CIS 700: Homework 3

Shubhankar Patankar

February 16, 2020

1 Sentiment Analysis

I used the TextBlob sentiment analyzer to predict the sentence with the more positive content as the sentence to end the story with. Using this simple scheme, the classifier performs better than chance:

- 2016 validation accuracy: 57.9%
- 2018 validation accuracy: 56.9%
- 2016 test accuracy: 57.7%

Next, I tried to take advantage of the available context sentences in order to choose a sentence closest in sentiment to the general sentiment of the story. I computed the mean sentiment of the context sentences, and compared the value to the sentiments of the options. Surprisingly, with this scheme, overall performance became worse:

- 2016 validation accuracy: 51.1%
- 2018 validation accuracy: 51.1%
- 2016 test accuracy: 50.2%

This classifier performs no better than chance.

Finally, I inferred that perhaps the last context sentence is a better predictor of the final sentence. Although performance marginally improved over random selection, it did not approach that of the first classifier:

- 2016 validation accuracy: 52.0%
- 2018 validation accuracy: 51.7%
- 2016 test accuracy: 51.8%

While this improves performance over the mean-based prediction, it is far worse than the baseline scheme. Since it barely outperforms random guessing, examples of instances that are incorrectly chosen are unlikely to be insightful.

Some of the stories incorrectly completed include:

```
    {'context': ['Stuart had three days off work.',
   'He wanted to go hunting during that time.',
   'Stuart invited his friends to a hunting trip.',
   'They all went deer hunting for two days.'],
  'label': 1,
  'options': ['Stuart and his friends did not go hunting.',
   'After the trip, Stuart felt much better.']}

{'context': ['Jackson is a painter.',
   'He often paints portraits for family members.',
   'One day a friend approaches him and asks him to paint his mother.',
   'Jackson happily obliges.'],
  'label': 1,
  'options': ['Jackson stops being friends with him.',
   "Jackson's friend loves the painting."]}

 {'context': ['Bill used to always love going to Cape Cod.',
   'He loved being by the ocean.',
   'But he also loved all the fresh fish.',
   'He would always go to same restaurant.'],
  'label': 0,
  'options': ['He liked to order the "Catch of the Day."',
   'The chef there was notorious for serving spoiled fish.']}
```

2 BERT Embeddings

2.1 Model Architectures

Default Architecture

- Layer 1: 512 dimensional, ReLU activated layer
- Layer 2: 768 dimensional, linear projection to BERT embedding size

Performance:

- 2016 validation accuracy: 63.8%
- 2018 validation accuracy: 64.1%
- 2016 test accuracy: 63.4%

Switching to a neural network already considerably improved performance. Next, I tried models with different neural network architectures. The simplest change to make was to add another layer and vary the activation function.

First Different Architecture

- Layer 1: 512 dimensional, ReLU activated layer
- Layer 2: lower dimensional projection to 256 dimensions, ReLU activated layer
- Layer 3: 768 dimensional, linear projection to BERT embedding size

Performance:

- 2016 validation accuracy: 62.9%
- 2018 validation accuracy: 62.8%
- 2016 test accuracy: 64.0%

Second Different Architecture Next, I tested a different non-linear activation in the second layer going from the ReLU to the softplus.

- Layer 1: 512 dimensional, ReLU activated layer
- Layer 2: lower dimensional projection to 256 dimensions, softplus activated layer
- Layer 3: 768 dimensional, linear projection to BERT embedding size

Performance:

- 2016 validation accuracy: 65.3%
- 2018 validation accuracy: 65.1%
- 2016 test accuracy: 64.6%

The two variants on the default architecture only marginally outperform it.

2.2 Model Hyperparameters

Since the model with softplus activation in the second layer seems to improve performance the most, I tested hyperparameter choices using this model.

First Hyperparameter Setting

In the first variant of the hyperparameter setting, I doubled the number of candidate 5th sentence classifiers that the model is required to choose between; going from 50 to 100. This considerably lowered performance despite using my best performing neural network architecture from above. Performance:

- 2016 validation accuracy: 48.3%
- 2018 validation accuracy: 47.5%
- 2016 test accuracy: 46.6%

Going the other way and reducing the number of candidates to match the batch size has a similar but less adverse effect. Performance:

- 2016 validation accuracy: 57.6%
- 2018 validation accuracy: 57.4%
- 2016 test accuracy: 58.9%

Intuitively, doubling the number of choices the model has to make ought to help the model not overfit. However, since the original model has a poor test accuracy, causing it to further under fit only has adverse effects.

Second Hyperparameter Setting

Next, I returned to the default settings in all hyperparameter choices but for the learning rate. I changed the learning rate to be 10 times higher from 0.001 to 0.01, which had an adverse effect: Performance:

- 2016 validation accuracy: 51.5%
- 2018 validation accuracy: 51.2%
- 2016 test accuracy: 51.3%

3 BERT Embeddings + Validation Data Training

The default architecture yields a test accuracy of 67.8%, which is better than any of the models that train only on the training data. This is unsurprising considering that the model now has many more training examples available to train on from the validation set.

Variant 1

In this variant, I used a model with the following architecture:

- Layer 1: 512 dimensional, ReLU activated layer
- Layer 2: lower dimensional projection to 256 dimensions, softplus activated layer
- Layer 3: 2 dimensional layer signifying binary classification

Hyperparameters were retained at their default values. With this model, performance improves by approximately 3% on the baseline to 70.9%.

Variant 2

In this variant, I retained the model architecture from Variant 1, and instead varied the number of validation data examples that the training process uses. Whereas the default model and the model in Variant 1 use 5000 validation data points during training, Variant 2 uses 2500. This significantly worsens performance to 51.3% indicating that the validation data split ought to be more towards using 5000 data points than towards none, confirming the need to peek into the validation set.