NLP Exercise 2: Aspect-Based Sentiment Analysis

C. Brun and S. Aït-Mokhtar

Deadline for project deliverable: March 22, 2020

Send deliverable to: caroline.brun@naverlabs.com and salah.ait-mokhtar@naverlabs.com

1 Introduction

The goal of this exercise is to implement a classifier to predict aspect-based polarities of opinions in sentences, that assigns a polarity label to every triple <aspect categories, aspect_term, sentence>. The polarity labels are positive, negative and neutral. Note that a sentence may have several opinions.

An example of a (small) dataset containing only 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 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 that opinion is expressed.

For instance, in the first line the opinion about the SERVICE#GENERAL aspect, which is associated to the term "wait staff", is negative.

In the second line, the sentence is the same but the opinion is about a different aspect and a different target term, and is positive.

There are 12 different aspects categories, which are:

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 has this format (5 fields) and contains 1503 lines, i.e. 1503 opinions.

File: traindata.csv

The classifier should be learned from this training set.

A development dataset is distributed to help you set up your classifier and assess its performance. It has the same format as the training dataset. You have to use it to apply your classification model and predict a polarity label. This polarity label can then be compared with the reference, i.e. the first field of the dev dataset.

File: devdata.csv (376 lines, i.e. 376 opinions).

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

2 How to proceed

1. Install and use **python 3.x** (required – python 2.x not accepted). Yu can use PyTorch, Tensorflow or Keras. Besides the standard python modules, you can use the following libraries: scikit-learn (>=0.19.1), pandas (>= 0.23.0), nltk (>= 3.3.0) and spacy (>= 2.0.12), gensim (>=3.8.0), transformers (>=2.0.0).
2. You can work on the project **in groups of 2, or 3 (max)**.
3. Download the **exercise2.zip** file (available on the NLP course website) and uncompress it to a dedicated root folder. The root folder contains 3 subfolders:
    a. **data**: contains traindata.csv and devdata.csv
    b. **src**: contains 2 python files: tester.py, classifier.py
    c. **resources** : (empty) where you can put your resource files if needed
4. Implement your classifier by completing the "Classifier" class template in src/classifier.py, containing the following 2 methods:
    a. The **train** method takes a training data file as input and trains the model
    b. The **predict** method takes a data file (e.g. devdata.csv) and should 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
5. You can create new python files in the src subfolder, if needed to implement the classifier.
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 measure.
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 3 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 contain **only** the family names of the authors of the deliverable. Example: Clouseau_Holmes_Velasquez.zip
11. The zip file size should not exceed 3 MBs.
12. Send the zip file by email (**deadline**: March 22, 2020)

3 Deliverable Content

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

Element	description
README.txt	A plain text file that should contain a couple of paragraphs describing: 1. Name of the students who contributed to the deliverable (max=3) 2. A couple of paragraphs describing your final system (type of classification model, feature representation, resources etc.) 3. The accuracy that you get on the dev dataset.
resources	(optional): containing resources in case you use resources (e.g. pre-trained word embeddings, polarity lexicons)
src	A subfolder containing ALL the python source files required to train and run your classifier using the unmodified **tester.py** : this file is used to run and evaluate your classifier.

Note:

- **Please make sure that when you cd to the src subfolder and launch `tester.py` (unmodified!) with a python3 interpreter, it runs without errors**: it trains the classifier on the train set and evaluates it on the dev dataset, outputting the accuracy.
- You can experiment with different classification models (deep and/or non deep models), using only the following ML frameworks: scikit-learn, PyTorch, Tensorflow and/or Keras.
- Using rich linguistic features is also allowed: lemmatization, shape, POS tags, and/or parsing dependency relations using the SpaCy parser (c.f. lecture on parsing), etc.