Can Ready-to-Use RNNs Generate "Good" Text Training Data?

Jia Hui Feng
Engineering, Computer Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand

***Abstract*—There is much research on the state-of-the-art techniques for generating training data through neural networks. However, many of these techniques are not easily implemented or available due to factors such as copyright of their research code. Meanwhile, there are other neural network codes currently available that are easily accessible for individuals to generate text data; this paper explores the quality of the text data generated by these ready-to-use neural networks for classification tasks. This paper's experiment showed that using the text data generated by a default configured RNN to train a classification model can match closely with baseline accuracy.**

***Keywords*—Neural networks; machine learning; text generation; classification; natural language processing; data augmentation; artificial intelligence**

I. Introduction

Training data is crucial for machine learning tasks. In cases where there is little availability of training data due to limited time and resources, it becomes difficult for machines to be able to learn.

In the field of computer vision, there have been successes and benefits from using different methods to generate new training data; this is called data augmentation. This is done by creating new training data based on the original labeled training data. An example of data augmentation in image data is manipulating labeled training data of images through transformations such as rotation and cropping. This generates new training data images for the machine to learn from.

However, in the field of natural language processing (NLP), data augmentation has been a challenging problem as there have been difficulties in establishing universal rules for transformations in textual data while maintaining the quality of the label [1]. For example, the sentence "this is good" is a statement with positive intent, but with a small transformation such as swapping the words to make it "is this good," it becomes a question with no positive intent; the meaning completely changes. There are also various studies on identifying effective ways to transform labeled textual training [1].

Another technique used to generate more training data is generative augmentation, where more new data is generated compared to switching or cropping images. There has been success in using GAN networks to generate more training data such as images of human faces [2].

Generative models have also been explored in the NLP field with existing language models [1]. There is ongoing research in generative models for both computer vision and NLP. However, for an individual who is building a machine learning system with classification tasks, these state-of-the-art techniques for generating text data can be quite difficult to implement.

Thus, this paper's intended purpose is to help individuals who want a "ready-to-use" method to generate more textual training data, by assessing the effectiveness of some of these available methods. The types of individuals that this is focused on are, for example, an entry-level data scientist who is hired in a small team for a machine learning and NLP project, a freelancer who is working on a personal project or a computer science student who is building a chatbot as part of their degree. A common task to implement is labeling data for training machine learning systems, thus classification is the task chosen for this experiment, allowing us to evaluate how "good" the generated training data is based on the accuracy results of the classification model.

There is currently much state-of-the-art research to help generate textual training data. However, much of this is not readily accessible due to copyright protection. Meanwhile, the state-of-the-art research that does have code available online is often complicated and difficult for entry level data scientists or coders to implement.

There are existing deep learning models available, such as TensorFlow [3] that are readily available for programmers to use; these are considered "ready-to-use" in this paper. There have been no studies though on the effectiveness of these neural network codes online. This research aims to perform an exploratory experiment on ready-to-use neural networks and the quality of the generated data.

II. Related Works

A. Generative Augmentation

Generative models are models that generate textual data; these models are often used in the research field in text generation. They have been explored in the use of data augmentation as they use the newly generated data as additional training data, this helps to increase the diversity of the text which can incorporate new information. Some generative models use pre-trained language models to improve their performance [1].

There are various state-of-the-art techniques, and research has been done on generative models for data augmentation. Although this is not the focus of this paper, most research on generative models in data augmentation shows minor improvements.

Rizos et al. [4] identified that one of the issues of online hate speech classification is not enough relevant training data that can be found online, and training on only a small amount of data will cause overfitting. Thus, they created an RNN that can generate data for online hate speech labeling to combat this issue. Their augmented data has improved 30% in hate speech class recall and 5.7% in macro-F1 score [4]. However, the authors state that further work can be done as the generative model was not the best performing augmentation method. In addition, it is only for short-text, and it is unknown whether the classification would be as effective for non-short text data.

Another technique, G-DAUG, was proposed by Yang et al. [5]. G-DAUG generates synthetic data using pretrained language models and then selects the most informative and diverse set of examples for augmentation. It has been shown to be effective on multiple common-sense reasoning QA setting benchmarks. However, the authors state that more improvements can be made to enhance the quality and diversity of the generated data [5].

Noise generation techniques are commonly used in generative methods as they help to solve the issue of class imbalance [1]; Qiu et al. [6] introduced a variation autoencoder (VAE) based method as a noise generation technique for text generation used in their paper. VAEs are autoencoders which transform input data into a latent representation. In their paper, however, it still only showed marginal improvements in classification tasks.

Generative Adversarial Network (GAN) is also another popular method used for data augmentation in computer vision [1]. There has been research in using GAN models for data augmentation in NLP, such as using the seqGAN architecture [7]. This is just like a GAN model, a generator, and discriminator, but used for textual data and the task of classification. There were, however, only minor improvements [1].

Many of the text generation techniques that have been explored in the research field are not readily available for the public to use. This is due to reasons such as copyright, or their research involves complex algorithms to be implemented in deep learning models, which may be time-consuming for individuals to implement. In addition, a lot of it is not for general use as their research findings are specific to a particular chosen field or topic.

Meanwhile, there has not been in-depth research on the quality of generated words from ready-to-use techniques.

III. Experiment Design

The experiment was conducted by using benchmark text classification data, AG News [8], then using a CNN built from Python as a classification model to obtain the baseline accuracy rate, then using a ready-to-use RNN [5] to generate synthetic training data. The same CNN model is then used to train and test the original dataset in combination with the new generated training data to evaluate the accuracy of the textual training data.

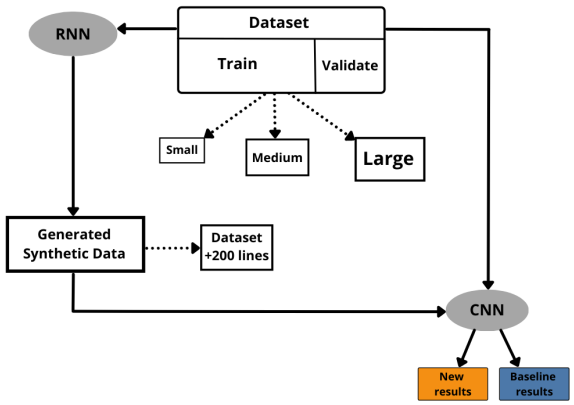

Fig. 1. Experiment Design Process.

As shown in the figure above (Fig. 1), there are three different sized datasets for this experiment. I first got the accuracy of small, medium and large baseline datasets, obtaining three baseline accuracy results. Then I used RNN to generate additional training data for each small, medium and large dataset. I then used the same CNN and tested the accuracy of the original dataset plus the newly generated synthetic training data.

A. Dataset

The dataset used was AG News which consists of four different classes, 1–4, where 1 is world, 2 is sports, 3 is business and 4 is science/technology [8].

- Dataset split ratio: 80% (training) and 20% (validating)
- Each label had an equal number of sentences
- Validating sets had a mixed number of labels

They were split into three different sized datasets:

- Size ratio: Sentences Total = Training/Validating
  a. **Small**: 50/40/10
  b. **Medium**: 500/400/100
  c. **Large**: 1000/800/200

An equal number of labels were in the training sentences. For instance, in the "medium" training dataset of 400, there were 100 (400/4) sentences from each label.

In some real case scenarios, 1,000 sentences of a total dataset split for training and testing may not be considered a "large" dataset, it is though considered "large" respective to this experiment.

The sizes of the dataset were randomly chosen as there is no "common sized" dataset used in the real world. The figure below (Fig. 2) shows the original dataset with labels that were augmented.

1,"ATHENS, Greece - Greek sprinters Kostas Kenteris and Katerina Thanou pulled out of the Athens Games on Wednesday, nearly a week after they missed a drug test and were later hospitalized following a suspicious motorcycle crash. ""I'm withdrawing from the Olympics,"" Kenteris said after meeting with the International Olympic Committee's disciplinary commission..."
4, "The chief scientist at the National Aquarium in Baltimore has launched a review of the dolphin breeding program after the death of a 4-month-old dolphin."
3,"Hewlett-Packard shares fall after disappointing third-quarter profits, while the firm warns the final quarter will also fall short of expectations."
2,"With Scott Williamson bracing for a potentially grim diagnosis of his latest elbow injury, the Red Sox last night appeared poised to move on without him as Curtis Leskanic inched closer to rejoining the club. The Sox initially indicated they would make an announcement last night on Williamson's injury but changed their position during the first winning of the game ..."

Fig. 2. Original Dataset with Label [8].

Each dataset generated 200 lines of new training data. When the CNN is trained on the new training data, it also includes the original dataset. For instance, the small dataset was fed into the RNN model and generated 200 additional lines. This small dataset along with the 200 new additional sentences were fed into the classification model to test accuracy. This was to mimic a real case scenario where someone would use their existing training data in combination with the synthetically generated data. Similar to the choice of the sizes of dataset, generating 200 additional lines was also randomly chosen.

B. RNN

The RNN model chosen was textgenrnn created by Max Woolf [9], it is a Python 3 module that is built on top of Keras/TensorFlow. This model can be configured by the size, layers and whether to use a bidirectional option. This was chosen as it is free, quick to implement and can be easily customized for an individual level.

For this experiment, the RNN model had not been configured or changed, it simply used the default configuration and settings. This was so I could see how good just the default configuration was as some individuals who are in early entry to this field may not know how to do configurations in an RNN but still want to generate text training data.

The default model takes an input that converts each character to a 100-D character embedding vector and feeds into two LSTM layers [9]. The architecture of this RNN is shown in Fig. 3.

C. Text Classification Model

The CNN was built from Python's TensorFlow [3]. The architecture was as follows: input of up to 50 words, 1D convolutional layer of 128 filters of size 5 with ReLU activation function with a softmax output layer. The metrics used to evaluate accuracy were chosen through the TensorFlow CNN's configurations which returns binary accuracy [3].

In addition, GloVe [10] was used to train the 100-dimensional word embedding.

IV. Results and Discussion

The results shown that for the small dataset the accuracy remains the same as shown in the table, at medium it becomes worse, but at the large dataset the accuracy catches up almost meeting the baseline. A visual comparison of the detailed results can be seen in Fig. 4.

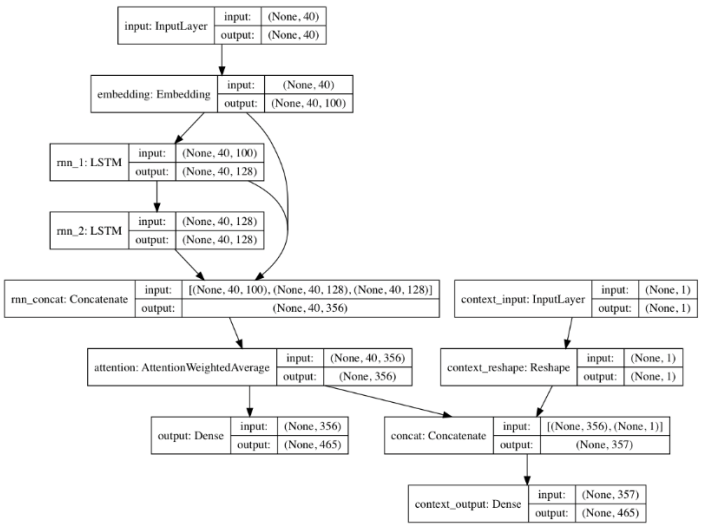

Fig. 3. Textgenrnn Architecture [9].

TABLE I. Experiment Results

Dataset	Baseline accuracy	New dataset	New dataset accuracy
Small	40%	250/40/10	40%
Medium	70%	700/400/100	60%
Large	73%	1200/800/200	72%

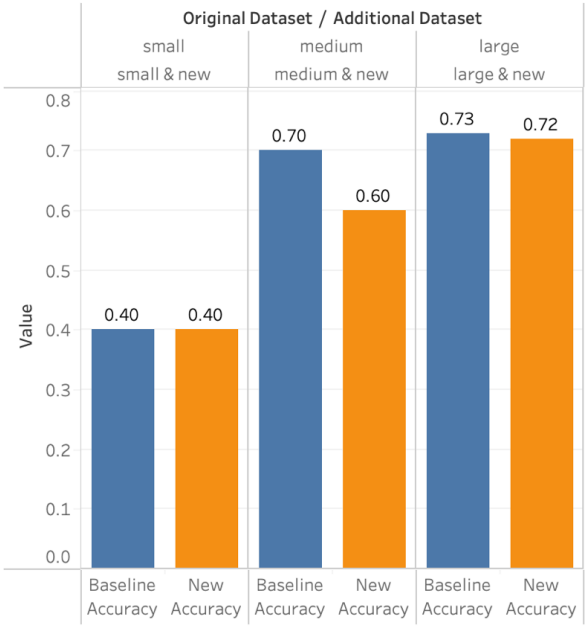

Fig. 4. Accuracy Visualization Comparison.

```
Southern Sunday hit the Brate Star victory and now skin.
```

Fig. 5. Label 2 Generated Example.

The results do not show a clear pattern, but it can be concluded (at least with this experiment) that this example

of a ready-to-use RNN of **default configurations** generates text training data that will likely closely match the baseline accuracy.

There was more investigation by looking at the generated text and a sample is given in Fig. 5. This is considered a label 2 which is "sports". At least with this example, it still seems that the generated data is still within the same topic.

In addition, the repository for textgenrnn [9] has made notes regarding the quality of the generated text data. The repository states that "results will vary greatly between datasets" [9] and that the best results are of datasets with at least 2,000–5,000 documents [9]. For smaller datasets, it is recommended to train it longer by setting *num_epochs* higher [5]. In this experiment, *num_epochs* was set at the default value of 1.

For the field of data augmentation in NLP in general, this experiment shows that more questions can be considered for generating text data, such as, "What are the characteristics of words that affect the CNN?" and "Are specific parameters of RNNs better for different types of text data?"

V. Conclusions and Future Work

Although the experiment did not show obvious patterns for different factors in generating text data, it has answered the question the paper posed: Can ready-to-use RNNs generate "good" text data? Based on this experiment and this dataset, if an individual imports and runs this example of ready-to-use RNN with default settings, the classification results will likely match close to the baseline accuracy. With some more tweaks and adjustments, it is likely that the accuracy will improve, but more research can be conducted on the quality of ready-to-use RNNs with adjusted configurations.

Thus, in the future there can be more research conducted to further investigate the quality of the textual data and into not using default configurations – instead investigating through an RNN with adjusted parameters.

In addition, since this experiment only included three different sized datasets, more research can also be done with more variety to discover more.

References

[1] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A Survey on Data Augmentation for Text Classification," p. 35.

[2] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," p. 19.

[4] G. Rizos, K. Hemker, and B. Schuller, "Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Beijing China: ACM, Nov. 2019, pp. 991–1000. [Online]. Available: https://dl.acm.org/doi/10.1145/3357384.3358040

[5] Y. Yang, C. Malaviya, J. Fernandez, S. Swayamdipta, R. L. Bras, J.-P. Wang, C. Bhagavatula, Y. Choi, and D. Downey, "Generative Data Augmentation for Commonsense Reasoning," *arXiv:2004.11546 [cs]*, Nov. 2020, arXiv: 2004.11546. [Online]. Available: http://arxiv.org/abs/2004.11546

[6] Y. L. Qiu, H. Zheng, and O. Gevaert, "Genomic data imputation with variational auto-encoders," *GigaScience*, vol. 9, no. 8, p. giaa082, Aug. 2020. [Online]. Available: https://academic.oup.com/gigascience/article/doi/10.1093/gigascience/giaa082/5881619

[7] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient," *arXiv:1609.05473 [cs]*, Aug. 2017, arXiv: 1609.05473. [Online]. Available: http://arxiv.org/abs/1609.05473

[8] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," *arXiv:1509.01626 [cs]*, Apr. 2016, arXiv: 1509.01626. [Online]. Available: http://arxiv.org/abs/1509.01626

[9] M. Woolf, "textgenrnn," *2017*. [Online]. Available: https://github.com/minimaxir/textgenrnn

[10] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. [Online]. Available: http://aclweb.org/anthology/D14-1162