Target-aspect-sentiment joint detection for SE-ABSA15 competition

Simone Rizzo, Lombardi Giuseppe
Master Degree in Computer Science.
s.rizzo14@studenti.unipi.it, g.lombardi11@studenti.unipi.it.
Human Language Technologies, Academic Year: 2021/2022

Abstract

Aspect-based sentiment analysis (ABSA) aims to detect the targets, aspects, and sentiment polarities in text. A published dataset from SemEval-2015 reveals that a sentiment polarity depends on both the target and the aspect. However, most existing methods consider predicting sentiment polarities from either targets or aspects but not from both. Thus they easily make wrong predictions on sentiment polarities. In particular, where the target is implicit, i.e., it does not appear in the given text, the methods predicting sentiment polarities from targets do not work. We propose a method for target-aspect-sentiment joint detection to tackle these limitations in ABSA. It relies on a pre-trained language model and can capture the dependence on both targets and aspects for sentiment prediction. SemEval-2015 restaurant datasets show that the proposed method achieves high-performance detecting target-aspect-sentiment triples even for the implicit target cases.

Contents

1 Introduction

Sentiment analysis aims to detect and classify the polarity of sentiments expressed in a text. The granularity of this classification goes from the overall polarity of complete documents to paragraphs, sentences or, as in Aspect Based Sentiment Analysis (ABSA), the sentiment about specific aspects being opinionated. The ABSA, mining opinions from a text about specific entities and their aspects, can help consumers decide what to purchase and businesses to monitor their reputation better and understand the needs of the market. This paper proposes a solution model for the SemEval-2015 Task 12 (SE-ABSA15).

Teams participating in the competition SE-ABSA15 propose models that individually predict the target, aspect, or polarity of a sentence. Only a few propose a model that jointly predicts the target and the aspect of a sentence [14]. We aim to build a model that detects the three elements (target, aspect, and sentiment) simultaneously, as proposed by [22]. The primary challenge is that the sentiment depends on both the target and the aspect. Most existing studies in ABSA do not handle this dual dependence. For example, some studies such as [24] and [25] predict sentiments from aspects alone, while other studies such as [18] and [19] predict sentiments from targets alone. Moreover, most existing studies such as [12], [23], [11] ignore implicate target cases. However, from the restaurant datasets, there are about one fourth of opinions that have implicit targets (see figure 7). Thus handling implicit target cases is also a non-neglectable challenge in ABSA. To detect a set of opinions given by three elements (target, aspect and sentiment), we develop a neural-based method for target-aspect-sentiment joint detection. The method separates the joint detection problem into two sub-problems based on aspect-sentiment pairs, where for every given aspect-sentiment pair, one sub-problem determines whether targets exist and is reducible to a binary text classification problem, and the other extracts all targets and is reducible to a sequence labeling problem. A single neural model, built upon the pre-trained language model BERT [4]), solves both sub-problems. The neural model is trained by minimizing a combined loss function about two subproblems.

Several ABSA methods have been proposed for domains like consumer electronics, restaurants, and movies. We have narrowed it down to building a specific model for restaurants, using domain-specific knowledge and transformers to improve our results (in-domain ABSA).

2 Problem Definition

Given a sentence S consisting of n words $s_1, \ldots, s_n$, an *opinion* is a tuple (t, A, p) [22], where:

- t is the *target* of the opinion, i.e. a sub-sequence of S to which the opinion refers;
- $A = (e, a)$ is the *aspect* category in $\mathcal{A}$, e and a are respectively the *entity* and the *attribute* of the aspect;
- p is the opinion *polarity* in P.

Our task aim is to detect all opinion triples (t, A, p) that a sentence entails in the natural language meaning. The text (the single consumer review) is divided into sentences, and opinions are produced for each of them. A sentence may or may not contain opinions, and the number of opinions varies. Moreover, the opinion target t can be empty and denoted by "NULL".

2.1 Problem Reduction

We reduce the problem of ABSA to a set of text classification problems and sequence labeling problems [22]. To this end, we divide a given problem which detects all opinions from a sentence S, a set $\mathcal{A}$ of aspects and a set P of sentiments into $|\mathcal{A}| \cdot |P|$ problems on the basis of all aspect-sentiment pairs.

Every resulting problem is further separated into two sub-problems, where one determines whether targets exist for the given aspect-sentiment pair, and the other extracts the targets corresponding to the given aspect-sentiment pair. The first sub-problem can be reduced to a binary text classification problem, with "yes" indicating that at least one target (including the implicit target) exists and "no" indicating that no target exists. The second sub-problem can be reduced to a sequence labeling problem using the BIO tagging (see chapter 4.4).

The results of the two sub-problems can be merged to get opinions. Given a sentence S and an aspect-sentiment pair (A, p), if the first sub-problem outputs "no", there will be no opinion of the form (t, A, p) that can be detected from S. Otherwise if the first sub-problem output is "yes", there are $n \geq 0$ sub-sequences of S output by the second sub-problem. In the latter case, f $n = 0$, there is only one opinion (NULL, A, p) with implicit target detected from S, otherwise there are n opinions (t, A, p) detected from S for t a sub-sequence output by the second sub-problem.

3 Transformers

In chapter 1 we have discussed Transformers and the pre-trained language model. The paper "Attention Is All You Need" [5] introduces a novel architecture called Transformer. A Transformer is an architecture for transforming one sequence into another with the help of two parts (Encoder and Decoder). However, it differs from the traditional sequence-to-sequence models. It does not imply Recurrent Networks (GRU, LSTM, etc.) because it exploits the attention mechanism. The attention mechanism looks at an input sequence and decides at each step which other parts of the sequence are essential. This mechanism simulates the human behavior in which, when reading the text, the reader always focuses on the word he reads. However, at the same time, the reader's mind still holds the essential keywords of the text in memory to provide context.

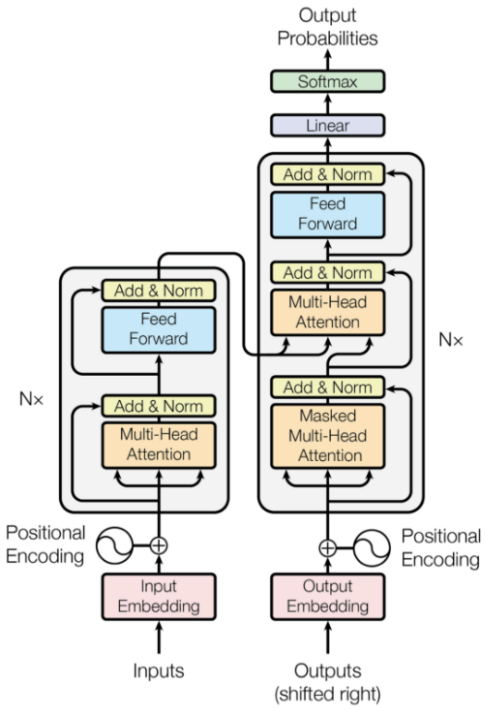

Figure 1: The Transformer - model architecture

The Encoder is on the left, and the Decoder is on the right. Both Encoder and Decoder are composed of modules stacked on top of each other multiple times, as described by Nx in the Fig. 1. The modules consist mainly of Multi-Head Attention and Feed Forward layers. The inputs and outputs (target sentences) are first embedded into an n-dimensional space since we cannot use strings directly.

One minor but essential part of the model is the positional encoding of the different words. Since we have no recurrent networks that can remember how sequences are fed into a model, we need somehow to give every word/part in our sequence a relative position since a sequence depends on the order of its elements. These positions are added to each word's embedded representation (n-dimensional vector).

Transformers are increasingly the model of choice for NLP problems, replacing RNN models such as long short-term memory (LSTM). Unlike RNNs, transformers process the entire input simultaneously thanks to the attention mechanism providing context for any position in the input sequence, allowing training parallelization on larger datasets. This characteristic led to the development of pre-trained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), which were trained with large language datasets, such as the Wikipedia Corpus and Common Crawl. These models can be fine-tuned for specific tasks having a great success in natural language processing (NLP). Many pre-trained models such as GPT-2, GPT-3, BERT, XLNet, and RoBERTa demonstrate the ability of transformers to perform a wide variety of such NLP-related tasks and have the potential to find real-world applications.

4 Dataset

SE-ABSA15 contains the exact domains as SE-ABSA14 restaurants and laptops. However, unlike SE-ABSA14, the input datasets of SE-ABSA15 will contain entire reviews, not isolated sentences. SE-ABSA15 consisted of two subtasks in which participants were free to choose the subtasks, slots, and domains they wished to participate in:

- Subtask1 in domain ABSA: given a review text about a laptop or a restaurant, identify all the opinion tuples Aspect Category, Opinion Target expression, and polarity.
- Subtask2 out of domain ABSA: In this subtask, the systems can be tested in a previously unseen domain, the hotel reviews for which no training data was made available.

We chose for our experiments the Subtask1 on the Restaurant dataset provided by the competition.

Restaurant Dataset	Training Data	Test Data
Review text	254	96
Sentences	1315	685

Table 1: Restaurant dataset provided for ABSA.

As we can see in the Tab.1 the dataset is composed of the training and test data. The dataset is composed of reviews that can contain one or more sentences, and for each sentence, we can have multiple opinions. The average number of sentences per review is 7, while the maximum is 17 and the minimum is 2.

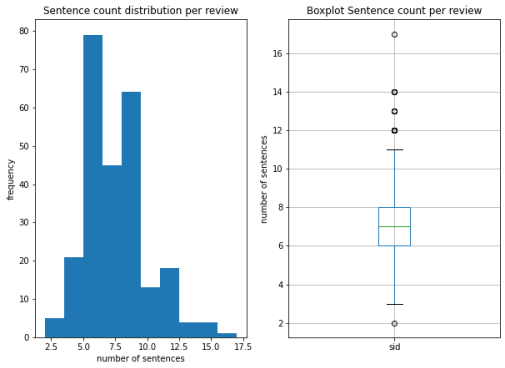

Figure 2: Sentences count distribution per review

As we saw in section 2, an opinion is a tuple of three elements. More specifically, for the "Restaurants" dataset:

- Aspect Category (Entity and Attribute): Identify the entity E and attribute A pair E#A towards which an opinion is expressed. An entity E is an element of the following set (FOOD, DRINKS, SERVICE, AMBIENCE, LOCATION, RESTAURANT) while

an attribute can be an element of the following set (GENERAL, PRICES, QUALITY, STILE&OPTIONS, MISCELLANEOUS)

- Opinion Target Expression (OTE): An opinion target expression (OTE) is an explicit reference (mention) to an entity E that is evaluated. This mention can be a named entity, a common noun or a multi-word term. The OTE is defined by its starting and ending offsets. When there is no explicit mention of the entity, the slot takes the value "NULL".
- Opinion Polarity: Each identified E#A pair has to be assigned one of the following polarity labels: positive, negative, neutral.

Two examples of opinion tuples values from the restaurants domain are shown below.

(a) The food was delicious but do not come here on an empty stomach. →
{category= "FOOD#QUALITY", target= "food", from: "4", to: "8", polarity= "positive"},
{category="FOOD#STYLE_OPTIONS", target="food", from: "4", to: "8", polarity= "negative"}

(b) Prices are in line →
{category: "RESTAURANT#PRICES", target= "NULL", from: "-", to: "-", polarity: "neutral"}

4.1 Datasets Format and Availability

The datasets of the SE-ABSA15 task were provided in an XML format. They are available under a non-commercial, no redistribution license through META-SHARE, a repository devoted to the sharing and disseminating of language resources [13].

4.2 Analysis

In this section, we analyze the dataset in order to study the distribution and the quality of the dataset. We can start by studying the aspect category distribution. As we can see in the Fig. 3 we have an unbalanced dataset in which the aspect category FOOD#QUALITY represents the 35% of the data in the training set and the 32% of the data in the test set. Instead, the classes with fewer supports are LOCATION#GENERAL and DRINK#PRICE, with around 1% of the dataset.

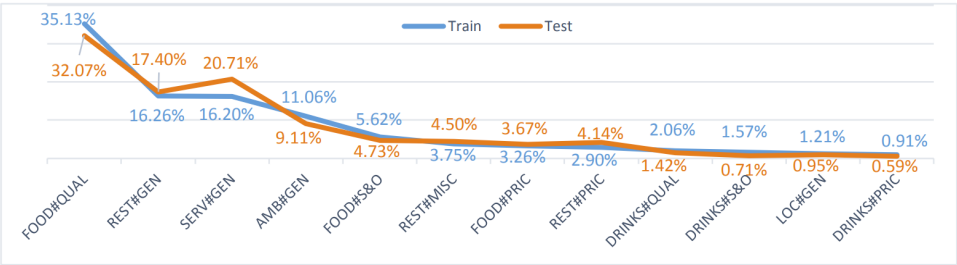

Figure 3: Aspect category distribution.

Let us now see the polarity distribution of each aspect category couple to have an idea of the distribution of the polarity for each category. As we can see in the Fig. 4 the polarity class positive has more occurrences than the others in all the categories. There are also aspect category couples with no neutral or negative samples. The global distribution of the polarity for all the aspect category couples is reported in the Tab. 2.

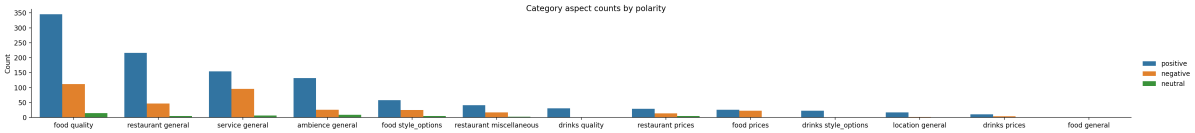

Figure 4: Polarity distribution of opinions.

We have an unbalanced dataset with positive opinions of more than 70%, while the neutral has only 3% (table 2). The unbalance dataset may make the model more likely to provide a positive polarity than a neutral one because, in the dataset, it saw more positive reviews than neutrals.

	Positive	Negative	Neutral
Training set	72,43%	24,36%	3,20%
Test set	53,72%	40,96%	5,32%

Table 2: Polarity distribution of the dataset.

From our 1315 distinct sentences, we analyze how many of them have opinions and how many of them do not have an opinion. The 85.2% of them express opinions while the remaining 14.8% does not contain an opinion (figure 5).

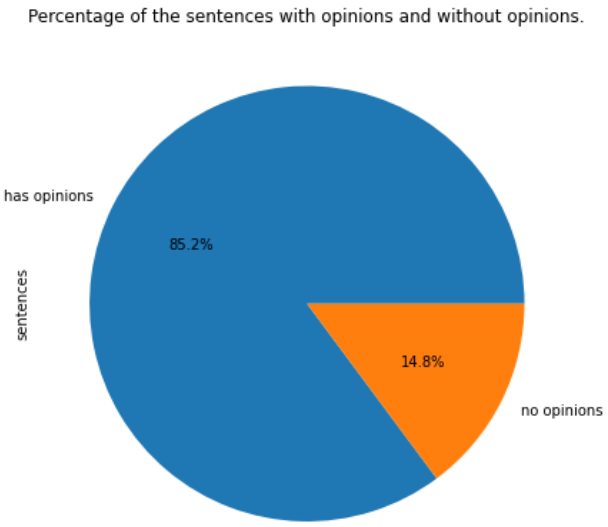

Figure 5: Percentage of sentences with opinions.

From these sentences with opinions, we are now looking for the distribution of the opinions count. In order to understand each sentence, we analyze how many opinions contain. As we can see from the Fig. 6 we have a maximum of 8 opinions in a sentence, a minimum of 1 opinion, and an average of 1 sentence. So on average, each sentence contains one opinion with some outliers that can reach 8 opinions.

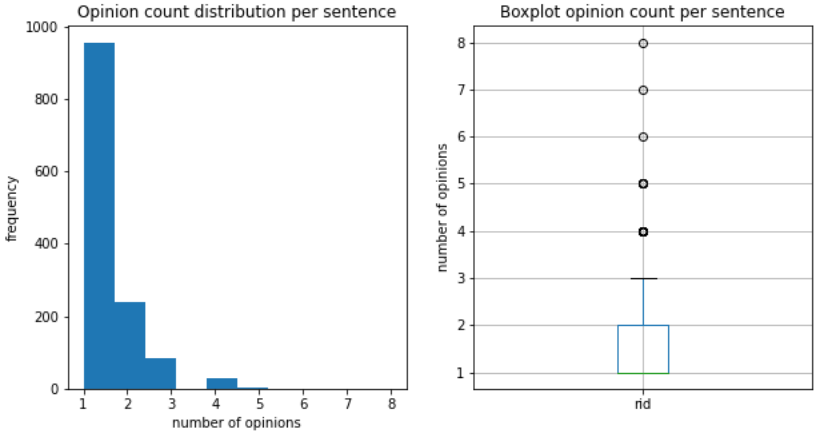

Figure 6: Opinion count distribution per sentence

From the current opinions, which are 1654, we analyzed how many of them contain a target and how many express an implicit reference (NULL target). From the figure 7 we can see that the 77.3% of the opinions contain a target while the remaining 22.7% express implicit opinions, so without a target.

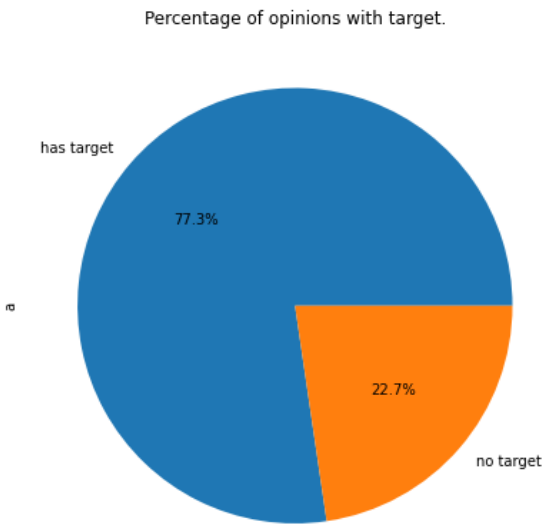

Figure 7: Percentage of opinions with target.

In order to use our data with an encoder, we also analyze the sentence's tokens length distribution to choose the best input size for our model. Fig. 8 shows the distribution of tokens for each sentence using the BERT tokenizer. We can see from the distribution that the minimum number of tokens is 1, the average is 16.23, and the maximum is 90 tokens.

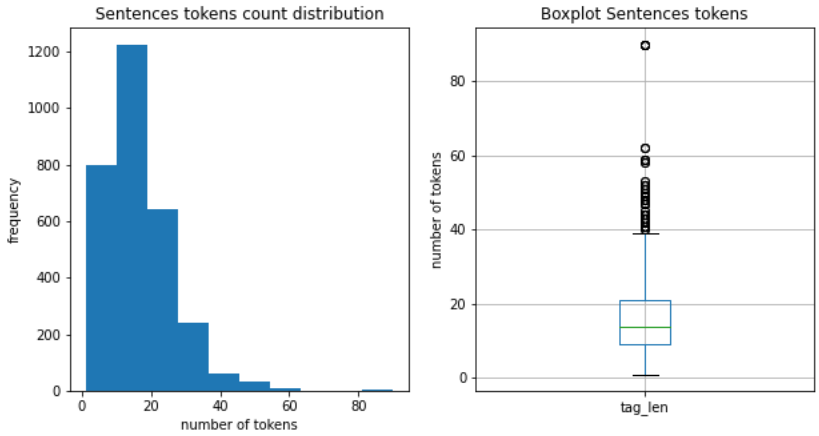

Figure 8: Sentence's tokens length distribution.

Thanks to this preliminary analysis, we can now know which distributions of our data and how big we should create the input size of our model. Furthermore, we can fix the unbalancing of our dataset by using balancing techniques.

4.3 Data Augmentation

In this section, we will see how we modified the dataset to align with the problem reduction expressed in the chapter above.

We carry out a data augmentation of our dataset by adding inputs for each sentence that does not have a specific category (the respective value of value f is "no"). In this way, we first calculated all the possible (entity#attribute, polarity) pairs present in our dataset resulting in a list of $|\mathcal{A}| \cdot |P| = 39$ possible pairs.

Subsequently, grouping by `sid` (sentence id), we add all the pairs (entity#attribute, polarity) that are not present with that sentence by setting the value of f no to indicate that the sentence does not have that (entity#attribute, polarity) pair. In this way, our dataset went from 1654 rows to 51434. In this way, we will have a resulting dataset composed of the following columns:

- txt: represents the sentence in textual format or string. For instance: The waiter was attentive.
- categ_txt: it represents the string composed of: entity + " " + attribute + " " + polarity. For example we have: service general positive.
- f: this column can be yes or no. Given the sentence (txt) and the categ_txt (entity + " " + attribute + " " + polarity) f represents whether the sentence expresses an opinion of categ_txt with the same polarity.

In this way, we have prepared our dataset to classify the sentence according to the aspect and polarity category. Below we are now going to extend our dataset by adding the tagging.

4.4 Tagging

Regarding tagging, we filtered our previous dataset to take only the sentences that express an opinion and that contain a target. By exploiting the from and to properties contained in the original dataset, we know precisely from which index to which index our sentence contains the indicated target. In this way, using the chosen tokenizer, we have tokenized the whole sentence, and using our algorithm, we create the list of tags for each token. As for tagging, we used the "BIOP" tagging where:

- B: indicates the start of a target.
- I: indicates the continuum of a target word.
- O: indicates a token with no target.
- P: indicates Padding or no token.

For example with the sentence txt:
"The duck confit is always amazing and the foie gras terrine with figs was out of this world.", we have:

- f: "yes";
- categ_txt: "food quality positive";
- from: 42 and to: 69;
- tagging: [O,O,O,O,O,O,O,O,O,O,O,B,B,B,I,I,I,I,I,I,I,I,O,O,O,O,O,O,O], which highlights the words: "foie gras terrine with figs".

4.5 Dataset Balancing

An essential problem in our augmented dataset concerns the imbalance for the feature f that is with 1509 "yes" and 49925 "no". To overcome this problem, we have downsampled the data with f equal to no to balance the dataset. To do this, we have divided the dataset by the attribute f, having one dataset with $f =$ "yes" and the other with $f =$ "no". Then, using the distribution in Fig. 4 we select several rows in dataset $f =$ "no" to balance our dataset on the feature f and, simultaneously, maintain the distribution of the (aspect, polarity) pair. Doing so will obtain a perfectly balanced dataset on feature f containing 3008 rows.

5 Model

In this section, we will describe the model structure and architecture to address the TASD task for the SEMEvalCompetition. We have chosen to use several popular pre-trained language model encoders to solve this problem, such as BERTbase, DistilBERT, and Roberta. Encoders are a great advantage as they allow us to compress the text into a latent space vector, valid for several tasks. The tab. 3 compares the architectures of the encoders used.

Transformers			
Model	**Encoder**	**Heads**	**Latent dim**
BERTBase	12 layers	12	768
DistilBERT	6 layers	12	768
RoBERTa	12 layers	12	768

Table 3: Architectural comparison of the ecoder used for TABSA.

BERT (Bidirectional Encoder Representations from Transformers) is a multi-layer bidirectional Transformer encoder introduced by Devlin et al. [4] and based on the original transformer implementation described in Vaswani et al. [21]. BERT is conceptually simple and empirically powerful, so we decided to use it in our project. However, the main problem is that the dimension is huge even in its base form (it contains twelve layers, as many as standard transformers), so it is not the most cost-efficient model for our task. For this reason, we also opted to use its distilled version, DistilBERT.

DistilBERT is a small, fast, cheap, and light Transformer model trained by distilling BERT. Model distillation is a compression technique in which a compact model is trained to reproduce the behavior of a larger model or an ensemble model. DistilBERT has 40% fewer parameters than BERT and runs 60% faster while preserving over 95% of the BERT's performances as measured on the GLUE language understanding benchmark. For those reasons, we used this model for our research work.

RoBERTa is a transformer encoder with the same architecture as BERT and developed by Facebook AI. As written by Liu et al.[10], the team found that BERT was significantly undertrained and could match or exceed the performance of every model published after it. They have just modified BERT key hyperparameters, removing the next-sentence pre-training objective and training with much larger mini-batches and learning rates. In 2019, when Facebook AI wrote the article, their best model achieved state-of-the-art results on GLUE, RACE, and SQuAD.

As we described in chapter 4, we have divided our problem into two tasks:

- Task1: text classification (yes/no);
- Task2: sequence labelling.

As we can see from the schema 9, we have developed a general architecture of the model which can be of two types:

- **Disjointed architecture**: In this architecture, we have the tokenized text as input. It passes within two different models: EncoderTask1 and a series of fully connected layers for task1 (that perform the task of text classification) and encoder2 with another series of fully connected layers (that are used for the sequence labeling task).

- **Jointed architecture**: Unlike the disjointed, here we have only one encoder for both tasks and two groups of fully connected layers disjointed for the two tasks. In this way, the central part of the encoder will be trained to minimize the loss functions on both tasks, resulting in a multi-output and multi-loss model.

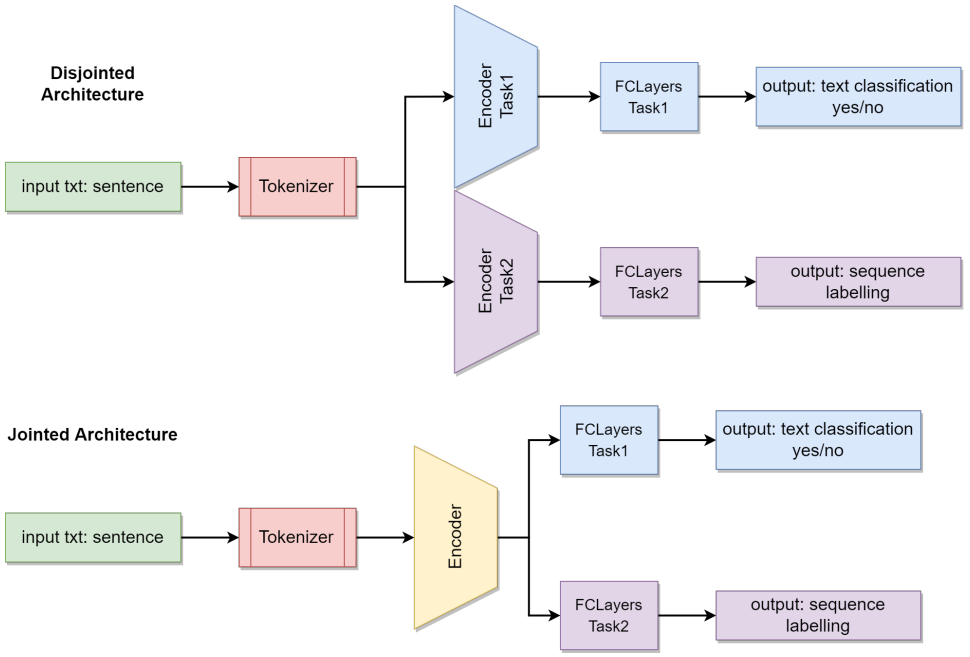

Figure 9: General architecture of the model: Disjointed or Jointed

5.1 Setup

Hardware Models were trained in `Google Colab` using `Google TPU v2-8`; it has 64 GB HBM memory and computes 180 teraflops. During our work, we noticed that TPUs outperform NVidia GPUs available in Colab by running ten times faster and with better power efficiency.

Software In order to train our model, we used `Tensorflow` with `Keras Chollet` [2] and the `huggingface.co` python library to retrieve pre-trained models that acted as encoders.

5.2 Tokenization and Input size

The dataset is composed of sentences written by users in natural language. Therefore, before training or inferring, we decide to tokenize sentences. The Huggingface library provides a Tokenizer class that can retrieve a pre-trained Tokenizer from a wide database. The chosen Tokenizer outputs a dictionary composed as follows:

- **input_ids**: a list of indices corresponds to the token positions in the vocabulary, e.g., "I love the pizza here!" $\rightarrow$ [101, 1045, 2293, 1996, 10733, 2182, 999, 102], where 101 and 102

are indices corresponding to the start and the end of the sentence. Padding starts from the right and is labeled as 0.

- **attention mask**: is a mask that allows the model to avoid focusing on padding tokens. If the token is not padding, the value will be 1; 0 otherwise. For example, if we set the maximum length of the input to 10, the attention mask for the sentence "I love the pizza here!" $\rightarrow$ [1, 1, 1, 1, 1, 1, 1, 1, 0, 0].

- **token_type_ids**: indicates the first and second portion of the input. It is mainly used in the *Question-Answering task*, whereas in input, we have two sentences written in natural language. By doing this, we make BERT understand the beginning and end of each sentence. The value 0 indicates the first sentence, and the value 1 indicates the second sentence. At most, the model can have two sentences in input.

The basic Tokenizer, which tokenizes only one sentence, is insufficient for our purpose. For this reason, we discard it. We need a Tokenizer that can tokenize two sentences together (e.g., "The pizza is really good" + "food quality positive"), like for the Question-Answering task. We first define some variables to indicate the size of the sentence and the size of the triple (entity, attribute, polarity).

- max_len: length of the number of tokens that the model takes as input. It is composed of: Sentece + PAD + Query + PAD where by query we mean the triple (entity, attribute, polarity) for example "food price negative" represented as a string (figure 10).

- query_len: represents the maximum length of the query (figure 10).

- sent_len: represents the number of maximum tokens for the sentence. If the limit length is exceeded, a truncation is performed (figure 10).

Therefore, we have adapted the tokenizer by implementing the *tokenize_couple* code function 5.2 to create the dictionary containing the sentence and the query with the padding and according to the rules defined above.

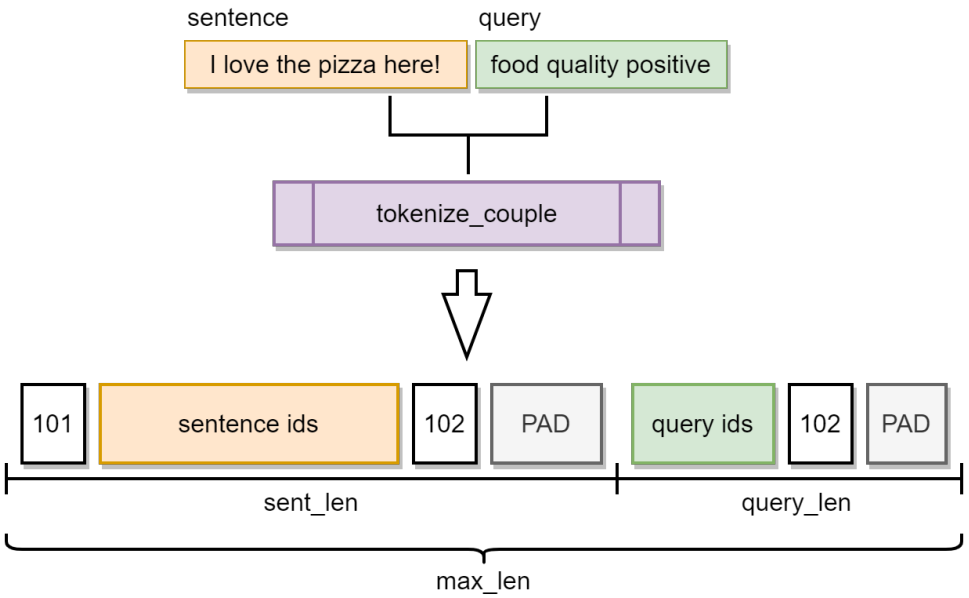

Figure 10: The custom tokenization with sentece and query.

Thanks to this tokenization technique, the model will generalize the task by learning to understand when the query is expressed within the sentence (in our case, the query is the opinion given by the triple (polarity, attribute, category)). The model result will be whether or not this opinion is expressed in the sentence. Thanks to this, the model can also work out of domain on queries of any type. The code 5.2 is an example of our Tokenizer and how "input_ids", "token_type_ids", and "attention_mask" provide all the information the model needs to distinguish the two-sentence and padding.

Listing 1: Code to tokenize a couple of sentences togheter

```
def tokenize_couple(ex):
  dct=slow_tokenizer(ex[0], max_length=sent_len, padding='max_length')
  dct2=slow_tokenizer(ex[1]+"[SEP]",add_special_tokens=False,
  max_length=query_len, padding='max_length')
  dct2['token_type_ids']=[1 for i in range(query_len)]
  dct['input_ids'].extend(dct2['input_ids'])
  dct['token_type_ids'].extend(dct2['token_type_ids'])
  dct['attention_mask'].extend(dct2['attention_mask'])
  return dct
```

Example

couple= ("I love the pizza here!","food quality positive")
tokenize_couple(couple) →
{
'input_ids': [101, 1045, 2293, 1996, 10733, 2182, 999, 102, 0, 0,2833, 3737, 3893, 102, 0, 0],
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
'attention_mask':[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0]
}

5.3 Model architecture and Training

Regarding the model architecture and training, we are now going to describe the final architecture used for the disjointed and jointed models. However, first, we start with the loss function. We used for both tasks the `SparseCategoricalCrossentropy` loss, provided by `Keras`. This loss is equal to the traditional `CategoricalCrossentropy` loss, but it does not need the one-hot-encode target.

The categorical cross-entropy is the default loss function for multi-class classification problems, where each class is assigned a unique integer value from 0 to (num_classes - 1). It will compute the average difference between all output classes' actual and predicted probability distributions. The categorical cross-entropy formula is described by the equation 1,where K is the number of classes, $\hat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value. The score is minimized, and an excellent cross-entropy value is 0.

$$Loss = -\sum_{i=1}^{K} y_i * log(\hat{y}_i). \tag{1}$$

This loss is a good measure of distinguishable two discrete probability distributions from each other. In this context, y_i is the probability that event i occurs, and the sum of all y_i is 1, meaning that precisely one event may occur. The minus sign ensures the loss gets smaller when the distributions get closer. The target must be one-hot encoded to be appropriate to the categorical cross-entropy loss function. Sparse categorical cross-entropy addresses this by performing the same cross-entropy calculation of error without requiring that the target variable be one-hot encoded prior to training. Regarding the loss for sequence tagging, a loss custom code function has been created (code 2), which uses the sparse categorical cross-entropy loss with a mask capable of not counting those tags that represent the padding ("P"→ integer value 3) to avoid learning to assign the padding tokens.

Listing 2: Masked loss for sequence tagging

```
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=False, reduction=tf.keras.losses.Reduction.NONE
)

def masked_ce_loss(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 3))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_mean(loss_)
```

Regarding the optimizer, we chose to use Adam with Epsilon equal to 1e-8 and clipping norm to 1 to avoid the gradient explosion. To train these models, we set the max sequence length to 266 with 256 for the sentence and 10 for the query length, the learning rate to 2e-5, and the maximum number of epochs to 10.

Disjointed Model architecture For the disjointed model, we trained the two models separately. First, we carried out a grid search to find the best hyperparameters for the activation

function, the number of neurons of the fully connected layer, and the dropout layer. This grid search (table 4) was performed for both models for the two tasks. For both tasks, the best activation function turned out to be tanh(·). The number of nodes for task 1 is 32, with a dropout at 0.1. For task 2, the number of nodes is 128, with a dropout of 0.2.

GridSearch	
Activation functions	[relu, sigmoid, tanh]
Number of neurons	[16, 32, 64, 128]
Dropout	[0.1, 0.2, 0.3]

Table 4: Hyperparameters grid search

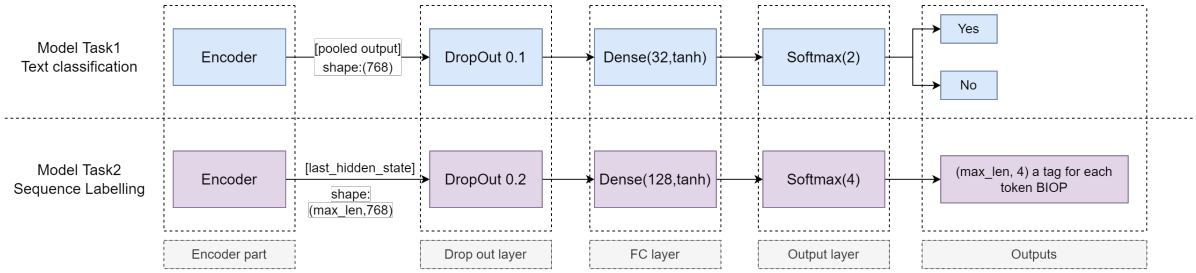

Figure 11: Disjointed model architecture

Jointed Model architecture Based on the jointed model architecture (figure 11), we use the embedding of the first token [CLS] to classify the sentence using a dropout layer of 0.2 and a fully connected layer of 32 with a function of activating Tanh and then ending up with a soft-max layer of two nodes. The first token was used to represent the entire sentence's meaning. While for sequence tagging, we use the embedding of tokens ranging from 1 to "sent_len-1". By doing so, we will always be sure to take the sentence and calculate the tags for any size it has; the loss function will exclude the padding tokens using the mask. We then go through a dropout of 0.2 and then in a fully connected layer Tanh of 64 nodes, and we end up in a soft-max layer of 4 nodes (figures 12, 13).

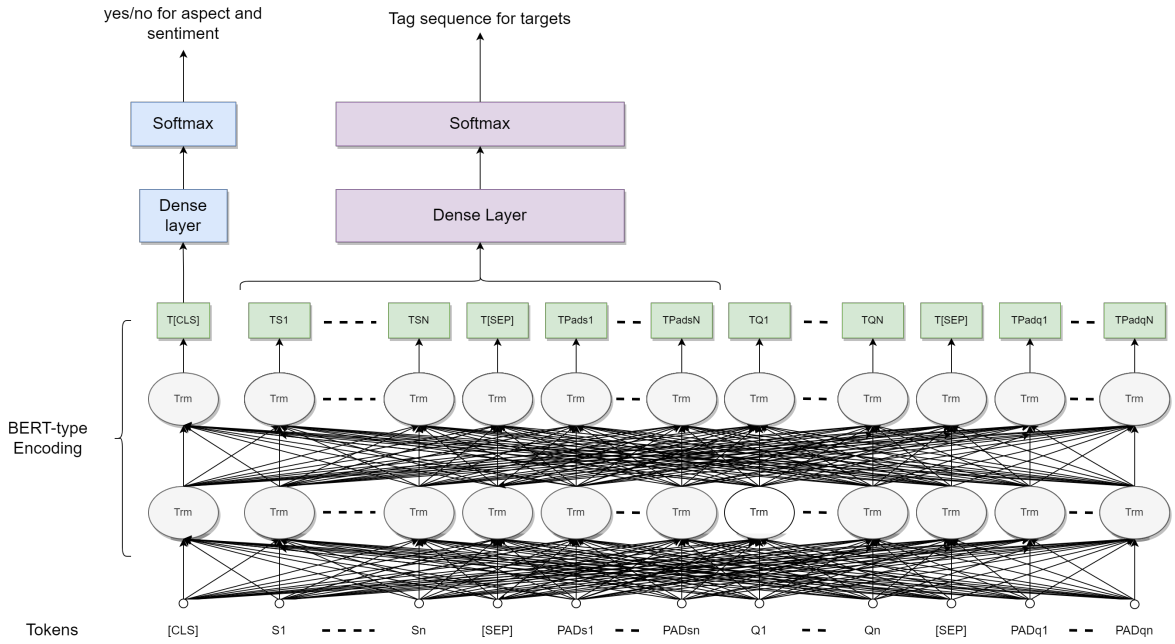

Figure 12: Jointed model architecture, highlighting the encoder and the input schema.

This model, unlike the disjointed one, this one provides the multi-output and implements two different losses at the same time. On Tensorflow it is possible to assign weights to multiple losses to give more importance to specific losses. In our case, we have decided to weigh the two losses in the same way as we want to try to minimize them both equally.

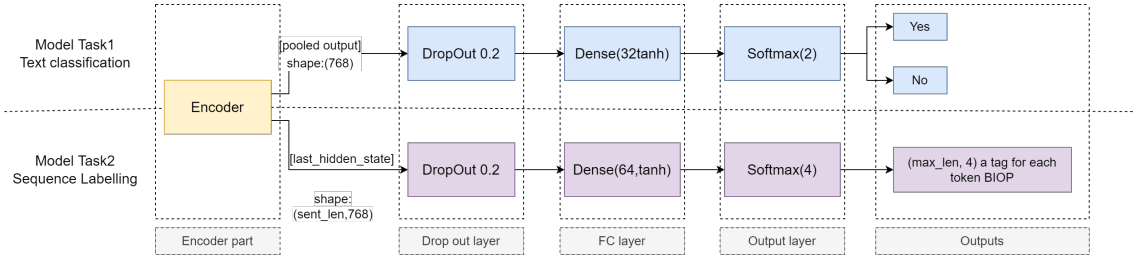

Figure 13: Jointed model architecture, highlighting the MLP architectures.

5.4 Workflow

Once we have trained our model and performed the model selection in this section, we will describe the global workflow for the task, considering the prepossessing and the post-processing. We can split the workflow into 4 steps:

- Step 1: Given a review, we split the text into different sentences using the python library `NLTK`.

- Step 2: For each sentence, create the tokenized couple with the sentence and all the possible combinations of aspect polarity as we saw in chapter 4.2.
- Step 3: We pass all these couples created into our models, and we get the outputs.
- Step 4: This step is also called the post-processing step. This step can be subdivided into different sub-steps:
  - Check the classification output: If the classification output is "false", we skip this output because no opinion is expressed in the sentence. Instead, if the output is "true", this means that the sentence contains the opinion explicitly or implicitly.
  - Check the sequence tagging output: If the opinion is presented, then we check the sequence tagging output. If the sequence contains all 'O', we have an implicit opinion, so we return the opinion with NULL as target. Otherwise, we look for all the different targets expressed through groups of tags starting with 'B'. This way, we know how many targets we have with the same (aspect, polarity) pair.
  - Given each target, we compute the "from" and "to" indices within the sentence.

5.5 Evaluation

Based on the problem definition described in the chapter 2, following the evaluating criteria of the SE-ABSA15 competition, each of the following three sub-tasks is evaluated:

- The first sub-task s_1 is the aspect category detection. The goal is to identify every entity e and attribute a pair, chosen from a predefined set, towards which an opinion is expressed in the text.
- The second sub-task s_2 is the OTE extraction. The goal is to extract the linguistic expression used in the text to refer to each reviewed aspect pair (e, a).
- The third sub-task s_3 is the sentiment polarity detection. Each identified (e, a) pair has to be assigned one of the following polarity labels: positive, negative, neutral (mildly positive or mildly negative sentiment).
- The jointed $s_1\&s_2$ detect the aspect category, OTE tuples of a sentence.

The metrics used for the assessment of the s_1, s_2 and $s_1\&s_2$ sub-tasks are the *F1-score* given by the computation of the *precision* and *recall* (equations 3, 4 and 5); meanwhile for the sub-task s_3 we use the *accuracy* (equation 2).

$$accuracy = \frac{t_p}{N} \tag{2}$$

$$precision = \frac{t_p}{t_p + f_p} \tag{3}$$

$$accuracy = \frac{t_p}{t_p + f_n} \tag{4}$$

$$F1 = 2\left(\frac{precision + recall}{precision \cdot recall}\right) \tag{5}$$

For the equations 2 3, 4 and 5 N is the number of target opinions, t_p, t_n, f_p, f_n are respectively the number of true positive, true negative, false positive and false negative.

In addition, we evaluate the models entirely, considering the SE-ABSA15 task as an opinion mining task, formally, the $s_1 \& s_2 \& s_3$. For this purpose, we compared the set of opinions predicted by the models with the set of target opinions for each review. Two opinions are equal if and only if they have the same aspect category, target, and sentiment polarity; that is, the model reaches the target for all sub-tasks (s_1, s_2, s_3). For the opinion mining task, the model predicts a set of opinions ($\mathcal{O}^S_{pred}$) for each sentence S, and we compare it with the relative set of target opinions ($\mathcal{O}^S_{target}$), computing the F1-score $F1^S$. After this we average the F1-scores over all sentences S obtaining the *overall F1-score* (equation 6).

$$F1_{overall} = \frac{1}{|S|} \sum_{S} F1^S \tag{6}$$

6 Results

For our results, we took our training set and separated it into the 80-20 proportion where we have 80% used for training while the remaining 20% for validation, while for the test set, we use the test provided by the challenge.

6.1 Comparison between models

We performed our internal comparison between the models to find the best one. We compare the different model architectures Jointed and Disjointed by changing the different encoders: BERT, DistilBERT, RoBERTa. The table 5 shows the results of the f1-scores for the training and validation sets.

Our Models		
Model type	Encoder	F1 (TR - VAL)
Disjointed	BERTBase	62.79 - 63.36
Jointed	BERTBase	64.08 - 65,89
Disjointed	DistilBERT	62.70 - 63,45
Jointed	DistilBERT	63.95 - 65,94
Disjointed	RoBERTa	62.88 - 63,74
Jointed	RoBERTa	64,81 - 66,20

Table 5: A quantitative comparison of the results obtained by the models we used.

We compared our models with the F1 score calculated with our evaluation method; we chose the final model according to the highest F1 score in the validation set because it represents the generalization capability on unseen data. These results show that the joint training framework in our method is superior to the traditional separate training mechanism, exploiting the dependence relationship between targets and aspects in target-aspect-sentiment detection. Another exciting thing is that the DistilBERT model performs better than the normal BERTbase; this

may cause the distilled one to be well regularized from the distillation technique, and it can provide better generalization power even if it has fewer parameters than the BERT.

The chosen model at the end was RoBERTa jointed trained on 3 epochs with a Batch size of 16 and a learning rate of 2e-5 because it reached the highest value of f1 in the validation set. The fact that this model performed better than the others did not surprise us because, as we know from the theory RoBERTa model is pre-trained with a way bigger dataset compared to the other, and this let the model provide a way better quality latent space that can represent more words and sentences.

6.2 Comparison with other models

In this section, we will compare our best model with models submitted by the teams for the SE-ABSA15 competition [14]. In the table 6, we report the F1-score (5) of the sub-tasks s1, s2, s1&s2 and the accuracy (2) of the sub-task s3. For each subtask, we show the scores of the baseline model, the best model, and the average score of the official submissions. For each sub-task, the baseline model is a Support Vector Machine (SVM) with a linear kernel. In particular, n uni-gram features are extracted from the respective sentence of each tuple that is encountered in the training data.

Our proposed model is inspired by the TAS-BERT model [22] and outperforms the old results due to the use of the encoder of a pre-trained language model. We also insert the TAS-BERT model [22] to make a comparison for better reference results. As we can see from the table 6, TAS-Bert has better performance with respect to our model due to a better fine-tuning and the use of the conditional random field (CRF) in addition to the soft-max layer and a "TO" tagging schema. We also compared the results of the opinion mining task ($s1\&s_2\&s_3$) which jointly evaluates the three main subtasks (table 7). The table 7 shows the overall F1-score (equation 6).

Comparison with other models				
Team	s1 (F1)	s2 (F1)	s1 & s2 (F1)	s3 Acc.
Our	0.70	0.72	0.59	0.83
Baseline	0.51	0.48	0.34	0.64
Best2015	0.62*	0.70*	0.43	0.79*
Average2015	0.53	0.52	0.37	0.71
TAS-BERT	0.76	0.75	0.63	
EliXa*		0.70		0.70
NLANGP*	0.62	0.62	0.40	
wnlp		0.58		0.71
SINAI*				0.61
SIEL*	0.57	0.54		0.71
TJUdeM*	0.52	0.52	0.38	0.69
UMDuluth	0.57	0.50	0.33	0.71
LT3*	0.54	0.50	0.36	0.75
IHS-RD.	0.50	0.63	0.43	
UFRGS*	0.52	0.49	0.35	0.72
Lsislif		0.62		0.75
V3*	0.42	0.46		0.69
Sentiue*	0.54	0.40	0.31	0.79
ECNU*				0.70
q-BDDA		0.36		

Table 6: Comparison between models. The table shows the F1 scores for s1, s2 and s1&s2 and accuracy for s3. Models with (*) indicate unconstrained systems models trained on additional external data.

Overall comparison with TAS-BERT model	
	s1 & s2 & s3 ($F1_{overall}$)
Our model	0.51
TAS-BERT	0.58

Table 7: Overall comparison of s1 & s2 & s3 with TAS-BERT model.

6.3 Demo of the model on Gradio

We created a demo of our final model by using `Gradio`. Gradio is an open-source python library that lets the customers publish their models and design a simple UI to let the people try the model. Once we upload our model with this library, it automatically provides us a link where all the people can access the model with a WEB interface (figure 14). This platform provides free hosting hardware like GPU for running the models and can also log all the errors that can occur.

Figure 14: Demo of our model on Gradio.

6.4 Target conflicts case of study

We observed after many inferences on the model that when we have a conflict between targets (more opinions with the same aspect and sentiment), our method fails to detect the opinion with the implicit target ("NULL" target). Our model construction causes this failure. We could tackle this kind of failure, but we decided to lose the "NULL" target opinion for the rare occurrence of this conflict. From the example below 3 we can see a sentence with multiple opinions in which we have a conflict in the target because we have the NULL target and another target at the same time with the same aspect and polarity. In this case, our model will lose the implicit target because the way we construct the model will find in the name entity recognition just one non-implicit target.

In a nutshell, our model either predicts no target (implicit target) or only explicit targets. For this reason, the implicit and explicit targets cannot coexist for the same (aspect, polarity) pair.

Listing 3: Example of target conflict

```
<sentence id="BFC#4:2">
    <text>It was totally overpriced- fish and chips was about $15.... </text>
    <Opinions>
        <Opinion target="NULL" category="FOOD#PRICES" polarity="negative"
            from="0" to="0" />
        <Opinion target="fish and chips" category="FOOD#PRICES"
            polarity="negative" from="27" to="41" />
    </Opinions>
</sentence>
```

7 Final considerations

Thanks to this challenge, we understood the importance of the ABSA task, particularly for companies that sell products or services. This task differs from the sentiment analysis task by producing not only one general opinion but several multiple opinions with a specific aspect

target polarity. So it is very accurate and gives the vendors the possibility of knowing the perception of the product by the consumers.

However, addressing this task from the machine learning point of view is very complex and requires much data for a better generalization. In the SE-ABSA15 competition, we had additional difficulty due to the small size dataset provided. To solve this task, we exploit the encoder models, notably the Transformers, the state-of-the-art models for most natural language tasks. From our experiments, we tried different encoders (BERT base, DistilBERT, RoBERTa) models, and we noticed the difference between them given by the dataset used for the training and the complexity of the models. Ultimately, we chose RoBERTa as the best model because it has better generalization capability on unseen data.

We could also improve our final model by trying a more extensive grid search of the hyperparameters, but we need more time or powerful hardware. For that reason, we trained our model with the TPUs offered by the Google Colab platform, and we noted that they are very efficient and faster than the traditional GPUs and way better than CPUs.

For future improvements to our model architecture, we can try to use a different tagging schema like TO or also change our output layer from a Softmax to a CRF(Conditional Random Field); we can fix the problem with tagging conflicts and use data augmentation and additional external data.

References

[1] Maryna Chernyshevich and Vadim Stankevitch. Ihs-rd-belarus: Identification and normalization of disorder concepts in clinical notes. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 380–384, Denver, Colorado, June 2015. Association for Computational Linguistics.

[2] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[3] Orphee De Clercq, Marjan Van de Kauter, Els Lefever, and Veronique Hoste. Lt3: Applying hybrid terminology extraction to aspect-based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 719–724, Denver, Colorado, June 2015. Association for Computational Linguistics.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Aitor García Pablos, Montse Cuadros, and German Rigau. V3: Unsupervised aspect based sentiment analysis for semeval2015 task 12. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 714–718, Denver, Colorado, June 2015. Association for Computational Linguistics.

[6] Satarupa Guha, Aditya Joshi, and Vasudeva Varma. Siel: Aspect based sentiment analysis in reviews. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 759–766, Denver, Colorado, June 2015. Association for Computational Linguistics.

[7] Hussam Hamdan, Patrice Bellot, and Frederic Bechet. Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 753–758, Denver, Colorado, June 2015. Association for Computational Linguistics.

[8] Salud M. Jiménez-Zafra, Eugenio Martínez-Cámara, M. Teresa Martín-Valdivia, and L. Alfonso Ureña López. Sinai: Syntactic approach for aspect-based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 730–735, Denver, Colorado, June 2015. Association for Computational Linguistics.

[9] Anderson Kauer and Viviane Moreira. Ufrgs: Identifying categories and targets in customer reviews. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 725–729, Denver, Colorado, June 2015. Association for Computational Linguistics.

[10] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[11] Huaishao Luo, Tianrui Li, Bing Liu, and Junbo Zhang. DOER: dual cross-shared RNN for aspect term-polarity co-extraction. *CoRR*, abs/1906.01794, 2019.

[12] Dehong Ma, Sujian Li, and Houfeng Wang. Joint learning for targeted sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4737–4742, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[13] Stelios Piperidis. The META-SHARE language resources sharing infrastructure: Principles, challenges, solutions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 36–42, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).

[14] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado, June 2015. Association for Computational Linguistics.

[15] Ravikanth Repaka, Ranga Reddy Pallelra, Akshay Reddy Koppula, and Venkata Subhash Movva. Umduluth-cs8761-12: A novel machine learning approach for aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 742–747, Denver, Colorado, June 2015. Association for Computational Linguistics.

[16] José Saias. Sentiue: Target and aspect based sentiment analysis in semeval-2015 task 12. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 767–771, Denver, Colorado, June 2015. Association for Computational Linguistics.

[17] Iñaki San Vicente, Xabier Saralegi, and Rodrigo Agerri. Elixa: A modular and flexible absa platform. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 748–752, Denver, Colorado, June 2015. Association for Computational Linguistics.

[18] Martin Schmitt, Simon Steinheber, Konrad Schreiber, and Benjamin Roth. Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1114, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[19] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. *CoRR*, abs/1903.09588, 2019.

[20] Zhiqiang Toh and Jian Su. Nlangp: Supervised machine learning system for aspect category classification and opinion target extraction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 496–501, Denver, Colorado, June 2015. Association for Computational Linguistics.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[22] Hai Wan, Yufei Yang, Jianfeng Du, Yanan Liu, Kunxun Qi, and Jeff Z. Pan. Target-aspect-sentiment joint detection for aspect-based sentiment analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9122–9129, Apr. 2020.

[23] Feixiang Wang, Man Lan, and Wenting Wang. Towards a one-stop solution to both aspect extraction and sentiment analysis tasks with neural multi-task learning. pages 1–8, 07 2018.

[24] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas, November 2016. Association for Computational Linguistics.

[25] Wei Xue and Tao Li. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2514–2523, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[26] Zhifei Zhang, Jian-Yun Nie, and Hongling Wang. Tjudem: A combination classifier for aspect category detection and sentiment polarity classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 772–777, Denver, Colorado, June 2015. Association for Computational Linguistics.