CITS4012 Project Report

Allison Lau (23123849), Peter Le (23193249)

Abstract

Aspect-based sentiment analysis (ABSA) is an important step in understanding sentiment in opinionated text. This project focuses on ABSA in human-written restaurant reviews, where the sentiment towards different aspects of a restaurant (e.g., food, service, ambience) needs to be considered. We propose a Long Short-Term Memory (LSTM) sequence-to-sequence (seq2seq) model with attention for ABSA, aiming to effectively incorporate aspect information into the sentiment classification process. Three different methods for integrating aspect into the models are explored: appending the aspect word to the end of the review sentence (Model 1), concatenating aspect embeddings with the word embeddings and the sentence hidden representations for computing attention weights (Model 2), and processing sentence and aspect embeddings separately then applying interactive attention mechanisms and finally concatenating both final representations (Model 3). Experimental results demonstrate that integrating aspect information into the model improves sentiment classification accuracy, with Model 3 outperforming Model 2, and Model 2 outperforming Model 1. These findings illustrate the importance of aspect integration for effective ABSA.

1 Introduction

There is a large amount of opinionated text generated daily, where people express sentiments towards various subjects such as survey responses, user reviews, online comments, and social media posts. Understanding the sentiment in this text is crucial for gaining insights into people's preferences and behaviours, and it supports decision-making in many domains. This project focuses particularly on human-written restaurant reviews. The sentiment polarity of a review is not only determined by its content but is also highly related to the specific aspect under consideration. Aspect-based sentiment analysis (ABSA) is a challenging task in Natural Language Processing (NLP) that aims to detect the sentiment polarity towards different aspects in the reviews. This task is challenging due to the need to understand the context and nuances of language to accurately capture the important contextual information for each aspect in a review. Failure to accurately capture this contextual information can lead to incorrect sentiment predictions. Therefore, correctly identifying and analysing contextual information for specific aspects in reviews is crucial for effective ABSA.

Sequence-to-sequence (seq2seq) processing is an NLP approach that involves converting an input sequence into an output sequence. In this project, we aim to design a model which can employ aspect information for sentiment classification. Hence, we implemented a Long Short-Term Memory (LSTM) seq2seq (many-to-one) model with attention for ABSA. We explored three different methods for integrating aspects into the model:

1. **Model 1**: Append the aspect word at the end of the review sentence.
2. **Model 2**: Concatenate aspect embeddings to the word embeddings before passing them through the LSTM network, then concatenate aspect embeddings with the sentence hidden representations for computing attention weights.
3. **Model 3:** Process sentence and aspect embeddings through separate LSTM networks, apply attention mechanisms to combine them interactively, and concatenate both representations.

We found that integrating aspect information in our model improves sentiment classification accuracy, with Model 3 performing the best, followed by Model 2 performing slightly better, and Model 1 performing the least effective. This confirms the importance of aspect integration for the task of ABSA.

1.1 Related Work

ABSA can be regarded as a text classification problem. Traditional classification methods like Support Vector Machines (SVMs) [1] can perform sentiment classification without considering the aspect. Then,

neural network methods such as LSTM, Gated Recurrent Unit (GRU), and Recurrent Neural Networks (RNNs) have shown promising results in sentiment analysis. However, these methods may overlook the contextual importance of the aspect which is crucial for accurately identifying the sentiment polarity of a review with regards to a particular aspect [2].

To incorporate the aspects into the model, Tang et al. [3] extended LSTM by using two LSTM networks, one forward and one backward, to capture the context surrounding an aspect term. Another approach proposed by Wang et al. [4] is the Attention Aspect Embedding (ATAE)-LSTM method, which combines word hidden states with aspect embeddings to generate attention vectors for the final representation. This method further enhances the context representation by appending the aspect embedding to each word embedding vector. Additionally, Ma et al. [5] introduced the Interactive Attention Networks (IAN) approach to interactively learn attention weights in both contexts and aspects, generating separate representations for aspect and context words. Recent advancements in Large Language Models (LLMs) and their capabilities in performing sentiment analysis have also prompted researchers to explore their application in ABSA [6]. This highlights the ongoing efforts to improve ABSA by leveraging advanced neural network architectures and LLMs.

2 Methods

2.1 LSTM Sequence-to-Sequence Model with Attention Mechanism

The LSTM network, introduced by Hochreiter and Schmidhuber [7], models sequential data and overcomes the vanishing gradient problem typically encountered in standard RNNs. Its ability to learn long-term dependencies by maintaining a memory state over time makes it suitable for tasks like ABSA where context over longer sequences, such as reviews, is important. In this project, we implemented a LSTM seq2seq model for ABSA, specifically using a many-to-one architecture. The model takes in a sequence of words from a review and outputs a single sentiment label for that review. The goal is to accurately classify the sentiment expressed in the review with respect to an aspect, such as "food" or "service."

In the LSTM architecture, there is a cell memory state, as well as an input gate i, forget gate f, and output gate o. These components work together to adaptively remember input vectors, forget previous history, and generate output vectors based on the current input and the memory of past inputs. The weights and biases are learned parameters of the LSTM network that are updated during training to minimize the loss function. In our model, given the input word embedding w_t, previous hidden state h_{t-1}, and previous cell state c_{t-1}, every cell state c_t and hidden state h_t are updated as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}; w_t] + b_i) \tag{1}$$
$$f_t = \sigma(W_f \cdot [h_{t-1}; w_t] + b_f) \tag{2}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}; w_t] + b_o) \tag{3}$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}; w_t] + b_c) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

The use of the dot product attention mechanism can improve the performance of our LSTM seq2seq model. This approach allows the model to focus on keywords in the input sequence. The attention mechanism assigns weights to different words in the input sequence, allowing the model to assign importance to the most relevant terms for sentiment analysis.

First, we compute the attention weights via the dot product method, which assign a number indicating the relevance of each word in the input sequence. If a masking parameter is used, it will ignore padding tokens by assigning a large negative value to their attention weights. This ensures that the attention is not drawn to the padding and instead ignores it. The weights are then normalised with a softmax function to make the attention weights sum to one, making it simple to interpret the importance of each word in the entire sequence. The attention output is then concatenated with the key vectors to create the final output of the attention mechanism.

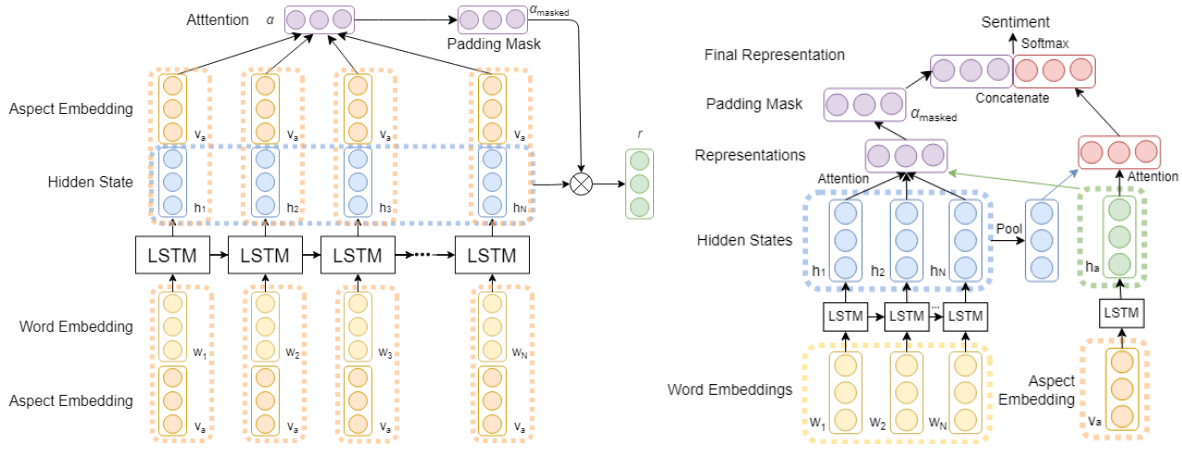
Then, the final output vector is fed to a linear layer, which transforms it into a vector whose length is equal to the number of class labels – in our case, 3. This vector is then passed through a softmax layer to estimate the probability of classifying the sentence into each sentiment label. The softmax function is calculated as follows, where C is the number of sentiment categories:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{7}$$

2.2 Model 1: Append Aspect to the end of Sentence

We first explore the integration of aspect information by appending the aspect word to the end of the sentence. The aim of this approach is to give the model contextual information from the aspect, so that it can focus on the particular aspect-related words. This model can be trained to recognise the sentiment of the sentence in relation to the supplied aspect by placing it at the end of the sentence. For example, given the sentence "Our waitress kept forgetting our drinks", and the aspect "service", the model's input would be "Our waitress kept forgetting our drinks `<SEP>` service." The model would learn that the aspect "service" is associated with the words in the sentence, which indicates a negative sentiment.

2.3 Model 2: ATAE-LSTM (Attention - Aspect Embedding LSTM)



(a) Model 2 (Adapted from [4])

(b) Model 3 (Adapted from [5])

Figure 1: Model Architecture Diagrams. $\{w_1, w_2, \ldots, w_N\}$ represent the word embedding vectors in a sentence whose length is N. $\{h_1, h_2, \ldots, h_N\}$ represent the hidden vectors. v_a is the aspect vector.

Model 2 is motivated by the ATAE-LSTM approach by Wang et al. [4], shown in Figure 1a. This method integrates aspect into the LSTM model by concatenating the aspect embeddings to the word embeddings, then again to the sentence hidden representations, which are used for computing the attention weights. The attention mechanism allows the model to focus on words related to the aspect.

During the forward step, the input sentence is first embedded and similarly, we obtain the aspect embedding. To match the word embedding sequence length, the aspect embedding is enlarged and altered. Through this, the aspect information can be integrated into every word. The step can be represented as follows: The word embeddings of each sentence are represented by w_n. The aspect embedding is represented as v_A and is reshaped so it can concatenate with the word embedding.

$$v_{concat} = \text{concat}(w_n, v_a^{reshape}) \tag{8}$$

The combined embeddings are created by concatenating the word embeddings with the altered aspect embedding. After the concatenated embeddings are passed through the LSTM network to produce hidden states capturing the contextual information of the input sequence:

$$H_n, (h_n, c_n) = \text{LSTM}(v_{concat}) \tag{9}$$

The aspect embedding is concatenated again with the sentence's hidden representations. This additional concatenation is crucial for the attention mechanism, as it helps the model focus on parts of the sentence's most relevant aspects

$$\text{Attention}(\text{concat}(H_n, v_a^{reshape}), \text{concat}(h_n, v_a)) \tag{10}$$

Here, the attention mechanism takes both the hidden states and the final hidden states from the LSTM model and concatenates them with the aspect embedding again, to determine the attention weights that will focus on the relevant sentence words.

2.4 Model 3: IAN

Model 3 is motivated by the IAN approach by Ma et al. [5], shown in Figure 1b. This method involves processing the word embeddings and aspect embeddings separately through different LSTM networks (equation 11). This generates two separate hidden states which contain the contextual information for the sentences and the aspects. Then, the average value of the hidden states of the sentences are used to create the sentence pools.

THeres are used in the computation of attention vectors to extract the important information. First, the sentence hidden states are processed with the aspect hidden state in the attention mechanism (equation 12). Then, the aspect hidden state is processed with the sentence pools created earlier (equation 13), enabling the model to focus on the sequences related to the given aspect. Both outputs from the two separate attention mechanisms are concatenated to capture the interactions between the sentences and the aspects. This final vector is linearised before being passed through the softmax function for the sentiment polarity prediction.

$$H_a, (h_a, c_a) = \text{LSTM}(v_a) \tag{11}$$

$$\text{Attention}(H_a, H_n) \tag{12}$$

$$\text{Attention}(\sum_{i=1}^{N}(H_n^i)/\text{N}, H_a) \tag{13}$$

3 Experiments

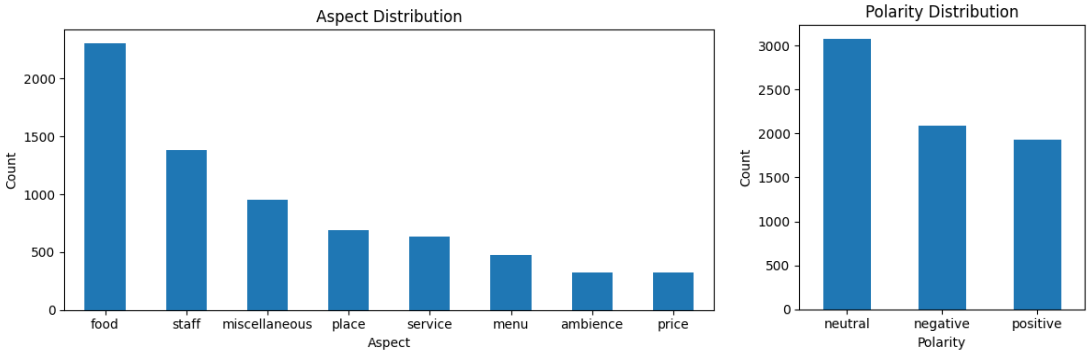
3.1 Dataset Description

3.1.1 Multi-Aspect Multi-Sentiment (MAMS) Dataset

The dataset used in this project is the Multi-Aspect Multi-Sentiment (MAMS)[1] dataset's version for aspect-category sentiment analysis (ACSA) [8]. It contains reviews annotated with at least two aspects and their corresponding sentiment polarities. It includes 8 aspect categories: food, service, staff, price, ambience, menu, place and miscellaneous, and 3 sentiment polarities: positive, negative and neutral.

3.1.2 Dataset Analysis

Upon analysing the training dataset, "food" appeared as the most frequent aspect, while "price" was the least frequent (Figure 2a). In terms of polarity distribution, there are more neutral sentiments (3000 occurrences) and a balanced distribution between positive and negative sentiments (2000 occurrences each), shown in Figure 2b. This indicates a slight bias towards neutrality in the dataset, but provides a fair representation of both positive and negative opinions. The aspect-polarity pair distribution shows varying frequencies for each aspect-polarity combination (Table 1), highlighting the diverse distribution of sentiments across different aspects. The most frequent pair was food-neutral (1298 occurrences), while the least frequent pair was menu-negative (39 occurrences), highlighting a significant disparity.



(a) Aspect distribution

(b) Polarity distribution

Figure 2: Bar graphs of distribution

[1] https://github.com/siat-nlp/MAMS-for-ABSA

Aspect-Polarity Pair	Count	Aspect-Polarity Pair	Count	Aspect-Polarity Pair	Count
(food, neutral)	1298	(food, negative)	255	(service, neutral)	128
(staff, negative)	922	(miscellaneous, positive)	227	(place, positive)	125
(food, positive)	754	(miscellaneous, negative)	196	(price, negative)	114
(miscellaneous, neutral)	531	(ambience, positive)	181	(ambience, negative)	90
(place, neutral)	430	(service, positive)	174	(price, positive)	72
(menu, neutral)	372	(place, negative)	139	(menu, positive)	64
(staff, positive)	332	(price, neutral)	136	(ambience, neutral)	53
(service, negative)	329	(staff, neutral)	129	(menu, negative)	39

Table 1: Aspect-Polarity distribution in descending order

3.2 Experiment Setup

3.2.1 Text Preprocessing

We conducted text preprocessing on the review sentences to prepare the data for training. This involved preserving emoticons, as they are essential for sentiment analysis. We then converted all text to lowercase, addressed contractions, removed punctuation and stop-words, and tokenised the text. We used the Spacy[2] library, particularly for its stop-words removal, lemmatisation, and tokenisation functions. We further removed some negation words with important sentiment meanings from the default stop words list.

3.2.2 Evaluation Metric

To assess our model's performance in executing ABSA, we use an Accuracy metric. This metric measures the proportion of all samples there were predicted correctly. It is calculated as follows:

$$\text{Accuracy} = \frac{T}{N} \tag{14}$$

where T is the number of correctly predicted samples, and N is the total number of samples. A higher accuracy indicates better performance of the model.

3.2.3 Hyperparameters Settings

In our experiments, we evaluated the performance of our model using both randomly initialised embeddings and pre-trained GloVe[3] word embeddings [9]. We initialised our word embedding weights by incorporating pre-trained word vectors from the GloVe dictionary vocabulary, by utilising GloVe's word weightings for our own vocabulary. For words not included in the GloVe vocabulary, we modify their weights by initialising a random weight that was selected from a uniform distribution with a range of -0.1 to 0.1 [5].

Hyperparameter	Search space	Selected value
Word embedding	randomly initialised, pre-trained	Pre-trained (GloVe)
Optimizer	'SGD', 'Adam'	'Adam'
Learning rate	[0.01, 0.001, 0.0001]	0.001
Number of epochs	[15, 20, 25]	20
Hidden size & Embedding size	[50, 100]	100
Loss function	'NLLLoss', 'CrossEntropyLoss'	'CrossEntropyLoss'

Table 2: Hyperparameter tuning

To optimise our models' performance, we explored different hyperparameter values, shown in table 2. We experimented with three learning rates (0.01, 0.001, and 0.0001) and number of epochs (15, 20, and 25). We also experimented with 50 and 100 embedding and hidden sizes for our model. Additionally, we compared the effectiveness of three optimisation methods, Adam, Adagrad and SGD, for updating model parameters during training. Furthermore, we also tested two different loss functions, CrossEntropyLoss and NLLLoss, to compute the loss during training. After conducting these experiments, the optimal hyperparameters that we selected for our model are shown in Table 2.

[2] https://spacy.io/

[3] http://nlp.stanford.edu/projects/glove/

4 Results

4.1 Quantitative Results

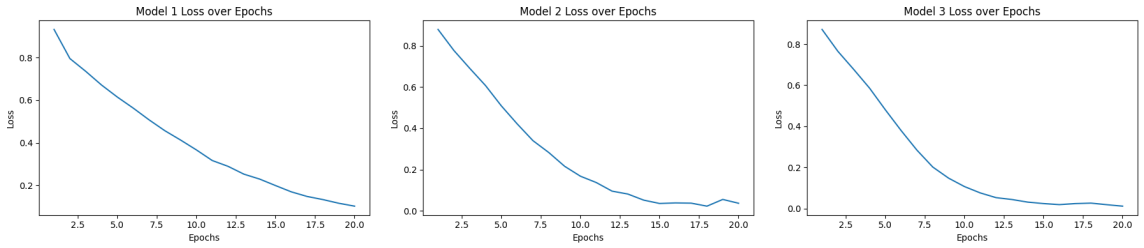
4.1.1 Comparative Analysis of Models

We evaluated the performance of the three model variants described in our methodology on the test set using the accuracy metric. The accuracy scores for each model are summarised in Table 3. Our results demonstrate that models incorporating aspect information into sentiment analysis outperform the model without aspect integration (Model 0), which achieved an accuracy of only 0.4573. Among the models with aspect integration, Model 3 achieved the highest accuracy of 0.7248, subsequently by Model 2 with an accuracy of 0.6815, and then Model 1, which achieved an accuracy of 0.6548. These findings highlight the importance of considering aspect information in sentiment analysis. By incorporating aspect information, these models were able to focus their attention on words or phrases relevant to the aspect, leading to more accurate sentiment predictions. Model 2 performs better by capturing the important parts of a sentence based on the aspects, utilising aspect embeddings to compute attention weights that gives higher importance to words that are more related to the aspect. In contrast, Model 3 performs slightly better than Model 2 by separately and interactively modelling the sentence and aspect, then taking the concatenated representation of both to refine its predictions.

Figure 3 illustrates the training loss for Models 1, 2, and 3, demonstrating a consistent decrease over the epochs. This indicates that as the models are training, they are becoming better at minimising the error on the training data by adjusting their parameters to reduce the prediction error. Figure 4 illustrates the validation accuracy, which shows an upward trend initially before stagnating, indicating that the models are improving their ability to generalise to unseen data until a point where they have learned most of the relevant patterns in the data.

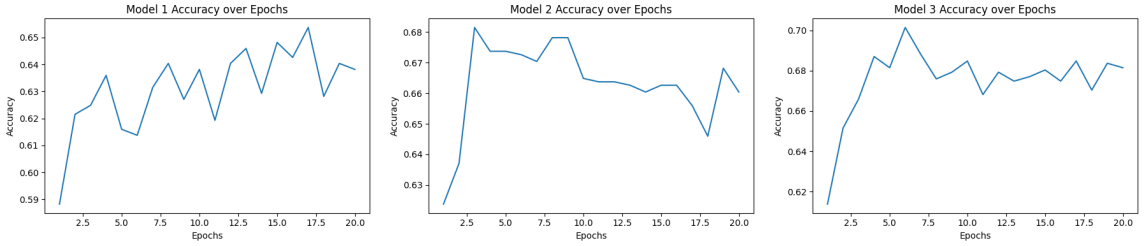
Model	**Accuracy**
0 (No aspect integration)	0.4573
1	0.6548
2	0.6815
2	0.7248

Table 3: Model performance results



(a) Model 1 (b) Model 2 (c) Model 3

Figure 3: Training Loss over Epochs during Model training



(a) Model 1 (b) Model 2 (c) Model 3

Figure 4: Validation Accuracy over Epochs during Model training

4.1.2 Ablation Study

The experiments detailed in Table 4 were conducted to assess the effect of various hyperparameters on our model's performance and on the training process, measured by the accuracy of the model on the test set. Each experiment was run multiple times and then averaged to ensure reliable results.

Varied Parameter	**Test**	**Used Parameter**	**Model 1 Accuracy**	**Model 2 Accuracy**	**Model 3 Accuracy**
Word embeddings	**1**	**Pre-trained GloVe**	**0.6548**	**0.6815**	**0.7248**
	2	Randomly initialised	0.6194	0.6382	0.6421
Optimiser	**1**	**Adam**	**0.6548**	**0.6815**	**0.7248**
	3	SGD	0.4335	0.5849	0.5700
	4	Adagrad	0.6354	0.6537	0.6582
Learning rate	**1**	**0.001**	**0.6548**	**0.6815**	**0.7248**
	5	0.01	0.6221	0.6460	0.6659
	6	0.0001	0.6499	0.6843	0.6654
Number of epochs	**1**	**20**	**0.6548**	**0.6815**	**0.7248**
	7	15	0.6427	0.6776	0.6765
	8	25	0.6443	0.6798	0.6737
Hidden/Embedding Size	**1**	**100**	**0.6548**	**0.6815**	**0.7248**
	9	50	0.6460	0.6632	0.6493
Loss function	**1**	**CrossEntropyLoss**	**0.6548**	**0.6815**	**0.7248**
	10	NLLLoss	0.6510	0.6932	0.6965
Attention mechanism	**1**	**Dot product**	**0.6548**	**0.6815**	**0.7248**
	11	Scaled dot product	0.6315	0.6732	0.6526
Separator (in Model 1)	**1**	**With separator**	**0.6548**	-	-
	12	Without separator	0.6493	-	-

Table 4: Ablation Study (Test 1 being the baseline)

The results indicate that using pre-trained GloVe embeddings led to overall higher accuracy (test 1) compared to randomly initialised embeddings (test 2). This improvement suggests that pre-trained embeddings are able to capture more semantic information and context from the text, hence improving the models' ability to understand and classify sentiment. Moreover, 'Adam' optimiser was found to most effectively update the parameters that minimise the loss during training, whereas 'SGD' and 'Adagrad' optimisers were not able to decrease the loss as much over the same number of epochs (test 3 and 4). 'Adam' optimiser computes individual adaptive learning rates for different parameters, allowing for faster convergence and efficient optimisation [10].

Additionally, a higher learning rate of 0.01 led to potential overshooting of the minimum and ineffective parameter optimisation (test 5). Conversely, a lower learning rate of 0.0001 caused the model to learn too slowly (test 6) and was unable to converge within the given number of epochs. Furthermore, a smaller number of epochs (15 in test 7) and a larger number of epochs (25 in test 8) were not able to obtain the better accuracy than running 20 epochs which yielded the highest accuracy for all three models. This is because a larger number of epochs can cause the model to overfit to the training data, and a smaller number of epochs can cause underfitting.

We also tested between 50 and 100 hidden sizes and embedding sizes. The model with a size of 100 (test 1) was able to significantly increase the accuracy of all models, especially Model 3, compared to using a size of 50 (test 9). This is because more hidden layers allow learning of more complex representations, capturing and generalising to patterns, but increases computational time. Moreover, we experimented with different loss functions to compute the loss. Using NLLLoss and CrossEntropyLoss yielded similar loss reduction during training and roughly similar model performance in tests 1 and 10. Other than that, the impacts of using dot product (test 1) attention mechanism versus scaled dot product (test 11) demonstrated similar performance outcomes. However, we decided to use the dot product attention mechanism due to its consistent performance across these experiments. Finally, we investigated the effect of removing the separator when appending the aspect words to the sentences in Model 1. Our experimentation showed that removing the separator did not result in any significant differences.

Given the outcomes of our ablation study, we can conclude that our selected optimal hyperparameters work well across the three models.

4.2 Qualitative Results

To determine how well Models 1, 2, and 3 perform at capturing words that were related to how they determine their aspect-specific sentiment, we analysed their attention weights. Poor attention performance was demonstrated by Model 1, which appends the aspect word at the end of the sentence (Figure 5). The model missed important contextual terms, as the attention weights were mostly focused on the added aspect term. As a result, the word would have a large influence on the sentimental prediction, making it biased.

In contrast, Models 2 and 3 (Figure 6a and 6b, respectively) showed an effective attention process. They were both able to perform well in allocating attention weights across important sentiment words based on their aspect. Models 2 and 3 are more successful compared to Model 1 because of their efficient attention allocation and ideal aspect integration, which improved their ability to predict aspect-specific sentiment.

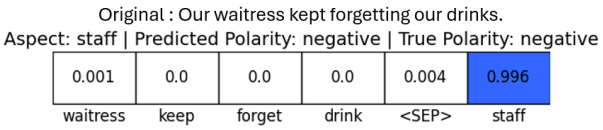
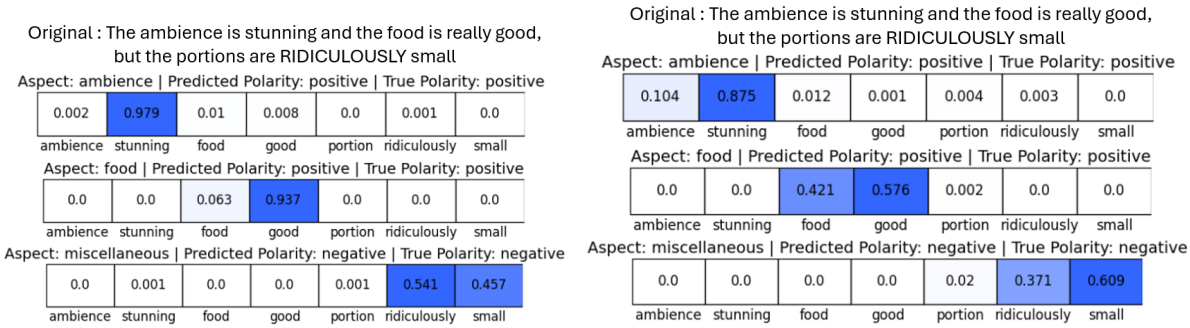


Figure 5: Visualisation of attention weights for words (Model 1)



(a) Model 2

(b) Model 3

Figure 6: Visualisation of attention weights for words

5 Conclusion

In conclusion, it is important to integrate aspects for the task of ABSA. This project uses an LSTM seq2seq model with various aspect integration techniques to improve sentiment classification in terms of a specific aspect. We were able to develop three different methods that successfully integrated aspect into our models, all of which improved the accuracy score, with the best-performing model achieving an accuracy score of 72%. The case study also shows that our models can reasonably focus on the words that are important when determining a sentence's sentiment polarity with regards to a particular aspect.

Our project has some limitations, a main problem during the training is the bias present in the dataset, as most of the sequences were significantly associated with a neutral polarity, and some aspects will be biased towards another polarity. This has an impact on the model's capacity to generalise in different situations. Especially in Model 1, where the aspect word that was appended to the end of the sentence has been given the most amount of attention, which resulted in a biased sentiment prediction based on the training data. Addressing the bias in the dataset is critical in future research. Improving the model's reliability and generalisation ability would require a balanced dataset considering the different aspect-polarity pairings. Another area of improvement is for the attention mechanisms to minimise the focus on less significant words while better capturing the relevant contextual information.

6 Team Contribution

In this project, we worked as a team of two members. We collaborated on all aspects of the project, from dataset analysis and preprocessing to model design and experimentation. Both members contributed equally to the project, actively participating in giving ideas, problem solving, decision making, and coding for implementing the neural network architecture. Additionally, both members worked together on data visualisation, result analysis and writing the project report. Our combined efforts and open communication allowed for a smooth completion of the project without conflicts.

References

[1] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *arXiv preprint cs/0205070*, 2002.

[2] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 151–160.

[3] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective lstms for target-dependent sentiment classification," *arXiv preprint arXiv:1512.01100*, 2016.

[4] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 606–615.

[5] D. Ma, S. Li, X. Zhang, and H. Wang, "Interactive attention networks for aspect-level sentiment classification," *arXiv preprint arXiv:1709.00893*, 2017.

[6] G. Negi, R. Sarkar, O. Zayed, and P. Buitelaar, "A hybrid approach to aspect based sentiment analysis using transfer learning," *arXiv preprint arXiv:2403.17254*, 2024.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] Q. Jiang, L. Chen, R. Xu, X. Ao, and M. Yang, "A challenge dataset and effective models for aspect-based sentiment analysis," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 6280–6285.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.