Assignment Two

Sentiment classification for social media

CS918: 2023-24

Submission: 12 pm (midday) Thursday 21st March 2024

Notes

a) This exercise will contribute towards 30% of your overall mark.

b) Submission should be made on Tabula and should include a Python code written in **Juypter Notebook** and a **report** of 3-5 pages summarising the techniques and features you have used for the classification, as well as the performance results.

c) You can use any Python libraries you like. Re-use of existing sentiment classifiers will receive lower marks. The idea is for you to build your own sentiment classifier.

Topic: Building a sentiment classifier for Twitter

SemEval competitions involve addressing different challenges pertaining to the extraction of meaning from text (semantics). The organisers of those competitions provide a dataset and a task, so that different participants can develop their system. In this exercise, we will focus on the Task 4 of Semeval 2017 (http://alt.qcri.org/semeval2017/task4/). We will focus particularly on Subtask A, i.e. classifying the overall sentiment of a tweet as positive, negative or neutral.

As part of the classification task, you will need to preprocess the tweets. You are allowed (and in fact encouraged) to reuse and adapt the preprocessing code you developed for Coursework 1. You may want to tweak your preprocessing code to deal with particularities of tweets, e.g. #hashtags or @user mentions.

You are requested to produce a standalone Juypter Notebook that somebody else could run on their computer, with the only requirement of having the SemEval data downloaded. Don't produce a Juypter Notebook that runs on some preprocessed files that only you have, as we will not be able to run that.

Exercise Guidelines

- **Data:** The training, development and test sets can be downloaded from the module website (semeval-tweets.tar.bz2). This compressed archive includes 5 files, one that is used for training (twitter-training-data.txt) another one for development (twitter-dev-data.txt) and another 3 that are used as different subsets for testing (twitter-test[1-3].txt). You may use the development set as the test set while you are developing your classifier, so that you tweak your classifiers and features; the development set can also be useful to compute hyperparameters, where needed. The files are formatted as TSV (tab-separated-values), with one tweet per row that includes the following values:

  tweet-id<tab>*sentiment*<tab>*tweet-text*

  where sentiment is one of {*positive*, *negative*, *neutral*}. The *tweet IDs* will be used as unique identifiers to build a Python dictionary with the predictions of your classifiers, e.g.:

  predictions = {'163361196206957578' : 'positive',
  '768006053969268950' : 'negative',
  ...}

- **Classifier:** You are requested to develop 3 classifiers that learn from the training data and test on each of the 3 test sets separately (i.e. evaluating on 3 different sets). You are given the skeleton of the code (*sentiment-classifier.tar.bz2*), with evaluation script included, which will help you develop your system in a way that we will then be able to run on our computers. Evaluation on different tests allows you to generalise your results. You may achieve an improvement over a particular test set just by chance (e.g. overfitting), but improvements over multiple test sets make it more likely to be a significant improvement.

You should develop at least 3 different classifiers, which you will then present and compare in your report. Please develop at least 2 classifiers based on traditional machine learning methods such as MaxEnt, SVM or Naïve Bayes trained on different sets of features (you could use Scikit-learn library). Then, train another classifier based on the LSTM using PyTorch (and optionally the *torchtext* library) by following the steps below:

a) Download the GloVe word embeddings and map each word in the dataset into its pre-trained GloVe word embedding.

   First go to https://nlp.stanford.edu/projects/glove/ and download the pre-trained embeddings from 2014 English Wikipedia into the "data" directory. It's a 822MB zip file named glove.6B.zip, containing 100-dimensional embedding vectors for 400,000 words (or non-word tokens). Un-zip it. Parse the unzipped file (it's a txt file) to build an index mapping words (as strings) to their vector representation (as number vectors).

   Build an embedding matrix that will be loaded into an Embedding layer later. It must be a matrix of shape (*max_words*, *embedding_dim*), where each entry *i* contains the *embedding_dim*-dimensional vector for the word of index *i* in our reference word index (built during tokenization). Note that the index 0 is not supposed to stand for any word or token -- it's a placeholder.

b) Build and train a neural model built on LSTM.

   Define a model which contains an Embedding Layer with maximum number of tokens to be 5,000 and embedding dimensionality as 100. Initialise the Embedding Layer with the pre-trained GloVe word vectors. You need to determine the maximum length of each document. Add an LSTM layer and add a Linear Layer which is the classifier. Train the basic model with an 'Adam' optimiser. You need to freeze the embedding layer by setting its *weight.requires_grad* attribute to *False* so that its weights will not be updated during training.

- **Evaluation:** You will compute and output the macroaveraged F1 score of your classifier for the positive and negative classes over the 3 test sets.

  An evaluation script is provided which has to be used in the skeleton code provided. This evaluation script produces the macroaveraged F1 score you will need to use. You can also compute a confusion matrix, which will help you identify where your classifier can be improved as part of the error analysis.

  If you perform error analysis that has led to improvements in the classifier, this should be described in the report.

To read more about the task and how others tackled it, see the task paper:
http://alt.qcri.org/semeval2017/task4/data/uploads/semeval2017-task4.pdf

Marking will be based on:

a. Your performance on the task: good and consistent performance across the test sets. While you are given 3 test sets, we will be running your code in 5 test sets to assess its generalisability. Therefore, making sure that your code runs is very important. [25 marks]
b. Clarity of the report. [20 marks]
c. Producing runnable, standalone code. [20 marks]
d. Innovation in the use of features and/or deep learning architectures (e.g.: BERT, prompting strategies with language models, etc.). [25 marks]
e. Methodological innovation. [10 marks]

Total: 100 marks