# NLP HW2: Text Classification

Zhang Naifu 2018280351
znf18@mails.tsinghua.edu.cn
funaizhang@github

Tsinghua University
May 8, 2019

## About

In this homework, we perform sentiment analysis on the *stanfordSentimentTreebank* with a **CNN**.

This folder contains the following.

- model.py - contains *build_cnn()* for building the model
- train.py - contains *train_model* for training the model, and the actual execution of the many variations of models
- test.py - contains *eval_models()* for testing the trained models
- utils.py - misc helper functions
- results.csv - summary of results
- plots - plots of training & validation losses & acc

## 1 Model Description

All the following model variations adopt this **CNN** architecture, similar to the one used in [1].

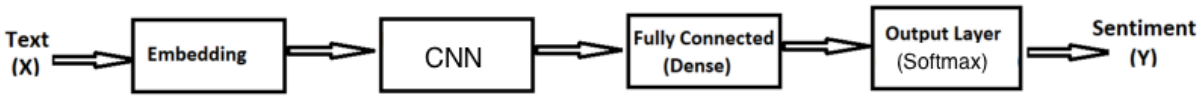


Figure 1: architecture

There are 4 parameters to vary, (namely: no. of hidden layers, hidden size, whether dropout is used, whether Glove embeddings are used). On top of this, I have also looked at regularizing the model. There are hence $2^5 = 32$ model variations in total.

In theory, a complete grid search should be performed over all variations. However, in the interest of brevity, I have studied only the few notable ones below. I have used *model 1* as a benchmark for the experiments.

| model | hidden_size | hidden_layers | dropout | regularize | trainable_embeddings |
|---|---|---|---|---|---|
| model1 | 512 | 1 | 0.2 | 0.05 | F |
| model2 | 256 | 1 | 0.2 | 0.05 | F |
| model3 | 512 | 3 | 0.2 | 0.05 | F |
| model4 | 512 | 1 | None | 0.05 | F |
| model5 | 512 | 1 | 0.2 | None | F |
| model6 | 512 | 1 | None | None | F |
| model7 | 512 | 1 | 0.2 | 0.05 | T |

All models are trained with the following hyperparameters. The limited attempts made to fine-tune these show that varying them within the reasonable range does not really impact results.

| Hyperparameters | |
|---|---|
| Batch size | 128 |
| Epochs | 10 |
| Max. no. of words per sentence | 60 |

## 2 Results and Discussion

Results are reported in the table below.

| model | train_acc | val_acc | test_acc |
|---|---|---|---|
| model1 | 93.11% | 42.51% | 44.52% |
| model2 | 88.03% | 44.05% | 44.03% |
| model3 | 26.94% | 26.25% | 28.78% |
| model4 | 95.21% | 43.23% | 44.89% |
| model5 | 96.10% | 42.23% | 43.57% |
| model6 | 97.02% | 41.14% | 42.26% |
| model7 | 96.06% | 42.23% | 46.38% |

Hidden size (*model 2*): 512 performs better, but really not much difference.

Hidden layer (*model 3*): Having 3 hidden layers hinders *test_acc* hugely. This should not be the case - it is possible that fiddling with the adjacent layers in the model could alter this result.

Dropout and Regularization (*models 4-6*): These prevent models from overfitting. Having these parameters set to non-zero improves *test_acc*. *model 6* is especially plagued with overfitting, with a high *train_acc* and low *test_acc*.

Trainable embeddings (*model 7*): This is in theory equivalent to not using pre-trained embeddings, but with a different initialisation. This model achieves the highest *test_acc*, meaning that using no pre-trained embeddings would eventually yield the best performance.

The loss and acc graph of *model 1* is plotted below. Please refer to "*plots*" folder for other models.

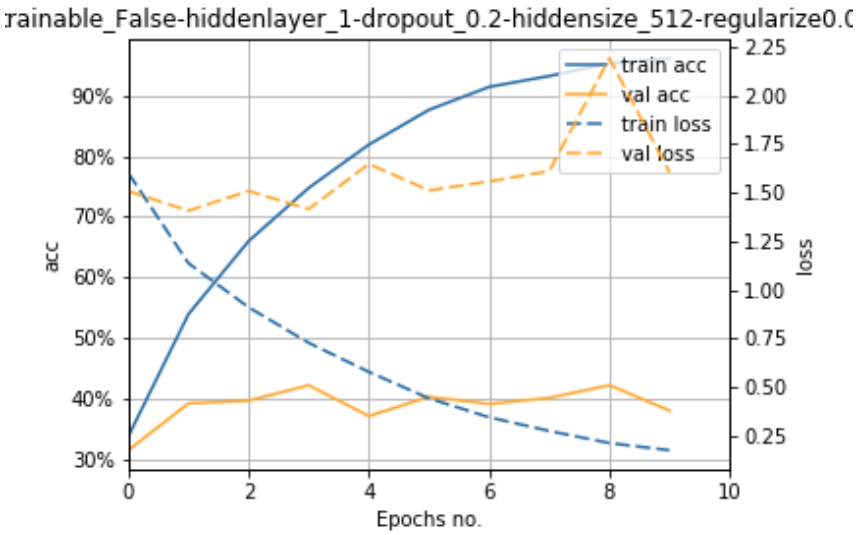


Figure 2: Training & validation loss & acc

# 3 References

[1] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification.

[2] Chollet, F. (2016). Keras blog. *https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html*