Iran University of Science and Technology
Deep Learning (Spring 2019)

Data Augmentation on Language Model

Fatemeh sadat Rezvainejad
Department of Computer Engineering
Iran University of Science
and Technology
rezvaninejad_f@comp.iust.ac.ir

Abstract

Having a small dataset for training is a necessary problem in many fields of deep learning, so we decided to investigate how we can augment it. Augmenting a dataset improves its robustness, reduces overfitting, in other words the results improves. We investigated on ways that introduced before on this problem and find other ways of augmenting data such as oppositing some words of text and for not changing the labels we added not, an other way that we suggest is to randomly deleting some words and randomly replacing some words in the text with their synonyms. For the end of the work we gave these into a Language Model to find out if it improves the predictions or not.

1 Introduction

Having a small dataset is a huge problem in some fields such as vision, text classification or even text clustering. There are so many works that has done for augmenting data on vision, speech and text classification. One of the approaches on augmenting data on text classification is EDA: easy data augmentation[1] that introduces four methods which are defined as below:
1. Synonym Replacement (SR):Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
2. Random Insertion (RI): Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this n times.
3. Random Swap (RS): Randomly choose two words in the sentence and swap their positions. Do this n times.
4. Random Deletion (RD): Randomly remove each word in the sentence with probability p. [1]. This approach code available in[1]

In this project we add two other methods:
5. First Deletion Second Synonym Replacement: In this method first we delete n/2 words and replace n/2 remained words which are not stop words.
6. Not + Opposite: We will find n words opposites and for not changing the labels we add not to the sentence.

Example of approach on SST-2 dataset:

[1] https://github.com/jasonwei20/

(LateX template borrowed from NeurIPS 2019)

Sentence: neil burger here succeeded in making the mystery of four decades back springboard for a more immediate mystery in the present.
RS: neil burger here succeeded in making the mystery of four more back the springboard for a decades immediate mystery in the present.
SR: neil burger here come through in the mystery of four decades back the springboard for a more immediate mystery in the present.

These methods produces augmented sentences for our training data. According to paper [1] n is the number of words that changes in a sentence and it gains from n=α l that α is a number preferly between 0.05 and 0.5 because it shouldn't be big in SR, because it will change too much words and it can change the label of the sentence. l is the length of the sentence. We gave the dataset to the language model without augmentation, with paper[1] augmentation method and with our augmentation method. We had other ideas for this project such as use twitter dataset for augmenting and feed it to the network but because twitter dataset has some words without meaning we had a huge problem for augmenting it cause of we couldn't find any synonym in wordnet for them so we used SST-2 and Text-Clustering dataset[2] which has 200 records of sentences about games and sports.
Another part of our approach was to investigate does improvements in small datasets augmentation is bigger than augmentation on big datasets.

2 Related work/Background

Previous approaches of augmenting small datasets are complex. Back-translation[2] which augments the data by translating data to another language and translate it back(in a part of this paper they translate data from English to German and then they translate it back from German to English) showed an improvement results, another approach is translational data augmentation [3] which targets low-frequencywords by generating new sentence pairs containing rare words in new have shown improvements in this field but both of them had high costs. So a simpler method showed in this paper.

3 Proposed method

For augment part of our approach we shuffled and sampled from both SST-2 and Text Clustering. After converting Text Clustering to the text file we gave both of our datasets to eda seperately to augment α =0.3 and n_aug(number of augmented sentences)=3. In eda code first we define stop-words and we checked them not to involve in methods , second we define our methods to do their jobs and the last part of it, is that the augmented sentences generated, In augmented code we print the augmentation sentences with theirs labels 0 and 1 which just means that which augmented sentences belongs to which. As default α = 0.1 and n_aug= 9, you can change it easily according to your need.
The seconed step was that we built a Language Model with 1 embedding, 2 lstm, 1 dense layer. The 2-lstm layers improves the result and lstm saves the sequence of data because of its memory and thats why we used it in our language model.
For comparing the effect of dataset size we generate 2 text file of Text Clustering which contains 5% and 10% of the dataset randomly. Our code is available in:[3]. We used softmax as activation function.

4 Results

We used part of SST-2 dataset and Text Clustering dataset. SST-2 is Stanford Sentiment Treebank dataset, it has sentences with their labels which we ignored the labels and we chose a part of it randomly.
The second dataset is Text Clustering which has 200 long sentences about games and sports which we randomly sampled 5% ,10% of it because of long runtime for each epochs ang comparing the results [figure1]. We had 2 datasets that each of them should run 3 times for non-augment, paper [1] augment method and our approach method so we ran every run with 5 epochs. Results have shown below:

[2]https://github.com/sharmaroshan/Text-Clustering
[3]https://github.com/fatemehrezvani/final-project

Approach	Text-Clustering of games data(10%)	Text-Clustering of games data(5%)
No augment	6.1	6.1
Our augmentation approach	5.8	6.2

Figure 1: Accuracy of language model with different different size of dataset

Approach	SST-2	Text-Clustering of games data
No augment	3.8	6.1
Augmentation with[1] approach	3.5	6.2
Our augmentation approach	4.0	6.3

Figure 2: Accuracy of language model with different approaches

we augment every of datasets and gave it to the language model and training accuracy are show in figure[2]. a sample of prediction of language model has shown in figure[3]. We show 3 of the implementation in figure[4], figure[5] and figure[6].

5 Discussion

We used 2 lstm network for our language model because lstm can save the sequence of words in a sentence and for language model that we want to predict likelihood of occurrence of a word. We present a simple method of augmentation while previous approaches had an expensive cost of implementation. We ran every of them in 5 epochs that each epoch takes about 1 hour.

References

[1] Wei, J. W. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.

[2] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages86–96. Association for Computational Linguistics.

[3] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 567–573. Association for Computational Linguistics.


```
1 print (generate_text( "it is too bad that this likable ", 1, max_sequence_len))
it is too bad that this likable  film
```

Figure 3: Prediction of language model

```
Epoch 1/5
8408/8408 [==============================] - 39s 5ms/step - loss: 7.1645 - acc: 0.0390
Epoch 2/5
  32/8408 [..............................] - ETA: 36s - loss: 6.1040 - acc: 0.0000e+00
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
8408/8408 [==============================] - 38s 4ms/step - loss: 6.6913 - acc: 0.0356
Epoch 3/5
8408/8408 [==============================] - 37s 4ms/step - loss: 6.6578 - acc: 0.0388
Epoch 4/5
8408/8408 [==============================] - 37s 4ms/step - loss: 6.6522 - acc: 0.0340
Epoch 5/5
8408/8408 [==============================] - 37s 4ms/step - loss: 6.6493 - acc: 0.0389
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 45, 10)            27770
_________________________________________________________________
lstm_3 (LSTM)                (None, 45, 150)           96600
_________________________________________________________________
lstm_4 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_2 (Dense)              (None, 2777)              280477
=================================================================
Total params: 505,247
Trainable params: 505,247
Non-trainable params: 0
```

Figure 4: Accuracy of not augmentation approach on SST-2 dataset

```
Epoch 1/5
34646/34646 [==============================] - 272s 8ms/step - loss: 7.1506 - acc: 0.0343
Epoch 2/5
/usr/local/lib/python3.6/dist-packages/keras/callbacks.py:569: RuntimeWarning: Early stoppir
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
34646/34646 [==============================] - 267s 8ms/step - loss: 6.8854 - acc: 0.0359
Epoch 3/5
34646/34646 [==============================] - 264s 8ms/step - loss: 6.8850 - acc: 0.0357
Epoch 4/5
34646/34646 [==============================] - 267s 8ms/step - loss: 6.8830 - acc: 0.0362
Epoch 5/5
34646/34646 [==============================] - 266s 8ms/step - loss: 6.8821 - acc: 0.0352
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 60, 10)            42900
_________________________________________________________________
lstm_1 (LSTM)                (None, 60, 150)           96600
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_1 (Dense)              (None, 4290)              433290
_________________________________________________________________
```

Figure 5: Accuracy of paper [1] augmentation approach on SST-2 dataset

```
Epoch 1/5
31147/31147 [==============================] - 170s 5ms/step - loss: 7.3625 - acc: 0.0405
Epoch 2/5
   32/31147 [..............................] - ETA: 2:51 - loss: 6.8265 - acc: 0.0312/usr
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
31147/31147 [==============================] - 166s 5ms/step - loss: 7.0541 - acc: 0.0398
Epoch 3/5
31147/31147 [==============================] - 165s 5ms/step - loss: 7.0533 - acc: 0.0414
Epoch 4/5
31147/31147 [==============================] - 165s 5ms/step - loss: 7.0561 - acc: 0.0404
Epoch 5/5
31147/31147 [==============================] - 164s 5ms/step - loss: 7.0516 - acc: 0.0407
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 61, 10)            44770
_________________________________________________________________
lstm_1 (LSTM)                (None, 61, 150)           96600
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_1 (Dense)              (None, 4477)              452177
=================================================================
Total params: 693,947
Trainable params: 693,947
Non-trainable params: 0
```

Figure 6: Accuracy of our augmentation approach on SST-2 dataset