NLP Assignment: Aspect-Term Polarity Classification in Sentiment Analysis

CentraleSupélec – NLP Course 2025 - S. Aït-Mokhtar

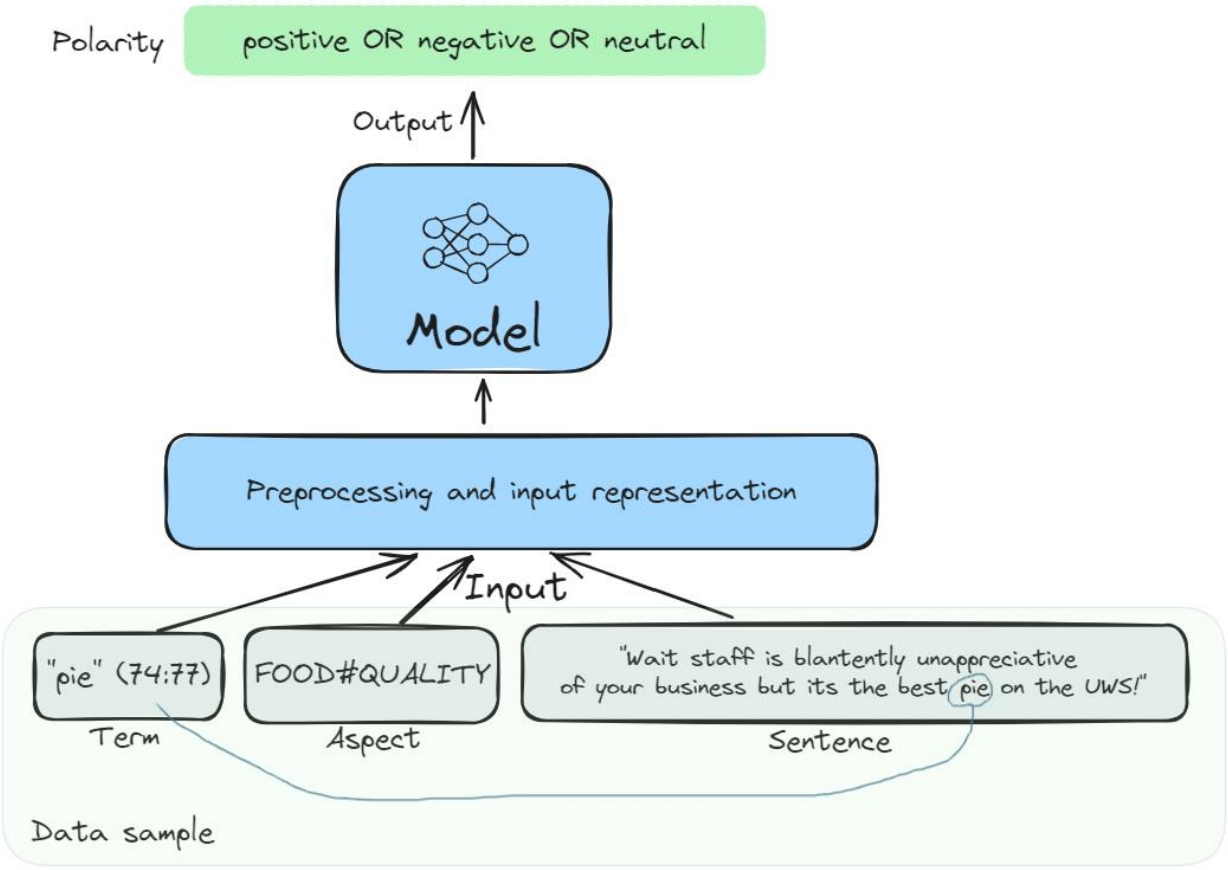
Deadline for project deliverable: April 22, 2025

Assignment for groups of 3 max.

Please when done, upload your deliverable as a zip file to the assignment page of the NLP Team on Microsoft Teams (see details on the name and content of the zip file in the sections below)

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. If you choose a method requiring training, then the training should be performed only with this training set.

A development dataset (filename: `devdata.csv`) is distributed to help you set up your model and/or 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 model 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 (very weak) baseline.

3 Python and libraries

1. Install Anaconda, Miniconda, Mambaforge or Miniforge.

2. Create a new **python 3.12** environment (let's call it `csnlp2025`).

3. Activate the csnlp2025 environment.

4. Install **PyTorch 2.6.0**. See https://pytorch.org/get-started/locally/ for the appropriate command depending on your machine configuration.

5. Install the following libraries (in the specified versions):

```
pip install transformers==4.50.3

pip install peft==0.15.1 trl==0.16.0 datasets==3.5.0

pip install sentencepiece=0.2.0

pip install lightning==2.5.1

pip install ollama==0.4.7

pip install pyrallis==0.3.1
```

For this assignment**, the only libraries that are allowed** are those listed above (with the specified versions), and the other librairies that get automatically installed when doing the installations listed above or when installing Anaconda, Miniconda or Miniforge, and python 3.12.

No other python library is allowed.

4 List of authorized pre-trained language models

No other model, besides those listed below, will be accepted.

4.1 Discriminative/classification methods with fine-tuning:

List of authorized encoder-only models, all from Huggingface repository (with their exact identifiers):

google-bert/bert-base-cased

google-bert/bert-base-uncased

google-bert/bert-large-cased

google-bert/bert-large-uncased

FacebookAI/roberta-base

FacebookAI/roberta-large

microsoft/deberta-v3-base

microsoft/deberta-v3-large

4.2 Generative methods, with fine-tuning:

List of authorized decoder-only models, all from Huggingface repository (with their exact identifiers):

facebook/opt-125m

facebook/opt-350m

4.3 Generative methods, WITHOUT fine-tuning:

List of authorized LLMs from the Ollama repository (with their exact identifiers, all quantized). These models can be used exclusively for simple in-context-learning, i.e. **no training of the model!**

gemma3:1b

gemma3:4b

llama3.2:1b

llama3.2:3b

qwen2.5:1.5b

qwen2.5:3b

If you use one of these Ollama models, the implementation of the **train() method should do nothing**, i.e. the body of the **train()** function must contain only the instruction "**pass**".

5 How to proceed

1. 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
2. Implement your system 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 and 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.
3. You can create new python files in the src subfolder, if needed to implement the model.
4. Your program 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).
5. To check and test your model, **cd to the src subfolder** and:
    python tester.py –g <gpu_id>
    or, if you are using an Ollama server:
    python tester.py -o <url_of_you_ollama_server>

It should run without errors and report the accuracy and speed metrics.

6. Please **do not modify tester.py**! Your program must run successfully without having to modify this file.
7. The exact content of the deliverable is described in section 4 of this document
8. Your project deliverable must be a unique **zip** file (a compressed folder). No gz, or other compression format.
9. The **name of the zip file** must consist of the family names of the authors of the deliverable. Example: Clouseau_Holmes_Velasquez.zip
10. **The zip file size should not exceed 1 MBs**.
11. **Upload your deliverable as a zip file to the assignment page of the NLP Team on Microsoft Teams**

6 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/or run your model 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. 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 if needed, and evaluates it on the dev dataset, outputting the average accuracy.
- The program must not require a library that is not listed in section 3.
- The program must not use a pre-trained LM that is not listed in section 4.