Homework 2: Aspect-Based Sentiment Analysis

Michele Conti
conti.1599133@studenti.uniroma1.it

1 Introduction

Aspect-based sentiment analysis (ABSA) tackles the problem of finding both aspects in a given sentence, and the polarities associated with those aspects. For example, given the sentence ”I love their pasta but I hate their Ananas Pizza”, the task is to first extract the two aspects ”pasta” and ”Ananas Pizza”, and to then to predict their associated polarities, in this case respectively ”positive” and ”negative”.

In this paper, I propose an end-to-end approach to attack this task, exploiting the combination of both context-independent (GloVe) and context-dependent (BERT) word embeddings, multihead attention layers, and the use of a BiLSTM layer, reaching satisfying results on the provided development dataset (macro f1 score of 0.544 on the Restaurants + Laptops development set).

2 Problem formulation

All the model variations proposed in this paper tackle the ABSA problem in an end-to-end fashion, meaning that both tasks (i.e., aspect sentiment extraction and aspect sentiment polarity evaluation) are solved simultaneously. To achieve this, the two-sided problem has been formulated as a sequence labeling task.

To that end, this is the general pipeline followed: during the preprocessing, each token of each sentence is tagged using either an IOB (i.e., Inside, Outside, Beginning) or a BIOES (i.e., Beginning, Inside, Outside, Ending, Single) tagging schema, used on top of the four polarities appearing in the datasets (example of the two tagging schemas in figure 1). Depending on the adopted tagging schema, the model is then trained to extract the right label from a set of either $4 \cdot 2 + 1 = 9$ (for IOB) or $4 \cdot 4 + 1 = 17$ (for BIOES) possible classes. Finally, the model output is translated back into its original form, where the tokens classified as ”O” are not considered and the aspect terms are merged together according to their tags (e.g., $\{$”Ananas” = B-negative, ”Pizza” = I-negative$\} \rightarrow \{$”Ananas Pizza” = negative$\}$).

3 Models

In this section, I will present all the modules that compose the proposed final model, starting by introducing the structure of the baseline network. Then, at each step, I will introduce the very next module used, describing its main functionalities and analyzing (possibly) the boost in terms of performance when added to the network.

To evaluate the performance of the following models, I will use their macro f1 score on the combined task of aspect term extraction and evaluation.

Moreover, IOB has been selected as the preferred tagging schema, and all the following model performances are computed using that. Several experiments have been conducted to find out the best one, but, due to space reasons, only the comparisons for the proposed model are reported here.

3.1 M0: Baseline

The first model used to solve the ABSA task is a very simple network, composed by an embedding layer to encode the tokens of the input sentences, followed by a one-on-one linear layer (i.e., input features equal to output features), two stacked BiLSTM layers and a prediction block, composed by a linear layer and a softmax activation (network architecture in figure 2). Additionally, I replaced the randomly initialized word embeddings in the embedding layer with the pre-trained word vectors of GloVe (Pennington et al., 2014).

In this first model variation, no contextual information is considered in the embedding phase, since the word embeddings provided by GloVe, being context-free, will produce the same vector representation regardless of the context. Nevertheless, the BiLSTM layers partially make up for this, encoding both the left-to-right and the right-to-left context for each word of a sentence.

This very simple architecture (let's call it M0) is still able to achieve decent results in terms of f1 score, reaching a maximum of 0.381 in terms of macro f1 score (loss plots, extraction f1 score, extraction + evaluation f1 score respectively on figures 4, 6, 8).

3.2 M1: BERT Embedder

To add contextual information to my model, I decided to use BERT (Bidirectional Encoder Representations Transformers) (Devlin et al., 2018). BERT is a language representation model, which uses bidirectional transformers to pre-train on a very large corpus over different pre-training tasks. Then, the pre-trained BERT model can be finetuned to be used on other tasks. In my case, I decided to use BERT as an embeding layer, as suggested in (Li et al., 2019). Differently from the traditional GloVe based embedding layer, which, as I already said, provide only context-independent representation for each token, using BERT as an embedding layer does provide context-dependant information about each token in a sentence.

To get the most out of BERT, I followed one of the approaches proposed in (Ács et al., 2021) as pooling strategy. In particular, the approach can be summarized in two steps: first, the BERT embeddings for the subwords of each token are computed. This is achieved by averaging the last 3 hidden layers of BERT. Then, the weighted average of the subwords embeddings are computed, based on the BPE length of each token.

This process can be carried out in two ways: keeping the BERT parameters freezed, or keeping them free and finetuning them during the training process. These two possible choices are the basis of a trade off between two possible scenarios: on the one hand, keeping the BERT weights freezed, we have a much easier time when training our model, in terms of both training time, and flexibility in the choice of the hyperparameters. On the other, learning also the parameters for the BERT model gives the chance to, potentially, achieve much better results.

Once the BERT embeddings have been computed, similarly to the GloVe embeddings, they are too passed into a linear layer with a ReLU activation, and subsequently concatenated with the GloVe pretrained embeddings. The rest of the network is the same. Adding context-dependent information with BERT and keeping its parameters freezed (let's call this model M1), the model reaches 0.466 macro f1 score on the dev set.

3.3 M2: Finetuning BERT

In order to finetune the BERT embedder layer, I had to tweak some of the hyperparameters I was using before. For instance, once the parameters of this embedding layers are free to be updated, the chosen learning rate really starts to make a difference. Decreasing it by an order of magnitude seemed to do the trick. This way, I was able to achieve much better results, reaching 0.497 macro f1 score on the development set.

3.4 M3: Self-Attention

Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. It has been made popular by (Vaswani et al., 2017), and, intuitively, it makes it possible for the network to focus on relevant parts of a sentence. This is achieved in the following way:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

To attend different parts of a sentence, multiple attention heads can be computed and then concatenated. This concatenation can be then linearly transformed into the expected dimension. This process is known as Multi-head Attention.

For my use case, I used Multi-head Attention on both the GloVe embeddings and on the BERT embeddings, respectively using 12 and 24 heads. Then, as in the previous versions, I concatenated the two resulting embeddings, and passed the concatenation to the stacked BiLSTM layers.

This was the trick that made me achieve the best results, achieving 0.544 macro f1 score on the development set.

4 Figures and tables

All figures and tables have been intentionally placed at the end of the document, in this section, so that they don't affect the maximum limit of three pages for the text.

Sentiment	Train	Dev
positive	2605	546
negative	1364	307
neutral	877	216
conflict	111	25

Table 1: Polarities support for targets (task A+B) in the training and development set (i.e., Restaurants + Laptops).

Category	Train	Dev
anecdotes/miscellaneous	941	191
price	268	53
food	1008	224
ambience	355	76
service	478	119

Table 2: Categories support (task C+D) in the training and development set (i.e., Restaurants).

Sentiment	Train	Dev
positive	1803	376
negative	672	167
neutral	411	89
conflict	164	31

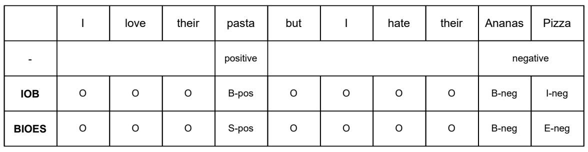
Table 3: Polarities support for categories (task C+D) in the training and development set (i.e., Restaurants).

Model	F1 Extr.	F1 Eval.
M0 = Baseline	0.715	0.381
M1 = M0 + BERT$_{freeze}$	0.805	0.466
M2 = M0 + BERT$_{finetune}$	0.801	0.497
M3 = M2 + ATT	**0.820**	**0.544**
M4 = M3 + CRF	0.813	0.493
M5 = M3 + POS	0.798	0.509

Table 4: Task A+B: Models variations top performance on development set (Restaurants + Laptops). For each variation, it has been selected the model with the highest f1 evaluation score on the validation set.
The baseline model (M0) is the one discussed in section (?). With "BERT" I refer to the BERT Embedder introduced in section (?) (BERT$_{freeze}$ means that the BERT Embedder is based on a pretrained version of the BERT model, and that there is no finetuning involved during the training. Instead, BERT$_{finetune}$ means that the BERT model is initially loaded from a pretrained version, but it is also finetuned during training). With "ATT" I refer to a multihead self attention layer, as discussed in section (?). With "CRF" I refer to Conditional Random Field, as discussed in section (?). With "POS", I refer to the use of an embedding layer for the part-of-speech tags of the tokens of each sentence, as discussed in section (?).

Model	F1 Extr.	F1 Eval.
M3$_{\text{IOB}}$	**0.820**	**0.544**
M3$_{\text{BIOES}}$	0.812	0.490

Table 5: Task A+B: Proposed model top performance comparison using two tagging schemas: IOB (i.e., Inside-Outside-Beginning), and BIOES (i.e., beginning, inside, outside, end, single).



	I	love	their	pasta	but	I	hate	their	Ananas	Pizza
-				positive					negative	
IOB	O	O	O	B-pos	O	O	O	O	B-neg	I-neg
BIOES	O	O	O	S-pos	O	O	O	O	B-neg	E-neg

Figure 1: IOB and BIOES tagging schemas example.

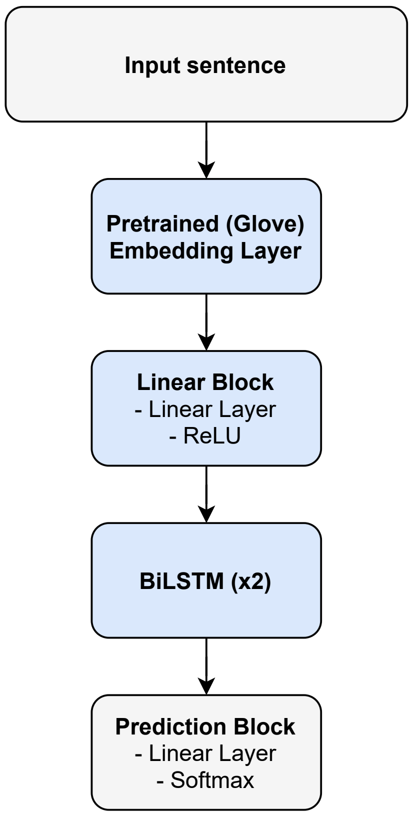


Figure 2: Baseline model (M0) architecture.

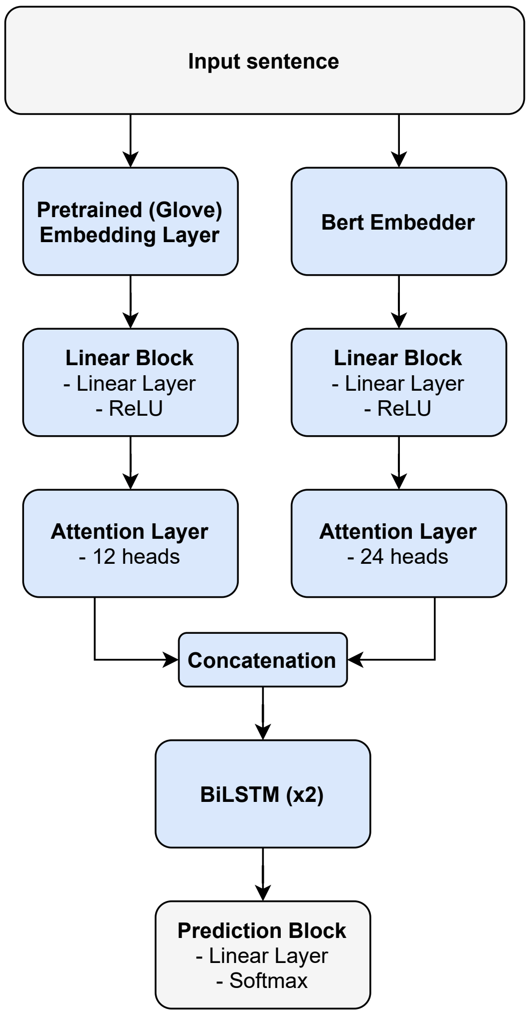


Figure 3: Final model (M3) architecture.

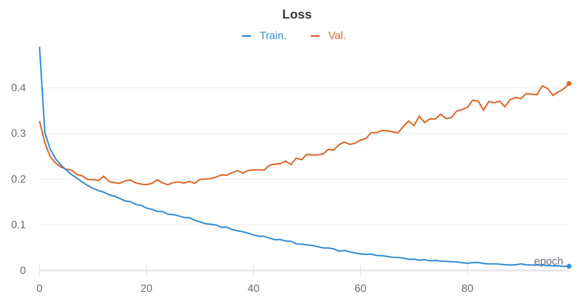


Figure 4: Task A+B: Baseline (M0) loss on the training and development set.

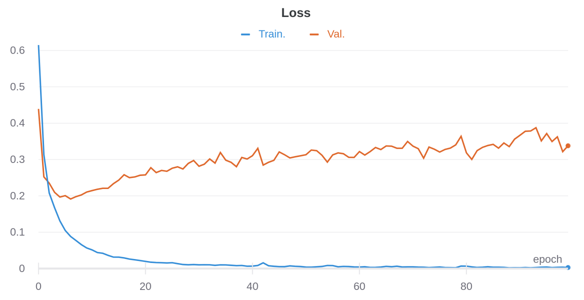


Figure 5: Task A+B: Final model (M3) loss on the training and development set.

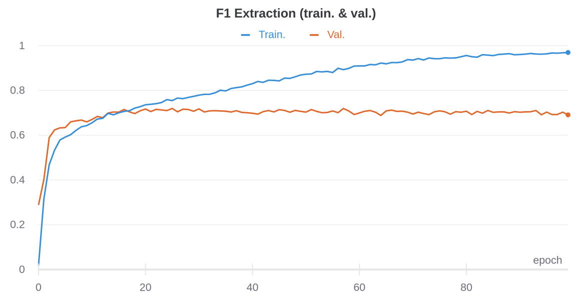


Figure 6: Task A+B: Baseline (M0) macro f1 extraction score on the training and development set.

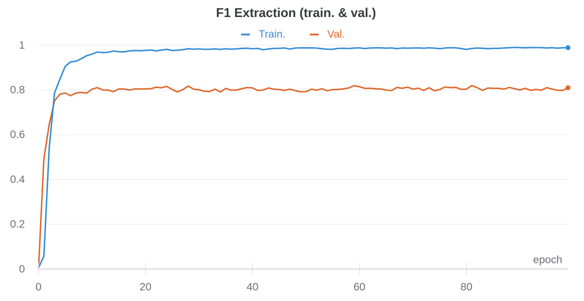


Figure 7: Task A+B: Final model (M3) macro f1 extraction score on the training and development set.

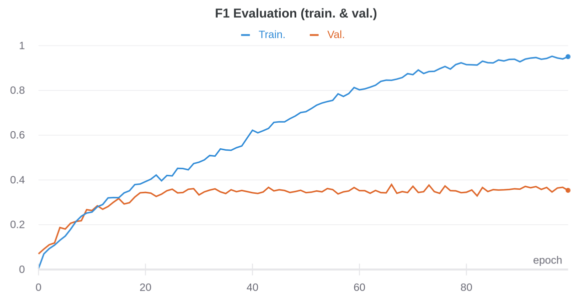


Figure 8: Task A+B: Baseline (M0) macro f1 evaluation score on the training and development set.

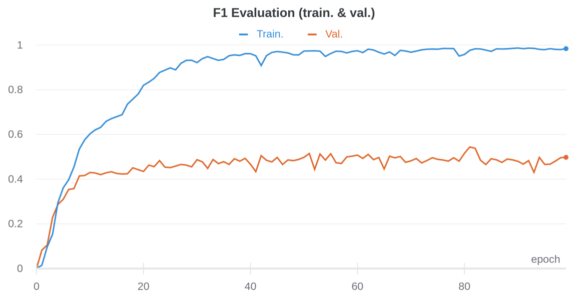


Figure 9: Task A+B: Final model (M3) macro f1 evaluation score on the training and development set.

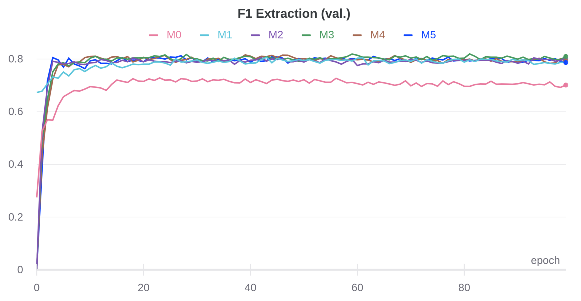


Figure 10: Task A+B: Comparative plot of f1 extraction score for the 6 model variations (M0, M1, M2, M3, M4, M5).

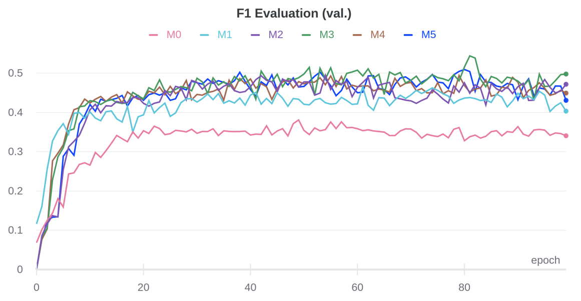


Figure 11: Task A+B: Comparative plot of f1 evaluation score for the 6 model variations (M0, M1, M2, M3, M4, M5).

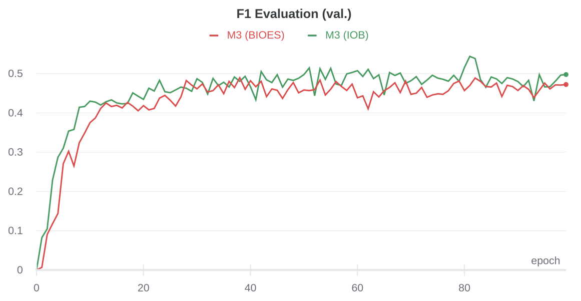


Figure 12: Task A+B: Comparative plot of f1 extraction score for the best model (M3) using two different tagging schemas (IOB, BIOES).

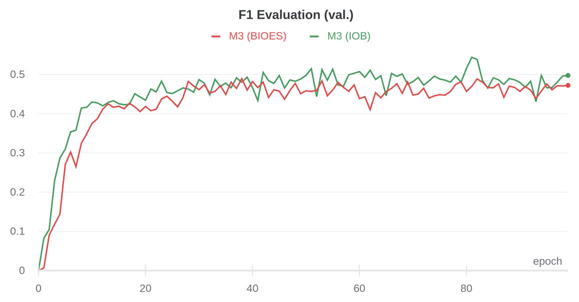


Figure 13: Task A+B: Comparative plot of f1 evaluation score for the best model (M3) using two different tagging schemas (IOB, BIOES).

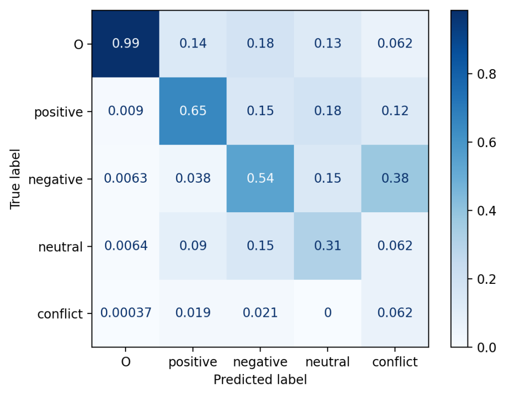


Figure 14: Task A+B: Normalized confusion matrix of the baseline model (M0).

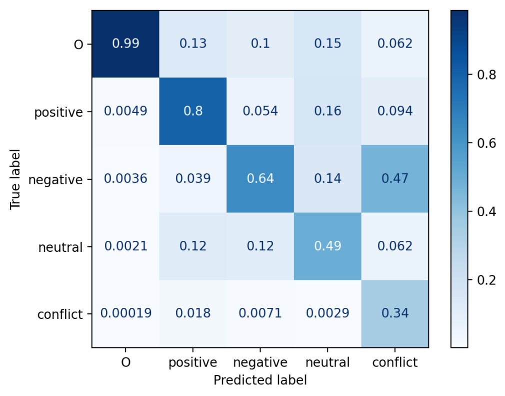


Figure 15: Task A+B: Normalized confusion matrix of the best model (M3).

References

Judit Ács, Ákos Kádár, and András Kornai. 2021. Subword pooling makes a difference. *arXiv preprint arXiv:2102.10864*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting bert for end-to-end aspect-based sentiment analysis. *arXiv preprint arXiv:1910.00883*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.