Deep Learning Classifier Sample

Objective: Create a classifier that is able to accurately gauge the sentiment of a user review.

Methodology:

Deep averaging network (DAN) has been used in this assignment. DAN has been discovered in 2015. It obtains near state-of-art accuracies on a variety of sentence and document-level tasks with just minutes of training time on average laptop computer. [1] In this assignment, DAN has two hidden layers and applies two layers of dropout. DAN has been composed of ReLU non-linearity combined with He et al. (2015) weights activation function. ReLU has been chosen because it is less computationally expensive than tanh and sigmoid, because it involves simpler mathematical operations. In He et al. method, the weights are initialized keeping in mind the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently. He et al. method works best with ReLU non-linearity. [2] The choice of optimization algorithm for the deep learning model can mean the difference between good results in minutes, hours, and days. Adam (2015) optimization algorithm has been used in this assignment. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Adam has been chosen since it is computationally efficient and has little memory requirements. [3] GloVe: Global Vectors for Word Representation (Stanford) was used to train the model. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. [4] All algorithms have been designed in Python using PyTorch and NumPy libraries. Currently there are two deep learning Python libraries: PyTorch designed by Facebook and TensorFlow designed by Google. PyTorch is more modern and Pythonic library, which gives better development and debugging experience and hence has been used here. [5]

Data processing:

The labels in the data has been changed from neg and pos to 0 and 1 respectively. The column order was changed to have the gold labels as the first column and the sentiment as a second column. The first row with the column names was removed.

The data has been split into 3 random sets: train, test and dev using NumPy in the proportion of 80%-10%-10%.

Framework code:

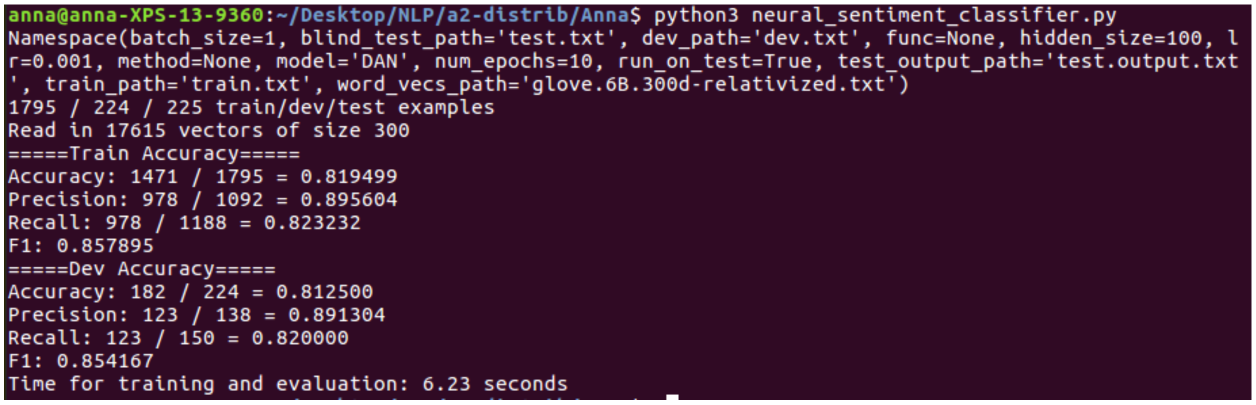
1. **neural_sentiment_classifier.py** is the mail class. It uses arparse to read in several command line arguments. This file also contains evaluation code. The main method loads in the data, initializes the feature extractor, trains the model, and evaluates it on train, dev and test, and write the test results to **test.output.txt**.
2. **sentiment_data.py handles** all data reading. It also defines a SentimentExample object which warps a sequence of words with an integer label (0/1).
3. **utils.py** implements an Indexer class, which maintains a bijective mapping between indices and features (strings).
4. **models.py** is the primary file which contains DAN setup.
5. **data_fixer.py** modifies and formats excel data for digestion into DAN. Splits data into training, validation and test data sets using NumPy: **dev.txt, train.txt, test.txt**

6. Pretrained Word Embeddings by GloVe algorithm in 50 and 300 dimensions: **glove.6B.50d-relativized.txt**, **glove.6B.300-relativized.txt**
7. Given data in different formats: **takehome.csv**, **takehome1.txt** (modified)

Results:

Accuracy of sentiment analysis is said to be acceptable if it matches that of human judgement, which is about 80%. Current models aim to outperform this baseline. There are a lot of challenging problems which seriously affect sentiment analysis accuracy: irony and sarcasm, types of negations, word ambiguity and multipolarity. [6]

The accuracy statistics are listed in the table below: Accuracy – 81%, Precision – 89%, Recall – 82%, F1 – 85%. These stats satisfy the modern requirement for sentiment analysis classifiers.



```
anna@anna-XPS-13-9360:~/Desktop/NLP/a2-distrib/Anna$ python3 neural_sentiment_classifier.py
Namespace(batch_size=1, blind_test_path='test.txt', dev_path='dev.txt', func=None, hidden_size=100, l
r=0.001, method=None, model='DAN', num_epochs=10, run_on_test=True, test_output_path='test.output.txt
', train_path='train.txt', word_vecs_path='glove.6B.300d-relativized.txt')
1795 / 224 / 225 train/dev/test examples
Read in 17615 vectors of size 300
=====Train Accuracy=====
Accuracy: 1471 / 1795 = 0.819499
Precision: 978 / 1092 = 0.895604
Recall: 978 / 1188 = 0.823232
F1: 0.857895
=====Dev Accuracy=====
Accuracy: 182 / 224 = 0.812500
Precision: 123 / 138 = 0.891304
Recall: 123 / 150 = 0.820000
F1: 0.854167
Time for training and evaluation: 6.23 seconds
```

HOW TO USE THE FRAMEWORK CODE

1. Make sure that Python3.7 with NumPy and PyTorch libraries are installed in the system.
2. Unzip the file into one folder.
3. Open command window and navigate to the unzipped folder.
4. Type python3 neural_sentiment_classifier.py and click "Enter".

References:

[1] Deep Unordered Composition Rivals Syntactic Models for Text Classification. Iyyer, Manjunatha, Boyd-Graber, Daume. https://people.cs.umass.edu/~miyyer/pubs/2015_acl_dan.pdf

[2] Random Initialization For Neural Network: A Thing Of The Past. https://towardsdatascience.com/random-initialization-for-neural-networks-a-thing-of-the-past-bfcdd806bf9e

[3] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[4] GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/

[5] PyTorch vs TensorFlow – spotting the difference. https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b

[6] Four Pitfalls of Sentiment Analysis Accuracy. https://www.toptal.com/deep-learning/4-sentiment-analysis-accuracy-traps