Aspect-Based Sentiment Analysis (ABSA)
Natural Language Processing course 2021 – Homework 2

Andrea Gasparini
Sapienza University of Rome
gasparini.1813486@studenti.uniroma1.it

1 Introduction

Aspect-Based Sentiment Analysis (ABSA) (Schouten and Frasincar, 2016) is the task of identifying aspect terms and categories from a given sentence, and to then associate a sentiment polarity to each of them. It can be also seen as composed by 4 different sub-tasks, namely: Aspect term identification (A), Aspect term polarity classification (B), Aspect category identification (C) and Aspect category polarity classification (D).

In this report are described several approaches and experiments carried out to jointly solve tasks A and B (A+B) and to then adapt the same architecture to also solve C and D (C+D) together.

2 Task definition

2.1 A+B

The first addressed task, namely A+B, is to simultaneously identify aspect terms and associate a sentiment polarity to them.

We can formulate it as a Sequence Labeling task, hence we preprocess the dataset by tagging each token using the IOB scheme (i.e. one may be either Inside, Outside or at the Beginning of an aspect term) and pairing the tag with the token polariy (fig. 1 shows a tagging example). Therefore, the possible 4 polarities (*positive*, *negative*, *neutral*, *conflict*) yield an output space with cardinality = 9.

When multi-word aspect terms with inconsistent polarities are predicted, the resulting polarity is identified as the most common one among them or as "conflict" in case all occurs the same amount of times.

2.2 C+D

The second task, namely C+D, is instead a multi-label classification task, where we want to identify the related categories (from a pre-defined set) and the associated sentiment polarity.

The output space in this case can be seen as one independent prediction for each category (*anecdotes/miscellaneous*, *price*, *food*, *ambience*, *service*) which ranges across 5 possible classes: the 4 given polarities and an additional label representing the absence of the category among the sentence's aspects. Its cardinality is therefore = 25.

3 Model architecture

In this section are described all the building blocks of the neural model architecture, which have been then composed together one on top of the other.

3.1 Static embedding layer

The very first block of the network is an embedding layer, to simply encode each token from the given input sentences to a vector representation. The pre-trained GloVe (Pennington et al., 2014) word vectors have been employed to already have a rich *static* embedding (context-free).

3.2 POS embedding

Since most of the aspect terms are represented by nouns, a subsequent embedding layer for part-of-speech (POS) tags seemed to be reasonable addition to enhance the performance, at least for the aspect terms identification task.

The consequent new input representation is obtained by concatenating the output of a randomly initialized POS embedding layer to the previously defined one (3.1). However, since the plain dataset did not include any POS information, a POS tagging procedure has been performed in advance with a pre-trained tagger from the NLTK library (Bird and Loper, 2004).

3.3 BERT contextualized embedding

Despite the already rich input representation obtained, static word embeddings are not suitable for

polysemous words (Hu et al., 2016) and contextual information may indeed be beneficial.

The choice of a contextualized embedding model fell on BERT (Devlin et al., 2019), which is a bidirectional transformer-based language model pre-trained on huge corpora over the two tasks of Next Sentence Prediction and Language Modelling. BERT learns contextual embeddings as a result of the training process and can be then used as an embedding layer which provides contextual representations for each token in a sentence.

For all the mentioned reasons, BERT can be effectively exploited for other tasks like ABSA (Li et al., 2019).

The final input representation is obtained by concatenating the contextualized embedding output of BERT with one or both of the previously defined ones (3.1 and 3.2)

3.3.1 Feature-based approach

We can use BERT as a frozen encoder with the feature-based approach, in which fixed features are extracted from the pre-trained model and directly employed as embeddings.

Pooling strategies Since the last hidden layer is too close to the pre-training target functions of BERT, it may be biased to those, and therefore it may lead to wrong representations. To overcome this issue and obtain a better representation, several layer pooling strategies have been applied, comparing their performance to pick the best one; the list of implemented strategies is shown in Table 1.

Different strategies have been tested for WordPiece pooling as well, but as was to be expected they did not really make much difference, since subword pooling effects heavily depend on the addressed task (Ács et al., 2021).

3.3.2 Fine-tuning approach

BERT can be also adapted on the downstream task by simply fine-tuning all its parameters during the training phase. This approach can help exploiting the strengths of BERT and improving the performance (Li et al., 2019), but there is a trade-off between training feasibility and performance to take into account, for which we also need to consider that a Transformer encoder architecture may not always easily represent all tasks (Peters et al., 2019).

3.4 LSTM layer

A Long Short-Term Memory (LSTM) layer (Hochreiter and Schmidhuber, 1997) has been employed to exploit sequence level semantics and capture dependencies.

The effectiveness of LSTMs can be further improved by stacking 2 layers, one going left-to-right and the other right-to-left, yielding a so-called bidirectional LSTM (BiLSTM).

3.4.1 Summarize output hidden states

In order to make use of the LSTM's output, we take into consideration different hidden states in task A+B and C+D. Since the former is a sequence labelling task (2.1), we use all the states to have a label for each token in the sentence. Intuitively, in this case a BiLSTM should better help having a consistent output. In the latter case, task C+D only needs a single "summary vector" of the sentence as a whole, and we achieve this by concatenating the last and the first hidden states when using a BiLSTM, or only the last otherwise.

3.5 Attention layer

In several NLP problems, such as ABSA, elements composing the source text may be characterized by having each a different relevance. For instance, some words could be relevant to some aspects under consideration, but not to others.

The commonest solution to this problem is a mechanism known as Attention (Galassi et al. 2021; Bahdanau et al. 2014), which aims to compute a representation that focuses on relevant parts of a sentence.

In the following, two different approaches to implement an Attention layer in the network are described. Anyhow, we exploit the output of the Attention layer by either giving it as input to the LSTM layer or concatenating it to the LSTM layer output; in the latter case both the LSTM and the Attention layers would have the very same input (produced by the embedding layers).

3.5.1 Multi-Head Attention

We firstly build our Attention layer based on Multi-Head Attention (Vaswani et al., 2017), where multiple attention functions are computed in parallel and the output is then reconstructed by concatenation and linearly projection to the expected dimension. Finally, we add a dropout layer (Srivastava et al., 2014) and layer normalization (Ba et al., 2016) for regularization.

3.5.2 Transformer encoder

Another experiment for the Attention layer has been to reproduce the encoder architecture of the Transformer (Vaswani et al., 2017) but using only a single layer, mainly due to computanional constraints, instead of the original 6 stacked identical ones.

3.6 Classification head

Lastly, the model architecture has a single linear layer classification head, which maps the output of the previous layer to the corresponding output space (defined in sec. 2). Task C+D also comprises a following softmax activation function.

4 Experimental setup

All the modules composing the architecture described in sec. 3 have been parametrized to make the model modular and have an effortless way to perform different experiments only with certain blocks of the network.

In order to evaluate the performance of the models, the main metric taken into account is the macro F1-score computed both for identification (tasks A and C) and classification (tasks B and D).

For task C+D, the dataset is always restricted to the subset annotated for the specific task, i.e. only the samples from the restaurant domain.

4.1 Training

During the training phase, the F1-score has been iteratively evaluated at each epoch end, along with the cross-entropy loss, to track the performance of the models on the development set. Furthermore, early stopping with a patience of 10 epochs has been applied on the same metric in order to prevent the model from overfitting. As for the optimizer, Adam (Kingma and Ba, 2014), with a learning rate of 1e-3, has been used to train all the models which results are shown in this report.

As a further way of considerably reduce the number of computations made by the LSTM layer and speed up the training, the widely used best practice of "packing" the batched input sequences has also been employed.

5 Experimental results

Table 2 shows a non-exhaustive list of experiments carried out with different model architectures. A comprehensive list of hyperparameters for both the proposed models is shown in Table 3.

An unforeseen discovery, as it is clearly visible from Table 2, has been that the addition of POS embedding (3.2) does not bring much improvement to the overall performance. However we have to take into account that the manual tagging may not be 100% accurate, and also that the BiLSTM and BERT may already infer enough information.

It is instead worth observing how adding context-dependent information (3.3) always yields a relevant increase in terms of macro F1-score; confirming the initial intuition on the benefits of employing it. Table 1 and fig. 2 also show a comparison of the different layer pooling strategies (3.3.1), which led to fix the "second-to-last" strategy for all the other experiments of task A+B and the "average" one for C+D.

Another explicit result that can be observed from Table 2 is that fine-tuning BERT did not make the cut to improve its performance. In addition to the already discussed reasons why an approach may work better than the other (3.3), this unsatisfactory behaviour may be also explained by the not so extensive experimentation performed on it, mainly due to the limited computational power; for instance, other training runs with several different learning rate values or partial unfreezing of the pre-trained weights may be beneficial.

As it was to be expected, the use of a bidirectional LSTM has proved to be always a key aspect in both tasks, whereas the combination with simple Attention layers did not bring very satisfactory improvements, yielding small increases for task C+D and none for A+B.

6 Conclusions

The modular nature of the proposed architecture made possible to easily perform an extensive experimentation on different models. We showed that a model can obtain satisfactory results by just combining static and contextual embeddings on top of a BiLSTM, reaching **47,41%** macro F1-score on the whole development dataset for task A+B, and 52,61% on its category-annotated subset for C+D; which also goes a little further by employing an Attention layer, reaching **53,59%**. See fig. 5 and 6 for a visualization of the proposed architectures.

As the experimental results (sec. 5) already point out, future work may include additional hyperparameter tuning, also specific to fine-tune BERT, and further investigation on how the Attention mechanism may be better exploited for this task.

References

Judit Ács, Ákos Kádár, and András Kornai. 2021. Subword pooling makes a difference.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2021. Attention in natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(10):4291–4308.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Wenpeng Hu, Jiajun Zhang, and Nan Zheng. 2016. Different contexts lead to different word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 762–771, Osaka, Japan. The COLING 2016 Organizing Committee.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for end-to-end aspect-based sentiment analysis. *CoRR*, abs/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.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks.

Kim Schouten and Flavius Frasincar. 2016. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

A Tables, plots and images

	Task A+B		Task C+D	
Pooling strategy	F1 Ident.	F1 Class.	F1 Ident.	F1 Class.
Last	80,75	45,54	82,60	49,87
Second-to-Last	**81,70**	**47,41**	83,99	52,20
Concat Last Four	80,73	44,37	82,82	49,56
Sum Last Four	79,77	44,19	82,09	48,04
Average Last Four	81,67	46,44	**84,36**	**52,61**

Table 1: BERT feature-based approach performances using different layer pooling strategies (based on a BiLSTM + GloVe architecture). Results are computed on the provided development set in terms of macro F1-score for both the Identification tasks (A and C) and the Classification ones (B and D). Histories of the performance at each epoch end are reported in fig. 2.

	Task A+B		Task C+D	
Model architecture	F1 Ident.	F1 Class.	F1 Ident.	F1 Class.
LSTM + GloVe	73,53	32,15	76,47	37,26
+ POS	68,55	30,20	70,32	35,71
+ BERT$_{frozen}$	79,96	40,53	82,09	44,95
+ BERT$_{frozen}$ + POS	74,45	35,23	77,37	41,02
BiLSTM + GloVe	75,26	38,82	78,04	44,12
+ POS	75,74	39,23	78,51	44,78
+ BERT$_{frozen}$	**81,70**	**47,41**	**84,36**	52,61
+ BERT$_{frozen}$ + POS	80,85	45,86	83,57	50,42
+ BERT$_{finetuned}$	79,78	40,75	82,86	47,58
BiLSTM + GloVe + Attention	74,17	36,40	78,37	39,59
+ Concat outputs	76,63	41,11	-	-
+ BERT$_{frozen}$	71,02	32,94	83,06	**53,59**
+ BERT$_{frozen}$ + Concat outputs	80,56	45,89	-	-
BiLSTM + GloVe + Transformer encoder	76,54	38,90	79,59	37,95
+ Concat outputs	77,41	41,66	-	-
+ BERT$_{frozen}$	65,27	30,27	81,33	48,94
+ BERT$_{frozen}$ + Concat outputs	80,97	46,00	-	-

Table 2: Non-exhaustive list of experiments carried out with different model architectures. BERT$_{frozen}$ refers to the *feature-based* approach (3.3.1) with the best layer pooling strategy from Table 1 (highlighted in bold), whereas BERT$_{finetuned}$ refers to the *fine-tuning* approach (3.3.2). In architectures with either "Attention" or "Transformer encoder" the default option is to give the attention output as input to the BiLSTM; when "Concat outputs" is specified instead, the two outputs are computed in isolation from the very same input and consequently concatenated.

Hyperparameter	Task A+B	Task C+D
Epochs	28	73
Random seed	42	42
Optimizer	Adam	Adam
Learning rate	1e-3	1e-3
Loss function	Cross Entropy	Cross Entropy
Batch size	8	8
Static embeddings	GloVe	GloVe
Static embeddings size	300	300
POS embeddings	False	False
POS embeddings size	-	-
LSTM layers	2	2
LSTM bidirectional	True	True
LSTM hidden size	128	128
LSTM input packing	True	True
Dropout	0,5	0,5
BERT model	bert-base-cased	bert-base-cased
BERT finetuning	False	False
BERT layer pooling strategy	second_to_last	mean
BERT pooled layers	-	[-1, -2, -3, -4]
BERT WordPiece pooling strategy	mean	mean
Attention	False	True
Attention heads	-	12
Attention dropout	-	0,2
Concat Attention out to LSTM out	-	False

Table 3: Hyperparameters' values of the final proposed models for both tasks A+B and C+D, which are the best performing ones (in terms of classification F1-score) from Table 2, highlighted in bold.

Tokenized sentence	The	hard	drive	crashed	as	well	so	I	bought	a	new	power	cord
Polarity		negative										negative	
IOB + Polarity	O	B-negative	I-negative	O	O	O	O	O	O	O	O	B-negative	I-negative

Figure 1: Task A+B tagging example of the sentence "The hard drive crashed as well so I bought a new power cord" based on the IOB (Inside, Outside, Beginning) scheme. Each token without a polarity is tagged with an "O", whereas the aspect terms have an associated polarity and are either just tagged with "B-*polarity*" when they are single-words, or also followed by $n-1$ "I-*polarity*" when composed by n tokens (multi-words).

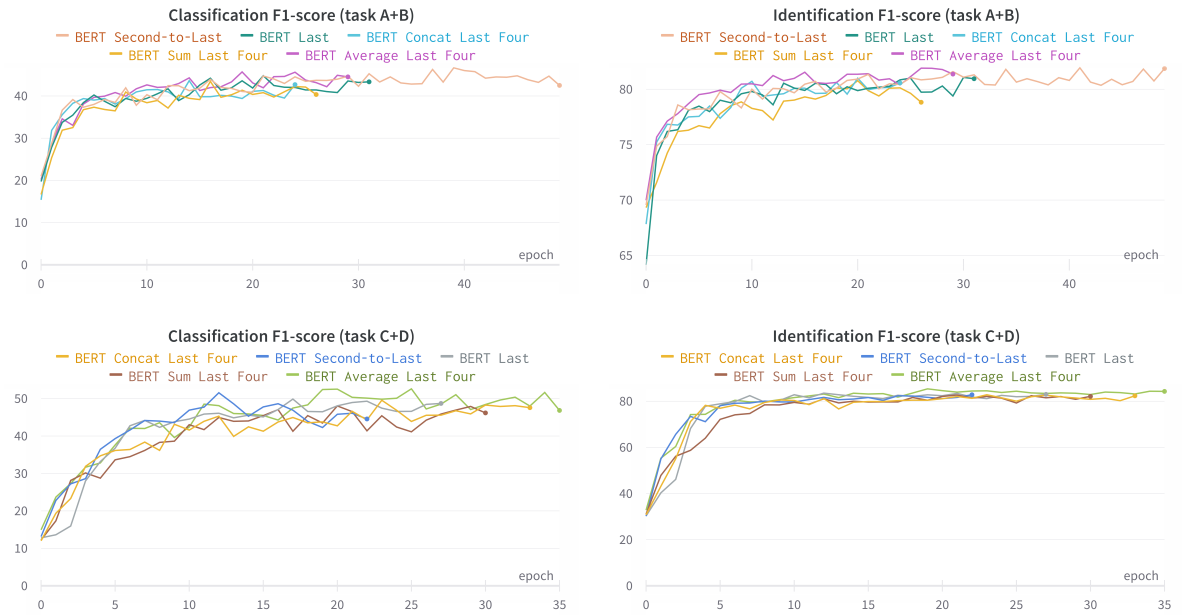

Figure 2: BERT feature-based approach performance histories using different layer pooling strategies (based on a BiLSTM + GloVe architecture). Results are computed on the provided development set in terms of macro F1-score for both the Identification tasks (A and C) and the Classification ones (B and D). Highest values for each strategy are reported in Table 1.

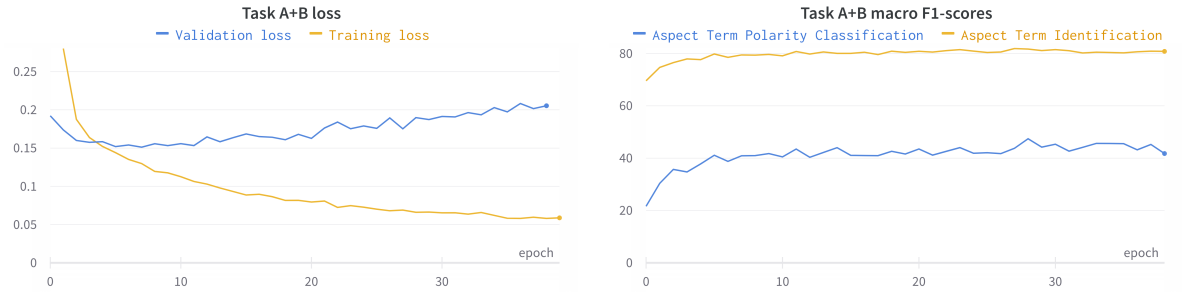

Figure 3: Task A+B loss and macro F1-scores histories of the proposed, and best berforming, model (BiLSTM + GloVe + BERT$_{frozen}$ from Table 2). Loss is computed on both training and development/validation sets, whereas F1-scores refers to the development/validation set.

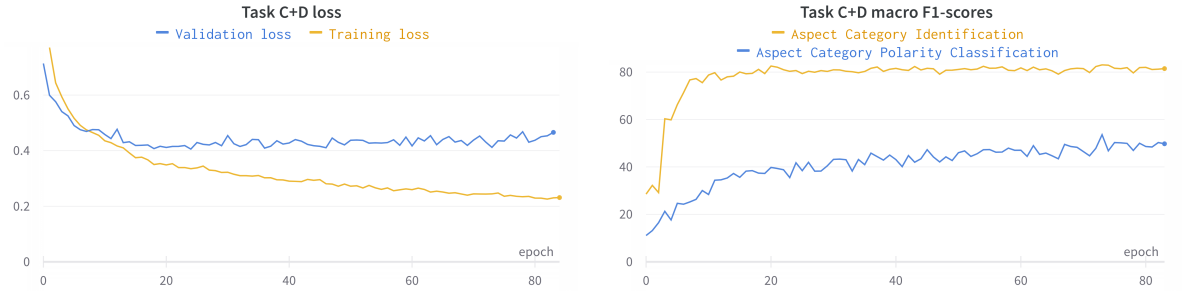

Figure 4: Task C+D loss and macro F1-scores histories of the proposed, and best berforming, model (BiLSTM + GloVe + Attention + BERT$_{frozen}$ from Table 2). Loss is computed on both training and development/validation sets, whereas F1-scores refers to the development/validation set.

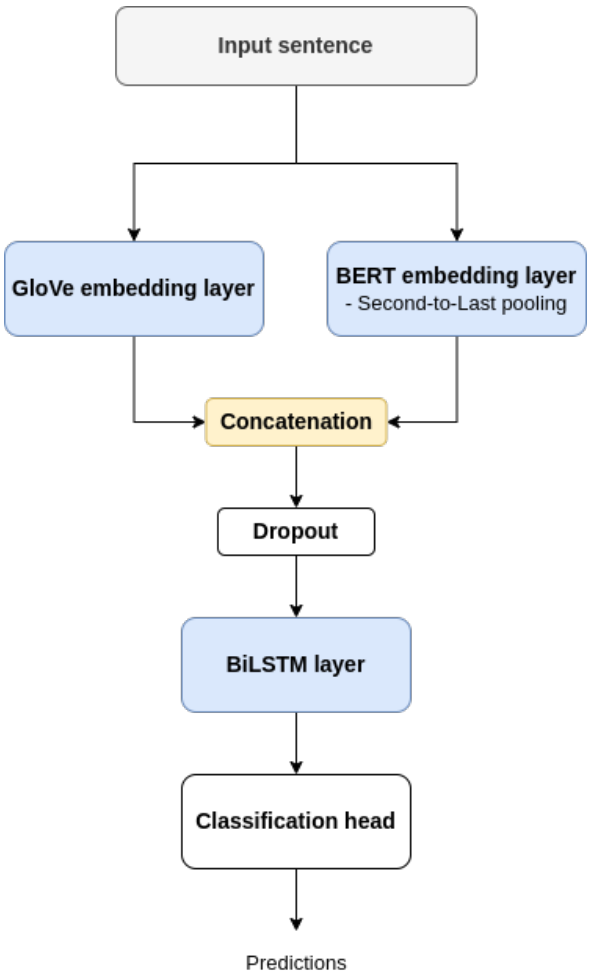

Figure 5: Architecture of the proposed model for task A+B (BiLSTM + GloVe + BERT$_{frozen}$ from Table 2).

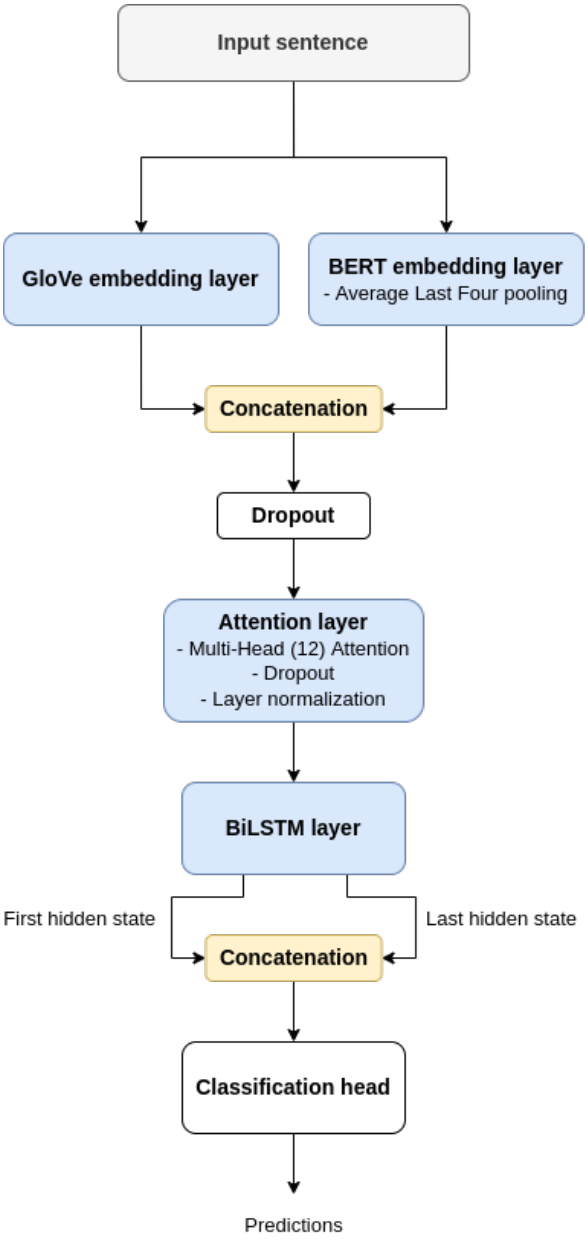

Figure 6: Architecture of the proposed model for task C+D (BiLSTM + GloVe + Attention + BERT$_{frozen}$ from Table 2).

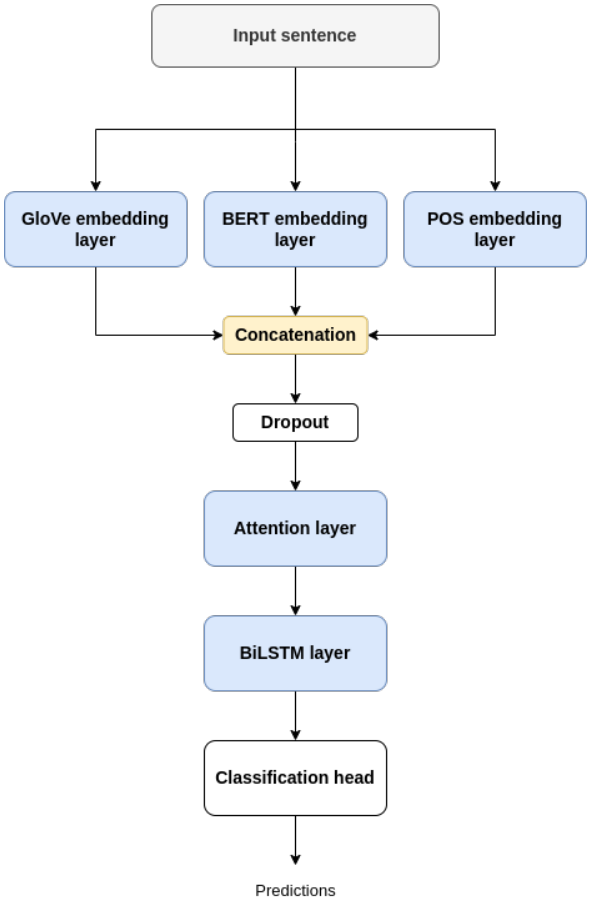

Figure 7: Model architecture comprising all the building blocks described in section 3, with the Attention layer output directly given as input to the LSTM layer.

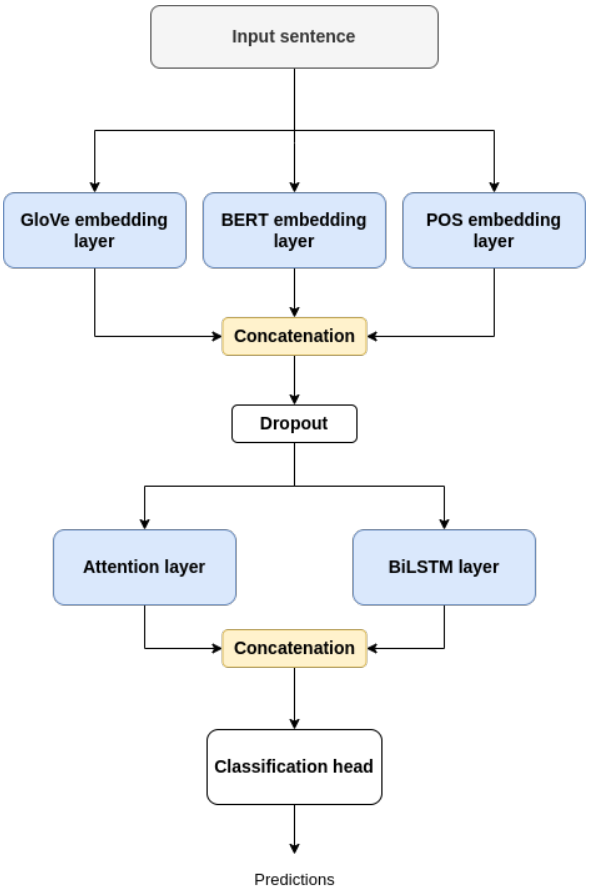

Figure 8: Model architecture comprising all the building blocks described in section 3, with the outputs of the Attention and the LSTM layers computed in isolation from the very same input and consequently concatenated.