

Homework 2: Aspect-Based Sentiment Analysis

Simone Rossetti

rossetti.1900592@studenti.uniroma1.it

1 Introduction

Aspect-Based Sentiment Analysis (ABSA) aims to identify the aspect terms of given target entities and the sentiment expressed towards the respective terms.

We will address the problem of performing ABSA on one of the publicly available ABSA benchmark dataset, namely the SemEval2014 Task 4¹, made of about 6K restaurants and laptops reviews.

Further details about the dataset and ABSA are treated here (Pontiki et al., 2014), in particular I will focus on the methodology I adopted to solve the 4 subtasks i.e. (a) aspect term identification, (b) aspect term polarity classification, (c) aspect category identification and (d) aspect category polarity classification.

Notice: extras are marked in bold.

2 Task A

Aspect Term Identification, also named Aspect Term Extraction (ATE), can be seen as a sequence labeling task. An aspect term may be a unique word or a sequence of words which are target of a sentiment. A very common strategy in NLP is to perform sequence labeling following the IOB (Inside-Outside-Beginning) tagging paradigm, introduced for the first time here (Ramshaw and Marcus, 1995), used in several tasks such as NER (Named Entity Recognition) or POS (Part-Of-Speech) tagging, with the aim of linking chunks of the same entity².

The task boils down to learning the probability distribution over the 3 tags classes: 'B' (Beginning), 'I' (Inside) and 'O' (Outside).

Sequence labeling is a well treated topic in the

literature, there exists several approaches (He et al., 2020). In general a precise pipeline is used: Embedding Module, Context Encoder and Inference Module. Most promising methods uses a well known inference model named Conditional Random Fields (CRFs), which are discriminative models used in general for many kinds of pattern recognition problems, which aims at maximizing the conditional distribution over the entire sequence of observations taking into account the whole context (i.e. not considering the independence assumption typical of the Hidden Markov Models (HMM)). I mostly focused on two different Context Encoder methodologies.

2.1 Method 1 - FastText+BiLSTM+CRF

Taking as reference this work (Giannakopoulos et al., 2017) I realized a similar pipeline to perform ATE. First of all, considering the very little corpus size on which I trained, I decided to implement a character-level embeddings module, made of a character Embedding Layer and a BiLSTM encoder. This strategy is **very common in NLP to overcome the problem of Out-Of-Vocabulary (OOV) words** when using fixed-size vocabularies. A different word-level embeddings module loads **pre-trained word embeddings from FastText³** trained on Common Crawl dataset and fine-tuned on the training set. At this point each token of a sentence is encoded by concatenation of its FastText embedding representation and the character-level one. A second BiLSTM encodes the context of the sentence within the token embeddings and outputs the final words representations. Finally a linear layer produces logits into the tag space over the IOB classes. A **CRF head⁴** learns the weights $w \in \mathbb{R}^{N \times N}$ of transitions $P(Y_i, Y_{i-1})$ and emissions $P(Y_i, X_i)$ where $i = 1, \dots, L$, L the sequence

¹<https://alt.qcri.org/semeval2014/task4/>

²[https://en.wikipedia.org/wiki/Inside-outside-beginning_\(tagging\)](https://en.wikipedia.org/wiki/Inside-outside-beginning_(tagging))

³<https://fasttext.cc/docs/en/crawl-vectors.html>

length, N the tag space dimension (3 in our case), $Y \in \{I, O, B\}$ the labels and $X \in \mathbb{R}^N$ the observed scores. CRF weights are trained maximizing the log-likelihood over the entire sequence, considering the dependencies within all random variables (actually⁴ uses Viterbi’s Algorithm which condition over preceding neighbor). Since the PyTorch-CRF head⁴ computes internally the log-likelihood, it’s negative value is taken (negative log-likelihood) and is minimized using stochastic gradient descent, namely Adam optimization with $1e - 3$ learning rate. See **Table 9** for more details.

2.2 Method 2 - DistilBert+BiLSTM+CRF

Intuitively from the previous architecture one can substitute the Embedding Module and the Context Encoder with a more **recent technology based on Attention**, namely a Transformer-based Encoder.

*”DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT’s performances as measured on the GLUE language understanding benchmark.”*⁵

These are the main reasons which motivates my choice to use the **DistilBERT Pre-Trained Encoder** module, which compared to other models, included BERT uncased version, runs pretty much ”faster” and can be **fine-tuned** on my aged Nvidia GeForce GTX 1070 (8GB RAM). Furthermore the results shown here⁶ and here (Sanh et al., 2020) are quite appealing. The model size is 260MB for a total of 66M parameters.

The DistilBERT’s contextualized encoded sequence is passed to a Linear layer which, as well as before, produces logits scores over the labels set, again a CRF head predict the most probable tagging. Notice that this task does not require the use of special tokens.

DistilBERT Encoder ”knowledge” and it’s multi-head self-attention layers improved the overall performance as expected.

One may opt to preserve the BiLSTM layer after the Transformer one and **this produces a very little improvement** but it looks somehow redundant as well. See **Figure 23** and **Table 10**.

⁴<https://pytorch-crf.readthedocs.io/en/stable/>

⁵https://huggingface.co/transformers/model_doc/distilbert.html

⁶<https://medium.com/huggingface/distilbert-8cf3380435b5>

3 Task B

Aspect Term Polarity Classification has been treated in many ways in the literature. Since the most promising methods uses Transformer-based architectures I decided to keep using DistilBERT Encoder as context encoder. A very intuitive way (in my opinion) to handle the presence of multiple aspect terms (with different polarities) within the same sentence is by positional extraction. Among the many possibilities, I decided to follow this approach (Wu et al., 2020), which to me really exploit the strengths of attention-based mechanisms, and stays as simple as effective and it works as follows.

First of all I created two new special tokens, ”[AS]” (Aspect Start) and ”[AE]” (Aspect End), then I inserted them respectively before and after each aspect term for each sentence. Once the tokenized sentence is encoded by the Transformer model, the two encoded special tokens, $h_{[AS]}^i$ and $h_{[ES]}^i$ with $i = 1, \dots, T^j$ and T^j the number of tokens in the j -th sentence, are extracted and concatenated, then the new representation, namely $[h_{[AS]}^i, h_{[ES]}^i]$, is fed to two subsequent Linear layers which produces probabilities over the four polarity classes, gathered by GELU activation function (Hendrycks and Gimpel, 2020) which resulted in being much more beneficial than others (e.g ReLU, ELU, etc.). See Figure 24 and Table 11.

Notice that while in the reference paper they only classify over the encoded state $h_{[AS]}^i$, **in my implementation I obtained improvements by classifying over the concatenation** $[h_{[AS]}^i, h_{[ES]}^i]$.

This approach permits to learn two embeddings for the two markers $h_{[AS]}$ and $h_{[ES]}$ which are common to all the aspect terms and that are not biased towards specific words representation (i.e. in case of positional extraction without the use of a specific marker).

Furthermore in this way we exploit the self-attention marginalization and the positional encoding of the Transformers architecture to give a specific polarity context to the markers.

Results are very promising but it suffers of data unbalancing (even if applying weighting loss mechanisms), in any case less than other methods I tested. In fact the ”conflict” class deteriorates the overall performance (in many papers the ”conflict” class is even not considered). Which frequency with respect to the most frequent class ”positive” is 1 : 23.

4 Task C

Aspect Category Identification is a sequence multi-labels classification task. I decided to fine-tune DistilBERT Encoder to produce a category embedding of the whole sentence over the first token (a pooled output), namely the "[CLS]" one, which is in general the opening token of each sample and **it is in fact used for sequence classification purposes** (e.g. see (Song et al., 2020)).

Again the category embedding of each sentence, namely $h_{[CLS]}^i$ of the i -th sentence, is processed by two Linear layers gathered by a GELU activation function. A Binary Cross Entropy Loss with sigmoid activation function is used to learn the multi-labels distributions. See Figure 25 and Table 12.

As expected the results are quite good. Abstract classes such as "ambience" results in being more difficult to predict with respect to more concrete concepts such as "price" regardless to the labels frequencies.

5 Task D

Aspect Category Polarity Classification is in general treated in the literature in the same way Aspect Term Polarity Classification is treated (e.g see (Wu et al., 2020)), **I decided instead to follow a simpler and intuitive strategy which came to my mind** which do not requires any sentence pairing, positional extraction or whatever; the reasoning is as follows.

We said before that the embedded state $h_{[CLS]}^i$ of the i -th sentence represent its category(-ies) embedding, thus we do not need anything else to perform polarity classification. And to deal with the multi-label nature of the problem we can build a multi-head classifier. Further more the correctness of this reasoning is given by the fact that each category is independent by the others.

Thus since we have in total five possible categories, we **parallelize the training of five classifiers heads specialized in learning the polarity distribution over each category independently**, predictions are then stacked together.

Each classifier has the same structure adopted for the classifiers of the precedent tasks, that is, two Linear layers gathered by GELU.

Of course the Cross Entropy Loss is then computed by taking the predicted polarity of the relative category label in input. See Figure 26 and Table 13.

6 Final Model

To preserve the simplicity and modularity of the problem I decided to **keep separates networks each specialized in one subtask**. To motivate further my choice I will address the following reasoning.

One may opt of course for collapsing the four modules in one (A+B+C+D) or maybe in two (A+B and C+D), which is of course feasible but, will it really works? There are multiple factors which should be taken into account according to this post⁷; one for instance is the model capacity, which in our case results in being reduced with respect to larger Transformer-based models, the second is the task covariance, that is the similarity of two or more tasks. Following this reasoning we can clearly assert that tasks A and B are very dissimilar with respect each other and all the others while instead tasks C and D may be good candidates for multi-task training.

But at the end what will effectively changes is the backbone training (in my case the DistilBERT Encoder module), since the heads are separate.

To me training a unique DistilBERT Encoder model and joint classifiers for the tasks C+D did not result in being beneficial.

For this reason the final model named ABSADB (used for inference purpose) is a module which join together the four submodules, one for each task, thanks to **pre/post-processing methods**. Accordingly one may decide to perform task A/A+B/C+D inference by selecting the opportune arguments.

7 Results

PyTorch is the background framework which I used to define the Models architectures. PyTorch Lightning (PL) framework has been used to facilitate the training process, prepare datasets, save models, tracking, compute metrics and evaluations.

Transformers library⁸ has been used to load pre-trained DistilBertModel, namely the "distilbert-base-uncased", and DistilBertTokenizerFast model.

To store the information relative to the training and evaluations history I used TensorBoard (PL natively integrates TB loggings).

Trainings are stopped after 5 epochs over-fitting is reached and top models are saved during epochs.

⁷https://hazyresearch.stanford.edu/blog/2020-03-01-multi_task_transfer_learning

⁸<https://huggingface.co/transformers/>

Results are shown from Table 1 to Table 8.

Metric	Avg
Precision	76.22
Recall	77.30
F1	76.76

Table 1: TASK A. Method 2. Evaluations on Laptops DevSet.

Metric	Macro avg	micro avg
Precision	85.04	85.40
Recall	86.52	87.33
F1	85.56	86.35

Table 7: TASK C. Evaluations on Restaurants DevSet.

Metric	Macro avg	micro avg
Precision	62.99	77.70
Recall	65.06	77.70
F1	63.72	77.70

Table 2: TASK B. Evaluations on Laptops DevSet.

Metric	Macro avg	micro avg
Precision	58.73	67.11
Recall	52.77	68.63
F1	54.32	67.86

Table 8: TASK C+D. Evaluations on Restaurants DevSet.

Metric	Macro avg	micro avg
Precision	47.17	57.27
Recall	49.65	58.22
F1	47.77	57.74

Table 3: TASK A+B. Evaluations on Laptops DevSet.

Metric	Avg
Precision	83.66
Recall	85.43
F1	84.53

Table 4: TASK A. Method 2. Evaluations on Restaurants DevSet.

Parameter	Value
Vocabulary Size	3206
Embeddings Dim	300
LSTM Directions	2
LSTM Layers	2
LSTM Hidden Dim	128
Batch Norm Type	LayerNorm
Dropout	0.3
Batch Size	128
Optimizer	Adam
CRF	True
lr	1e-3

Table 9: TASK A. Method 1 Model Summary.

Metric	Macro avg	micro avg
Precision	62.63	77.61
Recall	59.21	77.73
F1	60.44	77.67

Table 5: TASK B. Evaluations on Restaurants DevSet.

Metric	Macro avg	micro avg
Precision	50.71	63.46
Recall	47.70	65.00
F1	48.48	64.22

Table 6: TASK A+B. Evaluations on Restaurants DevSet.

Parameter	Value
Vocabulary Size	30522
Embeddings Dim	768
LSTM Directions	2
LSTM Layers	2
LSTM Hidden Dim	348
Batch Norm Type	LayerNorm
Dropout	0.3
Batch Size	96
Optimizer	Adam
CRF	True
lr	5e-5

Table 10: TASK A. Method 2 Model Summary.

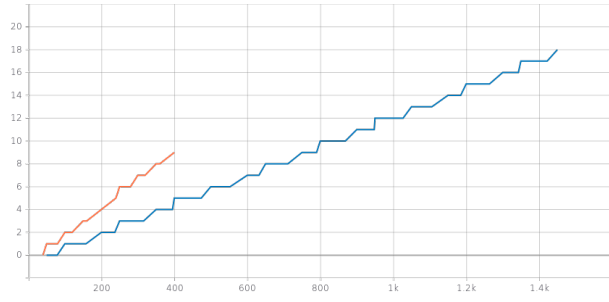


Figure 1: TASK A. Visited samples with respect to the epochs. Orange: Method 1, Blue: Method 2.

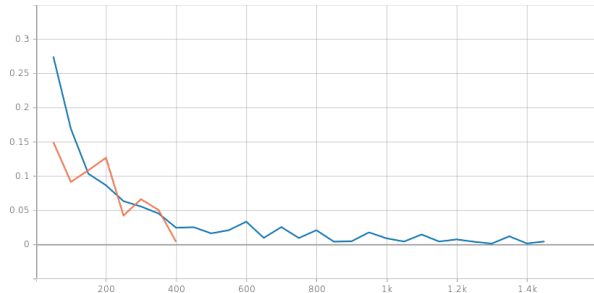


Figure 2: TASK A. Training losses. Orange: Method 1, Blue: Method 2.

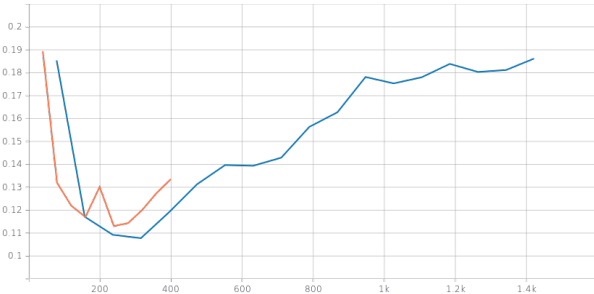


Figure 3: TASK A. Validation losses. Orange: Method 1, Blue: Method 2.

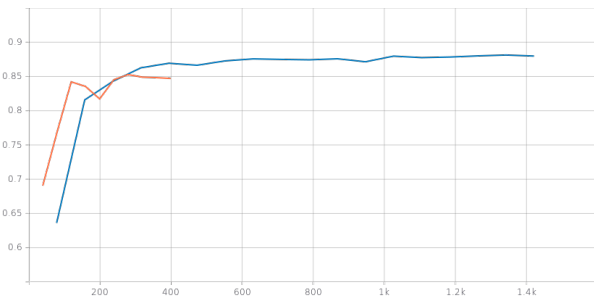


Figure 4: TASK A. F1 score. Orange: Method 1, Blue: Method 2.

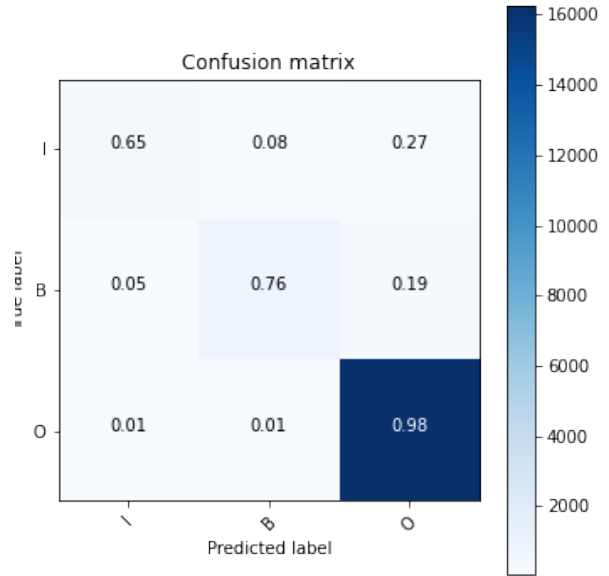


Figure 5: TASK A. Confusion Matrix. Method 1.

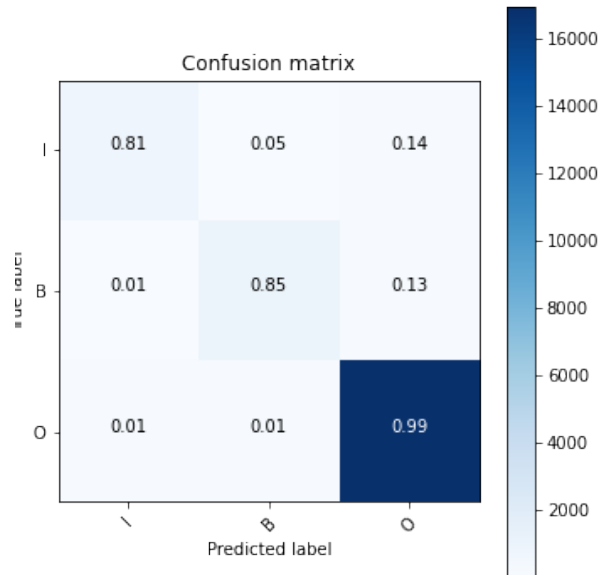


Figure 6: TASK A. Confusion Matrix. Method 2.

Parameter	Value
Vocabulary Size	30524
Embeddings Dim	768
Batch Norm Type	LayerNorm
Classifier Layers	1
Dropout	0.4
Batch Size	96
Optimizer	Adam
lr	5e-5

Table 11: TASK B. Model Summary.

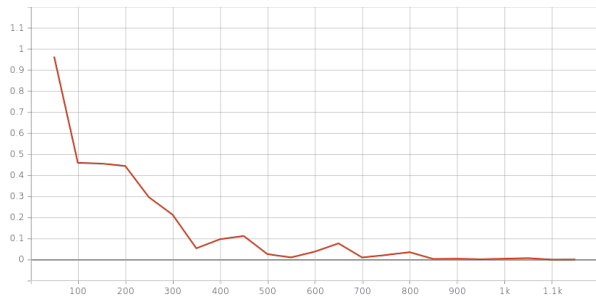


Figure 7: TASK B. Training loss.

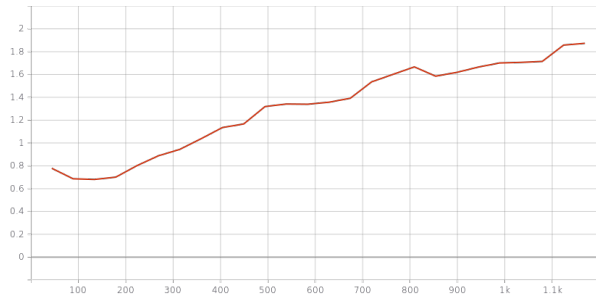


Figure 8: TASK B. Validation loss.

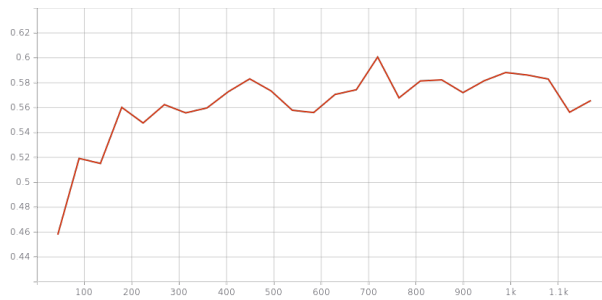


Figure 9: TASK B. F1 score.

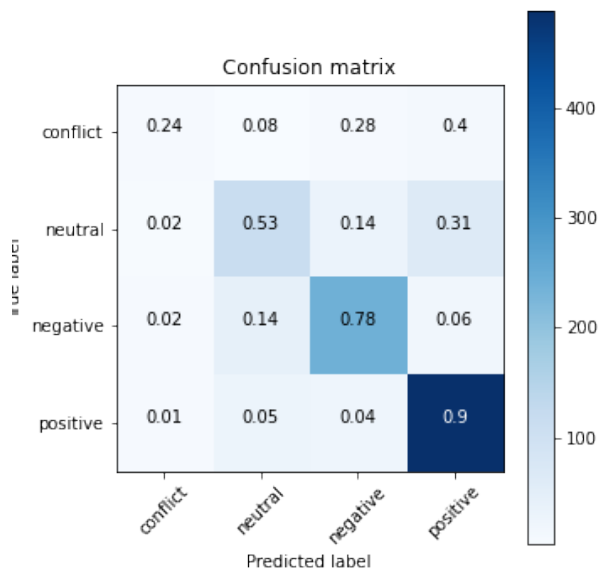


Figure 10: TASK B. Confusion Matrix.

Parameter	Value
Vocabulary Size	30522
Embeddings Dim	768
Batch Norm Type	LayerNorm
Classifier Layers	1
Dropout	0.4
Batch Size	96
Optimizer	Adam
lr	5e-5

Table 12: TASK C. Model Summary.

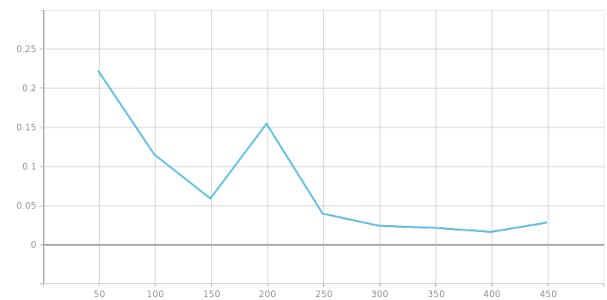


Figure 11: TASK C. Training loss.

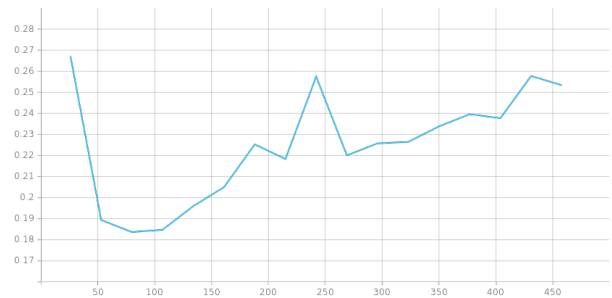


Figure 12: TASK C. Validation loss.

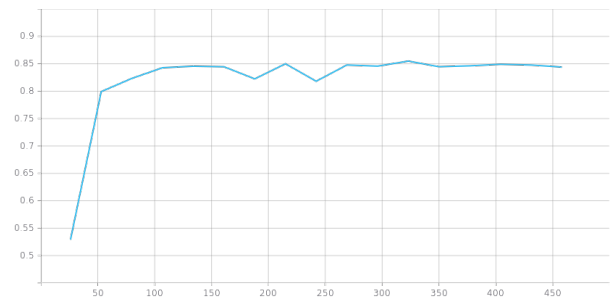


Figure 13: TASK C. F1 score.

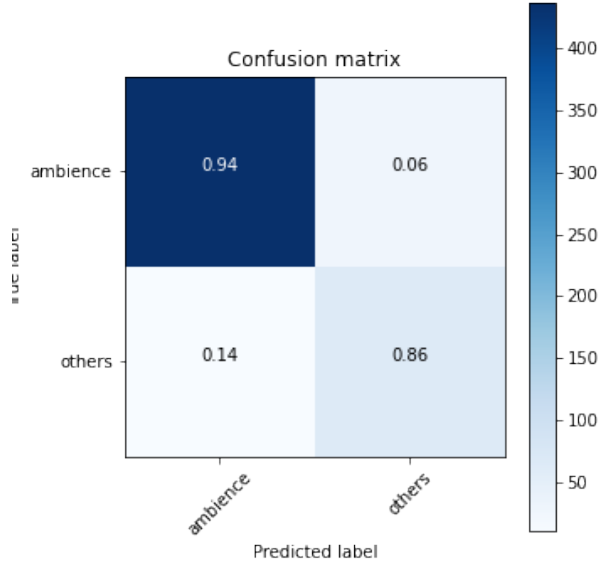


Figure 14: TASK C. Confusion Matrix. Ambience class.

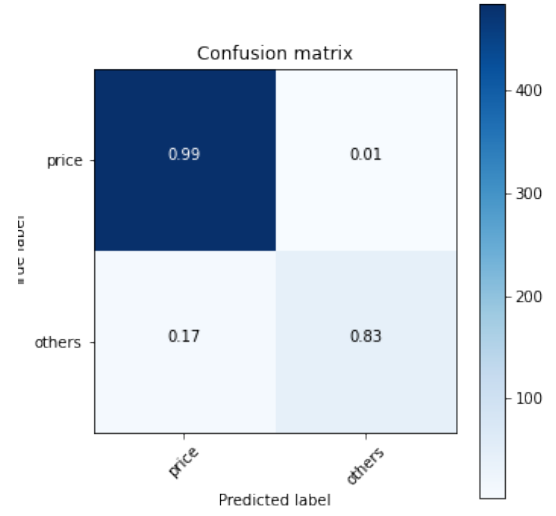


Figure 16: TASK C. Confusion Matrix. Price class.

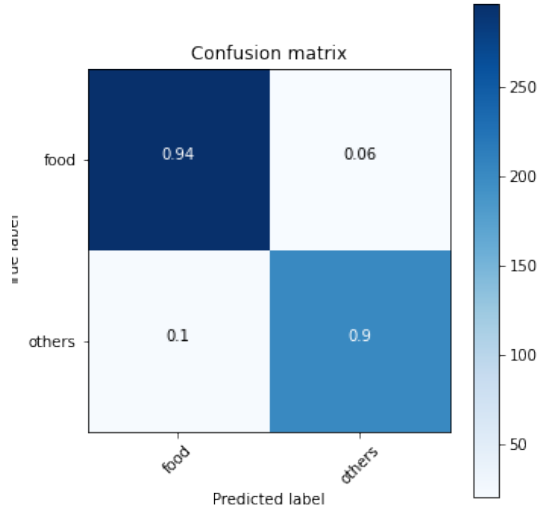


Figure 15: TASK C. Confusion Matrix. Food class.

Parameter	Value
Vocabulary Size	30522
Embeddings Dim	768
Batch Norm Type	LayerNorm
Classifier Layers	5
Dropout	0.4
Batch Size	96
Optimizer	Adam
lr	5e-5

Table 13: TASK D. Model Summary.

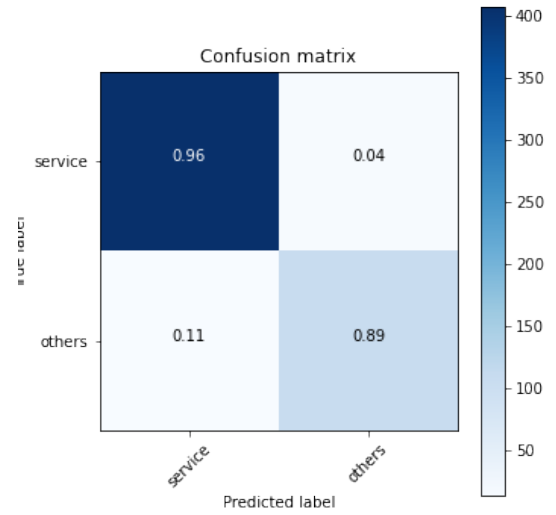


Figure 17: TASK C. Confusion Matrix. Service class.

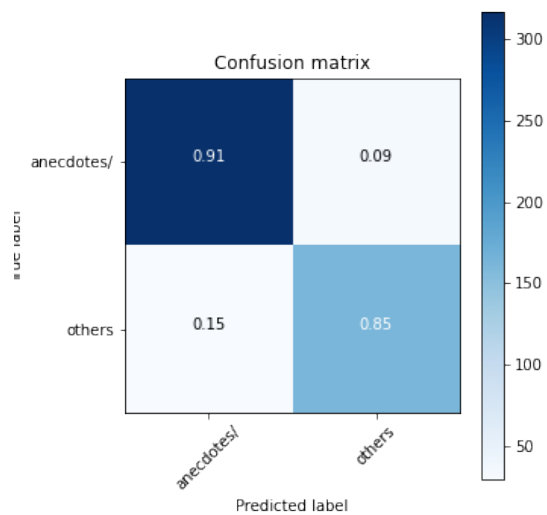


Figure 18: TASK C. Confusion Matrix. Anecdotes/miscellaneous class.

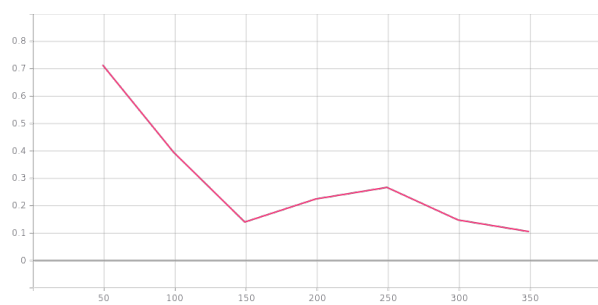


Figure 19: TASK D. Training loss.

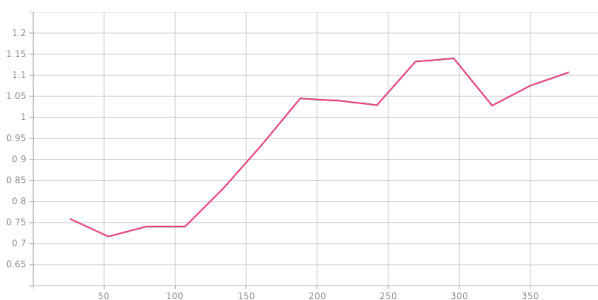


Figure 20: TASK D. Validation loss.

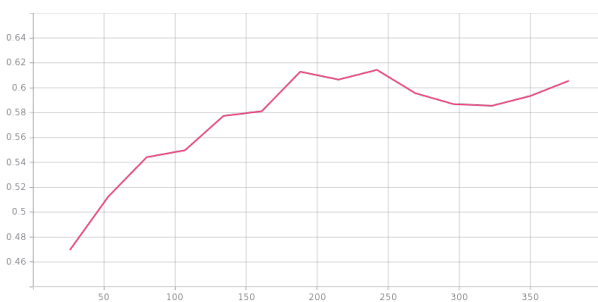


Figure 21: TASK D. F1 score.

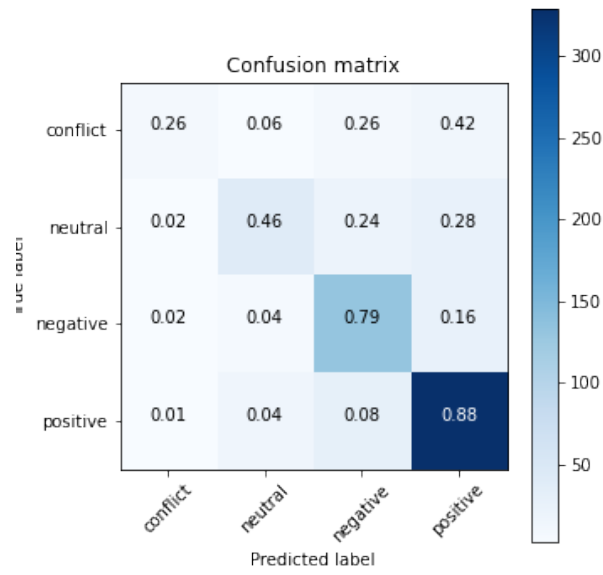


Figure 22: TASK D. Confusion Matrix.

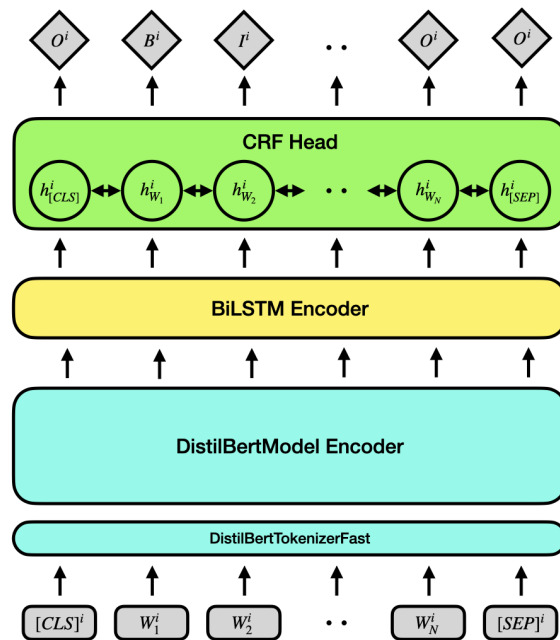


Figure 23: TASK A. Method 2 Model Architecture.

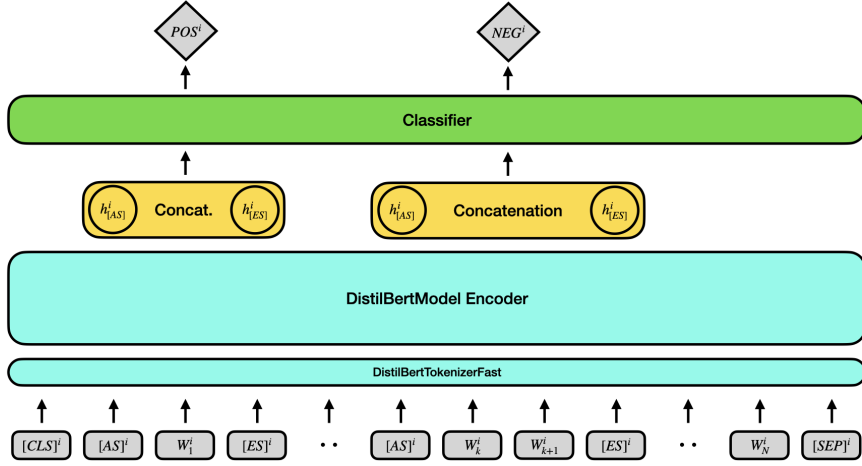


Figure 24: TASK B. Model Architecture.

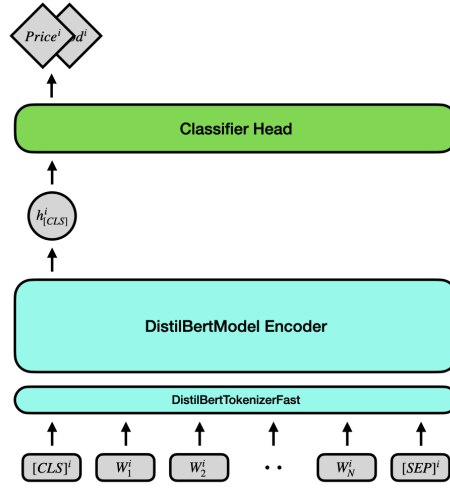


Figure 25: TASK C. Model Architecture.

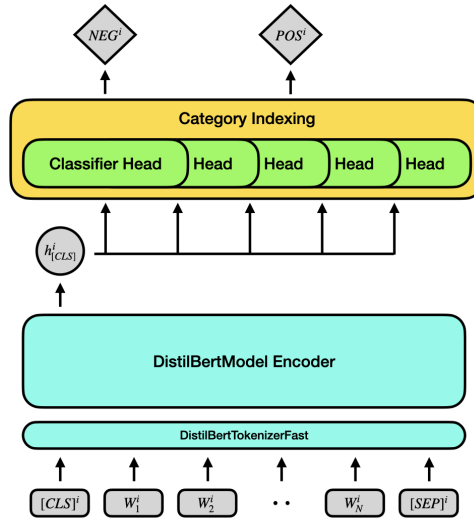


Figure 26: TASK D. Model Architecture.

References

- Athanasios Giannakopoulos, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2017. [Unsupervised aspect term extraction with b-lstm and crf using automatically labelled datasets](#).
- Zhiyong He, Zanbo Wang, Wei Wei, Shanshan Feng, Xianling Mao, and Sheng Jiang. 2020. [A survey on recent advances in sequence labeling from deep learning models](#).
- Dan Hendrycks and Kevin Gimpel. 2020. [Gaussian error linear units \(gelus\)](#).
- Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. [SemEval-2014 task 4: Aspect based sentiment analysis](#). In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland. Association for Computational Linguistics.
- Lance A. Ramshaw and Mitchell P. Marcus. 1995. [Text chunking using transformation-based learning](#). *CoRR*, cmp-lg/9505040.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).
- Youwei Song, Jiahai Wang, Zhiwei Liang, Zhiyue Liu, and Tao Jiang. 2020. [Utilizing bert intermediate layers for aspect based sentiment analysis and natural language inference](#).
- Zhen Wu, Chengcan Ying, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2020. [Transformer-based multi-aspect modeling for multi-aspect multi-sentiment analysis](#).