Artificial Intelligence "Option C" Coursework (SET09122)

Adel Luong

School of Computing, Edinburgh Napier University, Edinburgh UK
40454035@live.napier.ac.uk

1 Text Representation Technique

1.1 Word embeddings

The strength of embeddings lies in their ability to capture similarities in word meanings. The advantage of using words embeddings is their potential to detect and classify out-of-context word that are not included in the training data [1]. Word embedding is a distributed text representation technique, meaning the representation of a word is not independent or mutually exclusive of another word and their configurations often represent various metrics and concepts in data. The information about words is represented along real-valued vectors, whereas in discrete text representation, such as one-hot-encoding or bag-of-words, each word is considered unique and independent of each-other [2]. Embeddings map semantically similar words close on the embedding space. Since the purpose of our model is sentiment analysis, I chose to use word embeddings as they should prove to be a powerful tool for the above-mentioned reasons.

1.2 Word2vec

I chose Word2vec as the model's algorithm for representing word embeddings. In 1.1 I've mentioned how mapping works in embeddings. Positive words which are adjectives will be closer to each other and vice versa for negative adjectives [3]. Word2vec is a method where neural embeddings model is used to learn how to derive these vectors. I chose this algorithm specifically because of its ability to learn non-trivial relationships between words and its size of the embedding vector is very small. Word2vec uses two different architectures, CBOW (Continuous bag of words) and Skip Gram. I chose to implement the skip-gram method, for it can capture two semantics for a single word [4]. I also used Gensim, which is an open source python library that enables developing word embeddings by training our own Word2vec models on a custom corpus. In my Word2vec model, the training algorithm is set to skip-gram (sg=1). I trained my own W2v embedding with 10 epochs. To pass word2vec into the embedding layer in Keras, I created a vocabulary then an embedding matrix. One of the downsides of word2vec is it cannot handle out-of-vocab words very well, so these words will have to be assigned to a vector of all zeros in the matrix.

2 Machine Learning Model for Text Classification

2.1 LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture which allows better SGD (Stochastic Gradient Descent) training by better control of back-propagation gradients over large number of network layers. LSTMs can be stacked to multiple layers to improve their representation power, which is what we did by setting the model to sequential. First the word2vec embedding layer is added to the model. Then two LSTM layers are added to the model, with the dropout set to 0.4. Dropouts prevent overfitting by setting input units to 0 at random with a rate frequency at each step during training time. A flatten layer is added next, which merges all visible layers into the background to reduce file size. Finally, a dense layer is added. Here, we set the activation function to sigmoid and implement the L2 regularizer, which applies penalties to layers. We compile the model with the optimiser set to Adam, a type of adaptive optimiser that doesn't require a tuning of the learning rate value and is said to be the best among the adaptive optimisers [5]. I chose binary cross-entropy as the loss function as it is a binary classification loss function, which is said to be the default and preferred loss function for sentiment analysis problems [6]. Metric is set to accuracy. The sequential model consists of five layers in total which can be seen in the Tensorboard graph (See Fig. 1) and the plot made with Keras's own model plot utility (See Fig. 2).

3 Model Evaluation

With Adam, tuning of hyperparameters is not necessary, but I chose to do so to experiment and get the best result possible. I tried different epoch and batch sizes, the larger ones seemed to cause overfitting which is an error in statistics and can reduce the model's predictive power. When plotting the value of validation loss against accuracy at each epoch (see Fig. 3), the loss value increases, which is an indication of overfitting and the model isn't learning. Evaluating the model (see Fig. 4), we can also see that the difference between accuracy, precision and recall values of training and testing is 20% on average. When training accuracy is higher than testing accuracy, it means the model is overfit.

To find the best epoch size for this specific model, I implemented a callback called EarlyStopping. It monitors the performance measure and stops the training process when the model stops improving. The callback seemed to get triggered after one epoch (see Fig. 5). The evaluation metric scores of training and testing look a lot better with early stopping (see Fig. 6), I decided to focus on accuracy metrics wise, the difference is down to less than 10%. The model's performance is almost identical now on the testing and validation set (see Fig. 7) after implementing all the preventative measures against overfitting. I tried to find better fitting hyperparameters and tunings to increase its performance by using k-fold cross validation, compare different models on our training set. Cross validation also gives more accurate results of our model, which helps recognising under/overfitting. Using 5-fold cross validation, I got around 75/86% for loss/accuracy, my aim is to get it as close as possible to the values got from the evaluation without cross validation. However, the loss seems to be staying the same during the training within cross validation (See Fig. 8). Due to time limitations, I could not find a solution to this issue, as I have implemented many functions to avoid overfitting in the model, my only guess is the problem lies in the data normalisation process. The model performs slightly differently each time it is ran, generally about 72% testing and 85% training accuracy.

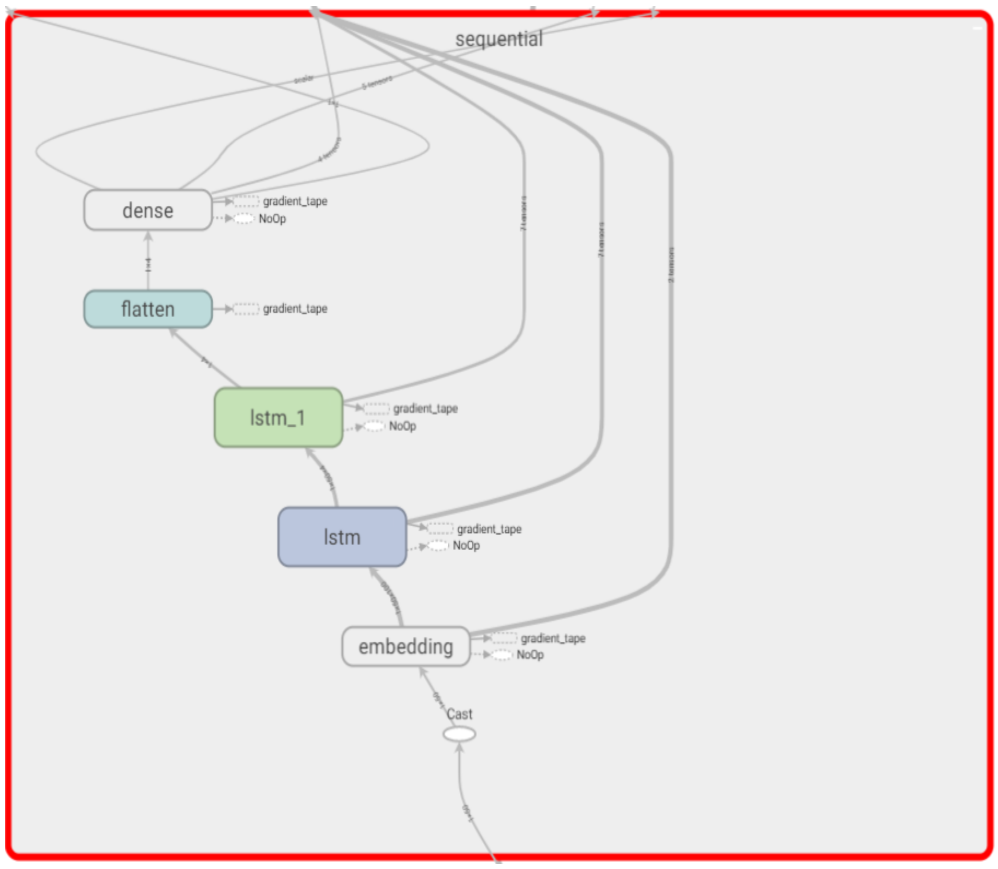


Fig. 1. Model Structure

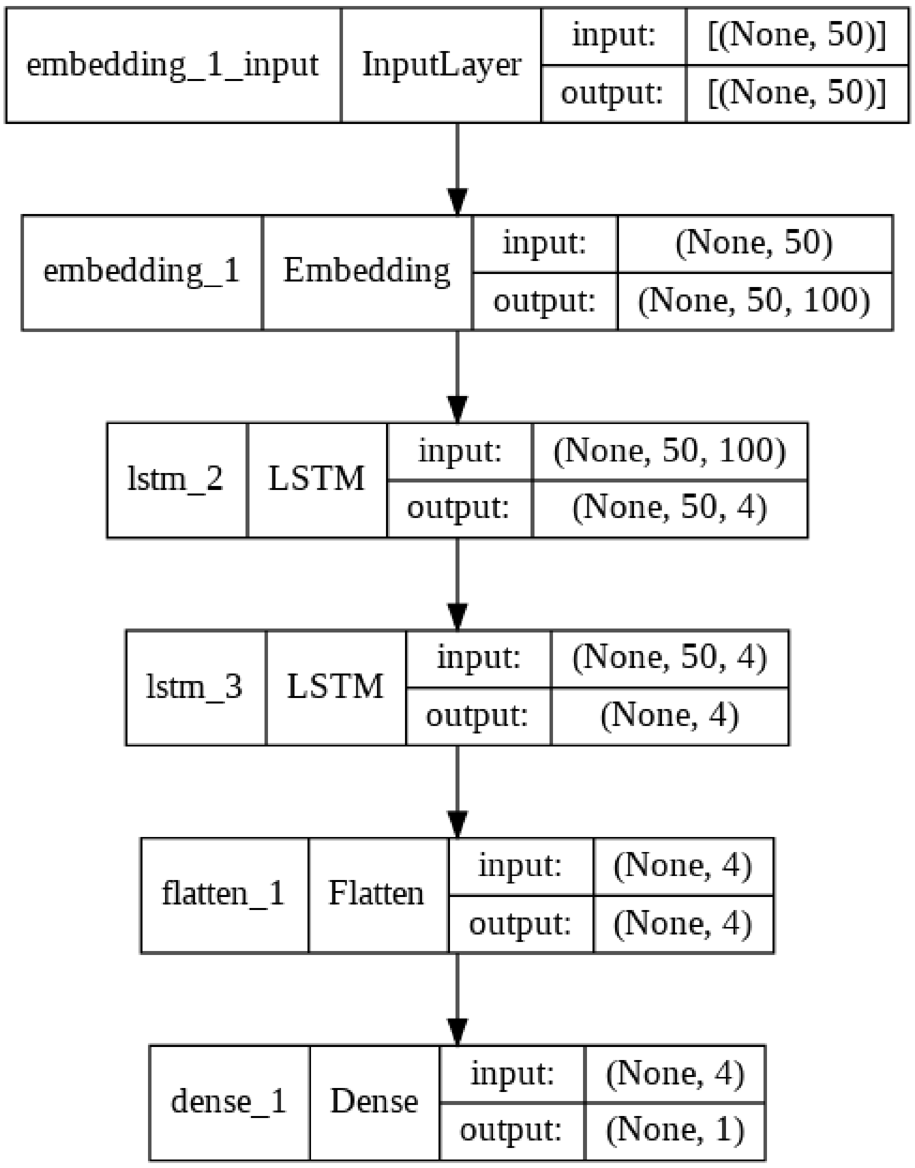


Fig. 2. Model Architecture

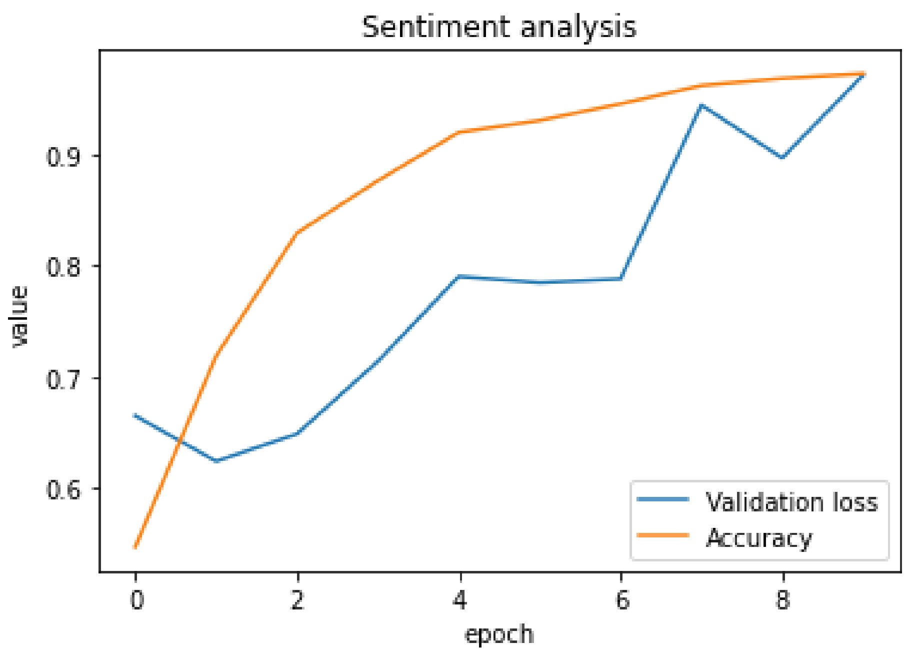
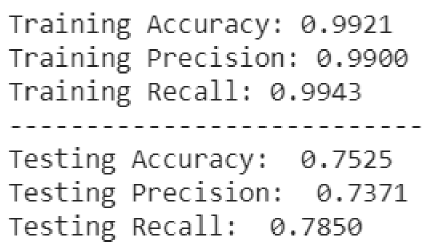


Fig. 3. Validation Loss vs Accuracy while Overfitting



```
Training Accuracy: 0.9921
Training Precision: 0.9900
Training Recall: 0.9943
---------------------------
Testing Accuracy:  0.7525
Testing Precision:  0.7371
Testing Recall:  0.7850
```

Fig. 4. Evaluation of overfit model

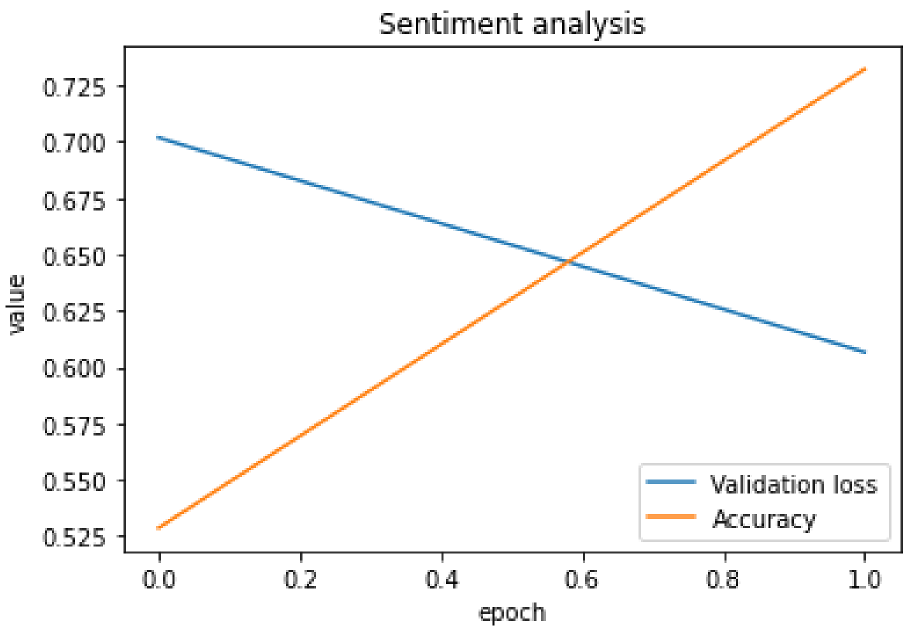
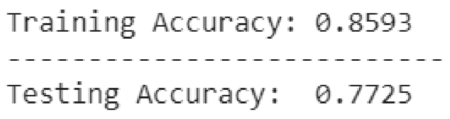


Fig. 5. Validation Loss vs Accuracy with EaryStopping



```
Training Accuracy: 0.8593
----------------------------
Testing Accuracy:  0.7725
```

Fig. 6. Evaluation of Model with EarlyStopping

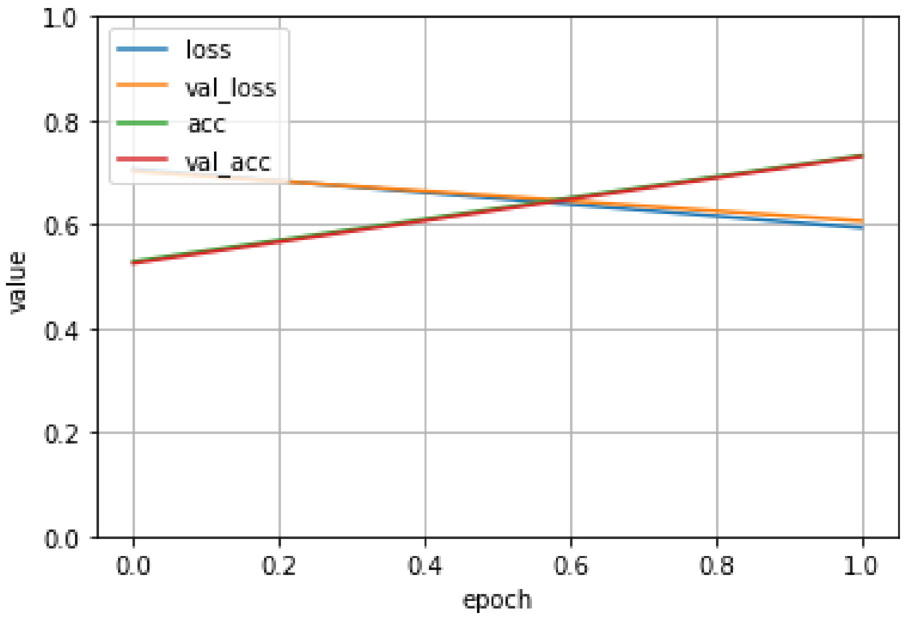


Fig. 7. Training vs Validation metrics

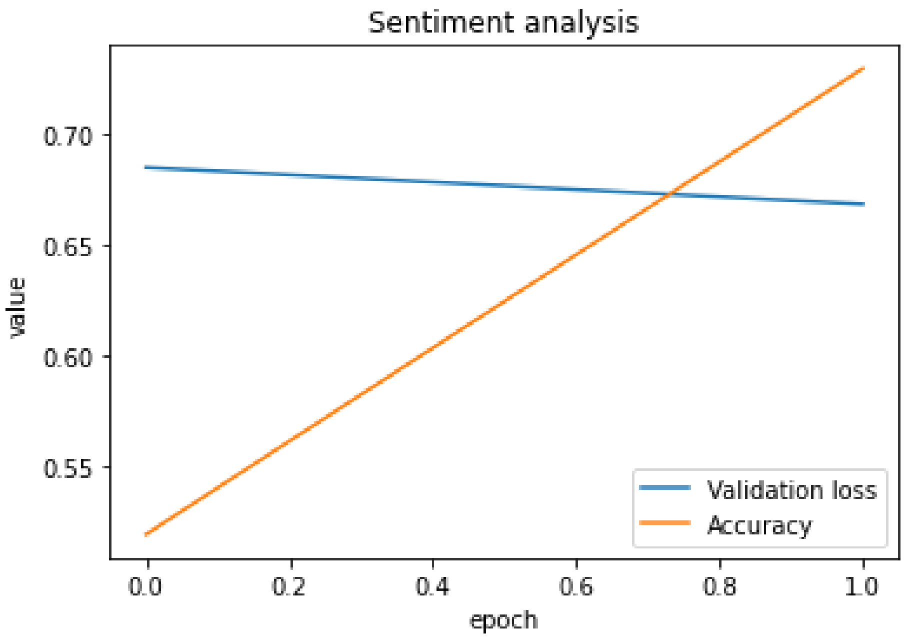


Fig. 8. Val loss vs Accuracy w. Cross Validation

References

1. Elena Rudkowsky, Martin Haselmayer, Matthias Wastian, Marcelo Jenny, Štefan Emrich & Michael Sedlmair (2018) More than Bags of Words: Sentiment Analysis with Word Embeddings, Communication Methods and Measures, 12:2-3, 140-157, DOI: 10.1080/19312458.2018.1455817, viewed 21 November 2021, https://www.tandfonline.com/doi/full/10.1080/19312458.2018.1455817
2. Sundaresh Chandran, 12 July 2020, Introduction to Text Representations for Language Processing — Part 2, viewed 21 November 2021, https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-2-54fe6907868
3. Dipika Baad, 2 March 2020, Sentiment Classification using Word Embeddings (Word2Vec), viewed 21 November 2021, https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca
4. Zhi Li, 30 May 2019, A Beginner's Guide to Word Embedding with Gensim Word2Vec Model, viewed 21 November 2021, https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92
5. Davide Giordano, 25 July 2020, 7 tips to choose the best optimizer, viewed 21 November 2021, https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e
6. Jason Brownlee, 30 January 2019, How to Choose Loss Functions When Training Deep Learning Neural Networks, https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/