SENTIMENT ANALYSIS

CLEANING

Inspection immediately showed that there was an imbalance in the training data. Positive reviews outweighed that of the negatives. The simplest solution was to take a random sample of the positives and combine with the negatives to give an even distribution of both. Duplicates were removed (rows which had identical sentences. I noticed the given sentiment was sometimes questionable:

"Matthew the chap who dealt with me was lovely but he had a nightmare my initial policy did not go through so after a lot of time and calls he did a refund and a week later we had to go through the whole procedure again."
= POSITIVE PROCEDURE

This highlights the objectivity of the dataset. Even though the staff member comes across as positive, and the procedure comes across as negative, the entire review is marked as positive. It is also clear that the entities given for each review are found via POS tagging (probably from NLTK or stemming), or some other entity-detection algorithm, as it has just picked up a noun of its own choosing:

"Also and this applies to all insurance companies it is very very annoying to have to to thru this procedure every year!!"
= NEGATIVE YEAR

I would mark the above entity as "insurance company", not "year". On the whole this is not a very clean dataset, which will have adverse effects in the coming stages. But at this stage I will treat it as ground-truth.
The dataset after distribution-balancing and duplication removal results in 1683 positive rows and 1683 negative rows.

TEXT NORMALISATION

In order to build a reliable vocabulary for the training, I normalised the text. That is, every word in the corpus of reviews has been converted to lower-case, had its punctuation removed, and lemmatized. Lemmatization normalizes the text because it removes all past and future tenses of a verb. I removed text contractions as a way to simplify text and correct potential spelling mistakes. I have opted to keep stop words in for this experiment as this can interfere with the sentiment of a sentence.

TOKENIZATION AND NUMERATION

In order to feed text into the neural net, it needs to be in a numeric form. I use Pandas dataframe functions to compute the maximum length of the sentences (maximum number of words in an entry), and also the number of unique words used throughout the data. By using Keras' inbuilt encoding function to assign each unique word a number, and then extend the length of each sequence of numbers to the length of the maximally long sentence, I am left with a series of lists that correspond to each sentence within the dataset:

- *"find it dishonest that i be charge more for my premium if i choose the highest voluntary excess"*
- *[9, 2975, 2397, 2234, 862, 2900, 2426, 387, 1874, 2991, 1618, 1242, 862, 2916, 2077, 567, 3088, 2023]*
- *[9, 2975, 2397, 2234, 862, 2900, 2426, 387, 1874, 2991, 1618, 1242, 862, 2916, 2077, 567, 3088, 2023, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]*

The sentiment of each sentence is also converted to 0 and 1 instead of "negative" and "positive for the same reasons.

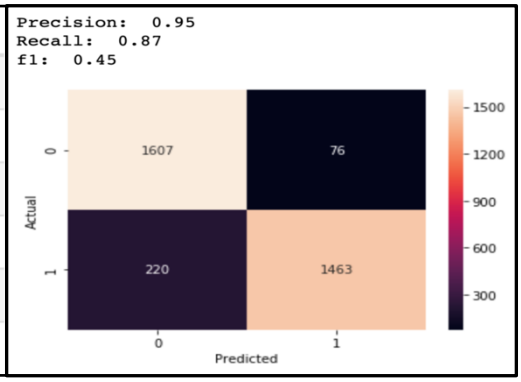
MODEL ARCHETECTURE

With previous experiments in neural net architecture, I realised that the best model is often the simplest. I have therefore opted to use Keras' embedding layer to encode the vectors of the words, without employing a 3rd party vector space. This method was fast and gave reasonably good results. I did not use any recurrent neural nets, nor did I use convolutions. The size and simplicity of the sentences could perhaps warrant it, but as a proof of concept I decided against it.

RESULTS

K-fold stratified cross validation is performed, and then averages computed. The only metric changed was the dimensionality in the Keras embedding layer. I used the early stopping call-back to allow the model to stop learning once validation loss converged. The model frequently converged with validation accuracies in excess of 80%, and validation losses on or below 0.40. Average epoch number for val loss convergence was between 6.2 and 7.4. 80 dimensions yields the overall best metrics:

Dimension	Epoch Avg	Val_Acc Avg	Val_Loss Avg
10.0	7.4	0.827989	0.398360
20.0	7.0	0.828584	0.397525
40.0	6.6	0.830074	0.401919
80.0	6.2	0.837183	0.392007
160.0	6.2	0.833903	0.432894
320.0	6.4	0.830037	0.472401



Example of results (left) showing averaged metrics taken over 5-fold stratified cross validation for differing embedding dimensionalities, and (right) a confusion matrix gained from applying the best model to the training data and comparing ground-truth with predicted.

Sentence	Entity	Prediction
Quick efficient service good product good value.	product	1
more steps required to achieve goals, more people required to achieve benefits eg cinema.	benefits	0
So far fantastic easy to use website and what's I can finally have protected ncd.	ncd	1
its okay but offer for motor insurance was a wild estimate compared with reentering details on website and getting a new quote, about 30% different.	insurance	1
By far the best quote for my first car insurance by miles!!	quote	1

A sample of predictions performed on the unlabelled 'test' dataset. The fourth entry is predicted incorrectly, and the rest are predicted correctly.

Sentence	Sentiment	Prediction	Result
quick and easy to input your details and prices you would not believe!	0	1	FP
He could not have improved the service that he provided today.	1	0	FN
\nWhen it came to renewal AXA's quote was Â£2 more expensive but with better cover and low excess.	1	0	FN
All done online so a relatively painless purchase experience ... no incomprehensible accents bonus.	1	0	FN
service was excellent spoke to alan who answered most of my questions if he didn't know he went out his way to find out.	0	1	FP

Prediction vs ground truth from the train dataset (FP, FN = false positive, false negative respectively). Incorrect predictions are coming from contradicting sentiment within the same sentence. The problem is reflected with multiple entities, whereby the model will incorrectly predict the sentiment of the named entity. RNNs could help with this as the sequence partly dictates the sentiment.

I have experimented with different optimizers and loss functions, but *'rmsprop'* and *'binary_crossentropy'* yielded the best results respectively. I believe limitations in accuracy and loss can be attributed to the following:

- Poorly structured data: objectivity/incorrectly detected entities will yield inaccuracies in the learning of the model. A more robust method is needed to build such data sets.

- Train on a larger dataset/corpus. The model should develop vector-relations with higher granularity and potentially outperform what is shown in this report. Similarly, using other vector spaces from Word2Vec, GloVe could give greater contextual relationships between words.

- Using RNNs would give the model the ability to learn sequences of words, and how that sequence can affect the sentiment. "There was no need to panic" vs "No, I needed to panic" have similar word composition, but the sequence gives vastly different sentiment.

SUMMARY

I would choose to redo this experiment, with a larger, more consistent dataset, and make use of an RNN and Word2Vec/GloVe. Ordering of the words, along with a greater corpus and higher-dimensional vector space would yield more accurate results.