NLU project exercise 2 lab 11

Luca Zanolo (20029226)

University of Trento

luca.zanolo@studenti.unitn.it

1. Introduction (approx. 100 words)

This report describes the implementation and use of a joint model for Aspect-Based Sentiment Analysis (ABSA) using a PyTorch neural network based on BERT (bert-base-uncased). The task is to extract aspect terms from the text and determine their sentiment polarity. To evaluate the results, I use the script provided in the assignment, including functions tag2ts and tag2ot for transforming tag sequences into sentiments and opinion targets. These functions are also derived from [1]. During data preprocessing, I adapted the dataset labels to be compatible with this methodology.

2. Implementation details (max approx. 200-300 words)

In the initial phase of data preparation, I use BERT's tokenizer to process sentences. This involves converting the sentences into tokens and subsequently aligning the labels with their respective tokens. For aspects, BIES (Beginning, Inside, End, Single) and O (Outside) labels were used, while for polarities, the labels NEG (Negative), POS (Positive), NEU (Neutral), and O (No polarity) were used. The label sequences are realigned to the tokens produced by tokenization. During this alignment process, the labels read from the dataset are initially maintained as 'S' for all tokens. After tokenization, these labels are transformed to also utilize B, I, and E (Beginning, Inside, End) labels for multi-token aspects, this was done in `align_tags` method of `Dataset` object.

In the model's architecture, I have integrated two linear layers over the BERT model: one is dedicated to aspect detection, while the other focuses on polarity classification. I introduce also a dropout layer with a probability of 0.05 immediately after the BERT layer. As optimizer, Adam is used, with a learning rate of 1e-4.

Training was conducted with 10-fold cross-validation with 5 runs for each fold. During training, the loss function is a weighted sum of the aspect and polarity losses, with a higher coefficient of 0.75 assigned to the polarity classification. The aggregate_loss() function computes the CrossEntropy loss separately for aspects and polarities, summing them to yield the total loss. An important aspect of training was the use of weights in the CrossEntropy calculation to more effectively balance the classes. The weights were calculated based on the frequency of classes in the dataset, assigning greater weight to less frequent classes. This approach has the goal to help mitigate the class imbalance problem, potentially enhancing the model's ability to recognize less represented aspects and polarities.

3. Results

For evaluation, the `evals.py` script from [1] was used, adapting the model's outputs to conform to this script. The performance was evaluated with `eval_loop` function, using the following metrics for both the tasks: F1, precision, and recall.

The metrics were computed at the fold level and represent averages across multiple runs. The performance for both OT and TS tasks for each fold is detailed in the next page, with the best model identified in fold 0 which scores are presented in the two tables below.

Fold	Run	OT Precision	OT Recall	OT F1
0	4	0.819	0.855	0.836

Table 1: *Best model performance metrics - Aspect detection*

Fold	Run	TS Precision	TS Recall	TS F1
0	4	0.656	0.645	0.651

Table 2: *Best model performance metrics - Aspect polarity*

The model exhibits a discrete performance in identifying aspects. However, its performance in correctly determining polarity is comparatively less effective. Additionally, there is a concern about potential overfitting. Despite efforts to fine-tune hyperparameters and modify the training procedure, the observed patterns in the training and validation loss charts indicate that the model is not generalizing effectively on validation data, as shown by the chart below.

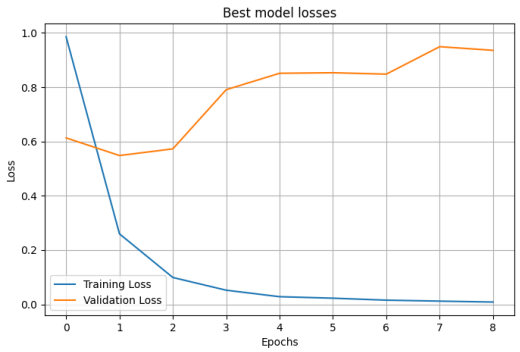

Figure 1: *Training and valdiation losses of the best model*

This suggests a need for further adjustments or a reevaluation of the model's architecture to enhance its generalization capabilities.

4. References

[1] X. Li, L. Bing, P. Li, and W. Lam, "A unified model for opinion target extraction and target sentiment prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 6714–6721.

Fold	OT Precision	OT Recall	OT F1	TS Precision	TS Recall	TS F1
0	0.789	0.849	0.818	0.629	0.633	0.631
1	0.782	0.848	0.813	0.613	0.629	0.620
2	0.781	0.846	0.812	0.612	0.620	0.615
3	0.767	0.849	0.806	0.612	0.631	0.621
4	0.766	0.850	0.805	0.598	0.627	0.612
5	0.757	0.853	0.801	0.588	0.625	0.605
6	0.783	0.835	0.808	0.601	0.596	0.598
7	0.788	0.839	0.812	0.615	0.615	0.615
8	0.784	0.845	0.813	0.603	0.615	0.609
9	0.765	0.852	0.806	0.597	0.624	0.610

Table 3: *Fold level performance metrics*