Level 4 project, part II

1 Feedback response

Most important

How much free beer? :-)

2 Things to do today

- Download the project skeleton template. This has:
  - Dissertation template
  - A suggested file structure
- Download the Hall of Fame projects and read them.
- Draw out a mind-map of your project.

3 How to act professionally

Professional conduct

10% of your project grade is professional conduct, as observed by your supervisor. This is a reasonable part of your grade.

But more importantly, a focus on effective professional conduct will help you make steady progress in the project, and provide useful training for your future career.

Professional conduct is a key part of the project. Do not neglect it.

Independence and leadership

You must demonstrate **leadership** and **independent thought**. This is one of the key skills we look for in assessing professional conduct. Doing what you are told is *not* enough.
This means you need to:

- Come up with ideas. **Not** *"What should I do?"*
- Work out how to implement them. **Not** *"How should I write this class?"*
- Follow them through. **Not** *"I started, but it wasn't working, so I stopped"*
- React to changes yourself. *"I started, it didn't work, so I came up with a new strategy"*

You should work *with* your supervisor, but you are the only one doing the work.

Time management

You will have to plan and manage your own time. You are assessed on your ability to get work done, and to manage the project process. You should:

- **record the time you spend**. This is **mandatory**.
- **aim to make steady progress**, and not rush everything at the end.
- **be clear about where you are** in the overall project plan at any given time; what remains to be done; how long it is expected to take.
- **be agile enough** to adjust the project to fit time constraints as they occur; cut out features sensibly as time runs short, or take on more ambitious tasks if you make fast progress.
- **be able to prioritise tasks** so that you complete your project without jeopardizing other modules.

Your supervisor should not have to manage your time, but they can give you useful feedback on how realistic your plans are.

Managing your time and balancing constraints

- Set an outline timeline at the start, with a few milestones. You can always change it. Weeks will go by much faster than you think.
- Consider how to manage tasks and milestones. Some people like tools like **Trello** to manage upcoming tasks.
- Aim to make steady progress. Make sure you don't waste weeks, and also avoid burning out with super-intense weeks.
- Monitor time spent and keep on top of where you are in the project timeline.
- Plan for risks to the timeline ("I'm not sure if the algorithm scales to 10M data points")

Time in the project

- You should spend around 15 hours each week on the project during term time.
- During Week 1 and Week 10–12 you should be spending 40 full hours each week [full time].
  - *If you are doing joint Honours, then halve those figures.*
- **Some students don't do this. Some students do way too much**
- You are supposed to produce a 40 credit "worth" output. 1 credit = 10 notional hours. You will be judged against this standard.
- Monitor how long you spend on the project. Don't get bogged down with assessed exercises. Don't kill yourself working on your project.

Using a project log to steer yourself

You are **required** to log the time you spend on the project. The log:

- will be submitted with your dissertation
- should be kept under version control with the rest of your project
- is taken into account when deciding your professional conduct grade
- failing to keep the log will count against you when assessing professional conduct.

Logging practice

- You have to conform to the format for the time log.
- You have a template timelog in the project template
- `/timelog.md`

This is a plain text Markdown file. Follow the format precisely.

Example

```
## Week 1

### 19 Oct 2019
* *4 hours* Read the project guidance notes
```

```
* *0.5 hour* Created GitLab repository and cloned the cookiecutter for the projects
* *1 hour* Modified dissertation template, compiled

## 20 Oct 2019
* *1 hour* meeting with supervisor
* *2 hours* writing initial version of test harness


## Week <number>

### <Day> <Month> <Year>
* *<duration to half hour>* Short description
* *<duration to half hour>* Short description
* ...
```

Logging

Log the hours you spend on the project in a formal, written log, hour-by-hour, as you do it. *Not afterwards.* **At *least* up date the log every day. This should take no more than a minute or two.**

- It is easy to show your supervisor what you've done.
- You can keep on top of your time allocation.
- You can see what you did when you write up.
- You can balance project work against other constraint (assessed exercises, part-time work, etc.)
- If there is any question if you have done the work in the project, the log will be evidence.
- Professional engineers in other fields *have* to keep time logs, and it is good discipline.

Logging time

You should be spending ~15 hours a week on your project during term time.

- If your log shows you are spending 5 hours/week, you'll store up a nasty surprise for later.
- If your log shows you are spending 100 hours/week, you'll end up in hospital.

4 Meetings

The purpose of meetings

- They are an opportunity for you to get advice and direction.
- You can report progress, discuss new ideas, help prune down options, go over technical details, ask for suggested references.
- Meetings are short: around 30 minutes a week. You will need to make the most of the time you get.

Not the purpose of meetings

- Your supervisor will not debug your code (probably). They can provide high level advice.
- Don't ask what to do. Have options prepared to discuss and ask for feedback.
- Don't ask what grade you will get, because your supervisor will not be able to give you a reasonable answer.
- It's fine (and a good thing) to ask for for more general feedback on your progress.

Good practice

- Be prepared.
- Don't waste time; or let your rambling supervisor waste your time :)
- Be clear, precise, honest and to the point.
- Have a goal for each meeting – know what you want to get out of it.
- Don't make excuses, and never deceive. State things as they are.
- Reflect on your progress. Think about where you might need help.
- Don't expect to be told what to do.

Slides

- Some supervisors will ask you to prepare documents (e.g. a status report, questions).
- Usually this should be submitted 24 hours in advance of a meeting
- You may be asked to present a short presentation at the start of each meeting
- e.g. three slides: what did you achieve, what questions do you have, what is your plan

Basic meeting organisation

- You should have the equivalent of 30 minute meetings each week with your supervisor.
  - This could be a weekly half-hour, or fortnightly hour, or even biweekly 15 minutes. Speak to your supervisor.
- Your supervisor might hold regular group meetings with other students, instead of regular individual supervisions. If this is the case, you can ask to arrange ad hoc individual meetings if you need them.
- Agree a regular schedule with your supervisor and attend meetings on the schedule.
- Typically, meetings at the start of the project will last the allotted time, but will get shorter as the project runs. Don't feel you have to fill the available time.

Rescheduling

- Supervisors and students will need to reschedule or cancel meetings from time to time, but try and rearrange/cancel with reasonable notice (ideally 24 hours).
- If you do cancel a meeting one week, send an email status report.
- **Don't miss more than one consecutive meeting.** This is a cause for concern, and it also will trigger the automatic Tier 4 warnings.
- It's worth going to a meeting even if you feel you are getting nowhere.

Status, questions, minutes, plan

There are records that you can use to make meetings more effective. **You will be expected to use these.** No-one will check up on you, *but you can be assessed on it* – it is in the professional conduct guidelines. The key things you should have at every meeting are:

Pre-meeting

- Status report
- Questions

Post-meeting

- Minutes
- Plan

Status

- Submit a *written* status report reporting what you have done that week. Send this to your supervisor 24 hours in advance of a meeting. **Refer to the plan/minutes from last week in this report.**
  18.04.2017 This week I have:
  Reviewed the metric calculation process to see how ID information can be considered.
  Performed evaluation of the system using Mask R-CNN only, Mask R-CNN + The Condensation Algorithm and Mask R-CNN + The Condensation Algorithm + Re-identification Network.
  Wrote a draft of the Proposed System section of the dissertation (see attached).

Questions

Come prepared with a *written* list of questions to ask

> I'm having trouble with NaNs in my network. Do you have any suggestions to debug this?
> I'm confused by channels in the CNN. What does this mean?
> Which model should I use for the first prototype? I was considering the quantized version, but I'm not sure if it will be stable.
> Should I describe the network architecture in the Design chapter or the Implementation chapter?

The more concrete your questions, the more value you will get.

Good questions to ask

- **Advice** on weighing up options and feedback on plans
- **Explanations** of concepts, words or ideas you don't understand
- **Help** expanding or making coherent ideas you have

Bad questions

- Anything better answered with a two minute Google search.
- Repeating yourself and ask questions you've covered before (because you didn't have records).
- Questions that are too vague to have a meaningful answer: *"what would be a good project?"*, *"will I get an A?"*

Minutes

- Take brief, dated minutes of all meetings. Record them somewhere electronically (Google Doc, GitHub, a Slack channel, or whatever). Share them with your supervisor.
- This avoids wasted time going over things that have been discussed before.

> 18.04.2017 We discussed the problems with the Arduino board.
> John suggested I look at the Teensy as an alternative.
> We discussed what sample rate would be needed, and agreed that either 400Hz or 1Khz ought to be enough.
> We ran through the plots of the frequency response, and we spotted that there was something weird in the windowing functions.
> John suggested I write up the section on processing before next week and get the 3D printed design finalised for the stethoscope head.

Plan

- Agree a short plan for the next week in a meeting. Record this electronically. Discuss the progress on the plan in the next meeting.

Plan before 25.10.2019:
Debug NaN issue.
Run experiments again, and plot classification results.
Implement the 16x model-based approach.

Managing your supervisor

- Try and keep your supervisor on topic, and bring things around if there is waffling.
- Use minutes to avoid repetition
- Your supervisor will have forgotten what your plan was last week. Bring them up to speed when you run through your status report.
- Your supervisor can't dictate sudden changes in the project. Once you've agreed what you are doing, you should pursue it unless you both agree an adjustment is needed.

Bad meeting

- vague, excuses, evasive,
- unprepared, no continuity,
- no leadership or originality, no reflection
- pointless questions

Good meeting

- precise, prepared,
- comes with new ideas, concrete questions, ready to discuss options
- realistic planning, reflective

5 Tool use

Software Tools

Part of being professional is using the right tools for the job. As a professional computer scientist or software engineer there are several tools that are absolutely standard:

Critical tools

- Version control **mandatory**
- Typesetting (LaTeX) **mandatory**
- Reference management **highly recommended**
- Build automation **highly recommended**

Tools

There may be other tools that are appropriate for your specific project. Make sure you are using modern, appropriate tools. Your supervisor can give guidance, but you should also be prepared to guide your supervisor, who might not be up-to-date with best practices.

Please don't:

- write source code in WordPad,
- try and build your software with arcane Bash scripts
- draw diagrams for your report in MS Paint,
- store versions of your project in folders named `Project_1`, `Project_1_new`, `Project_1_new_final_2`, etc. .

All of these examples are real!

Version control

You must keep your project under version control.

- Use an appropriate VCS tools
  - If you don't know a good reason to use an alternative, use Git
- **Public providers**: GitHub, GitLab, BitBucket
  - Make sure you make a *private* repository unless you are *sure* you want to make everything public.

- **Private servers**: set up your own Git or other VCS, or use an existing server (e.g. a client's)
  - Be careful with your own server. It's easy for this to go wrong. Make external backups.

  - Be careful with other people's servers. Don't break your client's repo! Make sure you engage with your client and work out whether there are branches you should work on, what their commit/PR workflow is, etc.
- Your supervisor might want to be able to `git clone <repo>; make` or similar to check your code works.
  - Expect to give access to your supervisor so they can see what you are doing

An example

One of my former students, Joe Cameron, has made his Level 4 project repository public on GitHub:

- https://github.com/JoeCameron1/IndividualProject

This is an good exemplar for a project repository, though there are many ways to organise one.

When to use VCS

- Keep your code under control.
- Keep dissertation under version control as well.
- Consider using branches to work on features in isolation
  - But don't go crazy and make a nightmare tree – this is a one person project!
  - Please don't branch dissertations. You'll go mad.
- Consider using other tools from VCS providers:
  - Issue trackers can be very helpful even if you're the only one filing issues.
  - Wikis can be a good place to keep notes, meeting records, etc.

When not to use VCS

- Do NOT check in anything to a repo unless you are sure you have permission.
- Do not check in private code or data (e.g. owned by client).
- Do not check in code that might be under NDA.
- Do not check in any personal data about other people (e.g. covered by GDPR), such as non-anonymised evaluation results
  - This includes names, addresses, IP addresses, phone numbers, email addresses, etc.
- Avoid polluting your repository with temporary files, generated code, etc.
  - Look into setting up `.gitignore` or equivalent

Make sure you have a backup

- At least on your machine, and on a remote repo, and *somewhere else*
- Ideally somewhere else too (e.g. sync to a shared filesystem in School regularly)
- **You lose your project – you get an H**. We expect you to have backups.
- Keep all data, code, dissertation backed up.

Reference managers

- You may have to work with many references in your project
  - For example, academic papers, books, and so on.
- If so, use a **reference manager** *from the very start*.
  - This can store a database of references, optionally with notes, links to the actual material, etc.
  - It can auto-generate bibliographies (e.g. export to BibTeX)
  - It can help you organise your research

Example reference managers

- **Papis** open source, fast, command line tool
- **Zotero** free cross-platform GUI tool (I use Zotero)
- **Mendeley** free (warning: makes it hard to migrate to other providers)
- **EndNote** (commercial)
- Note: BibTeX is *not* a reference manager – it just makes it easier to manage bibliography typesetting. Don't try and use it as one.

Other freely available tools

- Editing text or source? Consider **VS Code**, **Atom** or **Sublime Text** (commercial) if you don't like **vim**.
- Working on a LaTeX file? Consider **Overleaf** to allow remote editing in browser with live preview (GitHub integration)
  - The L4 project template is available as a pre-setup Overleaf template, and you can use GitHub integration.
- Tracking tasks and making plans? Consider **Trello**.
- Drawing a diagram like a flowchart? Consider **draw.io** for quick editing (GitHub and Google Drive integration)
- Creating a table in LaTeX? Use `www.tablesgenerator.com`
- Running experiments/doing analysis (in Python or R)? Consider **Jupyter notebooks** to write/test analysis scripts.

More tools

- Doing vector graphics? Consider **Inkscape** to create and edit SVG files. Or **tikz** to make diagrams directly in LaTeX.
- Editing images? Consider **GIMP** to create and edit bitmap images.
- Editing video? **DaVinci Resolve** is free (but not open source) and very powerful.
- Recording your screen? **Open Broadcast Studio** is an excellent tool to make screen-capture videos.
- Editing audio? **Audacity** is a useful open source audio editor.

- Creating notes? Consider writing them as **Markdown** for easy editing and in-line GitHub rendering.
- Convert **Markdown** to LaTeX or Word (and many other formats) with **Pandoc**

Break

Any other tools people recommend? Send them over YACRS now (with a short explanation of what the tool does).

Automation and reproducibility: Fire and Forget

Where possible, **automate** what you can.

- Use build tools (e.g. Make, automake, CMake, Maven, scons) to make builds of your software reproducible. Do this **from the start.**
- If appropriate, use continuous integration tools like Jenkins or Travis to check your builds pass tests.

Automation

- If you are doing experimental work **write scripts to do analysis**
  - You should be able to regenerate all data, visualisations and stats with a single command.
  - `make figures`
  - **DO NOT** do this by hand, tweaking parameters each time and forgetting how you made figures
    - *This is unacceptable science*
  - Jupyter Notebooks can be helpful in organising analysis (if you are using Python or R)
- Mantra: write it so someone else could build, run and analyse exactly as you did, with no risk of getting it wrong. **Don't hack a solution together that just works one time!**

Appropriate development tools

- A professional uses the right tools for the job.
- Use appropriate tools. This might be ones you are recommended by your supervisor, or you might need to do some research.
  - In some cases, you may have only one choice, in which case there isn't much to think about.
  - Avoid using out-of-date tools; this will look bad if the project is part of a portfolio. (*"I did all my version control using RCS!"*)
  - Avoid using cutting edge tools that are unstable; you might be stuck without bugfixes.
- Speak to other students. Many of you have experience of tools your supervisors will not be familiar with, and you can exploit this body of knowledge.

Dealing with clients and collaborators

- You may be working with external clients or collaborators: companies, other University departments or other organisations
- This might be a very direct link
  - For example, building a new system for a company, with an agreed specification
  - Or more informal; your supervisor works with a colleague in another School and you are contributing to a larger project
- These collaborations are often some of the best projects undertaken, and offer really valuable experience.
- **However: your interactions with clients are critical to professional conduct.**

Your project is the priority

- The purpose of the Level 4 project is to **educate you**.
  - This involves creating a product, but the product is not the end-goal.
- Watch out for collaborators, *particularly former/current/future employers* who want you to complete tasks that might not contribute to your project academically.
  - Tension between "implement this feature now" and "make the dissertation stronger"

Seek guidance from your supervisor if you have difficulty balancing this tension.

It is forbidden to do a project with a current employer without specific written permission from the project coordinator. This is a serious conflict of interest.

Reputation

- Remember that your actions with collaborators will reflect on you, your supervisor, the School and the University.
  - Unprofessional behaviour by project students can and does affect existing professional relationships.
  - On the other hand, we more often seriously impress companies and Schools with how effective our students are.
- Most important rule: **stay in communication** with collaborators. Attend meetings, keep records, follow up emails, and proactively communicate if things look like they are going wrong. This is all that can be expected of you.

When things go pear-shaped

- Occasionally relationships with collaborators go wrong during a project. This may well be caused by situations outside of your control
  - A company reorganises in the middle of a project and the contact person moves on.
  - A collaborator is unable to produce a dataset you were expecting to have.
  - A client decides to completely change what they want from a project three weeks before the deadline.
  - A collaborator stops communicating.

- You can't be blamed for a collaboration going wrong if you have followed good practice. **But** you will still be assessed on the quality of your project.
- You need to get a good project out; that has to be the priority. Plan for risks associated with collaborators.
- *Your project is unlikely to be a collaborators priority. Promises can fall through.*

Ethics

If you need to do any experiments with human subjects (e.g. test how well your software works), you must comply with ethics requirements. *If you capture data from anyone other than yourself, you must consider experimental ethics.*

- If you gather any data whatsoever from people during your project you must have ethical approval.

You are responsible for ethical approvals. Make sure you can get approval for what you need to do. Ask for advice from your supervisor if you are unsure.

- **Simple cases**, like testing a web app: *Ethics checklist* at http://www.dcs.gla.ac.uk/ethics. You must check this; sign it; have your supervisor countersign; include in the appendices of your dissertation.

More complicated ethics

- **Can't sign the checklist?** You need ethical approval from the School. Contact the School Ethics Chair (Stephen Brewster). This can usually be approved in simple edge cases, like slightly non-standard hardware or slightly increased risk.
- **School can't sign it off?** It will need to go to a College ethics panel. This will take at least a month, and you will need to provide:
  - Information sheets (what subjects will get when they do the experiment)
  - Consent forms (what subjects will sign to indicate informed consent)
  - Description of experiment
- Plan this well in advance, and seek guidance from your supervisor before making an application.

When you won't get ethical approval

If your research involves:

- children;
- vulnerable people;
- lack of active consent;
- activities that could be dangerous;
- deception;

you will likely not receive permission.

NHS

You will not receive permission from the University for any study involving the NHS (on NHS premises, involving NHS patients, NHS equipment or NHS staff).

You require NHS ethics for this – but it will take several months and a very very long form. Talk carefully with your supervisor if this might be an issue. Realistically, this won't happen in a Level 4 project.

Data protection

If you collect any personal data during your project, you will be subject to the GDPR regulations.

- The University takes data protection seriously. You must protect any data you acquire.
- "Personal data" is a very wide reaching term. It includes:
- Names, addresses, emails, IP addresses, phone numbers, student ID numbers
- Store data anonymously where possible.

GDPR guidance

- Do not store *any* personal data on cloud services (Google Drive, GitHub) except for the University's OneDrive service/CSCE storage
- Do not transfer personal data out of the EU.
- Do not capture personal data without explicit consent.
- Keep personal data encrypted.
- Remember that subjects have a right to view and delete data held about them.
- Do not include personally identifying information in your dissertation, presentation or video.

Read the University's GDPR guidance for further details.

Non-disclosure agreements (NDA)

Some projects are conducted under NDA with another organisation. These are *discouraged*, but are sometimes the only way to do a project with an interesting partner. If this applies to you:

- You must consent to this. You may have to change topics if you do not consent, but you cannot be forced to sign an NDA.
- This will **legally bind** you. Both you and the University may be financially liable if there is a leak.
- You must not talk about, distribute or publish at any part of your project without permission from the NDA signatories.

NDAs: Be careful

- Don't upload code under NDA to external sites like GitHub.
- Don't allow NDA code/data/text to move outside the UK without permission, even if apparently private or secure.

- Don't give public presentations about your work without permission.
- Don't talk to your classmates about what you are doing.
- Full-disk encrypt any mobile devices (like laptops) that might hold NDA data or code.
- Seek guidance from your supervisor.

NDA submissions

Students under NDA must follow careful procedures to submit their work for assessment.

- Speak to your supervisor.
- NDA project work must **not** be uploaded in the usual way.
- Encrypted, version-tracked USB sticks are usually used. Passwords must be sent separately and securely to markers.
- Presentations must usually be done privately in a closed environment.

Intellectual property

- **By default, you own all IP you generated during the project unless you explicitly waive that right.**
- You can commercialise, etc. at your own option.
- If this interests you, be careful around IP from other sources (e.g. background University IP, IP from collaborators)
- Some collaborators will insist you waive your rights (e.g. if working on a company's codebase). Be informed if you do so.

References

- **Your project supervisor will normally be your go-to referee when you graduate.** They will know you better than any member of academic staff and can write detailed comments on your project and performance in meetings.
- Any supervisor will be happy to write you a reference *assuming* they have something good they can say about you.
- Keep this in mind. What would your supervisor write about you that would be positive, and potentially stand out among other candidates?

Professional conduct: excerpt from the Assessment Criteria

Professional conduct varies according to supervisor style. Use your judgement to decide what aspects are relevant. But make clear to students what you expect, and how you will assess it if you deviate from these guidelines.

Good professional conduct should demonstrate independence; courtesy; organization; time management; adherence to legal and ethical guidelines; and technical project management.

Passing grades

- A1-A5: **Exceptional professionalism**. The student carried out independent work, was always prepared thoroughly, very effectively applied appropriate tools and made excellent use of your time. For high A grades (A1-A3), this should reflect the level of professionalism you would expect from a professional (if inexperienced) consultant, and be genuinely impressive on all counts.
- C1-C3: **Acceptable conduct**. A reasonable level of guidance was required with scoping required by the supervisor throughout the project. Meetings were not always useful or content had to be repeated. Use of tools was present but not clear student was using them effectively.
- D1-D3: **Barely acceptable conduct**. Significant help in planning and running project required. Unable to take forward project without step-by-step instructions. Suggestions for improvement not taken on board. Meetings were sometimes a waste of time and were missed on occasion. Very limited use of tools.

Failing grades

- E1-E3: **Unacceptable conduct**. Only a limited effort by the student in running the project. Meetings never minuted, little to no preparation for meetings, incorrect or inappropriate use of tools. Student was rude/uncooperative. Failed to follow basic professional practices (e.g. failed to get ethical clearance after being informed to do so).
- G1-G2: **Problematic conduct**. Student did not take any responsibility for the project. Student was hostile or effectively absent. No understanding of appropriate tools. Serious violations of basic practices, even after warning.

6 Status report (December)

The status report

In December, at the end of Week 12, you must submit a status report. This is a mandatory component, and the only requirement other than the dissertation and presentation. *If you don't submit a satisfactory report, this may be accounted for in your professional conduct grade.*

- This is not a heavy duty piece of work. It should take a couple of hours at most.
- It should be 1-3 sides of A4, usually 2.
- Keep it short, precise and professional.
- I would not start it until the last days of Week 12.

This is submitted on Moodle as an official record *and* emailed to your supervisor.

Contents of the status report

The report must cover:

- **Project Description**: clear description of the project, i.e. what it is that has to be done, a statement of the general problem.
- **Progress report:** what you have achieved as of time of writing
- **Plan of work:** a detailed plan of work, with deliverables/milestones, for the remainder of the project
- **Problems and risks:** any difficulties that you foresee or have experienced.

Use these as section headings. Bulleted lists are fine, except for the project description which should be prose. There is an exemplar on Moodle as well as templates. Please read the exemplar before writing your report.

7 Dissertation

Dissertation

Remember, you are working on a dissertation, not a product.

There will be a more extensive discussion of the dissertation in a future session. For now, some important guidance.

Dissertation must be logical

- You must state the general problem, without specifics, and then specialise it.
- You need to break down the problem from its abstract form into implementable parts, *showing a clear and logical train of thought*
- **Note** the abstract problem, and the train of thought, are not usually completely clear *until you have finished the project!* This is why a chronological report is not appropriate. You need to reflect on what you have done, pull out the key problem and form the argument.

Finishing the argument

At the end of this you will have:

- defined a problem;
- narrowed a problem to something implementable;
- implemented it.

You must then show that you have addressed the problem! This is the purpose of the evaluation. The evaluation must show that you have solved the problem. This is done by asking questions and seeking evidence to answer them.

An exercise

Write down the *general problem* you are solving, without regard (as far as possible) to specific technologies or paths to implementation.

- This **isn't** your project title.
- Phrase it as a question: *"How can cancer diagnosis be made more efficient by approximate clustering genome data?"*

Exercise Part II

Exchange with the person next to you. Discuss with them how you could show how you have addressed the problem.

- *"How can cancer diagnosis be made more efficient by approximate clustering genome data?"*
- **Maybe you could gather a dataset of cancer diagnoses and matched genome data, and running a clustering algorithm, and compare that with a classifier running without clustering?**

8 How to do research

How to do research

One of the first tasks in a project is to survey the background work. Your supervisor simply might tell you to "do background research". Here's a simple algorithm for research on the cheap.

The following is focused on collecting academic sources. Your project may involve other kinds of research, which we won't cover today, for example:

- requirements gathering from client interviews
- surveying and testing other existing software

Most projects will cite at least some academic literature, however.

Initial conditions

- Your supervisor might suggest some initial papers: "seed papers"
- If not, you have at least a topic.

Topic to seed papers

1. Derive some keywords from the topic.
2. Put them into Google Scholar.
3. Check the titles of papers that come back. Ignore ones that are clearly irrelevant.
4. Read the abstracts on the page and see if they talk about something to do with your project. If they are clearly irrelevant get rid of them.
5. Otherwise take a note of them (put them in a reference manager). **Use a reference manager!**
6. Skim read the papers you have taken a note of. Read the abstract, conclusion and look at the figures.
7. If it is clearly irrelevant, remove it from your reference manager list.
8. Read it a bit more carefully. Write a couple of sentence summarising the paper and enter these in your reference manager.
9. Are there any new keywords that have come up that seem relevant? Add them to your keywords to look for.

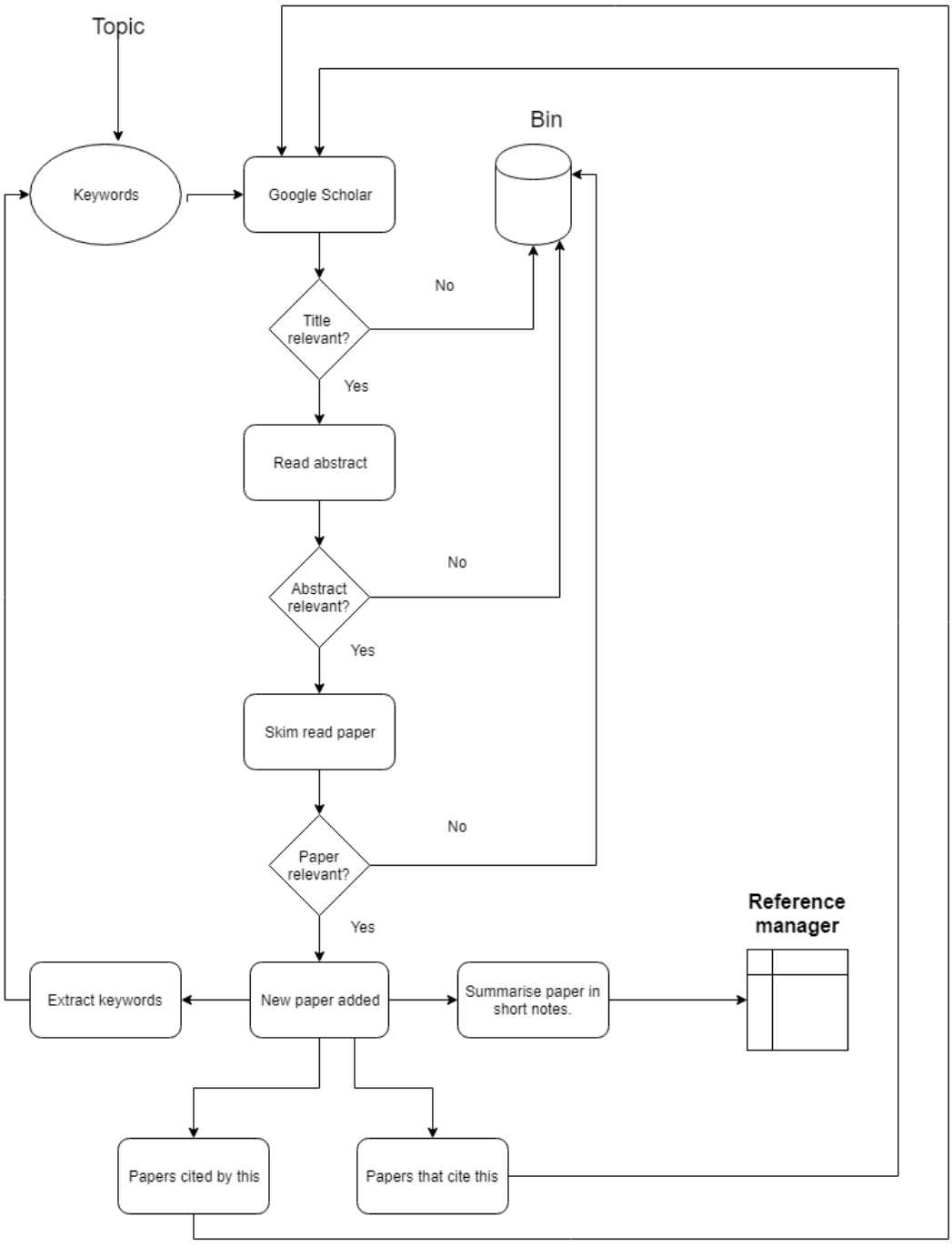

Figure 8.1: *The research algorithm as a flowchart*

Recursive search: seed papers -> all relevant papers

1. Look at all the papers that are *cited by* the paper you're looking at (backwards search) and all papers that *cite* the paper you're looking at (forwards search). Google Scholar will show you these.
2. For all of these papers, repeat the rejection process above, rejecting irrelevant papers.
3. You have a new set of papers. Read, reject, summarise the remaining.
4. Now you have about 200 tabs open in your browser.
5. Repeat this entire forward/backwards search process, using your set of papers as the new seed set. You will quickly find a set of "core papers" that really deal with the issues you are looking at and can focus your search there.
6. If you find new keywords but not direct citation links, use them to make a new initial search. You usually won't know the right language to use when you start, so you have to refine it.

Note

This is exhausting work. Make notes as you go along, use a reference manager, and take a break when you get worn out.

Summary of summer work

- You will need to email a very short summary of where you are to your supervisor by the start of next week.
- And submit this on Moodle, in the "Week 1" section.
- This should state for the record what, if any, work you have done over the summer or otherwise prior to the project starting.
- Use the template provided.
- Note that if you have done any prior work on the project (e.g. in an internship, a personal project), you must explicitly state this.

Remember: you are assessed on what you produce this year, not what you have produced beforehand.

9 Beyond the project

Taking the project forward

Your project is likely to be impressive. You should be proud of it. You should also be able to leverage it to open opportunities.

- **Networking**: establish links with employers (e.g. clients) and get a foot in the door.
- **Portfolio for employers**: show what you can do; perhaps via putting up a polished project on GitHub.
- **Marketisation**: polish the produce and put it on the App store or equivalent.
- **Commercialisation**: take it forward, start a company, or sell it to an existing one.
- **Research** set yourself up for a PhD or Master's extending the topic you worked on.
- **Publication** writing up your work either for an academic conference or journal, or publishing a blog article/series

There will be more on these topics in the next session, but think about what you might want to do with the project now, and near in mind the finished project might be something you want to show to employers.

Summary

The project is an important module. We've covered a lot of material. You don't need to absorb all of it today, but refer to this document when you are working on the project.

Action items

- Meet your supervisor.
- Download the project template from Moodle.
- Plan the project timeline, week-by-week.
- Set up version control.
- Fill in the time log and keep it up-to-date.
- Read the student guide carefully.
- Remember: each meeting, document electronically and durably:
  - minutes, status, questions, plan
- Think about the dissertation and evaluation *now*.

- Enjoy the year!

details

Summary of checklist

The rules, in summary (see link above for full details):

1. Participants were not exposed to any risks greater than those encountered in their normal working life.
2. The experimental materials were paper-based, or comprised software running on standard hardware. [not, for example, a VR headset!]
3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.
4. No incentives were offered to the participants.
5. No information about the evaluation or materials was intentionally withheld from the participants.

1. No participant was under the age of 16.
2. No participant has an impairment that may limit their understanding or communication.
3. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
4. All participants were informed that they could withdraw at any time.
5. All participants have been informed of my contact details.
6. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
7. All the data collected from the participants is stored in an anonymous form.

10 Professional Conduct Checklist

Meetings

- Did the student attend meetings regularly, and rearrange when required?
- Did the student attend meetings reasonably on time? (by academic standards!)
- Did the student come prepared to meetings?
  - With a clear status report?
  - With prepared questions to ask?
  - With demo/code ready to show (if appropriate)?
- Did the student record in some way (e.g. minutes) what was said in meetings? (or did you find yourself repeating yourself?)
  - Did you get a chance to review minutes taken?
- Was the student polite and courteous in meetings?
- Overall, did the student make good use of your time?

Independence and motivation

- Did the student take initiative in leading the work?
- Were they proactive in doing background research, exploring ideas and bringing their own thoughts to the project?
- Did the student complete the project without requiring step-by-step handholding?
- Did the student take on board suggestions intelligently?
- Did the student record the time they spent working on their project?
  - Did this correspond to reality?
- Did they honestly assess their own progress without attempting to deceive or offering excuses?
- Did the student consult you when things went wrong?
  - And make good use of advice offered?
- Did the student plan their time effectively?
- Did they make a reasonable attempt to adhere to this timeline, and manage obstacles that came up sensibly?

Professional behaviour

- Did the student follow applicable procedural guidelines or legal requirements (ethical approval, IP agreements/NDAs, confidentiality, data protection) carefully?
- If the project involved external clients, did the student interact professionally with the client?

- Were there instances where the student's professional behaviour could have reflected badly on the University, or negatively influenced collaborations you have established/are establishing?
- If the project involved collaboration, was the student active and involved in the collaboration? Or did they drop the ball or require you to keep things going?

Tools

- Did the student use appropriate version control?
- Was use of version control at an appropriate level of sophistication for the work? (neither too complex nor too simple)
- Did the student use appropriate tools for development (e.g. didn't edit their source code in WordPad)?
- Did the student use appropriate software to support development, writing and deployment (e.g. build systems, automated experimental analysis, deployment management systems) or was the project held together with hacky solutions?
- Was data stored, transmitted and managed in a secure way, e.g. with respect to data protection (only where applicable)?