NLP HW2: Aspect-Based Sentiment Analysis (ABSA)

Luigi Sigillo
sigillo.1761017@studenti.uniroma1.it

1 Introduction

ABSA aims to identify the aspect terms of given target entities (in our case restaurants and laptops) and the sentiment expressed towards each aspect. For example for this sentence: *"I love their **pasta** but I hate their **Ananas Pizza**"*. We want to identify the target words: pasta and Ananas pizza, and then classify their sentiment, so positive and then negative. Instead for this sentence: *"The people there is so kind and the taste is exceptional"*, the task is to classify categories and their sentiment associated so: **staff** negative and **food** positive. The task can be divided in: (A) Aspect term identification, (B) Aspect term polarity classification, (C) Aspect category identification and (D) Aspect category polarity classification. I followed two approaches to face these problems, adapted depending on the specific task (A,B,C,D): a RNN model, using Bi-LSTM and a transformers model, fine tuning BERT.

2 Preprocessing

We have two datasets containing texts of different macro categories, laptops and restaurants. An entry of the dataset is composed by the sentence, the targets with the polarities associated, and in the restaurants dataset also the categories with the respective polarities. I have noticed that a lemmatisation of the sentences does not increase the performance, but other NLP best practice like removal of stop-words and punctuation does, this is visible in `fig 1`. Following the idea of the IOB tagging (Ramshaw and Marcus, 1995), since we have to find the target words inside a sentence, I used the "B" to tag the target word if composed by only one word, and if it is the case the "I" to tag the continue of the word. The other words of a sentence are simply tagged as "O".
Only for task B, when using a WiC approach, I decided to wrap the target words to be classified between two tokens <**START** > and <**END** >. For this approach I classify only one sentiment per sentence, if there are multiple sentiments, I add a sentence for each sentiment (a sort of data augmentation), and then reconstruct the original sentence with the multiple sentiments associated to the targets. The architecture is showed in `fig 2`.
BERT uses WordPiece tokenization(Wu et al., 2016), this tokenization avoids to use many unknown tokens, splitting them in differents subtokens, this causes a different length of the original sentences with respect to a tokenized, and so I have handled this, taking only the first token of a splitted word.

2.1 Pretrained embeddings

When using Bi-LSTM approaches to vectorize the words of a sentence I tested different pretrained word embeddings: GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017) and Chargram (Wieting et al., 2016). Those pretrained embeddings are useful because they have been trained on larger corpura and are more precise with respect to an embedding layer that we could have trained from scratch.

3 Models architectures

I tried different approaches: only for task B a WiC approach, then for task A+B an approach similar to a NER tagging task so identifying tokens which correspond to a predefined set of "entities", firstly with a Bi-LSTM, then more powerful since it uses the BERT transformer model pretrained, that I have properly fine tuned.

3.1 Task A-B

3.1.1 Bi-LSTM

In this approach I used Bi-LSTMs (Graves and Schmidhuber, 2005), I have tried with different

embedding layers, one with the pretrained embeddings and one without it, using instead a random initialization of the embeddings. So the network is composed by this embedding layer, then a two layers Bi-LSTM and at the end a linear layer to classify the labels. I used two dropout layers, one before and one after the Bi-LSTM layer. I decided to use a window size (maximum number of word per sentences) of 25. The loss function used is the categorical cross entropy.

3.1.2 WiC approach for Task B

This approach is the one of the HW1 "WiC disambiguation". I tried with Bi-LSTM, that outputs one vector for each word, so I extracted the hidden states $(h_{t1}, ..., h_{tn})$ of the target words of the sentences, multiplied the two vectors and fed them into a classifier, the network uses an embedding layer with the GloVe embeddings and two recurrent layers for the Bi-LSTM. I tried this approach also with BERT, taking the keywords vectors from its last hidden state representation and then fed them into the MLP. The MLP for both is composed by two fully connected layer to classify using the Leaky ReLU as activation function and the CrossEntropy as loss function.

3.1.3 BERT for token classification

I fine tuned the Bert model (Devlin et al., 2019), in its token classification version to do NER tagging, and used it for the task A and B. I also tried with DistilBERT (Sanh et al., 2020) that is a smaller and faster version of BERT. The architecture is presented in `fig 4`.

3.2 Task C-D

3.2.1 DistilBERT for Sequence classification

I used DistilBERT for task C and D but for sequence classifciation because we have to assign to each sentence one or more categories, so uses a classification layer after BERT.
The base model outputs a vector of length 768 for each token, after it there is a linear layer of 20 outputs (5 categories $*$ 4 sentiments). The activation function is a sigmoid[1] while the loss function used is the binary cross entropy. The architecture is shown in `fig 5`.

[1] I binarized the classes to perform the multilabel classification

4 Experiments

Since using Bi-LSTM showed not very promising result with respect to the usage of the transformers, I decided to not tune much the approaches with Bi-LSTMs. Despite that I noticed an interesting thing: the pretrained embeddings performed worst or similar to a random assignment, visible in the `Figure 3` showing a graph of the F1 scores. I think because we are using, for example for the laptop dataset, very specific words, causing this to have a lot of UNK token in a pretrained approach. I tried with DistilBERT and the original BERT, in both cases with the *uncased* version because lower casing the sentences brought better results as shown in 1. I have done fine tuning, using the pretrained models, so the results are very promising after only 5 epochs, I noticed overfitting with an higher number of epochs. I used AdamW (Loshchilov and Hutter, 2019) as optimizer for both A-B and C-D with 0.00005 and 0.00002 as learning rate respectively. I find out that the batch size that gives the best results is 64 for both models.

5 Results

I considered separately the task of target and sentiment classification. So I will evaluate tasks: **A+B** (a model that does both task), **B** (model that does only B with targets provided), **A->B** (a model that does A and then use the B model with targets provided by A) and **C+D** (a model that does both task). A fine tuned BERT model performs better than the DistilBERT version, but not in all the cases. All the models have been trained and evaluated with the training sets joint together (restaurants and laptops), although using one dataset at time results in an increment of 2-3% in the macro F1.

5.1 Task A-B

For the two subtasks A and B, I basically used the same model adapted for aspect and sentiment. For the task B I have also see how the precision per sentiment is distributed, the values are showed in percentage in `Table 1`.

5.2 Task B

The difference in this task is that the target words are given, so we expected the performance to be better with respect to the A+B task, in fact I gained around the 15% in terms of F1 score as shown in `Table 2`. My baseline with DistilBERT performs better then the WiC approach with Bi-LSTM,

but using the WiC approach with DistilBERT I reached 49.8% of macro F1, so higher than the 35.4% reached with the NER tagging approach for the sentiment analysis.

5.3 Task C-D

The results obtained after 5 epochs, are interesting because the problem is huge, having 20 different labels. The problem could have been reduced using only the category tag and predicting the sentiment using another model for example like done in the B task, the results are showed in `Table 2`

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information.

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

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610. IJCNN 2005.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

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.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Tables

Model	Negative	Positive	Neutral	Conflict
Bi-LSTM	37.5	34.9	21.7	0
BERT	62.5	57.8	38.4	0

Table 1: Per class precision for the task B.

	Model	Target	Sentiment
Task A+B	Bi-LSTM	7.2	1.4
	DistilBERT	65.9	30.8
	BERT	69.2	35.4
Task A–>B	Bi-LSTM	7.1	1.1
	DistilBERT	68.7	28.5
	BERT	**71.5**	33.9
Task B	WiC Bi-LSTM	/	22.8
	DistilBERT	/	40.9
	WiC DistilBERT	/	**49.8**
	WiC BERT	/	48.5
Task C+D	DistilBERT	60.1	32.6
	BERT	50.8	34.1

Table 2: F1 scores in percentage for the different tasks using for evaluation both dataset together.

Figures

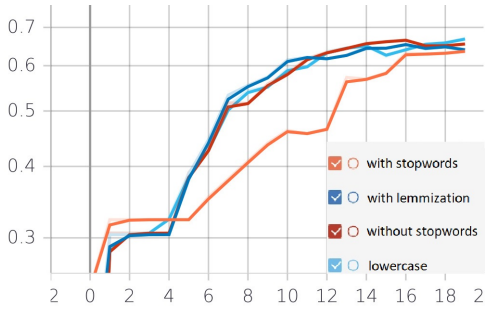

Figure 1: Comparison of overall F1 to see how stop words removal affects the model performances.

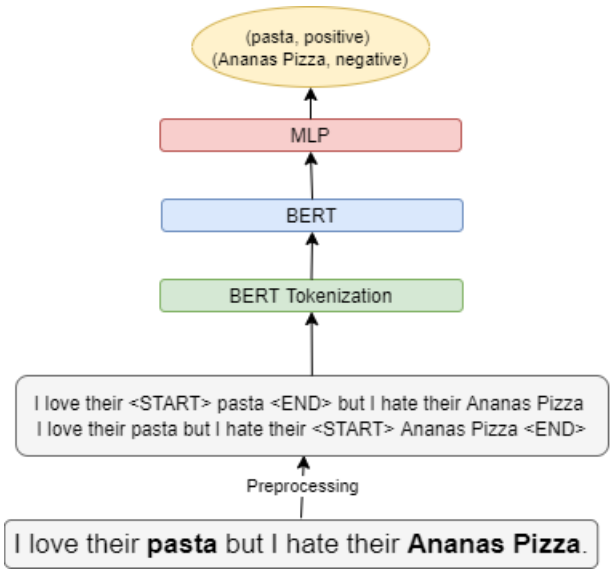

Figure 2: Architecture following WiC approach for task B

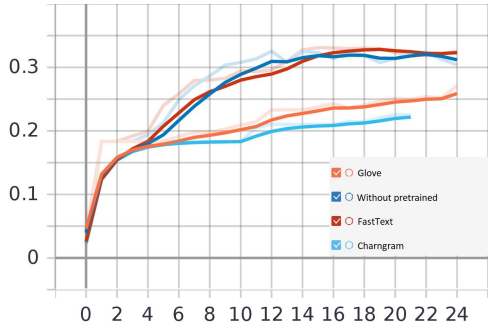

Figure 3: Comparison of F1 scores using different pre-trained embeddings.

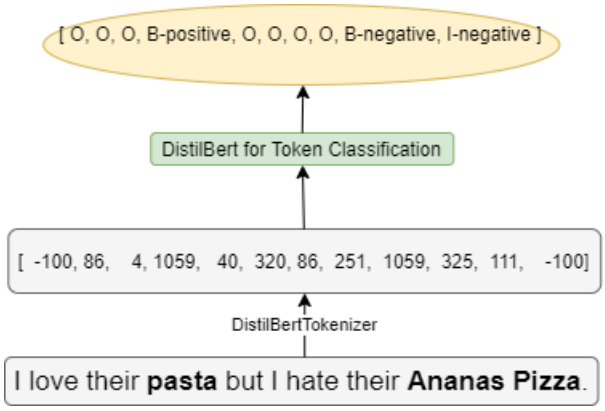

Figure 4: Architecture for task A+B using transformers BERT fine tuned.

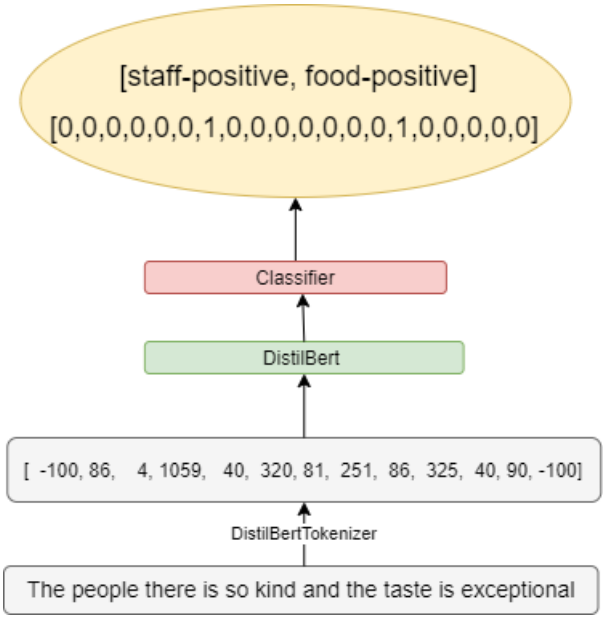

Figure 5: Architecture for task C+D using transformers BERT for sequence classification fine tuned