Report for task-24: Sentipolc

Leonardo Colosi
Sapienza Università di Roma
`colosi.1799057@studenti.uniroma1.it`

1 Dataset Description

The original *Sentipolc* dataset has been created to perform three different sentiment analysis tasks at message level of Italian tweets. An accurate description of the dataset is provided by the organizers of the challenge at the official website. The data are available in *.csv* format and have already been split into test and train sets. The size of the train set is of 7410 samples while the test set contains 2000 samples.

2 Format

Table [3] and Table [4] illustrate some examples taken respectively from the train and test splits of the original dataset. Train labels are represented as integer while Test labels as strings of numbers. The meaning associated to the values of each label is reported in the description of the dataset here.

3 Methodology

The code developed to re-frame the dataset works by iterating over each line of the original *csv* file while extracting the information needed to produce the new samples. Each new sample is represented by a *json* object, according to a standard template. The final resulta are two a collection of samples in *jsonl* format, for train and test data. The strategy used to extract the information differ with respect to the sub-task of interest.
Sub-task 1: Binary subjectivity classification. The value under the *subj* field was considered as the only relevant annotations to establish the subjectivity of the given tweet.
Sub-task 2: Multi-class polarity classification. All the possible combination of values under the *opos* and *oneg* fields has been used to determine the polarity of each sample.
Sub-task 3: Irony detection task. Also here just the value under the *iro* field was considered enough to determine the irony level of the associated text.

The elements inserted into the choices list are the results of a mapping from the binary labels to the corresponding natural language attributes. An additional field *topic* has been included to the *json* template in order to keep track of the context from which the tweet was extracted from. The mapping procedure is reported in Table [2]. Each Tweet text strings has been polished before being added to the new sample, meaning that educational escape characters has been removed. An example of the output is reported in Figure [1].

top	Topic
0	"generico"
1	"politico"
2	"socio politico"

Table 1: Mapping between top label and topic definition.

Task	Label Value	Choice
1	subj = 0	"oggettivo"
1	subj = 1	"soggettivo"
2	opos = 0; oneg = 1	"negativo"
2	opos = 1; oneg = 0	"positivo"
2	opos = 1; oneg = 1	"misto"
2	opos = 0; oneg = 0	"neutrale"
3	iro = 0	"serio"
3	iro = 1	"ironico"

Table 2: Mapping between original label and possible entries of the choices list.

```
{
    "id":       int,
    "text":     str,
    "topic":    str,
    "choices":  list[str],
    "label":    int
}
```

Figure 1: Standard template for a json object in the dataset.

idtwitter	subj	opos	oneg	iro	lpos	lneg	top	text
"122449983151669248"	1	0	1	0	0	1	1	""Intanto la partita per..."
"125485104863780865"	1	0	1	0	0	1	1	"@matteorenzi: Alle 10..."
"125633929217708032"	1	1	0	0	1	0	1	"Mario Monti sul Corriere:...

Table 3: Examples taken from the train set.

idtwitter	subj	opos	oneg	iro	lpos	lneg	top	text
"507074506880712705"	"0"	"0"	"0"	"0"	"0"	"0"	"2"	"Tra 5 minuti presentazione..."
"507075789456961536"	"1"	"1"	"0"	"0"	"1"	"0"	"2"	"@matteorenzi: Alle 10..."
"507081184518868992"	"1"	"0"	"0"	"0"	"0"	"0"	"2"	"#labuonascuola Eccolo...

Table 4: Examples taken from the test set.

4 Prompts

The prompts for the three sub-tasks has all been generated following three different styles. The first style consist into ask the model a simple straight forward question including also all the alternative as possible answers. The second style has more complex formulation of the prompt where the model is explicitly asked to focus only on the content of the given tweet. The third style can be considered as a free text generation approach. Here the model is promoted just with the source text and the general idea of the task but it is not constrained to chose its answer between a set of possible options. An additional modification to the prompt involved incorporating information about the context from which the tweet was sourced, a detail already present in the second style of the prompt.

5 How to run

In order to execute the code it is required to install some standard python libraries which are listed in the *requirement.txt* file. After this initial set-up will be possible to directly run the code inside the python file: *task_24.py*. This script will automatically download the original dataset from the online source directly into a *data/* directory. All the formatted partition of the new dataset will be saved into a *results/* directory, after the termination of the execution. It is also possible to test the correct generation of prompts by executing *prompt.py*. Run this script is by using the *-t* option, provide a number between 1 and 3 to identify the task. The code will then chose random samples, from the newly generated train and test splits, to be substituted inside the prompts templates.