Natural Language Processing for Healthcare: A Case Study of MIMIC III Dataset

Neal Cheng, PhD[1], Joshua Sherfey, BS[1], Oren Tevet, MS[1], Kevin Zhu, BS[1]
[1]Georgia Institute of Technology, Atlanta, Georgia

Abstract

In this project, we demonstrate a novel method of matching semantic information from physicians' clinical notes to ICD-9 codes using natural language processing techniques, such as the Universal Language Model Fine-tuning (ULMFit) architectures using pre-trained word vectors. We observed an improvement in the F1-score over the current state-of-the-art techniques, hereby establishing this methodology as a first-in-class neural network architecture for performing multi-class classification on ICD-9 codes.

Introduction

Recently, the use of Electronic Health Records (EHRs) has grown dramatically, which now include an unprecedented amount and variety of patient information, such as demographics, vital sign measurements, laboratory test results, prescriptions, procedures performed, digitized notes, medical images as well as mortality and survival outcomes. These records normally contain both structured data, such as a patient's weight and date of birth, as well as unstructured data (clinical notes written by doctors).

Clinical notes contain rich and valuable information such as observations, diagnosis, treatment plan and follow-up assessments in the form of unstructured data. While U.S. physicians spend one-sixth of working hours on administrative work[1], the clinical notes are generally not being systematically extracted to streamline this administrative work or generate meaningful clinical insights. The main challenge lies in the difficulty in manually extracting useful information from this unstructured data due to time and labor cost in an already costly and inefficient healthcare system. If this data can be processed, we can derive new medical insights and improved medical care, such as improved disease diagnosis, personalized treatment, or a more detailed understanding of treatment outcomes on both a personal and global scale.

In 1967 the World Health Organization (WHO) created the International Classification of Diseases (ICD) system to "monitor the incidence and prevalence of diseases, observe reimbursements and resource allocation trends, and keep track of safety and quality guidelines"[2]. Currently, ICD labeling is done manually by administrative personnel based on definitions and is subject to interpretation and errors .

The work on collecting EHR data has been ongoing for around two decades, dating back to 1996 when the first version of the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC) was compiled. Subsequently, two additional versions have been published to include textual data.

Structured data within the MIMIC include past histories of medications, diagnoses, procedures, and lab tests. Numerous studies have been performed using this aspect of the data including using Support Vector Machines (SVM) and Naive Bayes to predict heart-related diseases[3] and bidirectional RNN[4] with an attention mechanism.

Despite past success, structured data does not capture the richness of information within a clinician's note, and therefore unstructured data involving textual information can be used to improve prediction. Towards the end of the last decade, many papers have cited rule-based approaches, such as using a mapping of common words to diagnosis codes.[5,6,7] These systems are able to achieve a reasonable performance producing F-measures of 0.87 - 0.88, however, their construction require a significant amount of manual labor to optimize the performance metric. The drawbacks of rule-based approaches is that 1) these system are not robust to changes in the dataset, problem statement, and therefore cannot be transferred to a lateral problem 2) do not capture complex dependencies as it only tests for inclusion/exclusion of certain vocabulary. Since then, there have been papers using machine learning to

avoid the laborious process of constructing rule-based classifiers, using algorithms such as a Decision Tree Classifier, Maximum Entropy Classifier and hierarchical Support Vector Machines [6,8]

Neural networks and natural language processing techniques have seen a significant improvement in performance in the past few years, especially for unstructured datasets such as images and text. These approaches have been applied to a large variety of datasets, including the MIMIC dataset.

The Recurrent Neural Network (RNN) architecture was designed to capture past dependencies through its internal states. However, it suffers from numerous deficiencies, such the vanishing or exploding gradient, and the inability to capture longer longer-term dependencies. Much of the subsequent work in this field is to tackle the problem of capturing the relational information between non-adjacent words. The Long Short-term Memory (LSTM) recurrent neural network had been invented to address this issue, consisting of numerous forget gates to store temporal information. Another recent neural architecture is the Grounded Recurrent Neural Network (GRNN), which employs a modified GRU with dimensions dedicated to predicting the presence of individual labels.[10]

These types of neural network architectures have been applied to the multi-label classification task for assigning ICD-9 labels to textual medical notes, with results illustrating that a RNN and a LSTM demonstrating an improvement over a Binary Logistic Regression model.[9]

However, these aforementioned recurrent models, while popular, still lack from various issues. The sequential processing of the hidden states between the recurrent cells prevents parallelization, and therefore training these models can be time-intensive and computationally-expensive.

A new type of model, known as the Transformer, is claimed to be able to remedy this problem.[11] This model applies a small, constant number of steps, which at each step, it applies a self-attention mechanism which models all relationships between words of a sentence. The attention mechanism assists in the allocation of the relative importance of other words when processing the word of interest. This architecture is a huge advance in the field of text classification by tackling the problem of polysemy, where the meaning of certain words are dependent upon the context in which it is written.

The goal of this project is to transform physicians' clinical notes in the form of unstructured text data to ICD codes as a systematic and accurate way to classify and report diseases.

Data

MIMIC-III is an open-access relational database containing tables of data relating to patients who stayed within the intensive care units (ICU) at Beth Israel Deaconess Medical Center. It contains labels of discharge notes from the MIMIC-III Database into ICD codes. This public database of EHRs contains data points on about 41,000 patients from an intensive care units between 2001 and 2012, including notes on close to 53,000 admissions.

Following previous work, we focus on discharge summaries, which condense information about a patient's stay into a single document. This data is contained in the MIMIC-III database under the 'noteevents' table. There are a total of 2,083,180 rows in this table. Each admission is tagged by human coders with a set of ICD-9 codes, describing both diagnoses and procedures which occurred during the patient's stay. There are 8,921 unique ICD-9 codes present in our datasets, including 6,918 diagnosis codes and 2,003 procedure codes. Some patients have multiple admissions and therefore multiple discharge summaries; we will split the data by patient ID, so that no patient appears in both the training and test sets.

There are a total of 47,724 training documents in the MIMIC-III dataset, with a total vocabulary size of 51,917. There is a mean of 1,485 tokens per document and a mean of 15.9 labels per document. The total number of labels is 8,922. Table below gives the patient count and hospital admissions for the top 10 ICD-9 Codes and Categories based on hospital admissions.

ICD-9 Code	Patient Count	Hospital Admissions
4019: Hypertension	17,138	20,046
4280: Congestive heart failure	9,669	12,842
42731: Atrial fibrillation	10,053	12,589
41401: Coronary atherosclerosis	10,579	12,178
5849: Acute kidney failure	7,505	8,906
25000: Diabetes Type II	7,181	8,783
2724: Hyperlipidemia	7,324	8,503
51881: Acute respiratory failure	6,493	7,249
5990: Urinary tract infection	5,687	6,442
53081: Esophageal reflux	5,148	6,154

Table 1. Patient count and hospital admissions count for the 10 most frequent ICD-9 codes, sorted by hospital admissions.

Pre-processing

In this project, the clinical notes and ICD-9 Diagnoses of 38,597 distinct adult patients from 2001 to 2012 were retrieved from the publicly available multi-parameter intelligent monitoring in intensive care database MIMIC-III (Medical Information Mart for Intensive Care III).The goal of this study is to explore useful semantic information using unstructured data, therefore only the free-text clinic note section from the dataset, specifically the noteevents table, was used. Furthermore, we focused on the discharge summaries category as it contains actual ground truth and free-text compared to other categories. Since discharge summaries were written after the diagnosis was made, the notes are sanitized by removing any mention of class-labels (ICD-9 codes). In addition, all stop words were removed from the discharge summaries. For processing ICD-9 codes, we used the pre-process method developed by Huang, J., Osorio@, C., & Sy, L. W. (2018).[12] Our approach of pre-processing is to treat the ICD-9 code independently from each other, find the admissions (unique HADM_ID) for each ICD-9 classification, and consider only records related to the top 10 and top 50 common ICD-9 codes. Top 10 and top 50 were chosen because they cover a majority of the dataset, and for ease of results comparison. Evaluation will be separately performed on the two datasets. The two datasets will hereby be referred as top-10-code, top-50-code respectively. In order to properly evaluate and iterate our model, we split our dataset into training and testing datasets with an 80%/20% distribution.

Model Selection

One fundamental assumption adopted by traditional supervised learning algorithms is that each sample has only one label assigned to it. In our problem, each sample has multiple (one or more) ICD-9 codes attached to it. Generally, there are two main methods for tackling the multi-label classification problem: (1) problem transformation methods and (2) algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification or regression problems, multiple binary classifiers are trained separately for each label. Algorithm adaptation methods adapt the algorithms to perform multi-label classification in its full form and only one classifier are trained for all the labels. We had used the latter.

Experimental Methodology

To pre-process the data, we used a local desktop with Ubuntu 18.04 LTS, we installed Python 3.6.5 and Spark 2.4.1, PySpark and toree on it using Anaconda. Once the data is pre-processed, we used Google Colaboratory to develop our models. After models are developed and tested, We moved to cloud computing resources for the bulk of our experimentation. To accomplish this, we set up three Linux-based virtual machines in Google Cloud computing environment with GPU resources. Each virtual machine has 8 vCPUs, 52 GB memory, and one NVIDIA Tesla P100 GPU, Hence, we were able simultaneously evaluate multiple models. To ensure the dependencies of these instances are consistent, each VM was provisioned using ‘pytorch-latest-gpu’ image provided by Google in their

'deeplearning-platform-release' image project. Deep learning packages such as PyTorch are preinstalled in the image. We also used fast.ai library to create the aforementioned neural network architectures.

Approach

Our approach to model-training is based on a technique known as Universal Language Model Fine-tuning (ULMFit) proposed by Howard et al.[13] Essentially, the technique involves making use of a model pre-trained upon a larger corpus and fine-tuning of the model to adapt to the corpus of interest. Lastly, model is augmented with two additional linear layers to adapt the model to our classification problem.

In our model, we used the ASGD Weight-Dropped LSTM (AWD_LSTM) architecture [14] trained on the Wikitext 103 dataset, which contains a pre-processed large subset of English Wikipedia.[15]

The encoder of the AWD_LSTM was fine-tuned using the MIMIC III corpus. The training of the encoder was heavily influenced by the work of Smith et al.[16-18] The one-cycle policy described by the author was used to determine the optimal learning rate. In brief, the methodology involves the initialization of the learning rate at a small value (approximately 10^{-6}) and increased in an incremental manner after each mini-batch. Due to the divergent behavior of high learning rates upon the loss function, a minima can be observed when plotting the learning rate vs. the loss function. The author recommends a good maximum learning rate to be 5 - 10 times slower than the learning rate specified by the minima of this function.

We used an initial maximum rate of 10^{-2} for the first layer and upon unfreezing, 10^{-3} for the rest. The learning rate is implemented in a dynamic fashion, with the first half of the training linearly increasing until the specified maximum learning rate, and decreasing afterwards. The momentum is set in an inverse manner between 0.7 and 0.8, decreasing until the minima at the halfway point, and increasing afterwards. To avoid overfitting, we used a dropout layer.

After training the encoder portion of the network, another round of training was performed with the entire network.

Metrics

To evaluate the performance of the model, we used three metrics: F_1, Precision, and Recall. We define each of the metrics as follows:

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i|+|Z_i|} \qquad Precision = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Z_i|} \qquad Recall = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

where n is the number of samples, Y is the model's prediction labels, Z is the ground truth labels.

Experimental Results

Before arriving upon our final model, it's necessary for us to determine the best method to reach an optimal model. We evaluate performance based on two metrics: 1) number of epochs necessary to reach convergence 2) the validation loss at the last epoch. Our experimental approach was largely inspired by the approach discovered by Smith.[16]

Upon determination of the maximum learning rate, we had investigated the effects of the one-cycle policy at various learning rates and compared with using a fixed learning rate.

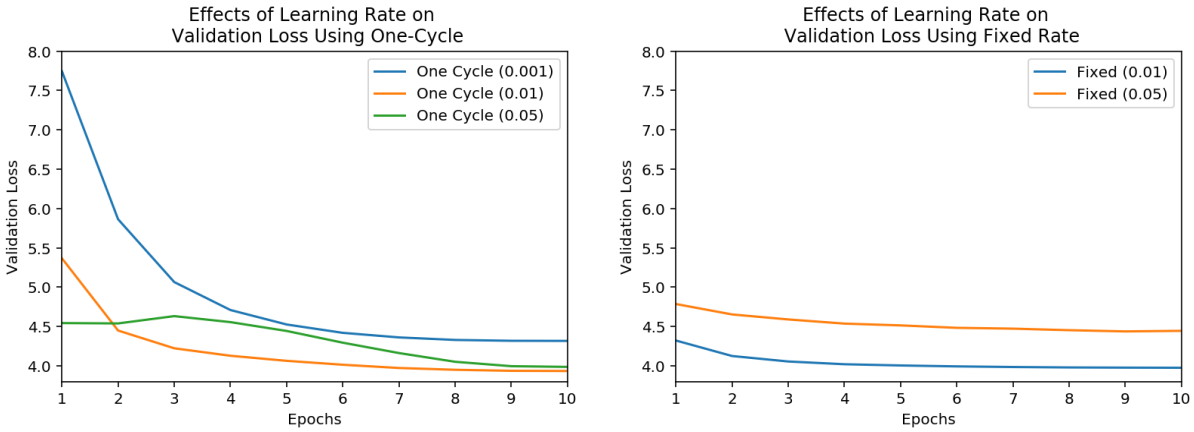

Figure 1. Effects of Learning Rate on Validation Loss (Language Model)

Figure 1 illustrates the result of varying the learning rate when fine-tuning the language model. The results show that a one-cycle learning rate of 0.01-0.05 is capable of reaching convergence within

As expected, too small of a learning rate prevents the optimizer from breaking out of a local minima whereas an overly excessive learning rate would prevent the optimizer from converging. However, when comparing between one-cycle and fixed rate, we have noticed no difference between the validation loss of the two at 0.01 at the end of training. What was even more surprising is that quicker convergence was reached with the fixed rate. We hypothesize that the pre-trained model was initialized relatively close to a "smooth" portion, therefore avoiding saddle points and local minima.

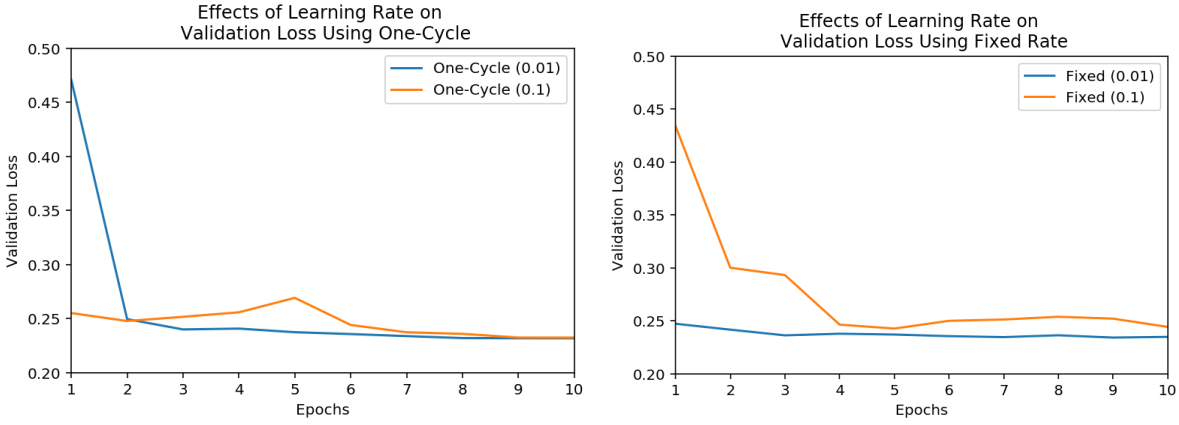

Figure 2. Effects of Learning Rate on Validation Loss (Classifier)

It can be observed from Figure 2 that a relatively stable minima can be obtained at a learning rate of 0.01 at 2 epochs for both techniques. We observe that the majority of the reduction in validation loss model training during the training of the language model rather than the classifier, which is intuitive and expected.

Evaluating the ULMFit approach results, we see that the model achieved an F1 score (micro) of 58.94%, an F1 score (weighted) of 52.64% and an ROC AUC score of 0.8830. These results are higher than previous works[12] (Huang et. al, 2018), which achieved an F1 score of 36.62% for the same task. (AUC scores for this task were not reported by the authors). The model performance for the most frequently occuring ICD-9 codes can be seen in Table 2. An an

analysis of weighted F-1 scores against number of features can be seen in Figure 1. According to the figure, the ULMFit model produced weighted F1-scores of 0.716 with the top 10 features and 0.527 with the top 50 features.

ICD-9 code	Precision	Recall	F1-score
4019: Hypertension	0.7	0.84	0.77
4280: Congestive heart failure	0.73	0.77	0.75
42731: Atrial fibrillation	0.85	0.84	0.85
41401: Coronary atherosclerosis	0.79	0.83	0.81
5849: Acute kidney failure	0.57	0.64	0.61
25000: Diabetes Type II	0.64	0.77	0.7
2724: Hyperlipidemia	0.61	0.63	0.62
51881: Acute respiratory failure	0.58	0.55	0.57
5990: Urinary tract infection	0.69	0.55	0.61
53081: Esophageal reflux	0.66	0.55	0.6

Table 2. Model performance for the 10 most common ICD-9 codes.

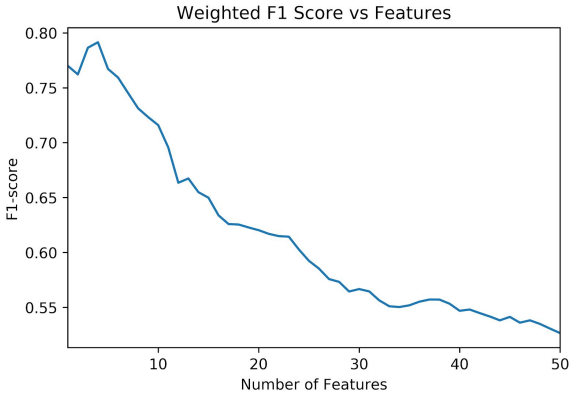

Figure 2. Weighted F1 score for the top n number of features.

Discussion

Despite the excellent results reported by FastAI on the one-cycle policy, we did not observe the phenomenon of "superconvergence" reported by Smith et al. On the contrary, a fixed learning rate performed better in certain cases. To the best of our knowledge, most training using the one-cycle policy is used upon Imagenet or CIFAR10. Given that the topology of the loss function between the two types of data can be drastically different, it can be that the current technique cannot be generalized to other forms of data, such as textual data in our case.

The ULMFit method proved to be an effective model. The weighted F1 score outperformed the GRU RNN results from a previous study[12] for 10 features (0.716 vs 0.6957) and 50 features (0.527 vs 0.326). Figure 3 shows the performance of ULMFit was marginally better than CNN, LSTM RNN, and GRU RNN for the top 10 features, and far greater than the rest for top 50 features.

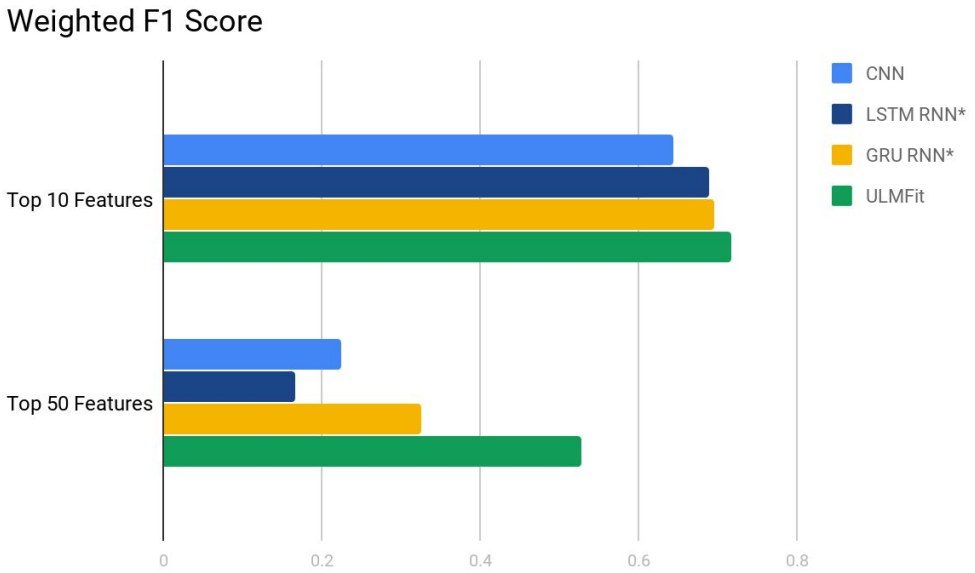

Figure 3. Weighted F1 scores for the CNN, LSTM RNN, GRU RNN, and ULMFit. *LSTM RNN and GRU RNN results are derived from previous study[12].

We considered whether the performance of the algorithm was truly class dependent, or rather due to algorithmic bias caused by the class imbalance between the labels. Figure 2 shows a graph of the F1 score as a function of the number of top codes ranked by size. We observe a monotonic decrease in the F1 score as the number of features increase, indicating poorer performance with the inclusion of minority codes, and conclude that the model performance by code is bottlenecked by the amount of data available.

In the future, we will look for ways to further tune our ULMFit model by adjusting various hyper parameters and improving our features selection techniques. Additionally, we will look to establish a more comprehensive baseline performance metric by incorporating other models (linear regression, random forest decision tree, feed-forward neural network, etc.). Moreover, we will look to improve the efficiency of our data pipeline by incorporating more PySpark into our preprocessing and modeling steps.

Conclusion

In this project, we used a novel method of matching semantic information from clinician notes to ICD-9 codes through natural language processing technologies. We used an LSTM architecture with pre-trained word vectors to improve F1-scores beyond current state-of-the-art techniques. Moreover, we utilized Google Cloud computing environment and big data technologies like Spark to increase data pipeline efficiency and reduce runtime. The project results demonstrate that this methodology is a cutting-edge neural network architecture for performing multiclass classification on ICD-9 codes.

References

1. Woolhandler S, Himmelstein DU. Administrative work consumes one-sixth of US physicians' working hours and lowers their career satisfaction. International Journal of Health Services. 2014 Oct;44(4):635-42.
2. World Health Organization. International classification of diseases (ICD). http://www. who. int/classifications/icd/en/. 2006.
3. Parthiban G, Srivatsa SK. Applying machine learning methods in diagnosing heart disease for diabetic patients. International Journal of Applied Information Systems (IJAIS). 2012 Aug;3:2249-0868.
4. Choi E, Bahadori MT, Schuetz A, Stewart WF, Sun J. Doctor ai: Predicting clinical events via recurrent neural networks. arXiv preprint arXiv:1511.05942. 2015 Nov 18.
5. Crammer K, Dredze M, Ganchev K, Talukdar PP, Carroll S. Automatic code assignment to medical text. InProceedings of the workshop on bionlp 2007: Biological, translational, and clinical language processing 2007 Jun 29 (pp. 129-136). Association for Computational Linguistics.
6. Farkas R, Szarvas G. Automatic construction of rule-based ICD-9-CM coding systems. InBMC bioinformatics 2008 Apr (Vol. 9, No. 3, p. S10). BioMed Central.
7. Goldstein I, Arzumtsyan A, Uzuner Ö. Three approaches to automatic assignment of ICD-9-CM codes to radiology reports. InAMIA Annual Symposium Proceedings 2007 (Vol. 2007, p. 279). American Medical Informatics Association.
8. Perotte A, Pivovarov R, Natarajan K, Weiskopf N, Wood F, Elhadad N. Diagnosis code assignment: models and evaluation metrics. Journal of the American Medical Informatics Association. 2013 Dec 2;21(2):231-7.
9. Nigam P. Applying deep learning to ICD-9 multi-label classification from medical records.
10. Vani A, Jernite Y, Sontag D. Grounded recurrent neural networks. arXiv preprint arXiv:1705.08557. 2017 May 23.
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In the Annual Conference on Neural Information Processing Systems (NIPS).
12. Huang J, Osorio C, Sy LW. An empirical evaluation of deep learning for ICD-9 code assignment using MIMIC-III clinical notes. arXiv preprint arXiv:1802.02311. 2018 Feb 7.
13. Howard J, Ruder S. Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146. 2018 Jan 18.
14. Merity S, Keskar NS, Socher R. Regularizing and optimizing LSTM language models. arXiv preprint arXiv:1708.02182. 2017 Aug 7.
15. Merity S, Xiong C, Bradbury J, Socher R. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843. 2016 Sep 26.
16. Smith LN. Cyclical learning rates for training neural networks. In2017 IEEE Winter Conference on Applications of Computer Vision (WACV) 2017 Mar 24 (pp. 464-472). IEEE.
17. Smith LN. A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820. 2018 Mar 26.
18. Smith LN, Topin N. Super-convergence: Very fast training of neural networks using large learning rates. arXiv preprint arXiv:1708.07120. 2017 Aug 23.