Effective Text Augmentation strategy for NLP Models

December 27, 2020

Abstract

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

1 Introduction

2 Related Work

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 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.

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.

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

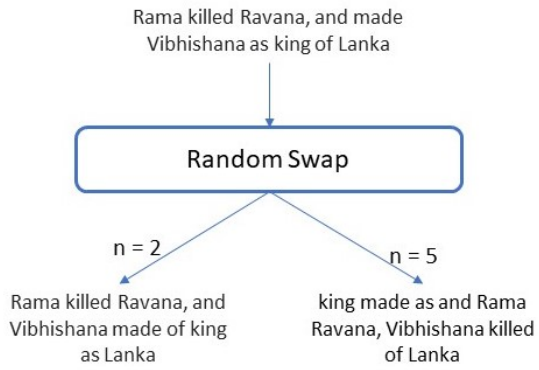


Figure (1) Random Swap

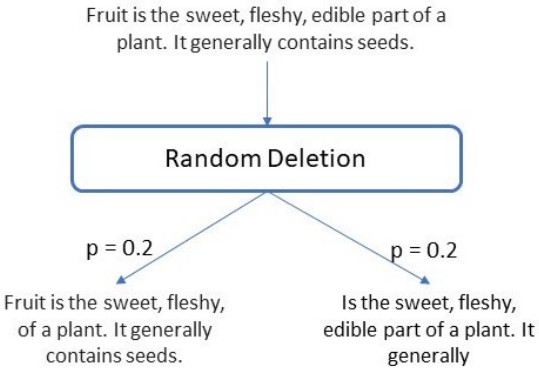


Figure (2) Random Deletion

3.1.3 Back Translate (BT)

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

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

3.1.4 Random Synonym Insertion (RSI)

This approach randomly inserts synonyms of of n words, which are not stopwords in a training example, x to generate a new training example. An example for Random Insertion with Synonym is shown in Fig. 4. The outcome of this

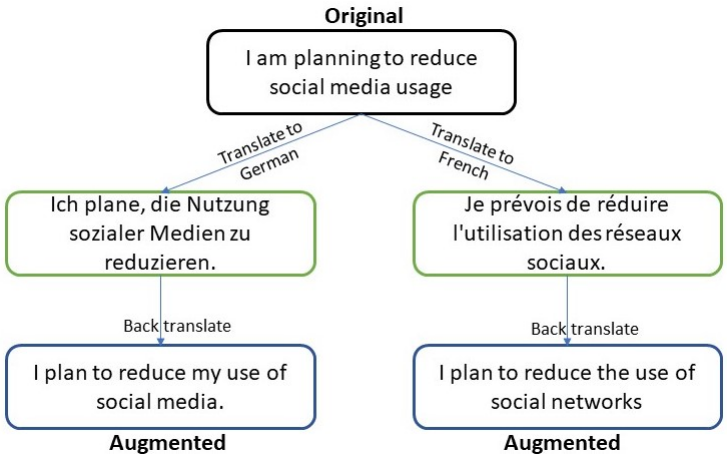


Figure (3) Backtranslation

method depends on the value of n. The suggestable value for n can be in the range of 1 to 3.

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

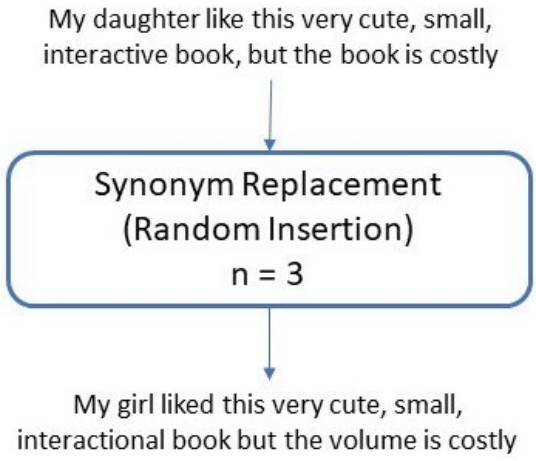


Figure (4) Random Insertion

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

(a) Pre-Augmentation

(b) Post-Augmentation

Figure (5) Initial methods to test all text augmentation strategies

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 $1..k$[1]. The classification model is constructed based on the training examples, and evaluated with the test set. Our paper used RNN language model based on Long Short Term Memory Network (LSTM) [2]. LSTM is better in analyzing emotion of long sentences and it is applied to achieve multi-classification for text emotional attributes [3]. This model is applied on the Apple Twitter Sentiment Dataset[1] to study the effectiveness of the selected text augmentation techniques in both the approaches and to come up with a best strategy for augmentation.

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, and cell state and hidden state 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 $/haty$.

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

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

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.

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 (5). Let $D_t : \{(x_i, y_i)\}_{i=1}^{M}$ is a set of M training examples.

$$D_{Nt} = D_t + T_t \tag{7}$$

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

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

Where, T is a transformation function, which augments a fraction, f of the M training samples to form new Training set, D_{Nt}. 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.

In the second approach the training samples $D_t : \{(x_i, y_i)\}_{i=1}^{M}$ in a mini-batch set at t^th training iteration 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. **??**). 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 augmentation happens randomly for 50% of the training samples, then the overall augmented training samples seen by the model in the training phase are $e * b * (05.m)$.

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.

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.

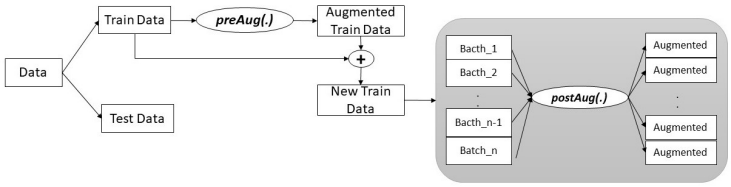


Figure (6) proposed method... write some text here

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

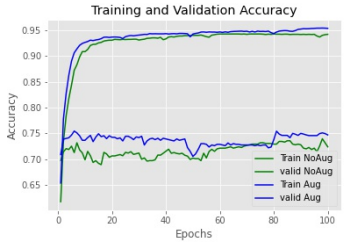
Algorithm 2: Post-Augmentation(x)

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

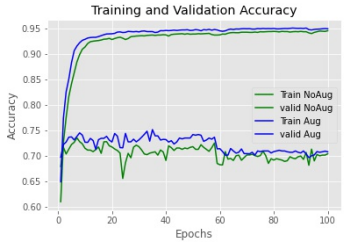
4 Experiment

4.1 Data

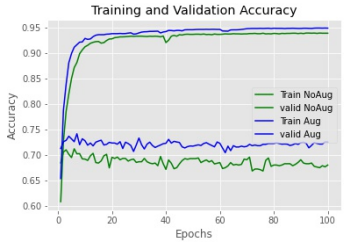
For our experiment, we use the Apple Twitter Sentiment dataset provided by kaggle for a competition called inclass prediction. This dataset is suitable for the experiment as we need to test augmentation strategy in limited data settings. ATS contains 3886 records in which 82 are not relevant. The tweets 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.**??**. 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.



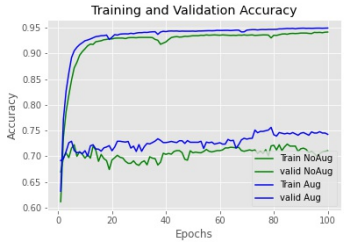
(a) Random Swap



(b) Random Deletion

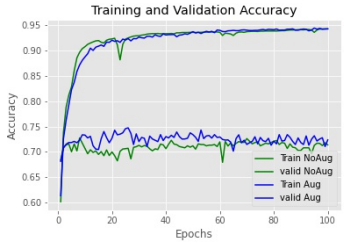


(c) Back translation

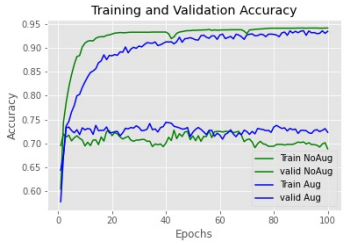


(d) Random Synonym Insertion

Figure (7) Approach 1 with RS, RD, BT, RSI



(a) Random Swap



(b) Random Deletion

Figure (8) Approach 2 with RS, RD

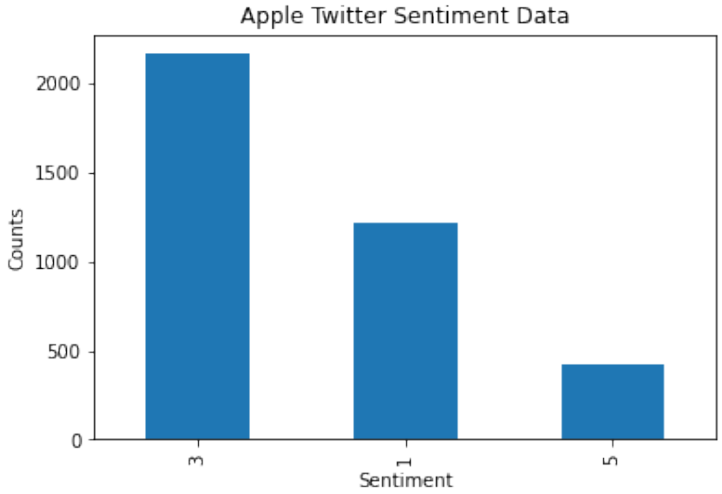


Figure (9) training examples class wise

4.2 Experimental Setup

The Data was tokenized with the help of *spacy*[4] tokenizer. *TorchText*[2] 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 /textbf8 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 the 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 preaument, which randomly chooses one of the four techniques is used to increase the training data before training and Algorithm 2 post augment, which randomly chooses either RS or RD while training were adopted as shown in Fig. 6. This approach

[2]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%

has resulted with 76.05%, which is an increase of +3.29, when compared with the baseline. The proposed approach outperformed all the simple approaches to augment the data for performance boosting.

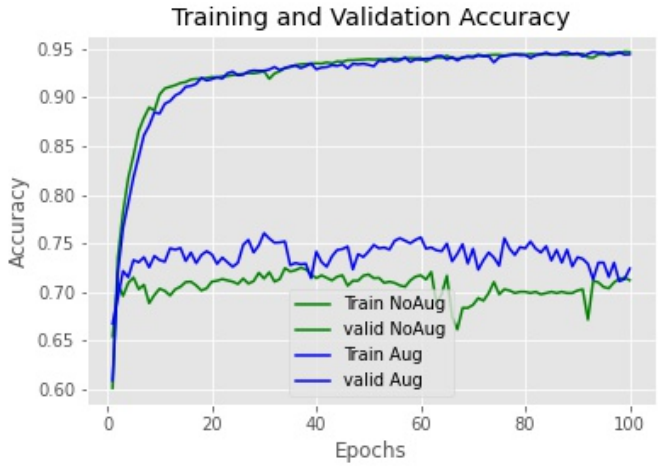


Figure (10) accuracy proposed

5 Conclusion

In this paper, we proposed a new data augmentation policy to increase the data before training and while training. The proposed approach best suits when the data is limited.

6 Future Work

References

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

[2] 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).

[3] 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.

[4] 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.