LSTM-based Models for Aspect Embedded Sentiment Analysis

Ethan Dai

CITS4012

Abstract

Aspect-based sentiment analysis can be used to extract the sentiment of a particular aspect from a subject of some sample text such as the quality of food at a restaurant. What makes this task difficult is the potential multipolarity of sentiments contained within a given sample text related to different aspects. Basic LSTM networks fail to capture this nuance within input strings as the hidden states and final output remains constant for any given sentence. Therefore, it has no ability to make representations about aspects within the network. This paper will introduce and test three aspect-embedded LSTM networks: an attentionless aspect-embedded model, an aspect-embedded model with attention, and a Seq2seq style model. The two models that contained attention were the most accurate but displayed some uninterpretable encoding in the attention mechanism.

1 Introduction

Sentiment analysis in NLP is the process of extracting the sentiment of a particular text. This problem was first introduced by Nasukawa and Yi where a primitive method was proposed relying on POS tagging and a sentiment dictionary (Nasukawa & Yi, 2003). The issue with this approach is that it was using a strictly defined method to extract information from a source that is usually fluid in semantic content as the semantic weight and meaning of any given word can shift given the sentence structure. This type of strictly defined hard-coded method gave way to neural networks which were able to capture the relationship between words to extract the sentiment.

The problem this paper will address is that of designing a model that is robust to multipolarity within the input regarding different aspects, as opposed to Nasukawa & Yi's aspect-agnostic mono-polar analysis. An example of these more complex sentence structures could be (keeping in theme with our restaurant review based training data) "the food was delicious, but the service was awful." This sentence expresses two polarities for different aspects, positive for the food but negative for the service.

A standard LSTM model would fail to capture this distinction when performing sentiment analysis as the aspect has no bearing on the network. The last hidden layer output would always be the same regardless of aspect resulting in the same prediction for sentiment. This is where aspect embedding becomes necessary within the network to differ-

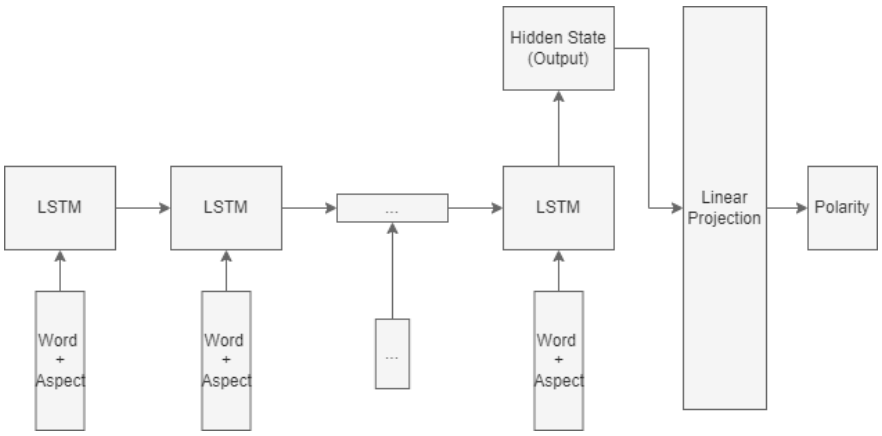

Figure 1: AlAElstm model diagram

entiate between aspects during the classification task. Wang et al. proposed LSTM models with aspect embeddings incorporated into different parts such as concatenating onto the input and onto the output tensors from the LSTM (Wang et al., 2016). The idea is that the network would learn the association between certain permutations of words and a given aspect. This paper will detail the implementation of three different LSTM network designs trained on a dataset comprising of multipolar restaurant reviews. The networks tested will be related as they are ordered as an ablated version of the next model. This paper will then perform a quantitative and qualitative analysis of the two most performant models.

2 Method

The networks designed in this paper include an Attentionless LSTM with Aspect Embedding (AlAElstm), Aspect Embedded LSTM with Attention (AElstm), and Sequence to Attention LSTM (Seq2Alstm). The AElstm network is a modified version of Attention-based LSTM with Aspect Embedding proposed in (Wang et al., 2016).

The AlAElstm model is an ablated version of the AElstm, removing the attention mechanism entirely but keeping aspect embeddings. The Seq2Alstm is a model which builds upon the AElstm and adds a decoder network similar to Seq2seq networks.

2.1 Attentionless LSTM with Aspect Embedding (AlAElstm)

This model is the most basic of the three using purely an LSTM network without attention. LSTM was chosen because it somewhat addresses the issue that RNN has with vanishing gradients over longer input sequences (Hochreiter & Schmidhuber, 1997).

The aspect embedding is concatenated onto the word embedding before being passed into the LSTM, both are which are of embedding dimension d, becoming a tensor $\in \mathbb{R}^{2d}$. After the hidden state is propagated through the recurrent network, the last hidden state is taken as the output of the LSTM and projected through a fully connected network consisting of two hidden layers before outputting a 3×1 tensor predicting the polarity.

The two projections from the last hidden layer h_N, N being input sequence length, to the prediction tensor $y \in \mathbb{R}^3$:

$$L_1 = \tanh(w_1 h_N) \tag{1}$$

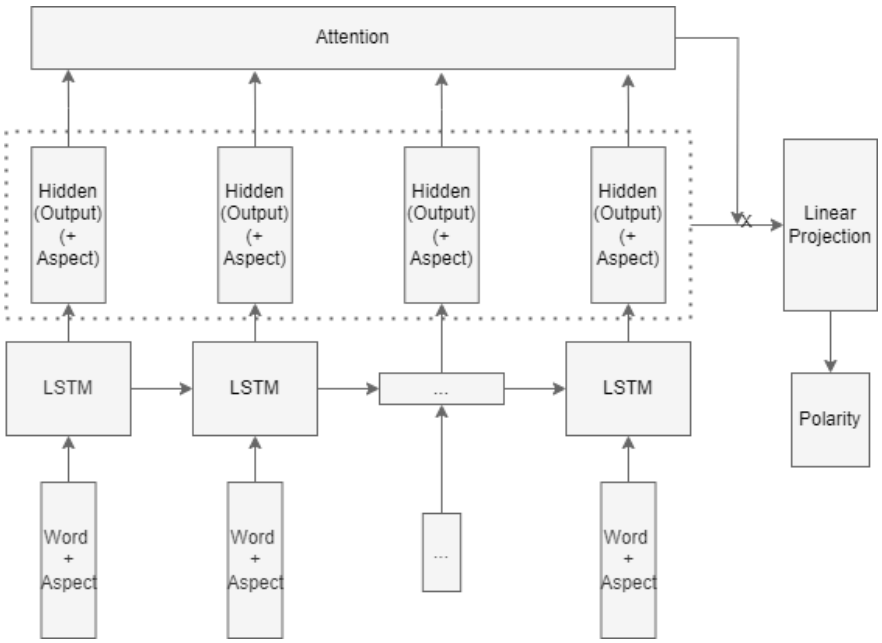

Figure 2: AElstm model diagram

$$y = w_2 L_1 \tag{2}$$

w_1 and w_2 being the linear projection layers within the network. The activation function tanh provides non-linearity in the network to better capture representations within the data.

2.2 Attention-based LSTM with Aspect Embedding (AElstm)

This next model is the unablated version of the AlAElstm model featuring an attention component within the network which allows for the semantic weight of specific words in the sequence to be learned as a discrete part of the network rather than being a portion of the weights of the final output tensor in a pure LSTM. This model is a modified version of the AT-LSTM model proposed by (Wang et al., 2016).

This attention is calculated by concatenating the aspect embedding onto the hidden states forming a tensor $h_n \in \mathbb{R}^{d+d_h}$, d_h being hidden size, before being projected with all other h_n (represented as the combined tensor of all h_n, $M \in \mathbb{R}^{N \times d+d_h}$) to the attention tensor $\alpha \in \mathbb{R}^N$. This α is then multiplied by the tensor of all hidden states H before being projected into the output dimension by $W_y \in \mathbb{R}^{d_h \times 3}$.

$$M = \tanh(W_h H \frown W_v A) \tag{3}$$

$$\alpha = \text{softmax}(wM) \tag{4}$$

$$y = W_y(H\alpha^T) \tag{5}$$

$W_h \in \mathbb{R}^{d \times d}$ and $W_v \in \mathbb{R}^{d \times d}$ are projection matrices to be learned during training which refines the embeddings.

2.3 Sequence to Attention LSTM (Seq2Alstm)

The Sequence to Attention model architecture is similar to the Sequence-to-Sequence architecture (Sutskever et al., 2014) but instead of decoding into a sequence of tokens, only a single decoding step is performed. As

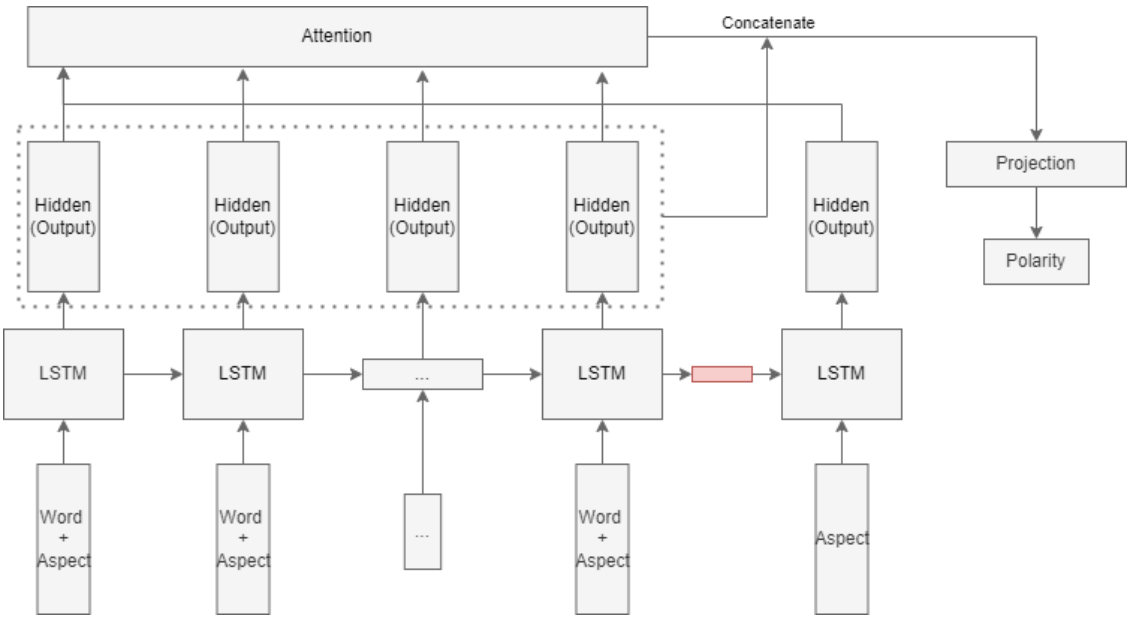

Figure 3: Seq2Alstm model diagram

per the two former networks, aspect embedding is concatenated onto the word embedding when being passed into the LSTM. After the final token of the input is propagated, the final hidden state is passed to the decoder.

The decoder takes only the aspect embedding as the input to produce output h_d and performs a scaled dot product attention with the h_d to produce attention α. Since the gradients can become large, the scaled dot product mitigates the effect of large values being passed into the softmax function for the attention calculation. α then concatenated with the last hidden state h_N, then projected into the output dimension with $W_y \in \mathbb{R}^{3\times 2d_h}$.

$$\alpha = H\frac{h_d^T}{\sqrt{d_h}} \tag{6}$$

$$y = W_y(\alpha \frown h_N) \tag{7}$$

3 Experiments

3.1 Dataset

The dataset used to train and test the models proposed comprises of restaurant reviews. The reviews are mostly multipolar in nature, having different sentiments about different aspects of the restaurant. As stated in the introduction, an example of these reviews could be "the food was delicious, but the service was awful." The same sentence appears as many times as it has aspects in the dataset. Each sentence is tagged with a corresponding aspect and polarity. Thus, the example sentence would appear twice tagged with the aspect *food* paired with *positive* and the aspect *service* paired with *negative*.

Aspect	Pos.	Neut.	Neg.	Total
Food	842	1478	277	2597
Staff	366	141	1041	1548
Service	199	148	368	715
Price	80	153	134	367
Place	150	472	160	782
Ambience	202	56	102	360
Menu	71	413	42	526
Total	2170	3465	2343	7978

Table 1: A breakdown of the training data for the models.

3.2 Experiment Setup

The training set and validation set were combined to form the complete training corpus. In the text preprocessing process, a contractions dictionary was used to expand contractions within the corpus. The punctuation is also discarded, and all words are converted to lower case.

All word and aspect embeddings used are from the pretrained Stanford GloVe 6B 300-dimensional embedding weights. The words are converted to the index reference for the word within the embedding dictionary. The aspects are also converted into the embedding index.

The polarity labels are converted into a target tensor of shape $(1, 3)$ with each position being representative of a classification: $[negative, neutral, positive] \in [0, 1]$.

The hidden size chosen for the LSTM of 300 was mainly as a compromise between representational ability and training time. Networks initialised with smaller hidden sizes performed worse than one with a larger hidden size in terms of prediction accuracy over larger numbers of iterations but trained each iteration more quickly.

The Adam optimiser was chosen, but initially Adagrad was tested as per (Wang et al., 2016). Adagrad failed to converge as prediction accuracy plateaued through training whereas Adam managed to maintain an upwards moving cumulative prediction accuracy across the entire training cycle. The learning rate used was 0.001.

Multi-class Cross-entropy Loss was used as the loss function for the models. The loss takes into account the prediction confidence where unconfident but correct predictions are punished as much as confidently wrong predictions. This is useful for classification tasks as the criterion for minimising the loss function includes not only correct predictions, but also confident predictions.

4 Results

4.1 Quantitative Results

The models' performance was tracked throughout training using both the loss and the cumulative accuracy. Cumulative accuracy is calculated with a running total of correct predictions divided by total samples seen. The unseen test set was then processed by the trained models and the accuracy of prediction was recorded.

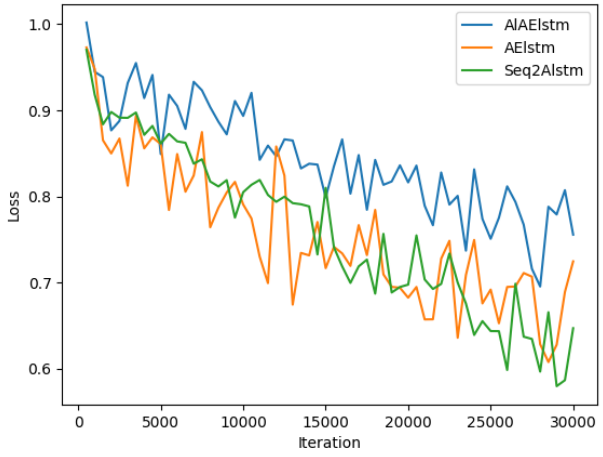

Figure 4: Loss of models over 30,000 training iterations

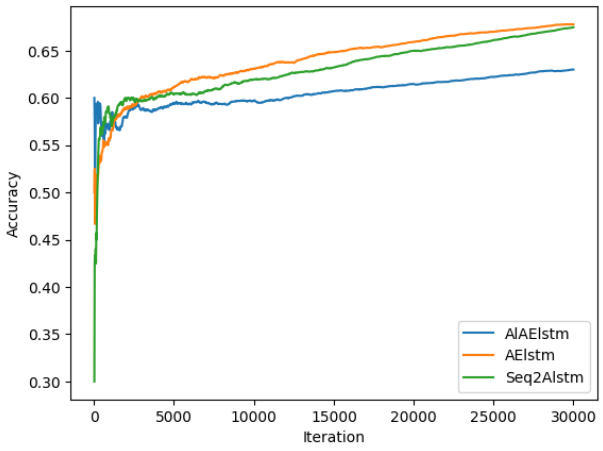

Figure 5: Cumulative accuracy of models over 30,000 training iterations

As explained in the method section above,

the models are ultimately derived from the Seq2Alstm model being ablated versions of it. The AElstm model is the Seq2A without a decoding network, instead using the last hidden state along with attention to predict, and the AlAElstm is the most ablated version, stripping away the attention mechanism leaving only the aspect embedding within the network.

Two important indicators for model performance are the loss and accuracy performance during training as seen in Figure 4 and Figure 5 respectively. Unfortunately, due to time limitations, none of the models properly converged to a local minimum during training. However, the speed of convergence is displayed through Figure 1 where the models with an attention mechanism (AElstm and Seq2Alstm) performed roughly on par as opposed to AlAElstm. Being the model most capable of learning representations as the most complex, Seq2Alstm achieved the most stability during training which may be beneficial for convergence over more iterations as the minimum is reached. It has to be noted that all models were still in the process of converging and over many more iterations it is possible that all models do converge similarly.

Looking at cumulative accuracy during training in Figure 5 yields roughly the same perspective as when looking at loss. However, the cumulative accuracy improvement during training seems to be slowing for AElstm compared to Seq2Alstm and at 30,000 iterations both models performed roughly on par with each other despite the latter performing worse initially.

Model	Accuracy (%)
AlAElstm	65.37
AElstm	70.81
Seq2Alstm	68.59

Table 2: Accuracy for each model on test set after 30,000 iterations in training.

The in-training results are reflected in the performance on the test set which indicates that the models generalised very well. Due to the cumulative accuracy keeping a history of earlier inaccuracies, it is a lagging indicator. The performance of the test set seems to be the cumulative accuracy $+\sim 5\%$.

4.2 Qualitative Results

Figure 6 and Figure 7 are representations of the attention tensor within the AElstm and Seq2Alstm models for a sample sentence. The sentence is "although our waitress was pleasant and accommodating the overpriced food was quite the opposite". The top attention tensor for each pair in the figures includes the aspect embedding for *food*. Even for humans, this sentence structure is rather complex to label in terms of word significance to the semantic meaning. We would have to give weight firstly to the positive sentiment towards the staff, then give weight to the word "opposite".

Both models seem fixated on the word "food" even though it doesn't seem to humans to carry any semantic content regarding the sentiment about food. Despite giving almost no weight to "opposite" or "although", the AElstm confidently predicted correctly that the polarity was negative, possibly giving great weight to "overpriced" (which may have been associated with negative even though there's a separate aspect for *price*). The Seq2Alstm model did seem to capture the semantic importance of words in the sentence when looking at its attention tensor, where it considers the word "although" and "opposite". However, the Seq2Alstm model predicted (albeit with low confidence) that the sentiment was neutral.

When looking at the bottom example of both pairs, the aspect embedding being *staff*, both models predicted extremely confidently (both at above 85%) that the sen-

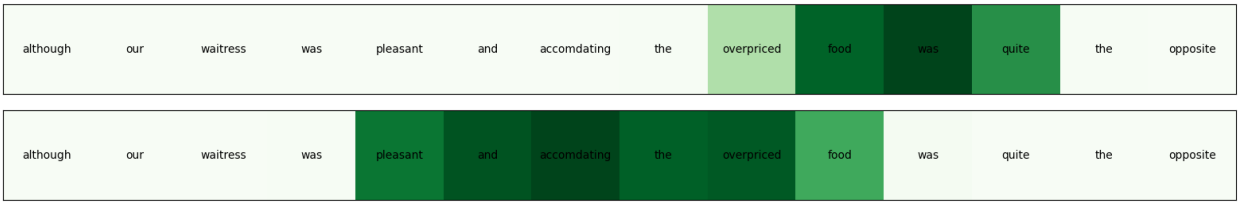

Figure 6: AElstm attention weights for a sample sentence. Top aspect: food; bottom aspect: staff.

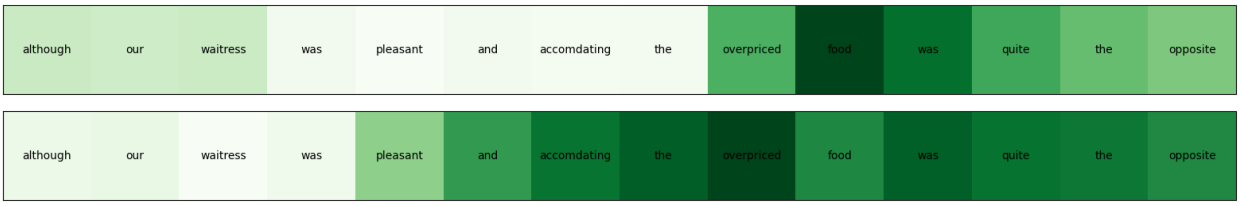

Figure 7: Seq2Alstm attention weights for the same pair of sample sentences.

timent was positive. When looking at the attention however, the AElstm seems to be more concentrated on the semantically important words, "pleasant" and "accommodating", which makes its prediction seem reasonable. However, the Seq2Alstm focused on "overpriced", albeit with emphasis on "pleasant" and "accommodating" as well. This attention is harder to explain. This could likely be a product of a lack of sufficient training iterations, or the models may have learned different semantic connections between words than humans.

5 Conclusion

Due to a lack of time as well as computing resources, the results weren't as conclusive as would be desired. However, it is conclusively evident that attention is a major aspect of sentiment analysis, at least in expediting the convergence of the model.

Seq2Alstm is at the best of my knowledge a novel implementation of the Seq2seq model architecture in sentiment analysis. It performed on par with AElstm which is an ablated version of AT-LSTM (Wang et al., 2016) which removed some linear projections.

When the results are extrapolated, it would seem that Seq2Alstm performs better than AElstm given more training as it seems to perform the gradient descent in a more stable manner. Its cumulative accuracy was also improving at a rate greater than that of the AElstm model at 30,000 iterations. However, this hypothesis will need to be tested in a future paper where the models are trained to convergence.

References

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. *Proceedings of the 2nd international conference on Knowledge capture*, 70–77.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with

neural networks. *Advances in neural information processing systems*, *27*.

Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016). Attention-based lstm for aspect-level sentiment classification. *Proceedings of the 2016 conference on empirical methods in natural language processing*, 606–615.