Effective Text Augmentation strategy for NLP Models

December 28, 2020

Abstract

Data Augmentation is proved to be effective in Vision tasks. We propose a strategy, through which training data can be increased in a meaningful way, and there by achieve good performance. We tested four augmentation operations for transforming text (Random Swap, Random Deletion, Back-translation, Random Synonym Insertion) in two different settings on a classification model. Our paper proposes a mixed augmentation strategy which involves pre and post augmentation by leveraging all the four operations. Experimental results show that our method achieves a significant improvement on datasets of limited training data.

Keywords : Data Augmentation, sentiment analysis, Back translation, Random Swap, Random Deletion, Synonym Replacement.

1 Introduction

Nowadays, highly advanced applications in the filed of Natural Language Processing (NLP) are ubiquitous and it involves the computational processing and understanding of human languages. Incredible progress has taken place, particularly in the last few years in deep learning based NLP. The field of NLP is relied on statistics, probability, and machine learning since the 1980s, and on deep learning since 2010s[1]. Machine Learning and Deep Learning have obtained significant results on the tasks ranging from Sentiment Analysis [2] to Question Answering [3].

Even though there are many advantages from deep learning, there are also more common challenges [4] when it comes to NLP, because of the lack of theoretical foundation, lack of interpretability of the model, and the requirement of a large amount of data and powerful computing resources. High performance of any model always depends on the size and quality of the data on which the model gets trained [5]. Data Augmentation (DA) is a technique to increase the training data, it helps to boost the performance of the model. Image data augmentation is a standard practice in Computer Vision tasks and they performed remarkably well on many tasks [6, 7], whereas Text data augmentation is rare in NLP tasks [9], due to the challenges it involves. The reasons for these challenges

is coming up with rules for language transformation is not thoroughly studied and experimented. Some methods have already been proposed to increase the amount of training data using simple text transformations or text generation through language models [10]. Therefore, text data augmentation for NLP tasks becomes appealing.

In this work, we propose a text data augmentation strategy based on increasing training data before model training and augmenting the data while training the model. For implementing this strategy, we are adopting four text augmentation methods like Random Swap (RS), Random Deletion (RD), Random Synonym Insertion (RSI) and Back-translation(BT). The proposed strategy is evaluated on Apple Twitter Sentiment (ATS) Dataset[1], a dataset for sentiment classification. The results show that our approach can obtain a significant improvement when the training data is limited.

The rest of the paper is organized as follows. Section 2 addresses the previous work happened in the text augmentation area. Section 3 skims through the adopted augmentation techniques, and how well they are working through pre and post augmentation strategies, and presents the proposed approach. Section 4 explains the experimental setup along with results and analysis and it is followed by Conclusion.

2 Related Work

Previous work has proposed some text augmentation techniques. A popular study called Back-translation, can generate new data by translating sentences from one languge to another and it is an effective method for Neural Machine Translation (NMT) to improve translation quality [11]. Synonym identification and replacement [12] is another study carried out to transform sentences into another with similar meaning. Data noising is another approach studied for augmentation, it is widely adopted in application domains like vision and speech [13]. Easy data augmentation (EDA)[14] proposed four techniques to do transformations in NLP, which includes synonym replacement, random insertion, random swap, and random deletion. EDA has shown significant performace improvement over text classification tasks.

In this paper, we are proposing an approach which augments the data before training and while training on batches. The proposed method follows a combination of RS, RD, BT, and RSI methods to perform text augmentaion. We evaluated the proposed method on a classfication task, and a significant improvement is achieved on a smaller dataset. Code is publicly available here[2].

[1] https://www.kaggle.com/c/apple-computers-twitter-sentiment2

[2] https://github.com/sridevibonthu/TextAugmentation

3 Our Approach

3.1 Augmentation Techniques adopted

3.1.1 Random Swap (RS)

This approach randomly selects two words and swaps them in a training example, x, and repeats this process for n number of times to generate an augmented example, $\hat{x}$. Fig. 1 illustrates this process with an example.

$$\hat{x} = RandomSwap(x, n) \tag{1}$$

This is a very simple approach to generate new training examples from the existing. Downside of this approach is it may cause adversarial text attack to fool the model especially if the sentence has nouns. For example "Rama Killed Ravana" is completely different from "Ravana killed Rama". This technique can be adopted based on the nature of the training examples.

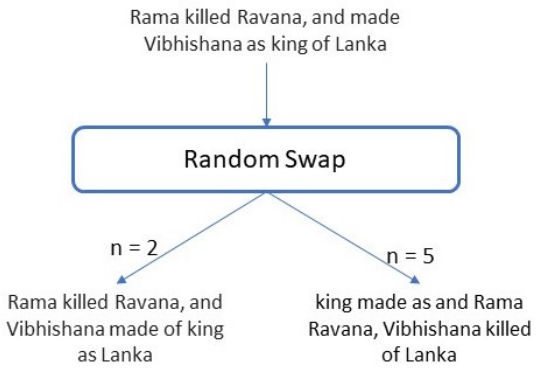


Figure (1) Random Swap operation generating two transformed examples from a single input x for the n values 2 and 5.

3.1.2 Random Deletion (RD)

This approach randomly deletes n number of words from the training example, x with a probability p and generates an augmented training example $\hat{x}$. A sample example is shown in Fig. 2. If the value of p is large, then it may result in meaningless sentences and sometimes the context may change completely.

$$\hat{x} = RandomDeletion(x, p) \tag{2}$$

3.1.3 Back Translate (BT)

This approach translates a training example, x from source language(SL) to some intermediate language (IL), and again back-translates it to source language. This technique generates synthetic data in four lines of code, but this

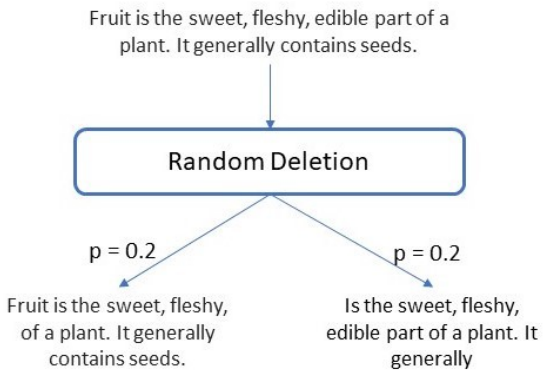


Figure (2) Random Deletion operation generating two transformed examples for a single input x, with a common probability value of 0.2.

is computationally expensive as it has to do language translation twice back to back. Fig. 3 shows two examples in which German and French are chosen as intermediate languages for translation.

$$\hat{x} = translate(translate(x, SL, IL), IL, SL) \tag{3}$$

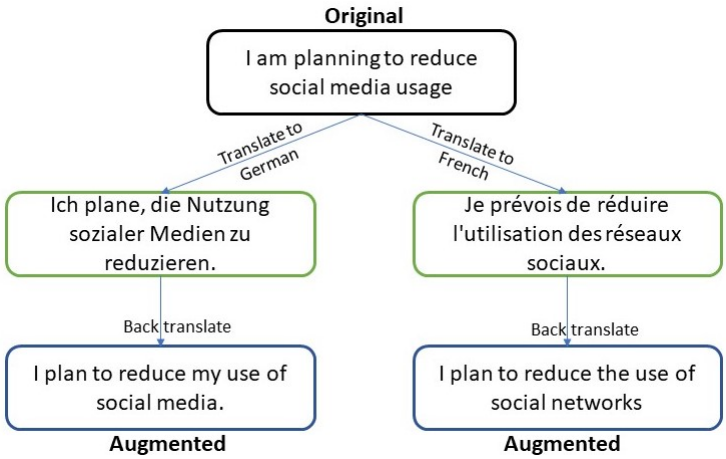


Figure (3) Back-translation operation generating two augmented examples for the same input, x, by taking two intermediate languages.

3.1.4 Random Synonym Insertion (RSI)

This approach randomly inserts synonyms of of n words, which are not stop-words in a training example, x to generate a new training example, $\hat{x}$. An

example for Random Insertion with Synonym is shown in Fig. 4. The outcome of this method depends on the value of n. The suggestable value for n can be in the range of 1 to 3.

$$\hat{x} = RandomInsertion(x, n) \tag{4}$$

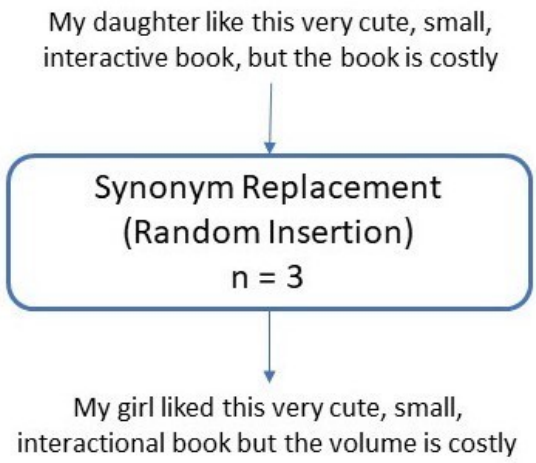


Figure (4) Random Insertion operation generating an augmented example in which *(daughter, interactive, book)* are replaced with *(girl, interactional, volume)*.

Random Insertion technique with synonym replacement can generate a new training example but it suffers with a deficiency. This may cause adversarial text attack as shown below.

input x $\rightarrow$*"True Grit" was the best movie I have seen since I was a small boy.* (Predicted as positive)

Random Insertion(x, n = 2) = Augmented $\hat{x} \rightarrow$*"True Grit" was the best movie I have seen since I was a* ***wee lad****.* predicted as negative

3.2 The Classification Model

A text classification problem can be defined as a set of training examples $D = \{x_1, x_2, ..., x_N\}$ in which every record is labelled with a class value drawn from a set of discrete class labels indexed by $1..k$ [15]. The classification model is constructed based on the training examples, and evaluated with the test set. Our paper used Recurrent Neural Network (RNN) language model based on Long Short Term Memory Network (LSTM) [16]. LSTM is better in analyzing emotion of long sentences and it is applied to achieve multi-classification for text emotional attributes [17]. This model is applied on the Apple Twitter Sentiment Dataset[3] to study the effectiveness of the selected text augmentation techniques in both the approaches and to come up with a best strategy for augmentation.

[3]https://www.kaggle.com/c/apple-computers-twitter-sentiment2

(a) Pre-Augmentation

(b) Post-Augmentation

Figure (5) Initial methods to test the adopted text augmentation strategies. (a) Approach 1 - pre-augmentation, which increases a fraction of training data. (b) Approach 2 - post-augmentation, which augments the data in the mini-batches while training.

The LSTM-RNN takes in a training example as a sequence of words, $X = x_1, x_2, .., x_T$ one a time and produces cell state, c, and hidden state, h, for each word. The network is used recurrently by feeding the current word x_t, cell state, c and hidden state, h from the previous word (c_{t-1}, h_{t-1}), to produce the next cell and hidden states, (c_t, h_t). The final hidden state, h_T obtained by sending last word in the sentence, x_T to the LSTM cell is fed through a linear layer f to get the predicted sentiment $\hat{y}$.

$$(c_t, h_t) = LSTM(x_t, h_{t-1}, c_{t-1}) \tag{5}$$

$$\hat{y} = f(h_T) \tag{6}$$

3.3 Evaluation of Augmentation Methods

The simple LSTM classification model is trained without applying any augmentation on the original data and received a baseline accuracy of 72.75%. Each of the four augmentation strategies (RS, RD, BT, RSI) were evaluated on the Apple Twitter Sentiment Dataset individually by following two approaches to understand how they are performing.

3.3.1 Approach - 1 (Pre-augmentation)

In the first approach the train set is increased by taking a fraction of the training examples, transforming using one of the augmentation technique from RS, RD, BT, and RSI (Fig. 5). Let $D : \{(x_i, y_i)\}_{i=1}^{M}$ is a set of M training examples.

$$D_{New} = D + D_{Aug} \tag{7}$$

$$D_{Aug} = T(\{(x_i, y_i)\}_{i=1}^{f.M}) \tag{8}$$

Where, T is a transformation function, which augments a fraction, f of the M training samples to form new Training set, D_{New}. The new training set will $(1 + f).M$ records after augmentation. This approach is followed for all the adopted augmentation techniques and all the methods improved the validation accuracy by 2 to 3% when compared with baseline. Fig. 5 depicts the training accuracy vs. validation accuracy for all these four experiments and it is very clear that Back translation consistently maintained good validation accuracy when compared with baseline accuracy.

3.3.2 Approach - 2 (Post-augmentation)

In the second approach the training samples in a mini-batch set at t^{th} training iteration, $D_t : \{(x_i, y_i)\}_{i=1}^{M}$ can be changed to $\hat{D}_t : \{(\hat{x_i}, y_i)\}_{i=1}^{M}$, by applying the augmentation techniques when they are fed into the LSTM network (Fig. 5). This process repeats for every batch of every epoch of the training process. In this approach, the model encounters plenty of augmented training examples. Let e be the number of epochs, and b be the number of batches and m, the number of training samples in every batch, and if the augmentation happens randomly for 50% of the training samples, then the overall augmented training samples seen by the model in the training phase are $e * b * (0.5 * m)$.

The augmentation techniques adopted to test this approach are RS and RD only. The reason for not adopting BT, RSI is they can work in sentence level, but not on token level, and in training the sentence is available in numerical format only. Fig. 8 depicts the training accuracy vs. validation accuracy for these two experiments. Both the methods helped improve validation accuracy, and Random Deletion also reduced overfitting.

3.4 Proposed Approach

By examining the performance of the text augmentation techniques adopted in the above two approaches, we have come up with a mixed augmentation startegy, in which a fraction of the original training data is transformed by using RS, RD, BT, RSI by following a randomized algorithm called *preAugment(x)* and again randomly applying transformation on batches by using RS and RD by following *postAugment(x)* algorithm. In this approach, Fig. 6, there is a chance to apply augmentation on the augmented text i.e, Random Swap operation may happen on the back-translated text.

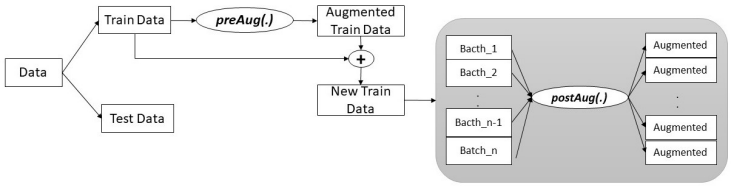


Figure (6) proposed method. Augmentation happens twice with preAug(.) and postAug(.) methods.

Algorithm 1: Pre-Augmentation(x)

Result: Transformed Exampmle $\hat{x}$ for the Training Example x
rate := getRandom(0,1) ; `// returns a number between 0 and 1`
if *rate* < 0.3 **then**
  $\hat{x} = RandomInsertion(x, n)$;
else
  if *rate* < 0.6 **then**
    $\hat{x} = translate(translate(x, SL, IL), IL, SL)$;
  else
    if *rate* < 0.8 **then**
      $\hat{x} = RandomDeletion(x, p)$;
    else
      $\hat{x} = RandomSwap(x, n)$;
    end
  end
end

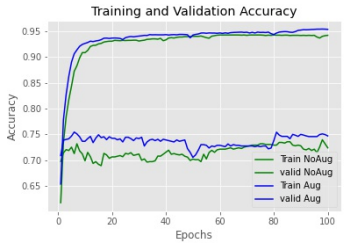
Algorithm 2: Post-Augmentation(x)

Result: Transformed Exampmle $\hat{x}$ for the Training Example x
rate := getRandom(0,1) ; `// returns a number between 0 and 1`
if *rate* < 0.2 **then**
  $\hat{x} = RandomSwap(x, n)$;
else
  if *rate* < 0.6 **then**
    $\hat{x} = RandomDeletion(x, p)$;
  else
    $\hat{x} = x$
  end
end

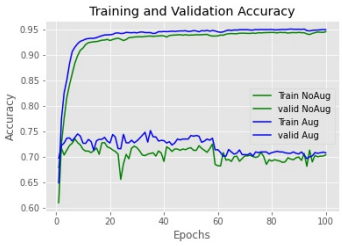
4 Experiment

4.1 Data

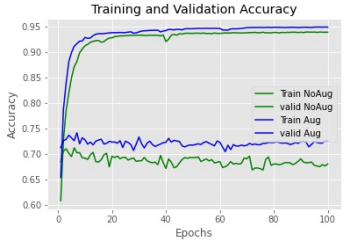
For our experiment, we use the Apple Twitter Sentiment dataset provided by kaggle for a competition called inclass prediction. This dataset is suitable for the experiment as we need to test augmentation strategy in limited data settings. ATS dataset contains 3886 records in which 82 are not relevant. The tweets can be either positive, negative or neutral. The original training records we adopted for experimentation with class labels were provided in the bar chart at Fig. 9. 80% of the data is taken as training data and the rest as validation data to perform the experiment.
@ mentions, #hashtag, RT (Retweet), hyperlinks were removed as part of preprocessing the data, as the adopted data is from twitter.



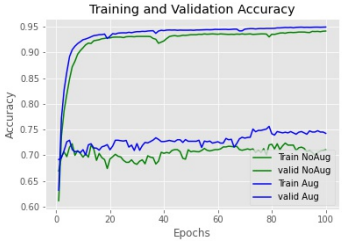
(a) Random Swap



(b) Random Deletion

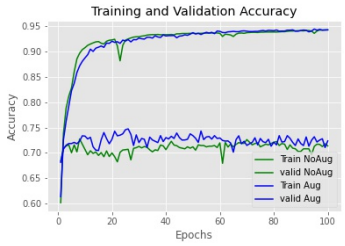


(c) Back translation

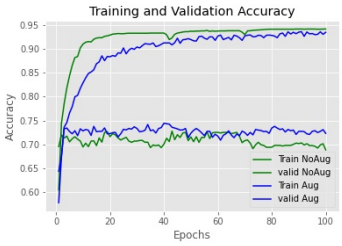


(d) Random Synonym Insertion

Figure (7) Training vs. Validation accuracy by following Approach 1 (pre-augmentation) with RS, RD, BT, RSI.



(a) Random Swap



(b) Random Deletion

Figure (8) Training vs. Validation accuracy by following Approach 2 (post-augmentation) with RS, RD.

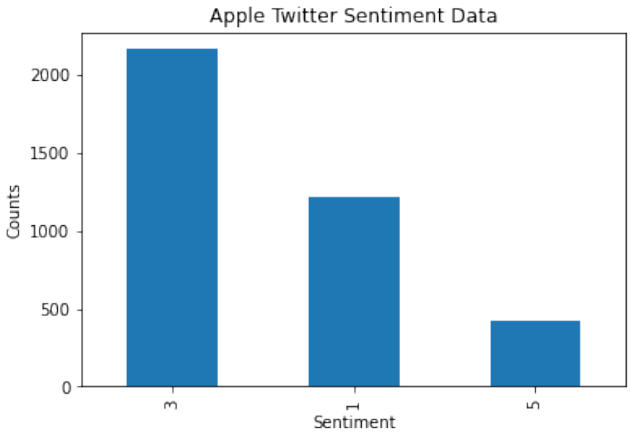


Figure (9) Class-wise training examples in ATS Dataset (1 - Negative, 3 - Neutral, 5 - Positive)

4.2 Experimental Setup

The Data was tokenized with the help of *spacy*[18] tokenizer. *TorchText*[4] library is utilized to complete the work. This library is part of PyTorch project, which contains data processing utilities and popular datasets for Natural Language Processing.

A simple classification model based on LSTM is adopted and same hyper-parameters are used for all the **8** experimentations, baseline without augmentations(1), preaugmentation approach (Fig. 5) for RS, RD, BT, RSI techniques (4), postaugmentation on batches (Fig. 5) approach for RS, RD techniques(2) and for the proposed approach (Fig.6)(1). The dimension of word embeddings is 300 and the number of hidden units is 100. Dropout rate is 0.25 and the batch size is 32. Adam optimizer is used with an inital learning rate of 0.001. All training consists of 100 epochs. We report accuracy of all the experiments.

4.3 Results and Analysis

The resultant accuracies obtained by applying a single augmentation strategy from the set of RS, RD, BT, RSI in the approaches mentioned above are present in Table 1. RS and RSI have performed well if training data is increased before training, RD reduced the overfitting if the data is augmented while training on batches. Based on these observations Algorithm1 pre-augmentation(x), which randomly chooses one of the four techniques is used to increase the training data before training and Algorithm 2 post-augmentation(x), which randomly chooses either RS or RD while training were adopted as shown in Fig. 6. This approach has resulted with 76.05%, which is an increase of +3.29, when compared with

[4]https://pytorch.org/text/stable/index.html

Augmentation Strategy	Approach - 1 pre-augmentation		Approach - 2 post-augmentation	
RS	75.45	+ 2.7	74.74	+ 1.99
RD	75.15	+ 2.4	74.41	+ 1.66
BT	74.74	+ 1.99		
RSI	75.51	+ 2.76		

Table (1) Comparison of adopted augmentation techniques with a baseline accuracy of 72.75%

the baseline. The proposed approach outperformed all the simple approaches to augment the data for performance boosting.

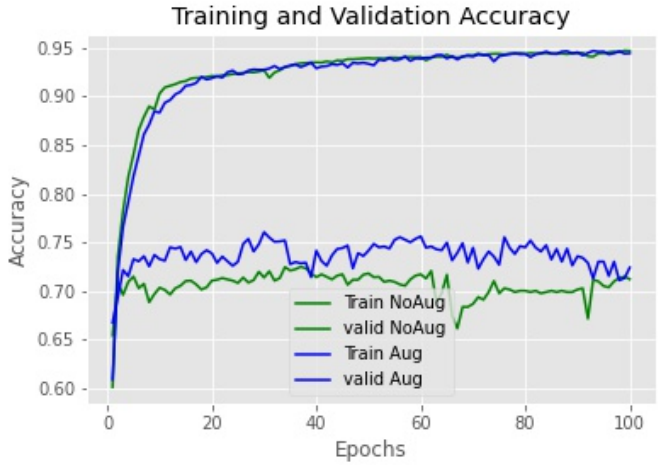


Figure (10) Training vs. Validation accuracy of proposed approach.

5 Conclusion

In this paper, we proposed a new data augmentation policy to increase the data before training and while training. Four augmentation methods are chosen in such a way that all are contributing for performance boosting. The proposed approach achieves a significant improvement in the accuracy and also reduced overfitting. This approach is best suitable when the training data is limited and it can be easily adopted to any task and dataset. This augmentation strategy can be further fine-tuned based on the increase or decrease in loss.

6 Future Work

References

[1] Otter, Daniel W., Julian R. Medina, and Jugal K. Kalita. "A survey of the usages of deep learning for natural language processing." IEEE Transactions on Neural Networks and Learning Systems (2020).

[2] Tang, Duyu, Bing Qin, and Ting Liu. "Deep learning for sentiment analysis: successful approaches and future challenges." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5.6 (2015): 292-303.

[3] Malinowski, Mateusz, Marcus Rohrbach, and Mario Fritz. "Ask your neurons: A deep learning approach to visual question answering." International Journal of Computer Vision 125.1-3 (2017): 110-135.

[4] Li, Hang. "Deep learning for natural language processing: advantages and challenges." National Science Review (2017).

[5] Pepe, Margaret Sullivan, et al. "Testing for improvement in prediction model performance." Statistics in medicine 32.9 (2013): 1467-1482.

[6] Perez, Luis, and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." arXiv preprint arXiv:1712.04621 (2017).

[7] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of Big Data 6.1 (2019): 60.

[8] Young, Tom, et al. "Recent trends in deep learning based natural language processing." ieee Computational intelligenCe magazine 13.3 (2018): 55-75.

[9] Young, Tom, et al. "Recent trends in deep learning based natural language processing." ieee Computational intelligenCe magazine 13.3 (2018): 55-75.

[10] Abonizio, Hugo Queiroz, and Sylvio Barbon Junior. "Pre-trained Data Augmentation for Text Classification." Brazilian Conference on Intelligent Systems. Springer, Cham, 2020.

[11] Fadaee, Marzieh, and Christof Monz. "Back-translation sampling by targeting difficult words in neural machine translation." arXiv preprint arXiv:1808.09006 (2018).

[12] Anders, Kelley L., et al. "Dynamic homophone/synonym identification and replacement for natural language processing." U.S. Patent No. 10,657,327. 19 May 2020.

[13] Xie, Ziang, et al. "Data noising as smoothing in neural network language models." arXiv preprint arXiv:1703.02573 (2017).

[14] Wei, Jason, and Kai Zou. "Eda: Easy data augmentation techniques for boosting performance on text classification tasks." arXiv preprint arXiv:1901.11196 (2019).

[15] Aggarwal, Charu C., and ChengXiang Zhai. "A survey of text classification algorithms." Mining text data. Springer, Boston, MA, 2012. 163-222.

[16] Can, Ethem F., Aysu Ezen-Can, and Fazli Can. "Multilingual sentiment analysis: An RNN-based framework for limited data." arXiv preprint arXiv:1806.04511 (2018).

[17] Li, Dan, and Jiang Qian. "Text sentiment analysis based on long short-term memory." 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI). IEEE, 2016.

[18] Srinivasa-Desikan, Bhargav. Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras. Packt Publishing Ltd, 2018.