks, our method greatly surpasses the state-of-the-art (it is the most recent model incorporating the best and latest technology), reducing the error on most datasets. Furthermore, with only a few labeled examples, it can match the performanceof training on 100× more data.
Keywords: NLP, ULMFiT, Fine-Tuning, Transfer learning, language model, Deep Learning.

I. INTRODUCTION

A. Problem Definition
Universal Language Model Fine-tuning for Text Classification.

B. Objective of the Project
We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses the issues of requiring large datasets, and days to converge, inductive transfer via fine-tuning has been unsuccessful for NLP and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet models: The same 3-layer LSTM architecture— with the same hyperparameters and no additions other than tuned dropout hyperparameters outperforms highly engineered models and transfer learning approaches on six widely studied text classification tasks.
Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. A language model (LM) is a probability distribution over sequences of words. This language model is used as a base model [as in Transfer learning] for various Natural language processing tasks including text classification, summarization, text generation, and more.
Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch but instead are fine-tuned from models that have been pre-trained on ImageNet, MS-COCO, and other datasets
Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection, emergency response, and commercialdocument classification, such as for legal discovery.
While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. For inductive transfer, fine-tuning pre-trained word embeddings (Mikolov et al., 2013), a simple transfer technique thatonly targets a model’s first layer, has had a large impact in practice and is used in most state-of-the-art models.

LMs make it easier to map context. Man to Woman mapping is equivalent to Uncle to Aunt and King to Queen. These mappings are converted to joint probability distribution among vocabulary. This is known as a statistical LM.
Statistical Language Modelling, or Language Modelling and LM for short, is the development of probabilistic models that are able to predict the next word in the sequence giventhe words that precede it

II. LITERATURE SURVEY

Transfer learning in the CV: Features in deep neural networks in CV have been observed to transition from general to task-specific from the first to the last layer (Yosinski et al., 2014). For this reason, most work in CV focuses on transferring the first layers of the model (Long et al., 2015b). Sharif Razavian et al. (2014) achieve state-of-the-art results using features of an ImageNet model as input to a simple classifier. In recent years, this approach has been superseded by fine-tuning either the last (Donahue et al., 2014) or several of the last layers of a pre-trained model and leaving the remaining layers frozen (Long et al., 2015a).
Hypercolumns: In NLP, only recently have methods have been proposed that go beyond transferring word embeddings. The prevailing approach is to pre-train embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers. This method is known as hypercolumns. In CV, hypercolumns have been nearly entirely superseded by end-to-end fine-tuning.
Multi-task learning: A related direction is multi-task learning (MTL) (Caruana, 1993). This is the approach taken by Rei (2017) and Liu et al. (2018) who add a language modelling objective to the model that is trained jointly with the main task model. MTL requires the tasks to be trained from scratch every time, which makes it inefficient and often requires careful weighting of the task specific objective functions (Chen et al., 2017).
Fine-tuning: Fine-tuning has been used successfully to transfer between similar tasks, for distantly supervised sentiment analysis, or MT domains but has been shown to fail between unrelated ones also fine-tune a language model, but overfit with 10k labelled examples and require millions of in-domain documents for good performance. In contrast, ULMFiT leverages general-domain pretraining and novel finetuning techniques to prevent overfitting even with only 100 labelled examples and achieve state-of-the-art results also on small datasets.
NLP Neural Networks: Many NLP tasks can be considered classification problems. A straightforward example is sentiment analysis, which can be as simple as classifying movie reviews as a positive or negative sentiment. A not-so-straightforward example is the NLP task of the next word prediction. When I want to predict the next word following ‘I like eating…’, this next word can be any word in my corpus. The way that next word prediction is to calculate the probabilities for all the unique words in my corpus, and then pick up the word with the maximum probability. SoftMax is usually used to normalize the probabilities of multiple output classes.
Over the past ten years, there have been significant advancements in the field of natural language processing, which has accelerated with the growing accessibility of deep learning. The level of competence acquired in the area of computer vision was ahead of it, though. The main reason for this was that transfer learning made many computer vision tasks much simpler than they actually were — pre-trained models like VGGNet[8] or AlexNet[9] could be fine-tuned to fit most computer vision tasks. These pre-trained models were trained on a huge corpus like ImageNet. They had been made to capture the general features and properties of images. Thus, with some tweaking, they could be used for most tasks. Hence, models did not have to be trained from scratch for each and every computer vision task. Moreover, since they were trained on a huge corpus, the accuracy or results for many tasks were exceptional and would outperform smaller models trained from scratch for the particular tasks.
On the other hand, models had to be trained separately and one by one for each NLP task. This was time-consuming and limited the scope of these models. “Recent approaches (in 2017 and 2018) that concatenate embeddings derived from other tasks with the input at different layers still train the main task model from scratch and treat pre-trained embeddings as fixed parameters, limiting their usefulness.” There was a lack of knowledge of properly fine-tuning language models for various NLP tasks.
In 2018, Howard and Ruder et. al.[1] provided a novel method for fine-tuning neural models for inductive transfer learning[1] — given a source task in which the model is trained, the same model is to be used to obtain good performance on other tasks (NLP tasks) as well.
In the simplest terms possible, the goal of the language model is to predict the subsequent word from a stream of input words. To overcome this specific issue, numerous methods have been employed in the past. Probabilistic models using Markov assumption are one example of this sort of model.

III. PROPOSED SYSTEM

A. Existing System

BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model by Google. It uses two steps, pre-training, and fine-tuning, to create state-of-the-art models for a wide range of tasks. BERT is a Machine Learning model based on transformers, i.e., attention components able to learn contextual relations between words.

Its distinctive feature is the unified architecture across different downstream tasks — what these are, we will discuss soon. That means that the same pre-trained model can be fine-tuned for a variety of final tasks that might not be similar to the task model was trained on and give close to state-of-the-art results.

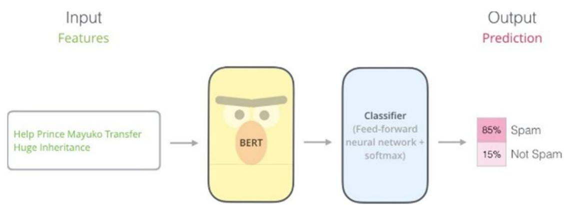

Input
Features
Help Prince Mayuko Transfer Huge Inheritance
BERT
Classifier
(Feed-forward neural network + softmax)
Output
Prediction
85% Spam
15% Not Spam

Fig 1: BERT Model

As you can see, we first train the model on the pre-training tasks simultaneously. Once the pre-training is complete, the same model can be fine-tuned for a variety of downstream tasks. Note that a separate model is fine-tuned for a specific downstream task. So single pre-trained models can generate multiple downstream task-specific models post fine tuning.

1) Fine-tuning BERT

Fine-tuning various downstream tasks is done by swapping out the appropriate inputs or outputs. In the general run of things, to train task-specific models, we add an extra output layer to existing BERT and fine-tune the resultant model — all parameters, end to end. A positive consequence of adding layers — input/output and not changing the BERT model is that only a minimal number of parameters need to be learned from scratch making the procedure fast, cost and resource-efficient.

2) Drawbacks of Existing Systems

BERT is a technology to generate “contextualized” word embeddings/vectors, which is its biggest advantage but also its biggest disadvantage as it is very compute-intensive at inference time, meaning that if you want to use it in production at scale, it can become costly.

This capability of contextualizing makes BERT compute-intensive and hard to bring intoproduction. That’s why it is powerful for a question or rewrite service but has limitations

B. Proposed System

We propose Universal Language Model Fine-tuning (ULMFiT), a method that can be used to achieve CV-like transfer learning for any task for NLP and discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing, novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning.

We significantly outperform the state-of-the-art on six representative text classification datasets, with an error reduction on the majority of datasets. Our method enables extremely sample-efficient transfer learning and performs an extensive ablation analysis.

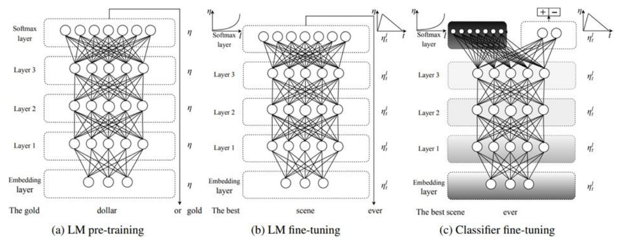


Fig 2: Stages of UMLFIT Model

Universal Language Model Fine-tuning (ULMFiT) pre trains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques. The method is universal in the sense that it meets these practical criteria:

1) It works across tasks varying in document size, number, and label type;
2) It uses a single architecture and training process;
3) It requires no custom feature engineering or preprocessing; and
4) It does not require additional in-domain documents or labels.

In our model, we use the state-of-the-art language model AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, short-cut connections, or other sophisticated additions) with various tuned dropout hyperparameters. Analogous to CV, we expect that downstream performance can be improved by using higher-performance language models in thefuture.
ULMFiT consists of the following steps, which we show in Figure 2:

a) General-domain LM pre-training

An ImageNet-like corpus for language should be large and capture the general properties of language. We pre-train the language model on Wikitext-103 (Merity et al., 2017b) consisting of 28,595 preprocessed Wikipedia articles and 103 million words. Pretraining is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples. We leave the exploration of more diverse pretraining corpora to future work but expect that they would boost performance. While this stage is the most expensive, it only needs to be performed once and improves the performance and convergence of downstream models.

b) Target task LM fine-tuning

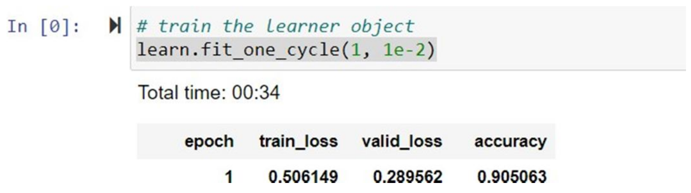
No matter how diverse the general-domain data used for pretraining is, the target task data will likely come from a different distribution. We thus fine-tune the LM on data of the target task. Given a pre-trained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows us to train a robust LM even for small datasets. We propose discriminative fine-tuning and slanted triangular learning rates for fine-tuning the LM, which we introduce in the following. Discriminative fine-tuning: Since several layers gather various sorts of information, they need to be tweaked to various degrees. To achieve this, we suggest discriminative fine-tuning, a cutting-edge method of fine-tuning.

c) Target task classifier fine-tuning

Finally, for fine-tuning the classifier, we augment the pre-trained language model with two additional linear blocks. Following standard practice for CV classifiers, each block uses batch normalization (Ioffe and Szegedy, 2015) and dropout, with ReLU activations for the intermediate layer and a SoftMax activation that outputs a probability distribution over target classes at the last layer. Note that the parameters in these task-specific classifier layers are the only ones that are learned from scratch. The first linear layer takes as the input the pooled last hidden layer states.

C. Modules

Universal Language Model Fine-tuning (ULMFiT), which pre-trains a language model(LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques.
ULMFiT consists of the following steps:



```
In [0]:   # train the Learner object
          learn.fit_one_cycle(1, 1e-2)
```

Total time: 00:34

epoch	train_loss	valid_loss	accuracy
1	0.506149	0.289562	0.905063

- General-domain LM pre-training
- Target task LM fine-tuning and
- Target task classifier fine-tuning.

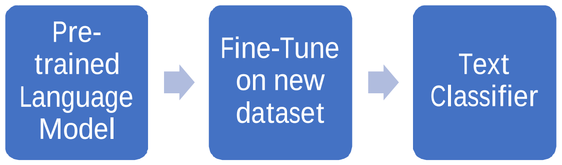


Fig 3: Steps in UMLFIT

D. Architecture

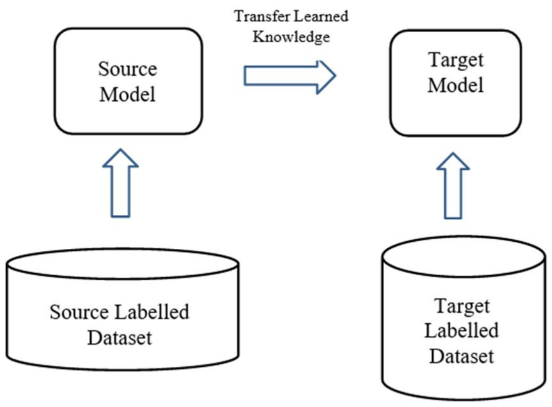


Fig 4: Architecture of UMLFIT

IV. RESULTS

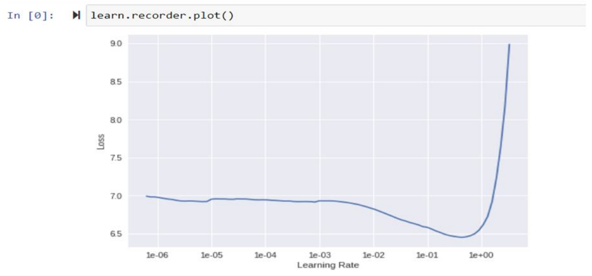
A. Fine-Tuning the Pre-Trained Model and Making Predictions

We can use the data_lm object we created earlier to fine-tune a pre-trained language model. Wecan create a learner object, 'learn', that will directly create a model, download the pre-trained weights, and be ready for fine-tuning

The one cycle and cyclic momentum allow the model to be trained on higher learning rates and converge faster**.** The one-cycle policy provides some form of regularization**.**

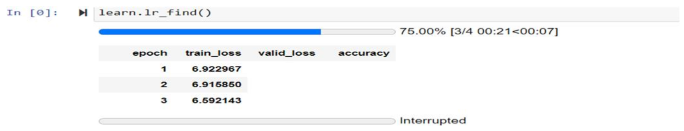
B. LR Finder is Complete

The Graph



```
In [0]:    learn.recorder.plot()
```

We will save this encoder to use it for classification later.



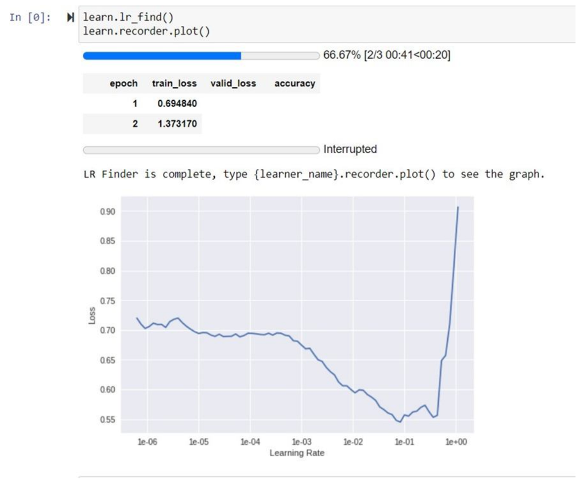
```
In [0]:    learn.lr_find()

           75.00% [3/4 00:21<00:07]

           epoch  train_loss  valid_loss  accuracy
               1    6.922967
               2    6.915850
               3    6.592143

           Interrupted
```

Let’s now use the data_clas object we created earlier to build a classifier with our fine-tunedencoder.



```
In [0]:  learn.lr_find()
         learn.recorder.plot()
```

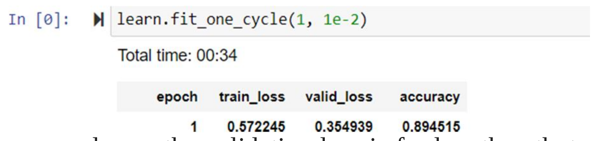
66.67% [2/3 00:41<00:20]

epoch	train_loss	valid_loss	accuracy
1	0.694840		
2	1.373170		

Interrupted

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

We will again try to fit our model.



```
In [0]:  learn.fit_one_cycle(1, 1e-2)
```

Total time: 00:34

epoch	train_loss	valid_loss	accuracy
1	0.572245	0.354939	0.894515

We got a result with increase in the accuracy and even the validation loss is far less than thetraining loss. It is a pretty outstanding performance on a small dataset.

V. CONCLUSION

We have proposed ULMFiT, an effective and extremely sample-efficient transfer learningmethod that can be applied to any NLP task. Our method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks. ULMFiT-based models (which have been pre-trained) perform very well even on small and medium datasets compared to models trained from scratch on the corresponding dataset. Thisis because they have already captured the properties of the language during pre-training. The newly proposed methods for fine-tuning the LM and the classifier prove to give better accuracy than using the traditional methods of fine-tuning a model.

While we have shown that ULMFiT can achieve state-of-the-art performance on widely used text classification tasks, we believe that language model fine-tuning will be particularly useful in the following settings compared to existing transfer learning approaches

REFERENCES

[1] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. arXivpreprintarXiv:1801.06146.

[2] Figure1:http://jalammar.github.io/images/BERT-classification-spam.png

[3] Figure2: [1801.06146] Universal Language Model Fine-tuning for Text Classification(arxiv.org)

[4] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. arXiv preprint arXiv:1611.01368 .

[5] Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). volume 1, pages 562–570

[6] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017a. Regularizing and Optimizing LSTM Language Models. arXiv preprint arXiv:1708.02182 .

[7] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv:1605.07725 .

[8] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709 .

[9] Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on. IEEE, pages 464–472.

[10] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10):1345–1359.