Natural Language Processing

Report for Assignment 2 (Sentiment Analysis)

Vikanksh Nath (MT19AI024)

Objective: We require to build a deep learning model for sentence level sentiment analysis of the text. For testing the performance we build a web crawler for the website "https://www.livemint.com/topic/international" to extract news articles for predicting their sentiment score.

Data-Preparation: We used the reviews data set from following link : https://archive.ics.uci.edu/ml/machine-learning-databases/00331/sentiment%20labelled%20sentences.zip , It contains the amazon, yelp and imdb reviews for products, movies, and restaurants. Sentences are labelled with positive (1) or negative sentiment (0). The whole data set contains 2748 reviews. We split this data set in 70 %, 30% training and validation data sets respectively.

	reviews	sentiment
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1

Figure 1. Snap of Training Dataset from Notebook

During data preparation, we also ensured that each word token is free of any punctuation, special character and numbers. We also used *packed padded*

sequences, which will make our RNN only process the non-padded elements of our sequence, and for any padded element the output will be a zero tensor. As we are using the **PyTorch** platform**,** We do this by setting include_lengths = True for our TEXT field.

Word Embeddings: instead of having our word embeddings initialized randomly, they are initialized with these pre-trained vectors. We used the "**glove.6B.100d**" vectors". glove is the algorithm used to calculate the vectors. 6B indicates these vectors were trained on 6 billion tokens and 100d indicates these vectors are 100-dimensional.

```
txt_field.vocab.vectors[txt_field.vocab.stoi['product']]
# txt_field.vocab.vectors[txt_field.vocab.itos[1]]
```

```
tensor([ 0.1280,  0.3413,  0.3311, -0.0267, -0.0227, -1.0228,  0.6519, -0.1420,
         0.2910,  0.5614, -0.1294, -0.7779, -0.0147, -0.0082,  0.1977,  0.4230,
         0.6420,  0.8920,  0.2820,  0.0382, -0.0661, -0.3985, -0.0251,  0.4593,
        -0.4563,  0.3667,  0.5693, -0.1560, -0.8231, -0.4675,  0.3595,  0.9756,
        -0.0480, -0.4706,  0.6593,  0.6621,  0.1840, -0.0525, -0.6372, -0.5337,
         0.5093, -0.5586,  0.0120,  0.0967,  0.0535,  0.2957, -0.1554, -0.4062,
        -0.5804, -0.9215,  0.6170, -0.0199, -0.1937,  0.7281,  0.0768, -1.6533,
        -0.6374, -0.0603,  1.9839,  0.1353,  0.4741, -0.1415, -0.3758,  0.1504,
         0.8950, -0.0732,  0.6373, -0.3346,  0.9764, -0.4185,  0.2639,  0.6476,
        -0.0575,  0.0053,  0.3126, -0.0048, -0.2146, -0.1834, -0.5071, -0.1076,
         0.1550,  0.1201, -1.0232,  0.5519, -1.4446, -0.3760,  0.1186, -0.7098,
        -0.1665, -0.4338,  0.2761,  0.5844, -0.3868, -0.4611,  0.2423, -0.2198,
        -0.2088, -0.3773,  1.2899,  0.1322])
```

Figure 2. Snap of Embedding of word ‘product’ from Notebook

Model Architecture: In the first stage of implementation we tried Bidirectional LSTM with 2 Layers. The architecture is given in below Fig 3. We used dropout with 0.2 rate, as a regularization to the model too. The output of the model is the tensor with sentiment score of each sentence of the fed input text as shown in below Fig. ***The optimum validation accuracy found to be 84.25 %*** . We saved the weights at the optimum value of validation accuracy. We used Adam as a solver. Learning rate = 0.01.

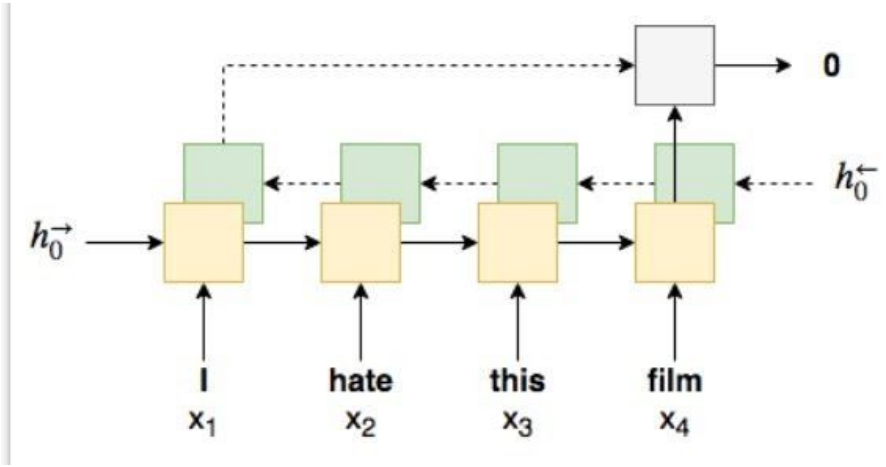

Figure 3. BiLSTM architecture

```
    if(epoch%10==0):
      print(f'Epoch: {epoch+1:02}')
      print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_
      print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_

# print(f'\nOptimal Epoch: {opt_epoch+1:02}')
# print(f'Best Training Accuracy achieved: {best_training_accurac
# print(f'Best Validation Accuracy achieved: {best_validation_acc
```

```
Epoch: 01
        Train Loss: 0.566 | Train Acc: 71.68%
         Val. Loss: 0.426 |  Val. Acc: 81.61%
Epoch: 11
        Train Loss: 0.015 | Train Acc: 99.69%
         Val. Loss: 0.818 |  Val. Acc: 83.05%
Epoch: 21
        Train Loss: 0.000 | Train Acc: 100.00%
         Val. Loss: 1.269 |  Val. Acc: 81.73%
Epoch: 31
        Train Loss: 0.000 | Train Acc: 100.00%
         Val. Loss: 1.365 |  Val. Acc: 82.45%
Epoch: 41
        Train Loss: 0.005 | Train Acc: 99.90%
         Val. Loss: 1.302 |  Val. Acc: 81.85%
```

Figure 4. Snap of Training Phase from Notebook

Web-Crawler: We used BeautifulSoup for making the web crawler. The following information is extracted from the https://www.livemint.com/topic/international , 1. News headline, 2. Date of Publication, 3. Link to Article, 4. Complete Text. We saved this information in a .csv file. We are submitting a web-crawler in a separate notebook.

```
df.head()
```

	News headline	Date of Publication	Link to Article	Complete Text
0	India's IT spending to drop by 4.5% compared t...	30 Apr 2020	https://www.livemint.com/news/india/india-s-it...	IT spending in India is expected to drop by 4...
1	Lifestyle International appoints Rishi Vasudev...	16 Mar 2020	https://www.livemint.com/companies/news/lifest...	NEW DELHI : Retail company Lifestyle Internat...
2	Geneva International Motor Show is cancelled o...	28 Feb 2020	https://www.livemint.com/news/world/geneva-int...	MUMBAI : This year's Geneva International Mot...
3	Delhi's IGI Airport becomes first single-use p...	17 Feb 2020	https://www.livemint.com/news/india/delhi-s-ig...	New Delhi: GMR-led Delhi International Airpor...
4	Sudan to hand over ex-President Omar al-Bashir...	11 Feb 2020	https://www.livemint.com/news/world/sudan-to-h...	Sudan has agreed to hand ousted autocrat Omar...

Figure 5. Snap of Extracted "Web-Page information" from Notebook

Testing-Phase: To test our model performance, we fed these extracted news from the saved .csv file to the model. As it is seen from below Fig 6. Every Tensor is corresponding to every article, and tensor elements show the sentiment score of each sentence of the article. To get an article (document) level sentiment score, we summed up the sentence score given by elements of the tensor and divided by the total number of the sentences (i.e. tensor length). Now we get the normalized score ranging (0,1) for every article. So, a score nearer to 0 indicates more negative sentence/article, nearer to 1indicates more positive sentence/article and nearer to 0.5 indicates neutral sentence/article.

We manually saw the first 12 Articles from Livemint/International and found the sentiments predicted by the model are very consistent with what we manually thought after reading for these first 12 articles.

Moreover we evaluated model performance on the downloaded data set also with train, val, test split. ***We found the testing accuracy to be 82 %.***

We run all models on Google Colab GPU.

```
for i in range(df.shape[0]):
  newsdf = pd.DataFrame(nltk.tokenize.sent_tokenize(df1[i]),columns=['reviews'])
  newsdf['sentiment']=''
  newsdf1 = clean_lines(newsdf)
  newsdf1 = newsdf1[:-2]
  # newsdf1
  newsdf1.to_csv('sentimentanalysis/news.csv', index=False)
  testdata = data.TabularDataset(
           path="sentimentanalysis/news.csv", # the file path
           format='csv',
           skip_header=True, # if your csv header has a header, make sure to pass this
           fields=dataset_fields)
  news_it = Iterator(testdata, batch_size=32, shuffle=False, sort=False, sort_within_b
  os.rename(filename, os.path.join(path, 'captured' + str(i) + '.jpg'))
  test(model, news_it, criterion)
```

```
[tensor([0.7140, 0.5093, 0.7814, 0.7527, 0.6724, 0.5742, 0.6019, 0.4493, 0.6495,
         0.5799, 0.6819, 0.5128], device='cuda:0')]
[tensor([0.4932, 0.6997, 0.5135, 0.4973, 0.5527, 0.4727, 0.4709, 0.2213, 0.0653,
         0.2919, 0.6776, 0.7128, 0.5811, 0.5814, 0.1956, 0.4638],
        device='cuda:0')]
[tensor([0.5932, 0.6391, 0.4756, 0.5713, 0.6646, 0.6870, 0.5834, 0.4922, 0.6704,
         0.6646, 0.4161, 0.6975, 0.7921, 0.9000, 0.3868], device='cuda:0')]
[tensor([0.5264, 0.6388, 0.5070, 0.6812, 0.3297, 0.6992, 0.7778, 0.4774, 0.6311,
         0.5441], device='cuda:0')]
[tensor([0.5465, 0.6928, 0.5977, 0.6198, 0.6184, 0.5619, 0.6160, 0.4247, 0.7625,
         0.6621, 0.4160, 0.8305, 0.6673, 0.7154, 0.7622, 0.6769, 0.6348, 0.6681,
         0.7910, 0.5965, 0.6311, 0.5441], device='cuda:0')]
[tensor([0.5822, 0.4557, 0.7879, 0.6386, 0.3915, 0.5238, 0.6521, 0.7403, 0.2706,
```

Figure 6. Snap of "Sentence-level Sentiment Score shown by Tensor of an Article"

In our submitted sentiment.csv file, first 12 news are extracted from https://www.livemint.com/topic/international repeated twice,after that all news from next pages of "https://www.livemint.com/topic/international/page-i" are extracted and stored with article as well as sentence level sentiment score. One may specify the number of pages of the website in the "web crawler notebook file" from which one wants to extract news The format for sentence sentiment score is 'sen1 (score:0/1)', 'sen2 (score:0/1)'..and so on for an article.