MMAI 5400 Assignment 3 -- Sentiment classification with a neural language model

This assignment has the same goal as assignment 2, that is, to classify the sentiment of reviews from Trustpilot.com. You will even use the same data. However, this time, you have to beat your previous performance (specifically, you need to get more than 60% accuracy on the test set). To do this you will be encoding the reviews with a neural language model followed by sentiment classification. The strategy is similar to the lecture 9 tutorial (`MMAI5400_class09_umlfit_textclass.ipynb`).

Submission

The assignment should be submitted as Python 3 code and uploaded to Canvas as a single `PY` file (**not** a Jupyter Notebook) and the trained model. The due date is on July 14 at 8:30 am.

The code will be tested and should produce the output specified below.

Data

Use the file `reviews.csv` , containing close to 2000 reviews. To read `reviews.csv` with `pandas` it might be good to specify `,` as the delimiter, as follows:

```
data = pd.read_csv('reviews.csv', delimiter=',')
```

or

```
data = pd.read_csv('reviews.csv', sep=',')
```

The two are equivalent.

Task

Your task is to do sentiment analysis of the reviews. You will do this by training a neural language model connected to a classification head with `RatingValue` as labels. `RatingValue == 0` corresponds to negative reviews, `RatingValue == 1` to neutral and `RatingValue = 2` to positive reviews.

Remember that the classes in `reviews.csv` are unbalanced while they are balanced in the test set (not available to you), so it might be good idea to balance the data.

The resulting table should look as follows:

	RatingValue	Review
1	`1`	`"It shows ..."`
2	`0`	`"Disgusting..."`
3	`2`	`"Yummy..."`
...	...	...

Split the data into training and validation sets and store them as `training.csv` and `valid.csv`. You should use the validation data for model selection and evaluation.

Finally, the test data on which your model will be evaluated will be comma (`,`) separated and have the same format as the above example table.

Save the model with `model.export(fname="fine_tuned.pkl")` so that it can be loaded with `fastai.text.all.load_learner(model)`.

Deliverable

You need to submit a single Python file (`PY` **NOT** `IPYNB`) and the trained model/weights. The `PY` file should do the following:

- Load the trained model(s) with `fastai.text.all.load_learner(model)`.
- Evaluate the model on an unseen test set (`test.csv`) with the format described above.
- Print the following:

```
accuracy: "accuracy on the test set"

F1_score: "f1-score on the test set"

Confusion_matrix:
              negative neutral positive
      negative   a       b        c
      neutral    d       e        f
      positive   g       h        i
```

Together with the `PY` file you should also submit the model/weights.

However, the code for the initial pruning/balancing of the data in `reviews.csv`, splitting into training and validation set or the `CSV` files should **not** be submitted.

Your code should follow the PEP 8 style guide. See the original PEP 8 style guide, an easier to read version, or a short PEP 8 YouTube intro. Practically, adding a PEP 8 plugin to your text editor (e.g. Falke8) will make it easier to follow to style guide.

Grading

For grading the model will be evaluated on unseen test data (`test.csv`). For full marks, the code has to be bug-free, PEP8 formatted and the model has to get an accuracy > 60% on the test data (not provided).

Good luck!