# Natural Language Processing

## Aspect-Term Polarity Classification in Sentiment Analysis

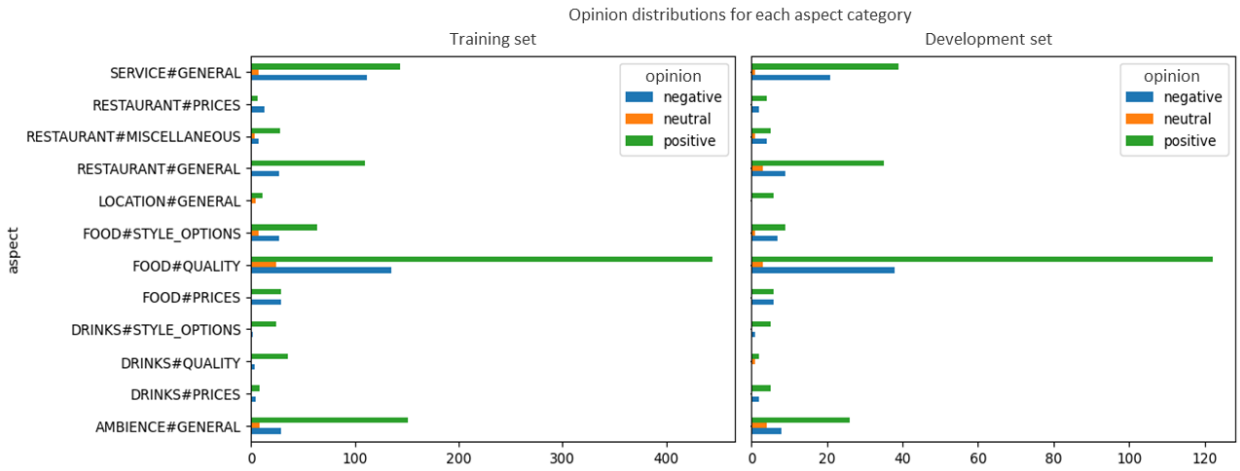
By: Ian Moon, Adel Remadi, Lasse Schmidt

Within: MS Data Sciences & Business Analytics

At: CentraleSupélec & ESSEC Business School

## Introduction

The aim of this project has been to implement a classifier that predicts opinion polarities (positive, negative or neutral) for a given set of aspect categories related to a sample of customer reviews. The dataset of interest was composed of a training set and a development set, which both presented class imbalance among the 12 aspect categories to be tackled, as illustrated below.




## 1. Implementation Details

The input data was composed of three elements: the aspect category, the target term and the sentence. These elements were preprocessed into enriched concatenated strings containing separators and marker tags in order to structure the model's input and highlight the target term's position in the sentence. Several enrichments were considered in the tuning phase and the eventual final format was of the form:

'<start> aspect <marker> target </marker>? <sep> sentence <sep> <marker> target </marker> </end>'

Below is a concrete example from the training dataset:

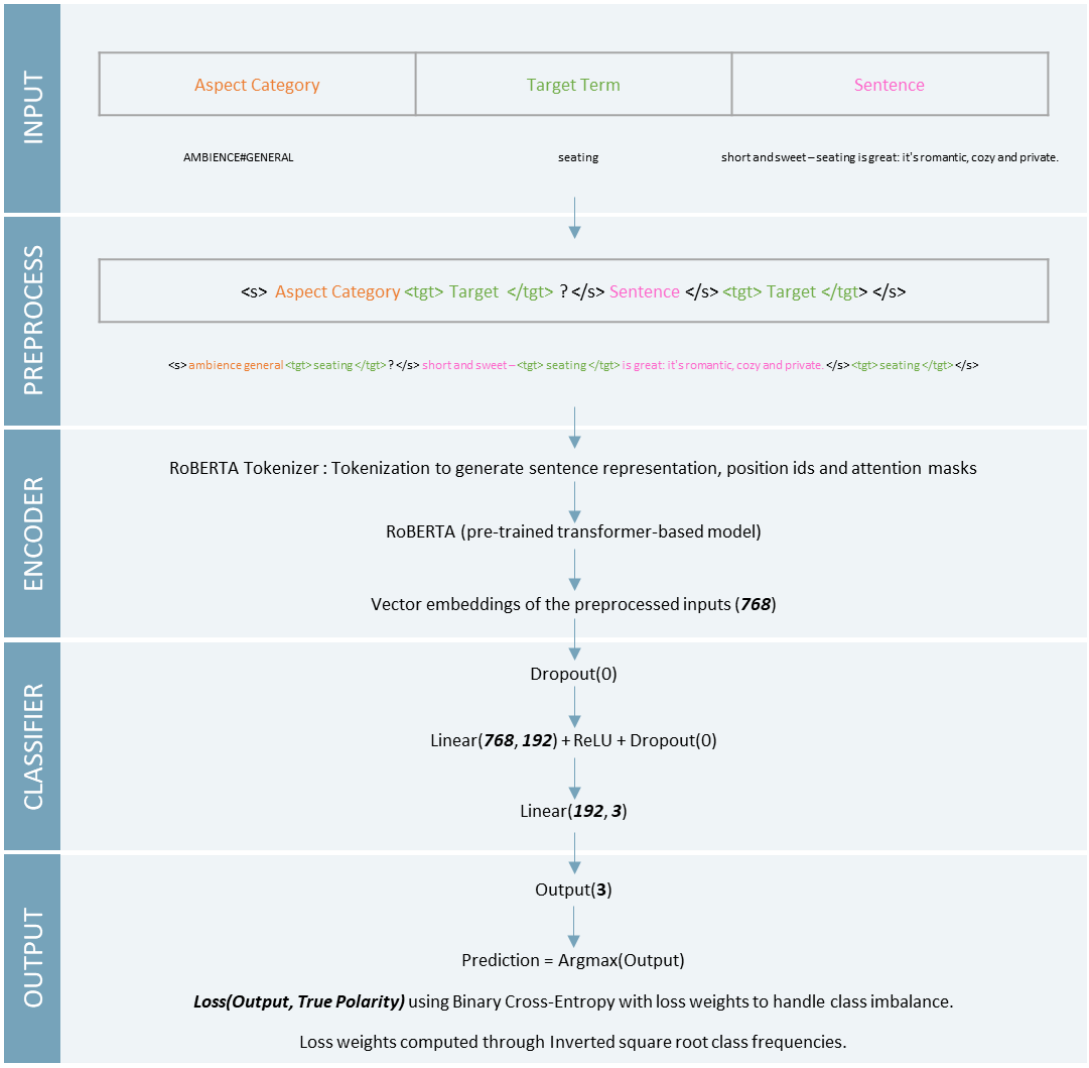
<s> ambience general <tgt> seating </tgt> ? </s> short and sweet – <tgt> seating </tgt> is great: it's romantic, cozy and private. </s> <tgt> seating </tgt> </s>

The final model selected was composed of the association of a model named RoBERTa with a classifier network. BERT (Bidirectional Encoder Representations from Transformers) is an encoder-only transformer-based pre-trained language model. RoBERTa is an optimized version of BERT and follows a similar input structure (with small differences in tokenization and special tokens).

The final architecture as well as all the hyperparameters of the classifier were determined using a series of grid-searches. Eventually, among the tested configurations, the optimal architecture for the classifier occurred to be composed of 2 linear layers, 192 hidden units and ReLU as the activation function of the first layer. Dropouts were also considered for tuning at two locations: on the input of the classifier and just after the activation of the first layer. However, the optimal architecture was obtained with no dropout.

Another critical aspect of the pipeline was the choice of the loss function. Similarly to the input enrichment or the model's architecture, several loss functions were considered for tuning. Eventually, the Binary Cross Entropy was selected. In order to tackle the class imbalance mentioned in the introduction, loss weights were computed, and the optimal results were obtained with inverted square root class frequency.

The following diagram illustrates the overall pipeline that was implemented:

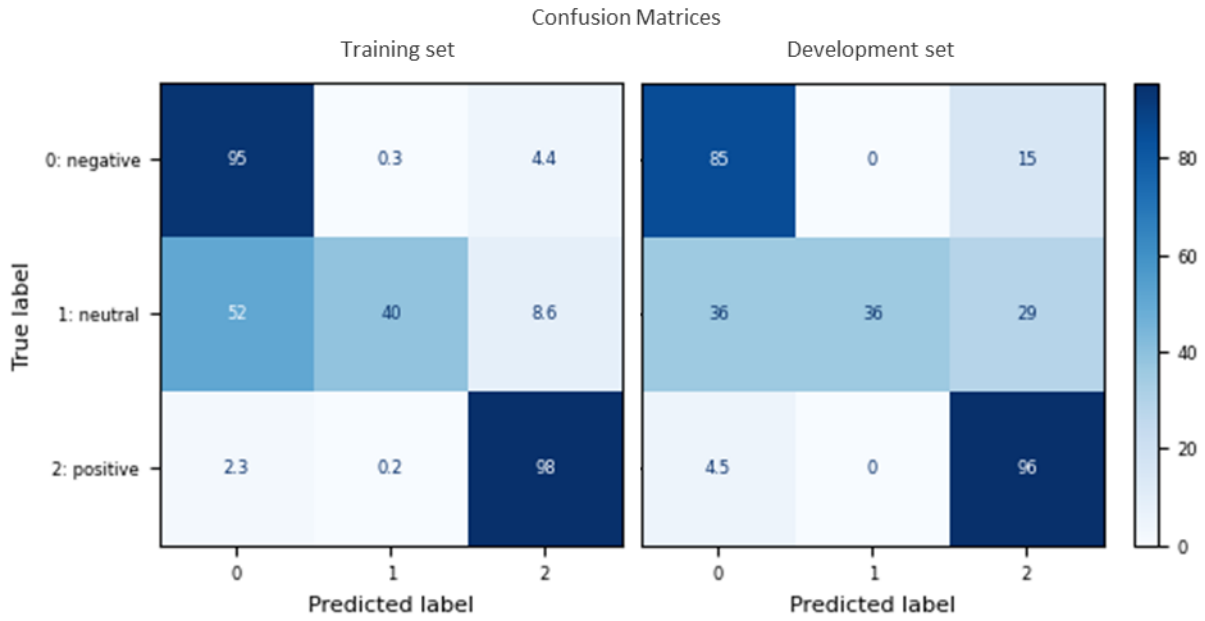


## 2. Main Results

The selected model described above has obtained the following performance on the Dev Set (5 runs):

| Pre-trained Model | Machine | Number of epochs | Average Dev Accuracy (5 runs) | Standard Deviation | Total Runtime (5 runs) |
|---|---|---|---|---|---|
| Roberta-base | Local Machine (RTX 3070 8GB) | 20 | 90.32 | 0.57 | **20 min** (1176 s) |

The following confusion matrices indicate that most misclassifications occur for the neutral polarity, which is the most underrepresented class. In addition, it can be noted that the proportion of negative comments misclassified as positives is higher on the development set. Other than these two aspects, the performance of the predictions on the train and dev set show similar patterns, which seemed to indicate a good potential for generalization.




Several variants of the model presented above were tested and achieved very good results, as illustrated in the table below. Some configurations even outperformed the submitted one, but were eventually discarded to ensure a better tradeoff between average accuracy and runtime. As an example, using Roberta-large as a pre-trained model, on 20 epochs seemed very promising and actually obtained the highest performance of **92.39** with a really low standard deviation, but took 2h30 to complete the 5 runs on Google Colaboratory.

| Pre-trained Model | Machine | Number of epochs | Average Dev Accuracy (5 runs) | Standard Deviation | Total Runtime (5 runs) |
|---|---|---|---|---|---|
| Roberta-base | Local Machine (RTX 3070 8GB) | 10 | 88.56 | 0.97 | 11 min (650 s) |
| Roberta-large | Colab (uses 11.6 GPU RAM) | 10 | **92.02** | 0.81 | ~ 1h15 (4451.22 s) |
| Roberta-large | Colab (uses 11.6 GPU RAM) | 20 | **92.39** | **0.27** | ~ 2h30 (9260.80 s) |

## 3. Hyperparameter tuning

As mentioned previously, the obtained results were achieved through a series of grid-searches that covered a large number of hyperparameters stored in the 'config' dictionary and illustrated in the below code snippet:

```
36          self.config = {
37              # basic infos
38              "verbose": False,
39              "max_epochs": 20,
40              "batch_size": 32,
41
42              # data preprocessing
43              "input_enrichment": "short_question_sentence_target",
44
45              # pre-trained language model (transformer)
46              "plm_name": "roberta-base",
47              "plm_freeze": False,
48
49              # classifier (linear layers)
50              "cls_depth":          2,
51              "cls_width":          192,
52              "cls_activation":     "ReLU",
53              "cls_dropout_st":     0, # maybe try out 0.1 sometime
54              "cls_dropout_hidden": 0,
55
56              # optimizer
57              "lr": 5e-6,
58              "wd": 15e-3,
59
60              # scheduler
61              "lr_s": "linear",
62              "warmup": 0,
63
64              # loss function
65              "crit": "BCE",
66              "crit_w": "invSqrtClassFreq",
67          }
```

Input Enrichment refers to the preprocessing step described section 1. Several enrichments were considered and their performances were tracked and compared to identify the most appropriate one. The other options involved the use of fully worded questions, such as 'What is the ambience like regarding the *$target_term$* ?'.

The pre-trained models that were considered for tuning are ‘bert-based-cased’, ‘roberta-base’ and ‘roberta-large’. The whole pipeline, from the preprocessing step to the prediction was designed so that it could handle any choice of model among those three. The specificities of each models were therefore taken into account at each step, which facilitated strongly the tuning process.

As mentioned in section 1, the classifier’s architecture was determined through several grid-searches on the following hyperparameters: Depth, Width, Activation function, Dropout on inputs, and Dropout after first layer. Additionally, the choice of the optimizer, its learning rate and of a scheduler was also the result of the tuning process.

Three loss functions were experimented, namely Binary Cross-Entropy, Focal Loss and Top-k loss. The former showed the most promising results and was then selected in the final model. The class imbalance of the dataset was addressed through the computation of loss weights aiming at steering the model to classify correctly the underrepresented classes. Three options were considered in the tuning phase: unweighted (standard), inverted class frequency and inverted square root class frequency. The former being the case where all misclassifications would have the same impact on the loss, while the two other instead aimed at increasing the impact of misclassifying underrepresented classes like ‘Neutral’ in order to compensate for their smaller counts in the dataset. It eventually turned out that inverted square root class frequency was the most promising option.