Task 1: Simple Classifier

Before you begin, please read the general instructions, and download the MLRD skeleton. The skeleton contains the data files and utils you need for the first part of the course, including a template for this tick.

Part 1: Manual Classification

In Moodle under Task 1 you will find four film reviews. Please read each of them and record your own opinion about whether the sentiment of the review is positive or negative in the database.

We expect there to be some disagreements in these opinions. We will use the database with the judgements later in the course.

Part 2: Sentiment Lexicon Database

One very simple way of assigning sentiment automatically is to use a **lexicon** with words that we expect to be associated with particular sentiments. For example, we might expect the words *excellent* and *boring* to provide good clues.

Please open the sentiment lexicon in the data folder (from the MLRD skeleton) and look at the entries for *excellent* and *boring*. What do you think the entries mean?

Based on the reviews that you looked at in part 1, please write down in your lab notebook 10 words which you think would be useful for the task of classifying the sentiment of film reviews and say whether you expect each to be positive or negative. Then upload a text file (with .txt extension) containing these 10 words. We will need them later.

Now check whether your words occur in the sentiment lexicon. Do you agree with the lexicon as to whether they are positive or negative?

Part 3: Simple Classifier Virtual programming lab

The main task in this session is to use the sentiment lexicon to automatically classify a large number of film reviews as positive or negative. You will create some code (in `tick1.py`) and upload it to the automatic tester.

To tokenize the review texts (convert them from plain text to a list of words) we use the NLTK tokenizer, found in `utils/tokenizer.py`. To run the code on your local machine make sure to install the NLTK package (Natural Language Tool Kit). You can do this by running `python3 -m pip install nltk`. Unlike many other packages, nltk requires you install additional data. To do this, open a python shell, and run `import nltk` followed by `nltk.download('punkt')` to install all required resources for the tokenizer.

Don't edit anything in the utils or data folders. This includes the lexicon. You won't upload these files to Moodle when you submit, only your edited version of `tick1.py`; if you make local changes, these won't be applied when your code is automatically tested.

The actual sentiment (**ground truth**) is in the file `data/sentiment_detection/reviews/review_sentiment`.

1. Classification. Write a program that reads in the sentiment lexicon, and then reads in the review files (sentiment dataset/reviews). For each review file, it records how many positive lexicon words it contains, and how many negative ones, and then arrives at a decision about whether the entirety of the text is positive or negative (as described in the slides). For this experiment, in the case of a tie between positive and negative, resolve in favour of positive.
2. Evaluation. Your program will not be right in all cases. In order to find out how close to the truth it gets, you will write a short routine that **evaluates** the algorithm, i.e. gives it a score expressing how close to the truth it gets. Write the evaluation program that reads in your output and the truth and tells you the proportion of times your program made the right decision. This number is called **accuracy**.
3. Improve your classifier. After recording the accuracy of your first attempt, you should try some simple ways to raise the accuracy of the system. For instance, can you improve the way that ties are handled? You have a choice as to exactly how to try and improve the system, but try the simplest things first, and record everything you try in your lab notebook. The automatic grader on Moodle uses a hidden test set of reviews for evaluation, so changes that show an improvement on the set you have downloaded may not be as effective on other data.

To check your code locally, you can run `python3 exercises/tick1.py` from the root directory (`mlrd/`). If this doesn't work, check the Python FAQs (on Moodle).

Once everything is working, you may submit your code to the automatic tester. Please only submit the `tick1.py` file. Once your code has passed, please follow the procedure for obtaining the tick explained in the general instructions (on Moodle).