Subjectivity Classification Experiment
-Ritwik Kulkarni

1. **Introduction:**

This experiment implements and measures the performance of two neural network architectures of a subjectivity classifier model. Subjectivity classification, in its simplest form, pertains to the binary classification of text as subjective or objective. By definition, subjective texts contain opinions, beliefs or view points of either the author or a third party. On the other hand, objective text is supposed to contain factual information. For example:

- Subjective: "Mango is the best fruit in the world"
- Objective: "Mango is a tropical fruit occurring in different parts of the world"

This classification however, is not a trivial difference in how the text is expressed. Just as in most Natural Language Processing tasks, humans have a multitude of ways to express a thought and make use of the combinatorial properties of syntax and semantics (Kulkarni *et al* 2013). Literature shows that this issue has been studied in several ways which use varying techniques including probabilistic models, graphical methods and neural networks. At the core of this task lies 'feature extraction' i.e. to obtain features from the text that can reliably classify a text as subjective or objective. Features can either be learned or be manually defined as "rules". Since this is a classification task, it requires annotated data which is used to train an algorithm.

2. **Dataset**:

The dataset consists of 10,000 sentences with a 50-50 split into subjective and objective sentences. The sentences were compiled by Pang *et al* 2004, and obtained from movie reviews where sentences from reviews were considered as subjective while sentences from movie plots were considered to be objective. More information about the data can be found at http://www.cs.cornell.edu/people/pabo/movie-review-data/.

For the purposes of this experiment the data is already downloaded and the raw files can be found in the packaged project file under 'data/'. The following set of pre-processing steps are applied to the text, a) removal of non alpha-numeric characters, b) converting all text to lower case, c) tokenising the sentences into words tokens, d) removal of stopwords (as provided by the nltk library). The data is split into 80% train and 20% test sets.

3. **Models:**

We evaluate the performance to two neural network architectures which are known to show decent performance on text classification tasks. The two architectures are
a) a combination of a Convolutional Neural Network (CNN) and a Long Short Term Memory network (LSTM) and
b) a multi-channel CNN model.
Both the models have been implemented in more or less their vanilla forms.

a) CNN+LSTM:

This model has a CNN at its base that uses a single kernel with 1000 filters and a stride of 1. The input layer is an embedding layer of size 128 which takes one hot encoded input vectors. The information from the filters, after a max-pooling operation is passed on to an LSTM network of hidden layer size of 128. Finally a feed-forward network compresses the final state of the LSTM into a single unit binary layer to obtain the final output. Dropout regularisation is used at various stages of the network including the recurrent weights. The model is trained for 20 epochs with a batch size of 100. Performance of the model is measured using 4 metrics, namely, 'Accuracy', 'Precision', 'Recall' and 'F-score'. A mean over these metrics is calculated over a 10-fold cross validation experiment.

b) Multi-channel CNN:

This model implements a three channel CNN network where each channel independently learns filters over the input text and finally combines their outputs as a concatenation operation. A fully connected feed-forward network then compresses the layers, as in the previous model, into a single unit binary layer. As before, a one-hot input is transformed into an embedding layer for each of the channel. The three channels use different kernel sizes of 1, 3 and 5 respectively and 640 filters each. Thus, effectively the model considers unigrams, trigrams and pentagrams as base features. The model is trained for 20 epochs with a batch size of 100. Performance is again measured in the same way as a mean over a 10-fold cross validation of 4 metrics mentioned in the previous section.

4. **Results**:

Performance of the two architectures is shown in the table below. Each value is the mean over a 10-fold cross-validation with the standard deviation reported in brackets.

Model	Accuracy	Precision	Recall	F-score
CNN+LSTM	0.73 (0.05)	0.75 (0.07)	0.73 (0.08)	0.73 (0.05)
Multi-channel CNN	0.64 (0.05)	0.62 (0.07)	0.69 (0.15)	0.64 (0.07)

5. **Discussion:**

Both models perform above the chance level, which is 0.5 for this task, indicating that they do capture some of the basic features required to separate the sentences into subjective and objective. In other words the models learn to recognise stable patterns in the data which separate the two categories. However for a binary classification task the results are quite low to get a reliable prediction in a large scale system. Moreover, certain experiments in the literature have obtained an accuracy over 0.9, indicating there is much scope of improvement and investigation in the internal processes of separating the two domains.

One of the impressive architectures is presented by Liang *et al* 2016 by using an assymetric convolutional bi-lstm network (AC-biLSTM). The focal point of the work is in transforming the dimensions of the output of a multi-channel CNN so that all the resultant outputs are of the same dimension. This allows all the channel outputs to be combined as a single sequential stack which can then be fed into a bi-directional LSTM network.

Implementation ‘a’ in this experiment (CNN+LSTM) is a crude “baby” version of the AC-biLSTM model. It uses a single channel and a uni directional LSTM which vastly reduces the complexity of the network but with a heavy cost in performance. Applying a bi-LSTM in this model, in fact, resulted in a reduced performance by 3%. The multi-channel CNN on the other hand, despite using three different kernels resulted in a reduced performance compared to the CNN+LSTM model, pointing towards the possible importance of considering sequential information in classifying the text. Both these models, however, have been implemented in their basic forms. There is much scope to explore the performance of the models in terms of hyper-parameter tuning, regularisation techniques, optimisation functions and other design considerations such as bottle-necking. There are different ways to tokenise input text, one can consider character level inputs such as character n-grams or a combination of character and word inputs. Such variations are known to affect model learning abilities. Another variation can be the use of pre-trained embeddings such as word2vec, Glove etc which carry a lot of semantic information in the base vectors.

It may be worthwhile to consider traditional NLP techniques to extract features manually from the data, using dependency parsing, semantic role labelling etc. There is an advantage of using hand crafted features in addition to using “discovered” features, in many cases as we are able to target specific areas of the text that are potentially useful. The dataset used in this experiment may contain some specific features, e.g. the use of adjectives and their frequencies in distinguishing subjective and objective text.

As in any modelling experiment, everything starts with the right data. It is crucial to obtain real life data in order for the models to be applicable “on-field” as working smart systems. It is very likely that the current models trained in this experiment, exclusively only applies to this movie review database without having wider applicability to distinguishing between subjective and objective text. One may consider transfer learning tasks to increase the scope of the models. Another way to obtain real life data is to use crowd-sourcing annotation tools such a Mechanical Turk or conduct a well crafted annotation exercise.

Thus to conclude, we implemented two architectures to classify the subjectivity movie review database and found the CNN+LSTM architecture to perform better than the multi-channel CNN. Both the architectures despite being able to successfully pick up patterns in the data, require further investigation to improve performance.

References:

Kulkarni, Ritwik, Susan Rothstein, and Alessandro Treves. "A statistical investigation into the cross-linguistic distribution of mass and count nouns: Morphosyntactic and semantic perspectives." *Biolinguistics* 7 (2013): 132-168.

Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, p. 271. Association for Computational Linguistics, 2004.

Liang, Depeng, and Yongdong Zhang. "AC-BLSTM: Asymmetric Convolutional Bidirectional LSTM Networks for Text Classification." *arXiv preprint arXiv:1611.01884* (2016).