Writing a capstone project report

Goals of this document

To summarise useful advice for writing and structuring your skripsie report. Most of the advice here overlap very strongly with the content from Sharon Goldwater's website as well as my own notes:

- http://homepages.inf.ed.ac.uk/sgwater/writing/content.html
- http://homepages.inf.ed.ac.uk/sgwater/writing/ug4_report.html
- https://www.kamperh.com/notes/writing

Read it! The things I write below are simply guidelines; there are very few hard rules when writing your report. Also, I might be wrong about certain (maybe many) things, so if you have different insights, different experiences, or other things that help you with writing your report, let me know. Here is another helpful link, which is more directed at papers, but still relevant:

- http://approximatelycorrect.com/2018/01/29/heuristics-technical-scientific-writing-machine-learning-perspective/

Structure of your report

Below, I give rough chapter headings. Again, the structure of your report will be different depending on your specific project, how you approach it, and what you focus on specifically. For instance, you might not have a "Background" chapter, but incorporate required background literature at the start of each content chapter; this is especially the case if you had several separated components or aspects that you considered.

In general, I think it is important to keep things concise, keep the reader interested, and have an overarching storyline. The report should start at a high level, become more detailed as you progress to the middle (the meat), and then become more higher-level again towards the conclusion. Every chapter, and even every subsection, can follow this pattern.

Abstract

Super important! The abstract needs to summarise the entire project. There should be enough details so a reader would know whether it is worth reading your project or not. Read this: https://users.ece.cmu.edu/~koopman/essays/abstract.html. A lot of it overlaps with the introduction (below), but here you need to say *only* what is necessary to make those points (i.e., you will be super concise).

Introduction

Should answer the question: *What did you do and why should I care?* You can do this by describing the following:

- What is the problem? What is/are the goal(s) of your project?
- Motivate why this problem is important.
- Describe how you solved the problem.
- Your contributions: What did you achieve? What are the results? Without giving full details, you can give the main conclusion of the work.

Background

- Describe related work, i.e. cite relevant literature.
- Keep things fairly general (but more details than in the Introduction).
- Very important: *Have an argument!* Don't just list a bunch of things that are related to what you did, but rather have a point (otherwise it gets extremely boring to the reader).
- Unpack the motivation from the Introduction a little.
- Setup the reader for the next chapter, which describes your approach.

Model description / System design

I sometimes like calling this chapter "Model for solving problem X" or "System design for task X". This chapter should give all the details of your system or model. How did you tackle your problem? But don't give all the implementational details (see comments below); you can give some implementational detail, but only those that are necessary to understand the system or model.

Experimental setup / System implementation

In the previous chapter, you described your model, but you left out details of, for instance, what specific data set you used, or what programming language, tools, or packages you used to implement your software. All these details should be given in this chapter. After this chapter, the reader should be able to replicate your model or system (more-or-less).

Experiments / System evaluation

Have a story, and only include results that contribute to the point that you want to make. It is far better to have fewer experiments that make a strong point, than many experiments without a goal (this will simply overwhelm the reader).

Summary and conclusion

The conclusion is sometimes the most difficult to write, since in some way, you don't want to mention anything that you have described in detail before, but you also don't want to write anything that you haven't stated before. Really, you want to summarize the most important points: remind the reader of some awesome things that happened through the course of your report. There shouldn't be any new surprises here or new revelations that you haven't talked about elsewhere.

I sometimes use this as a rough guideline (this is not a strict rule at all, but it helps): if the examiner reads *only* your Abstract, Introduction and Conclusion, they should have enough details to actually know (almost) exactly what you did, how you did it, and what you achieved (i.e., they should be able to give you a rough mark).

A disconnected list of writing tips

Below is a rant of things that students often do wrong. Many of these are specific to Latex, but go through them even if you plan to use Word.

- Choose either British or American spelling, and stick to it. Also, tell your supervisor.
- If you are using LaTex, learn to use tabularx and booktabs to make pretty tables.
- If you are using LaTex, break up your paper into separate .tex files. Then in the main file, use \input to pull in the separate .tex files. This also helps with version control and collaborative editing.
- With whatever you do (spelling, figures, table layout, capitalisation), be consistent!
- Abbreviate references consistently. Do not trust bibtex files you obtain from Google Scholar or applications like Mendeley: you need to go through these to make sure they are correct.
- Try to put tables and figures at the top or bottom of a page. This is not a hard rule. But a hard rule I do use is that I do not split two paragraphs within a section with a figure. This means that, if the figure is not at the top or bottom, it will be in-between the end of a section and the heading of the next section.
- If you are using LaTex, put each sentence on a new line. Even partial sentences can be put on newlines. This is useful since if you then do a diff later, you only see the individual sentences that changed (not whole paragraphs). This is especially useful if there are multiple authors of a paper and you are editing collaboratively using a central (git/svn) repository. Even with overleaf I think this is useful.
- Make sure that .pdf, .aux, .bbl, etc. files (anything that is not source or figures) are not included in your repo. Learn to use .gitignore files.
- Some LaTeX editors can really break collaborative editing on a common repository. If a file is open in the editor, and you pull in the command-line, the files in the editor are sometimes not reloaded. When you then save the file, you overwrite any changes that you have pulled in from the repository.
- Figure captions are always full sentences. The caption should be as self-contained as possible, i.e. it should be possible for the reader to get a fairly good understanding of the figures simply by reading the caption. This means that "Model diagram" or "Illustration of the AwsmeSrc approach" are not good captions.