Final Assignment **October 2022**

Task 1 (100%): Opinion Miner
Your task is to build a product review opinion miner using the data provided in **Data.zip**. The background of the task (with paper references) and its key specifications are summarised on the Engage webpage of the Final Assignment. This document details the workflow steps to help you structure the project and explains our expectations of your submission and its marking.

General information

Being the only summative assessment of the unit, the marking of the Final Assignment will be **strict** requiring a **rigorous write-up**. The assessment is **report-based** meaning that even the most brilliant implementation without an accompanying report will receive no more than a pass mark (40%). At the same time, your implementation must work and actually realise what the report claims it does.

In-code comments do not count as academic reporting. Your (hopefully clear) code is an essential part of the submission but you still need to provide an academically sound write-up with references to theory.

Project Workflow: steps

You will need to apply the following steps from the generic NLP project pipeline discussed in the course. Further guidance on preparatory steps covering *task and data analyses* and *preprocessing* will be released at an appropriate time early in the unit.

- *Analyse the data and the task:*
  Explore the data you have been given for the task. Consider the task and break it down into constituent parts. Come up with a system-level outline of your opinion miner solution.
- *Apply relevant data pre-processing steps:*
  This involves data ingestion and its conversion into useable forms for *specific* downstream processes using NLP preprocessing techniques learnt in the course.
- *Extract relevant information:*
  Identify keyphrases that describe product features. Relevant techniques from the

course that can be applied at this step include PoS tagging, chunking or parsing (e.g., for the identification of noun phrases).

Identify words and/or sentences that contain opinions (i.e., are sentiment-bearing) about the product features.

- *Apply a relevant algorithm:*
  That step applies to multiple sub-systems of the pipeline: e.g., product feature candidate pruning/ranking as well as sentiment detection/classification. You can refer to the two reference papers (see Engage page) for some inspiration (note: the papers mention the use of *association rules* to identify meaningful phrases; instead, you could use *pointwise mutual information*, which is discussed in Week 7 of the course, as the *word association measure*).

  An algorithm has to be implemented (with justification!) for each sub-system using the knowledge and skills acquired during the course and independent research. To compete for higher marks, you need to perform a **comparative experiment** (see details further on) of two algorithms for **one** sub-system.

  Note that if you experiment with different features, settings or algorithms in a machine learning framework, you need to split the dataset into training and test sets and run the experiments on the same splits to make different runs of your algorithm comparable. For example, in the sentiment analysis step you may choose to implement a baseline model (e.g., the simplest algorithm) and experiment with at least one meaningful model extension or entire approach modification, comparing the results of your extended model(s) / modified approach to the baseline one in a principled way.

- *Report evaluation results:*
  Apply relevant evaluation metrics and report the results at different steps of your implementation. Each sub-system needs to be validated independently, as well as the entire pipeline. You have the responsibility to define and explain a principled evaluation procedure for each component (don't forget explicit metric definitions!).

  For example, you are encouraged to report precision and recall in various evaluation contexts using the annotations in the provided datasets converted into your groundtruth. Groundtruth definition from annotations is an integral component of your evaluation methodology design.

  Note that you will not be assessed on the basis of your evaluation results: i.e., if you correctly implement a reasonable algorithm that attempts to solve the task but achieves low performance scores, you will not be penalised for that.

  Please remember that there is qualitative as well as quantitative evaluation. For the final evaluation of your opinion miner, print out the required opinion "summaries" presenting the features of the product and the break down of the corresponding positive

and negative sentiment counts. There is no need to print and discuss all 17 opinion tables in the evaluation: it is enough to show the representative ones.

Consider all the results obtained. Summarise and discuss the observed performance trends leading up to conclusions and any suggestions for possible future work.

Academic writing, Report Structure and the Comparative Experiment

As has been mentioned previously, a lot of emphasis in this assignment is placed on the report, so there are certain rules you need to follow to conform to good academic writing standards.

The first key element of the report is **proper methodology descriptions** at both the pipeline and task levels. If you use a known algorithm (self-implemented or from a library) as part of your methodology, it is not enough to name it: please explain what it does and what its underlying operating mechanism is (how it works, the type of algorithm it is etc.).

The second key element of the report is **theoretical explanations and justifications** of design choices. You must demonstrate your understanding of theory to prove that the choices are well grounded.

The third essential element of the report is explicit **evaluation methodology definition** and **performance trend description**, **analysis and discussion** of the results.

Comparative experiment. Let us consider the comparative experiment you need to perform to compete for higher marks. In line with the three academic writing principles outlined above, you need to properly present it. It is not enough to say: "I used *method X* for this task and now I will try *method Y* to compare. Here are the two sets of results". You should provide a thorough theoretical justification of why you choose these two methods to compare. In other words, you formulate a theoretical hypothesis on the relative performance of the two methods to set up the comparative experiment. What makes you think *method Y* would be better? Without the theoretical hypothesis, the comparative experiment will not count.

Further, please choose two algorithms that are sufficiently different to compare. There may be deductions if the difference between the approaches in the comparison is too incremental. You should make a case for whatever you choose to compare in your report (choose something interesting!). Although it is not mandated, we recommend you looking to the sentiment analysis component for your comparative experiment in the first instance as it is perhaps somewhat easier to identify two sufficiently different approaches to compare in this context.

Suggested report structure. We suggest the following approximate structure for the report, which aligns to the marking criteria (see further on):

1. Task and Data Analyses (including system/approach outline)
2. Data preprocessing
3. Product feature extraction (**to clarify:** extraction of product aspects people talk about in review texts)
4. Sentiment analysis
5. Evaluation and discussion

We would recommend writing your report in markdown cells of the jupyter notebook with your implementation so that you can refer to the generated results easier. You must submit the notebook already pre-executed showing all the results you discuss in the report (also think about relevant intermediate outputs of the pipeline).

We suggest rather than mandate the report structure because your individual system design, preferences and needs of the report narrative may require some variations on it: e.g. a common thing to do is to provide sub-system evaluation already in their corresponding sections, before the part 5 dedicated to the overall evaluation and discussion. This is fine - if the marker can easily find relevant work, they will credit it towards the right marking criterion, provided you globally speaking stick to the prescribed structure (with the 5 parts identifiable) and have any reasonable variations from the structure clearly sign-posted.

Plagiarism prevention. Finally, beware of plagiarism in all its forms in your submission.

Text. Copying entire sentences or paragraphs *verbatim* from papers and blogs into the report instead of self-generating the content is not acceptable. No marker will ever award points to the student for an explanation that is copied. So to be absolutely clear, this practice is unacceptable even if there is a reference to the source in the report. To get credit and avoid being reported for plagiarism, the report must be written from your own understanding and only in your own words, i.e., not be a collage of snippets from third-party texts.

There are *very rare* occasions when *verbatim* quoting is appropriate, e.g. to give a strict (mathematical) definition of a concept - such inserts must always be enclosed in direct speech quotation marks to keep it separate from your own text. The source of the direct quotation must be fully referenced. Excessive use of direct quotes is bad academic writing and does not allow the marker to gauge your understanding, which is essential to award credit.

Code. The guidelines in this assignment are as follows. You may use academic papers and online tutorials for algorithmic / implementation inspiration for parts of system design, but obviously not download and use complete solutions. If you use a source for algorithmic / implementation inspiration, it must be referenced and you must show full understanding of the algorithm you have adopted. The same rule applies to any third-party library functions used to build your system.

Generally speaking, code presented without an explanation typically raises red flags with the marker. So there should always be a high-level explanation of the implementation in addition to any clarifying in-code comments (which the marker may or may not consult).

We thank you for following high standards of academic integrity in your submitted work!

Information about marking

The assignment is assessed out of **100 points**, with each report section listed above being worth **20 points**. Being an open-ended assignment, there is no further breakdown into sub-criteria as there are multiple ways to get credit in each section. Further breakdown would also give away potential solutions.

The assessment will be based on the clarity of the description and motivation of the work done, steps implemented and evaluated, demonstration of the skills and knowledge acquired during the course, and insights gained. Assessors may run your code, but you will not be assessed on the quality of your code writing, nor will you be assessed on the basis of where your system's results rank amongst the results achieved by the systems submitted by other course participants or results reported in published papers.

No more than a **pass (40%)** will be given for any submission that implements all 5 project workflow steps but fails to provide sufficient detail (description / discussion / reasoning / design decision justification etc.) in the report. Please refer to the previous section of this document to understand our expectation of the academic rigour in the write up. A submission with such a minimal report also failing to implement (or submit the implementation of) some steps will receive a mark below 40%.

The mark will be capped at below the **merit threshold (60%)** for any submission not attempting a comparative experiment (details discussed previously). So, for example, if you write a good report on a single pipeline without variations, the maximum mark you can get is 59%. Just to clarify, inclusion of a comparative experiment does not automatically guarantee a mark in the higher range as the assessment depends on the overall quality of your submission, but it prevents the automatic cap at below 60%.