NLP Assignment: Aspect-Term Polarity Classification in Sentiment Analysis

DSBA Master – NLP Lecture 2024 - S. Aït-Mokhtar

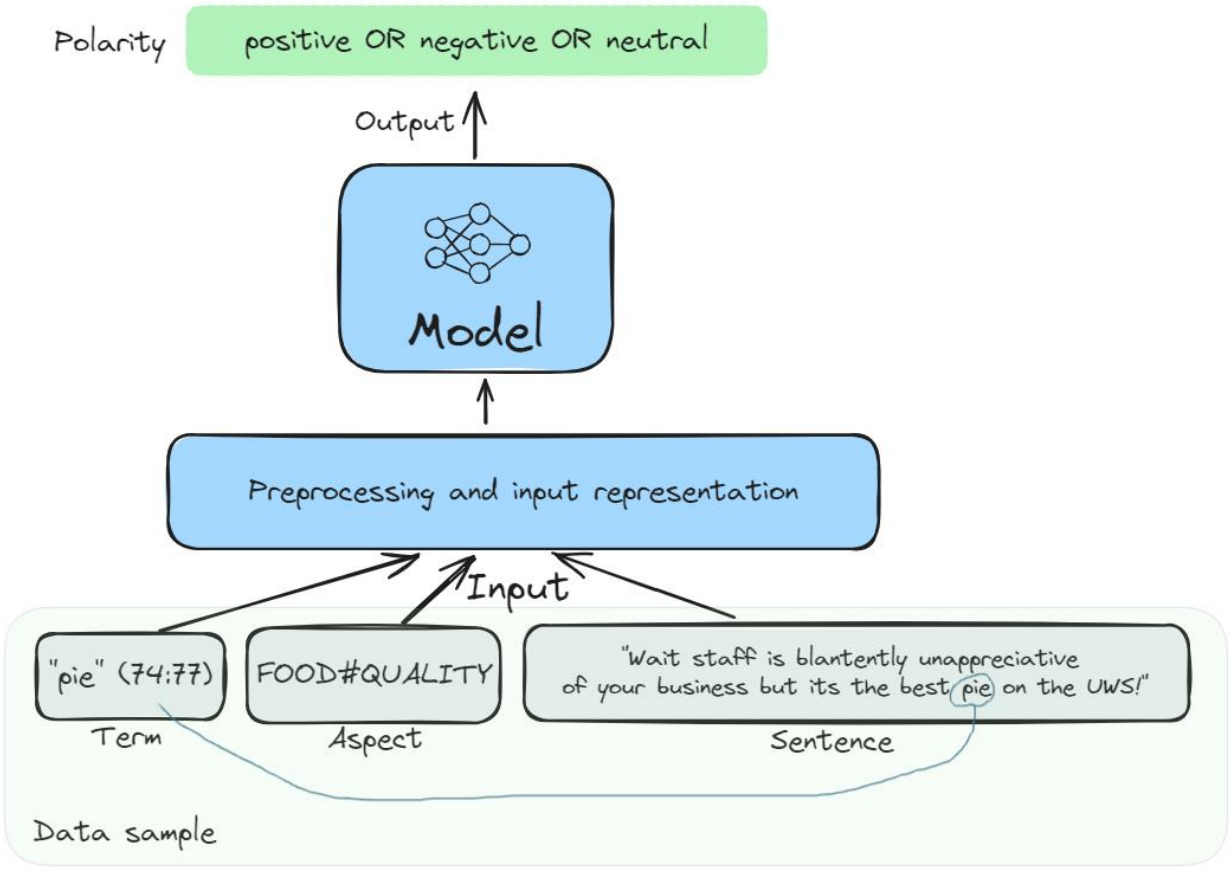
Deadline for project deliverable: April 5, 2024

Assignment for groups of 4 max.

Please send deliverable to: salah.ait-mokhtar@naverlabs.com

1 Introduction

The goal of this assignment is to implement a model that predicts opinion polarities (positive, negative or neutral) for given aspect terms in sentences. The model takes as input 3 elements: a sentence, a term occurring in the sentence, and its aspect category. For each input triple, it produces a polarity label: *positive*, *negative* or *neutral*. Note: the term can occur more than once in the same sentence, that is why its character start/end offsets are also provided (74:77 in figure below).



2 Dataset

The dataset is in TSV format, one instance per line. As an example, here are 2 instances:

negative	SERVICE#GENERAL	Wait staff	0:10	Wait staff is blantently unappreciative of your business but its the best pie on the UWS!
positive	FOOD#QUALITY	pie	74:77	Wait staff is blantently unappreciative of your business but its the best pie on the UWS!

Each line contains 5 tab-separated fields: the **polarity** of the opinion (the ground truth polarity label), the **aspect category** on which the opinion is expressed, a specific **target term**, the **character offsets** of the term (*start:end*), and the **sentence** in which the term occurs and the opinion is expressed.

For instance, in the first line, the opinion polarity regarding the target term "***wait staff***", which has the aspect category **SERVICE#GENERAL**, is ***negative***. In the example of the second line, the sentence is the same but the opinion is about a different aspect and a different target term (***pie***), and is *positive*.

There are 12 different aspects categories:

AMBIENCE#GENERAL
DRINKS#PRICES
DRINKS#QUALITY
DRINKS#STYLE_OPTIONS
FOOD#PRICES
FOOD#QUALITY
FOOD#STYLE_OPTIONS
LOCATION#GENERAL
RESTAURANT#GENERAL
RESTAURANT#MISCELLANEOUS
RESTAURANT#PRICES
SERVICE#GENERAL

The training set (filename: `traindata.csv`) has this format (5 fields) and contains 1503 lines, i.e. 1503 opinions. The classifier should be learned only from this training set.

A development dataset (filename: `devdata.csv`) is distributed to help you set up your classifier and estimate its performance. It has the same format as the training dataset. It has 376 lines, i.e. 376 opinions.

We will perform the final evaluation by measuring the accuracy of the classifier on a test dataset that is not distributed. The majority class of the dev set is about 70% (*positive* labels), and will be considered as a (weak) baseline.

3 How to proceed

1. Create a python environment and install/use **python = 3.10.x** (required). Besides the standard python modules, you can use the following libraries:
    a. pytorch = 2.1.x
    b. transformers = 4.34.1

    c. tokenizers = 0.14.1
    d. datasets = 2.14.5 (just the huggingface library 'datasets', no labelled data)
    e. scikit-learn = 1.2.1
    f. numpy = 1.26.0
    g. pandas = 2.1.1
2. Download the **nlp_assignment.zip** file and uncompress it to a dedicated root folder. The root folder will contain 2 subfolders:
    a. **data**: contains traindata.csv and devdata.csv
    b. **src**: contains 2 python files: tester.py, classifier.py
3. Implement your classifier by completing the "Classifier" class template in src/classifier.py, containing the following 2 methods:
    a. The **train** method takes training data file and a dev data file as input, and trains the model on the specified device
    b. The **predict** method takes a data file (e.g. devdata.csv), it should run on the specified device return a python list of predicted labels. The returned list contains the predicted labels in the same order as the corresponding examples in the input file
4. You can create new python files in the src subfolder, if needed to implement the classifier.
5. The classifier must use the device specified as a parameter in the train() and predict() methods. **Please do not use a default device (like 'cuda' or 'cuda:0')!** Also, **the model should not require more than 14GB of memory to run on the data** (that's the limit of the GPU device on which the program will be evaluated).
6. To check and test your classifier, **cd to the src subfolder** and **run tester.py**. It should run without errors, training the model on traindata.csv and evaluating it on devdata.csv, and reporting the accuracy and speed measures.
7. Please **do not modify tester.py**! Your program must run successfully without having to modify this file.
8. The exact content of the deliverable is described in section 4 of this document
9. Your project deliverable must be a unique **zip** file (a compressed folder). No gz, or other compression format.
10. The **name of the zip file** must consist of the family names of the authors of the deliverable. Example: Clouseau_Holmes_Velasquez.zip
11. **The zip file size should not exceed 1 MBs**.
12. Send the zip file by email to: **salah.ait-mokhtar@naverlabs.com**

4 Deliverable Content

When uncompressed, the main folder must contain the following elements:

Element	Description
README.txt	A plain text file that should contain: 1. Names of the students who contributed to the deliverable (max=4) 2. A clear and detailed description of the implemented classifier (type of classification model, input and feature representation, resources etc.) 3. The accuracy that you get on the dev dataset.
src	A subfolder containing ALL the python source files required to train and run your classifier using the unmodified **tester.py**, including the completed classifier.py file. You can put in this folder other code files (that you import from classifier.py), and any potential resources your model requires (e.g. list of polarity words or other manual features, if you use such features). Please do not include pre-trained LM files in the deliverable (the deliverable size is limited to 1 MB).

Note:

- **Please make sure that when you cd to the src subfolder and launch `tester.py` (unmodified!) with the python interpreter, it runs without errors**: it trains the classifier on the train set and evaluates it on the dev dataset, outputting the average accuracy.
- You can use any type of model, including generative models, and any kind of method, as long as:
    - The training and inference can run on a GPU with 14GB of memory
    - The program does not require a library that is not listed in section 3.1.