Abstract

This paper explains about stance determination of a particular target topic with respect to a text using data mining techniques, pre-training with the help of deep learning models like DistilBERT and tools like transformers using the Natural Language Processing(NLP) where an effective way of understanding the linguistics of the dataset entries is performed using the encoding and decoding of the transformers on the data, and then the relation between between the tokens thus generated via transformers are used for the stance detection with the help more efficient DistilBERT(as with more speed and accuracy). The dataset that is considered is similar to the Fake News dataset with the labels for the stance(here between the headline and articlebody) being used in here are agree,disagree,unrelated of the dataset

Introduction

The Stance in linguistics is a practice of positioning a particular statement or action with another statement or action in respect to the epistemology, terms of evaluation, intentionality,.. That is, if a speaker is describing about an object in a way that expresses the attitude or relation to that object then it the speaker is taking a stance against the object

And thus with the tremendous increase in data available it has become a need for classifying and analyzing the data(data mining) especially in the fields of corpus and analyzing consumer behavior based on the feedbacks,.. which can be made easier using the stance analysis which basically includes how a statement or action is standing against another statement or action i.e; if it is opposing the things that the other action is trying to convey or is it supporting the action or statement or may be is it not at all involving with that action(like how strong does the action is supporting an another action is also important)

The previous papers like “Text Stance Detection Based on Deep Learning” by Xu Zhang, Chunyang Liu, Zhongqin Gao, Yue Jiang. Used CNN models to design models for the stance detection using techniques like the independent encoding,.. with improved accuracy and efficiency.

Later Further improved versions of the NLP based data analyzing techniques were built based on the tools like the transformers which helped in maintaining the self-attention of the words in the statements or the actions along with the independent encoding and decoding techniques, which further improved the accuracy of solving the NLP based data mining problems

Thus the later versions of the BERT(Bi-directional Encoder Representations from Transformers, 2018) like the RoBERTa(Robustly Optimized BERT Pre-training Approach) and also the DistilBERT(a Distilled version of BERT) were made used in this paper for better accuracy and efficiency

For example - The Apples in the sentences (Apple is launching a new product . Apple a day keeps doctor away) are treated different by using the **transformers** along with the BERT models, thus helping in much efficient and accurate classifications of the NLP based data mining problems

Literature Survey

"Text Stance Detection Based on Deep Learning" by Xu Zhang, Chunyang Liu, Zhongqin Gao, Yue Jiang . This paper mainly discusses about the effect of deep learning models on the stance detection using a hybrid encoding-CNN models and their advantages, as though the machine learning models can give accurate results, they are not efficient for larger datasets because of the high labeling cost and weak model generability

This paper mainly worked on the independent encoding-CNN and conditional encoding-CNN models based on its previous related work which worked on two datasets, SemEval2016 (An English language dataset) which has highest accuracy rate on a CNN model and NLPCC (A Chinese language dataset) which has the highest accuracy on the BiLSTM neural network model which have faced the problems mainly in the tapping of text target information with the stance, hence the text target information is encoded firstly and then worked on, a Twitter dataset is used in this paper for the better observation of the user responsive information.

Initially it used the static attention for the words of the test and target combined using the word2vec CBOW model based on the hierarchical softmax optimization(based on the frequency of the words, similar to tensorflow's Adam, like considering the weights of the words similar to the weighted average technique) for word-embedding pre training and then for the CNN part it used different features for distinguishing the text target stance aspects by evaluating with different filters for each feature on the word-embedding matrix. It also uses the ReLU activation to generate new required word vectors which are later combined to form the sentence vectors

Finally the feature maps are used in mapping with the functionalities and is verified for both the independently and conditionally encoding CNN models clarifying the importance of independent encoding CNN model

Methodology

The paper uses the *DistilBERT* model for training and testing on the dataset. Distilbert is used for natural language processing and is based on transformers neural architecture.

1.Model

Distilbert tries to maintain as much performance as feasible while optimizing the training by scaling back BERT and speeding it up. Distilbert, in particular, is 40% smaller, 60% quicker, and 97% functionally equivalent to the original BERT-base model.

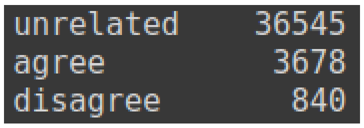
In order to imitate BERT, having the vast neural network into a smaller one, DistilBERT employs a method known as distillation. The basic idea is that when a big neural network has been trained, its whole output distributions may be roughly represented by a smaller network.

2. Dataset

The dataset has 2 tables or csv files and has 3 main fields, that are, *headline, articlebody and stance,* in the 2 tables. Before using the dataset tables are merged based on the *bodyID* field given the 2 csv files. For the stance field it has 4 labels, that are *agree, disagree, discuss and unrelated*. For simplicity, the *discussion* field will not be included in the final dataset.

3. Pre-processing

The given csv files are merged taking *bodyID* as the common field and the *discuss* field is removed. The total length of the dataset becomes 41,063 and data is distributed between the fields such that the distribution of stance is



```
unrelated    36545
agree         3678
disagree       840
```

Fig. Dataset label count

The disagree,agree and unrelated labels are removed and labels 0,1 and 2 are used instead of them respectively.

As nlp model training takes a lot of time, this dataset of length 41063 is further divided into 4 parts of length 10000 each with the last one of length 11063.

4. Logic of the Code

The code is importing models and tokenizers for the respective model from the hugging face library. After this the importation of the dataset is taking place.

This dataset is passed into tokenizer's encode_plus function and tokenIds and attention mask are given as the output. The encode_plus function also takes a few other parameters like max_length, pad_to_max_length, truncation etc. The max length is set to 256, pad_to_max_length is set to true so that all the tokenIds and an attention mask which has a length of 256 for a given heading and articleBody. Max_length is chosen to be 256 because 128 and 512 seem to be extreme. The length of the tokenIds given from the tokenizer was around 250 length,choosing 512 will pad the tokenIds and attention mask, which can hamper the performance of model in training the increase the time taken for both training and validation and choosing max_length to be 128 would have removed the important data for stance detection. The input_ids(token_ids) and an attention mask are stored into arrays and these are passed to the Dataset class of the tensor library and then divided into 0.7 and 0.3 size for training and validation.
This training and validation dataset is then passed into DataLoader class, to convert it into batch size of 16, which is neither too big nor too small.

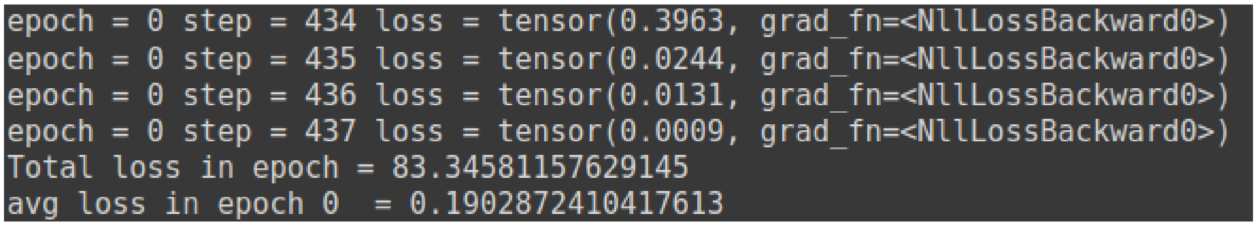
5. Training
The model is trained on 2 epochs. In the epoch loop code is setting the model for training using model.train(). In the training loop code id going through every batch of size 16 each and putting its data into the model to train the model. This gives the output, which has loss, and logits for that batch during training.
The optimizer used in AdamW optimers. In the next line the loss for every batch and write optimization steps are written.

6. Validation
In the validation step, the model is set to validation mode using model.eval(). In the validation loop the validation data loader is given to the model, which does the validation and gives the output back.
This output contains loss taken at every validation of every batch.

Results and Analysis :

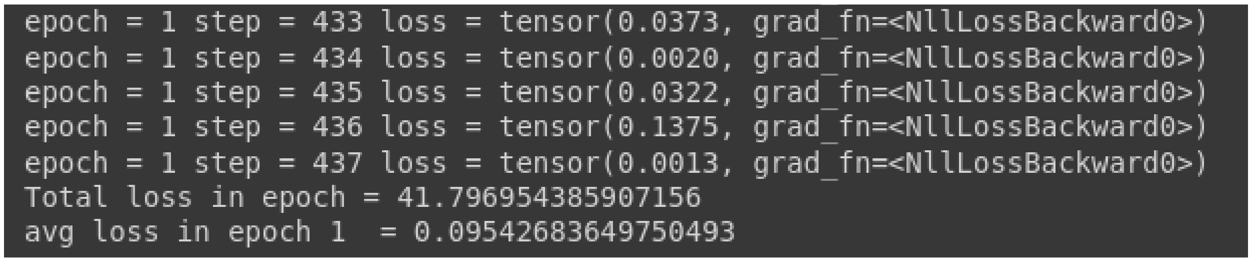
1. **Training**

In the training loop code is outputting the loss for every epoch and batch. At the end total epoch loss and average epoch loss is printed



```
epoch = 0 step = 434 loss = tensor(0.3963, grad_fn=<NllLossBackward0>)
epoch = 0 step = 435 loss = tensor(0.0244, grad_fn=<NllLossBackward0>)
epoch = 0 step = 436 loss = tensor(0.0131, grad_fn=<NllLossBackward0>)
epoch = 0 step = 437 loss = tensor(0.0009, grad_fn=<NllLossBackward0>)
Total loss in epoch = 83.34581157629145
avg loss in epoch 0  = 0.1902872410417613
```

Fig.Total and average epoch loss for 1st epoch

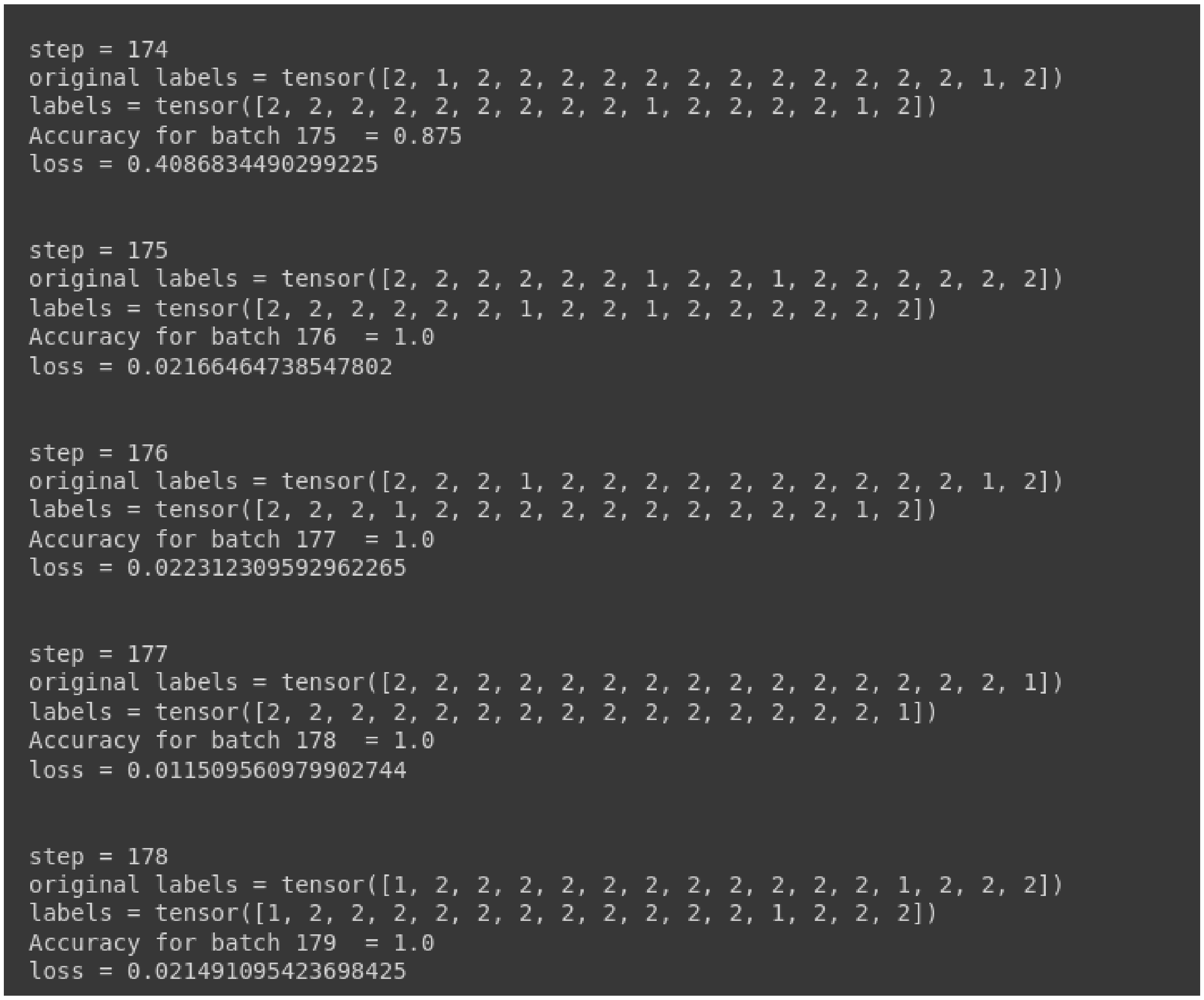


```
epoch = 1 step = 433 loss = tensor(0.0373, grad_fn=<NllLossBackward0>)
epoch = 1 step = 434 loss = tensor(0.0020, grad_fn=<NllLossBackward0>)
epoch = 1 step = 435 loss = tensor(0.0322, grad_fn=<NllLossBackward0>)
epoch = 1 step = 436 loss = tensor(0.1375, grad_fn=<NllLossBackward0>)
epoch = 1 step = 437 loss = tensor(0.0013, grad_fn=<NllLossBackward0>)
Total loss in epoch = 41.796954385907156
avg loss in epoch 1  = 0.09542683649750493
```

Fig.Total and average epoch loss for 2nd epoch

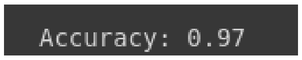
As can be seen from above screenshots the total epoch loss is reduced to half in the second epoch training and the average loss in training in the epoch is reduced by around 20 times.

2. **Validation**
   The code outputs original labels, predicted labels, accuracy and loss for every batch in the validation loop.



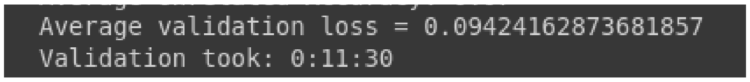
```
step = 174
original labels = tensor([2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2])
labels = tensor([2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2])
Accuracy for batch 175  = 0.875
loss = 0.4086834490299225


step = 175
original labels = tensor([2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2])
labels = tensor([2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2])
Accuracy for batch 176  = 1.0
loss = 0.02166464738547802


step = 176
original labels = tensor([2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2])
labels = tensor([2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2])
Accuracy for batch 177  = 1.0
loss = 0.022312309592962265


step = 177
original labels = tensor([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1])
labels = tensor([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1])
Accuracy for batch 178  = 1.0
loss = 0.011509560979902744


step = 178
original labels = tensor([1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2])
labels = tensor([1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2])
Accuracy for batch 179  = 1.0
loss = 0.021491095423698425
```

Fig. validation loop output for few batches



```
Accuracy: 0.97
```

Fig. Average accuracy for validation loop



```
Average validation loss = 0.09424162873681857
Validation took: 0:11:30
```

Fig. Average validation loss and time taken for validation

As seen the average accuracy for validation loop is coming to be 97% with an average validation loss of 0.0942. Also the time taken for validation is 11 minutes and 30 seconds.

Conclusions:

This implementation is successful in implementing a stance analyzer for the English language using DistiltBERT with an accuracy of 97% and with a very low validation loss. The model is successful in reducing the loss by a significant factor of around 20 between the two training epochs. This implementation can be used in widespread use of stance analysis.

References:

1. The previous papers "Text Stance Detection Based on Deep Learning" by Xu Zhang, Chunyang Liu, Zhongqin Gao, Yue Jiang
2. "Samia Khalid", "BERT Explained-A Complete Guide with a theory and tutorial ", https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c(accessed on Aug 1, 2022)
3. "Julien Chaumond, Clément Delangue and Thomas Wolf", "BERT", https://huggingface.co/docs/transformers/model_doc/bert(accessed on Aug 1, 2022)
4. "Meta Platforms", "PYTORCH DOCUMENTATION", https://pytorch.org/docs/stable/index.html (accessed on Aug 5, 2022)