Coursework commentary 2017–18

CO3320 Final project report

Introduction

This report relates to the Final Project Report (FPR) submissions for CO3320 in the academic year 2017–18. Before reading this, be sure to read the accompanying report on the Preliminary Project Report (PPR) submissions. Much of what is written in the PPR report applies to the FPR submissions too, and will not be repeated here.

The comments below are organised into a number of categories:

- Organisation and presentation
- Project subject
- Literature review and background research
- Software design and development
- Experimental design
- Presentation of results
- Evaluation of results
- Discussion and conclusions

Each of these topics is discussed below. Note that most of these issues come up every year in the FPRs. If you are reading this report at an early stage in thinking about your project, and take time to ensure that each of these issues is addressed, then you should be in good shape for producing your FPR.

Organisation and presentation

While there are no rigid limits on the length of an FPR, generally speaking, most reports come in at somewhere around 40–80 pages of main text (excluding appendices, reference list, etc.). Of course, this will depend on font size, line spacing, etc., but most students are able to adequately describe their work within that kind of range.

This year, like every year, we saw a handful of reports that were much longer than this, some in the region of 150 pages of main text. It is rarely warranted to have such a long report – almost without exception, a report of over 100 pages is a sign that the student does not have a good sense of how to selectively present the important details in the main text, how to organise supplementary details in appendices, and how to refrain from including details of no direct relevance to the project.

On the other hand, we also saw a few projects where important information (such as discussion of survey results) was completely relegated to an appendix, even though it constituted an important component of the project. Think carefully about what the reader needs to know to understand what you have done, and be sure that the important information is included in the main text.

After deciding what information needs to be presented, care should be given to how it should be presented. Try to think from the perspective of a reader

who knows nothing about your project. Be sure to explain acronyms and abbreviations when you first use them – several reports were weak in this area.

Another notable weakness this year was that a significant number of reports did not include a self-evaluation by the student at the end of the report. This is a required part of the report, in which we expect to see the student reflecting on their experience of doing the project and on what they have achieved.

Be sure to follow the guidelines for FPR structure as set out in Section 6.2 of the CO3320 Subject Guide (pp.29–34).

Project subject

This year it was very gratifying to see a small number of candidates who had identified an existing piece of work published in the academic literature as the starting point for their project. While it is not necessary to do this, if you are able to do so, it can give you some confidence that the project topic is suitable. One or two students even contacted the authors of the previous work to ask questions, request further data, *etc*. This shows a high level of academic maturity.

As we see every year, some projects scored poorly because the candidate never defined clear aims – and this is key to a good project. This year in particular we saw some projects which gave differing statements of their aims at different points in the project report, leading to a lack of clarity about what the candidate was hoping to achieve. As always, we also saw some candidates who confused objectives with aims – the aims are the high-level overall goals, ends or intentions of the work.

Literature review and background research

While many candidates provided good literature reviews, some struggled to find the appropriate level of detail. This section of the FPR should provide an overview of existing work and research of specific relevance to the project topic. It should not be a general tutorial or text book, but should only discuss those aspects of existing work and theory that are specifically relevant to the work which has been built upon.

Projects that are mainly focussed on software development should certainly include in their review a discussion of any existing software or apps available on the market. Several of this year's FPRs were weak in this area.

A good way to get a better idea of the level and breadth of detail required in the literature review is to look at some of the good projects submitted in previous years, in the Project Library on the VLE (see the end of this report for details on how to access this).

As part of the planning and background research stage of the project, you should also consider what resources you will need to complete the project, and be sure that you will have access to what you need. This may include, for example, access to appropriate datasets (especially for machine learning projects). For software development projects, you should also consider what third party libraries are available and whether they might be useful to help you complete your project. This year we saw several projects where the candidate only discovered at a late stage a software library that might have made their work much easier. Many had a similar problem obtaining appropriate data for their neural network project, etc.

One word of warning about the literature review and background research: this year we saw several candidates who spent too much time on these aspects of the project, to the extent that they ran out of time to complete the whole

project. These reports generally start off very well indeed, but then do not end strongly. Although background research is an important part of the project, be sure to allocate an appropriate amount of time to it such that you can complete all other aspects of the project as well. Similarly, we also saw some candidates spend too much time on the user interface or other aspect of their project, such that they ran out of time to complete the whole project. The lesson here is that there may be some parts of the project that particularly interest you, but don't let that derail the project as a whole.

Software design and development

While many candidates did a good job at describing their process of software design and development, a significant number of reports were deficient in one or more aspects of this.

Some reports only presented the requirements for their software ("my software is going to do x, y, and z") without any discussion or justification for why these features were necessary or desirable. The most straightforward way to provide justification for features is by reference to what you have discovered in your literature review or survey of existing software and/or from the results of any requirements gathering stage. Many candidates engaged in surveys or questionnaires with potential users in order to define and justify their software requirements, which was good to see. But whether or not such surveys are appropriate, there should be some clear discussion of why each feature is needed. Furthermore, especially for complex software projects, it is usually wise to prioritise requirements and work on the most important ones first. That way, if development falls behind schedule in the project, at least the most important aspects have been tackled.

A handful of candidates experienced problems relating to the fact that they were developing software for a client (*e.g.* either it was a work-related project, or they were developing software to help a local company). Problems arose when the client changed their mind about what they wanted from the software midway through the project. This problem (known as "feature creep") can seriously impact the successful completion of a project. The way to avoid this is to have a clear process of requirements gathering at the start of the project and to use that to draw up an explicit list of requirements that is agreed between the student and the client at the start.

For any significant software development, the examiners expect to see a clear explanation of the software design, probably including UML diagrams, wireframes, *etc*. While most candidates provided this, others gave very few details of how their systems worked.

This year it was gratifying to see some candidates make a very good job at iterative development approaches, completing multiple rounds of prototype design, development, testing and feedback. Many candidates say in the report that they will pursue an iterative development approach, but it is quite rare to see this done well in practice.

Candidates often use the project as an opportunity to get hands-on experience with a new programming language or environment that interests them. For example, a significant number of projects involved the development of an Android mobile app. This year, as in previous years, some very impressive apps were developed. However, it is important to think carefully before embarking upon a project that relies upon the mastery of new skills. As in previous years, we saw several candidates who did not manage to get to grips with Android development (or

other new languages) to a sufficient degree to allow them to complete their projects – these candidates typically submit an incomplete project, or are forced to switch to a different environment (*e.g*. Processing) at the last minute and rewrite their code in a very short space of time. When considering a project that requires new skills to be learned, rigorous project planning and management is vital. You should consider alternative approaches you could switch to if things don't work out as planned, and you should monitor your progress against your plan at regular intervals.

Many candidates make use of third-party software libraries or tools in their projects, and this is a perfectly permissible thing to do. Indeed, evaluating what tools might be available to help you, and the pros and cons of using them versus developing your own code, is an important aspect of any project that involves any significant software development. But it is very important to remember that any third-party code used must be clearly acknowledged in your report – you should clearly state what libraries or code you have used, and you should make it clear which parts of your submitted code are your own and which parts have come from other sources. In a few cases, candidates submitted large sections of code copied from other sources without any indication or acknowledgement that they had not written the code themselves. The most serious of these cases led the examiners to flag the reports for formal investigation for plagiarism. It is essential to remember that proper acknowledgement of the use of other people's code is just as important as proper citation and referencing in written work.

Another thing to watch out for when using third-party libraries – and this is particularly relevant to projects involving neural networks or other kinds of machine learning – is that you should still have a good understanding of the techniques being used, and demonstrate that understanding in your project report, even if you have not implemented all of the code yourself. A weakness in some reports this year, as in most years, is when a candidate used a machine learning technique with little or no apparent understanding of how it worked. As a consequence, the technique was often used inappropriately, or its parameters were not tuned correctly, *etc*.

A few project reports jumped straight from software design to results and analysis, with nothing said about the process of software development and testing. The examiners expect to see some description of these stages, covering, for example, what (if any) third-party libraries were used, what (if any) problems were encountered during development, and how the code was tested to ensure it was working as expected.

Experimental design

Many projects involve some kind of experiments, and the design of these must be carefully considered and discussed in the FPR.

For projects that involve some kind of machine learning technique (e.g. neural networks), careful thought must be given to the design of appropriate training and testing phases, including what subset of the available data will be used for training and for testing. The data should also be examined prior to using it in experiments to ensure that it is as expected (e.g. that it does not contain any missing or nonsense values). Justification should be given for the choice of parameter settings for learning algorithms. This year, as with most years, many projects involving machine learning were rather weak on these aspects.

Elsewhere, in projects involving testing or surveys with human subjects, full details and justification should be given about how the subjects were recruited, the selection criteria (*e.g*. were you targeting a particular demographic?), how you decided how many people should be involved, etc.

In projects involving more open-ended feedback rather than a specific survey, a fairly common problem is to provide little or no details of exactly how the feedback was obtained (e.g. was it a face-to-face discussion, what questions were asked, etc.). A disappointingly large number of projects gave very little discussion of these kinds of details.

Presentation of results

Careful thought should be given to how the results of the project should be presented. This year, as ever, we saw some projects with very nice presentation of results, and others which left considerable room for improvement.

Among those that did not do a good job in this area were projects in which numerical results from experiments were presented in page after page of numeric data tables. While there may be some justification for including these raw results in an appendix, it is not appropriate to include them in the main text, as the reader can learn very little from them. It is much better to present appropriate graphs showing the results in summary form, which the reader can understand much more easily.

Examples of projects that did a good job in presenting results include those that included a video walk-through / demo of the developed software (uploaded as a video file along with the main FPR submission), and others that included a full user manual for their software.

A weakness in some projects involving software development was the lack of any screenshots or other evidence of the finished product. Be sure to include sufficient information in your report to give the examiners a clear idea of what you achieved and to show your work to its best advantage.

What technique you choose will depend upon the nature of your project, but you should be aiming to communicate your results to the reader as clearly and simply as possible.

Evaluation of results

Having performed some experiments or having produced and tested a new piece of software, the candidate needs to be able to convincingly discuss whether the results were good, and therefore whether the project has been successful in achieving its aims.

This year we were gratified to see various projects that displayed considerable thought and effort in these areas. For example, some projects used test and control sets of users to measure the difference in performance of a task between users who used the developed software to help them, compared with those who did not.

Some other projects used focus groups of people who had used the developed app, to gather honest feedback and suggestions for improvements.

There are many ways in which evaluation can be done, but it is an important aspect of most projects, and should be thought about carefully when devising the project plan.

Discussion and conclusions

It is very common that projects don't turn out as originally planned, and that the candidate is unable to achieve everything that they had originally envisaged. This is quite normal, and does not in itself prevent the project from receiving a high mark. In these cases, the examiners will take into consideration the extent to which the candidate could have reasonably foreseen any problems that arose, how they planned for things not working,

and how they dealt with unforeseen problems if and when they arose.

In the discussion and conclusion sections at the end of the report, the examiners are looking for evidence that the student has a realistic appreciation for what has been achieved, and the strengths and weaknesses of the work. A common problem in this year's reports (as is the case every year) was that candidates who did not achieve great results present an overly rosy assessment of what they achieved. Candidates whose projects did not work out as planned will gain credit for providing an honest and realistic assessment of the strengths and weaknesses, of what went wrong and of what could be improved upon if they were to do the project again. Candidates who gloss over obvious weaknesses in their work, or show no awareness for the weaknesses, tend to be marked down.

In general, the 2017–18 FPRs spanned a very wide range of standards, from the weak to the truly outstanding. The preceding comments have highlighted some of the common problems. Further advice on how to produce a good FPR can be obtained in the following ways:

- Read the CO3320 Project Subject Guide.
- Look at examples of good projects from previous years in the Project Library section of the VLE.
- Discuss problems and questions with fellow students on the Discussion forum of the CO3320 Project page on the VLE.

Below is a pie chart showing the distribution of interim grades (see Appendix E in the Regulations for Assessment criteria) for the FPR in 2017–18.