CIS 400/401 Writing Guide -- Appended

The purpose of this document is two-fold: (1) To address common issues seen in the progress reports that should be corrected in the final report, and (2) To speak briefly about novel aspects of the final report and how to proactively author these aspects/sections correctly. It should be emphasized there are already "Writing Guidelines" published to the course website. This document builds upon that prior one and is a more focused (and less comprehensive) guide. However, the advice of that document still holds and some points are re-iterated as necessary:

Common Proposal Issues

Respect for Feedback

- We store the detailed markup we provide on your documents. We revisit this markup before we grade a subsequent report. When non-controversial suggestions are not heeded (*e.g.*, obvious misspellings and typos) it sends a very negative message. This should not be interpreted to mean *every* suggestion must be followed; you have ultimate control over project direction

Figures and Graphs

- In general, figures are too large and too generic. The length requirements are quite stringent, and just as textual space should not be wasted, the same holds true with images. Do not waste white space. Moreover, figures should help the reader, not simply satisfy rubric requirements. Every team has the standard "block diagram system model", but few have constructed images to exemplify/explain complex and/or subtle processes.
- So you have created a cool GUI application? That does not mean full-size screen shots are the best way to show it off. True screens shots are likely to have illegible small text. Sometimes a simplified and manually drawn "mock-up" will achieve your objectives better.
- Unless color is an absolute requirement in your graphs/figures, stick to using greyscale. Sure, its not that exciting, but it is most often the way these papers are printed and read. This means you should use different line/point styles in graphs or use different hatching patterns for filled areas.
- For many, Microsoft Excel seems to be the preferred way of producing graphs. While we would prefer you investigate tools like Matlab, GNUPlot, *etc*. -- this is not inherently problematic. However, if you go this route you must format them appropriately so that text and images do not become anti-aliased and pixelated.

Citations and Related Work

- There is a tendency to cite only conference papers, but ignore references for libraries/tools/*etc*. You don't need to cite super common things like programming languages, but elements like Weka, NTLK, that SDK you are using, and Mechanical Turk probably warrant a citation. You can point just to the generic homepage, but a white paper would be even better.
- Technical writing is not an English paper and citations should not be held off until the end of paragraphs. If you mention an author, paper, or library it should be cited immediately, even in the middle of the sentence: "We will be using the Weka [23] library for machine learning... . As West and Lee [24] exemplified in their use case..."

Academic Tone

- Do not editorialize! Technical writing is no place for your personal opinions or story. Design decisions should be justified by facts and evidence. It is the readers job to determine whether or not you were successful, not your responsibility to self reflect.
- Similarly with storytelling! Research is a destination and we are not terribly interested in the chronology that got you there. Unless your failures are a result worth mentioning in and of themselves, do not mention them. Development cycles and methodologies are just a means to get to your research destination and are also not interesting. It may be hard for you to bury those bugs you fought, that concept you abandoned, and all those meetings you had --- but this is just the reality of the situation.
- Use formal speech. There has been lots of colloquial writing and phrasal verbs (*e.g.*, showed up, figured out). Change first person to third person pronouns wherever possible.
- Be subtle in your criticism of related work. You *should* highlight ways you improve upon related systems, however this does not mean the related system is "bad", "poor", or "did not go far enough". Realize that the other authors might have had very different objectives.

Technical Detail & Organization

- Finding the right level of technical detail in the "model" and "implementation" sections can be tricky. Unless you are doing a programming languages project, code snippets are inappropriate (similar with DB schema). Specific library/function calls are on the fringe of being too low level. While some overburdened us with these details, others were too high level and vague. A competent technical person should be able to *approximate* your results/output by reading your report. Their system internals may be different, but your report should guide them through the significant design decisions.
- Similarly, there were some odd divisions where the "model" vs. "implementation" line was drawn. The "model" section should be generic. Imagine someone is re-using your general technique with a different use-case, hardware, or data set. Your model should work just as well in their paper. You should be as detailed as possible within this constraint. Mathematics, algorithmic discussion, block diagrams, *etc*. are all common. Then, in the "implementation" customize the model to your project, talk about the individual libraries, get into scalability and the like. Ask a member of the teaching staff if you need further help with this division.
- Both the "abstract" and "introduction" section should summarize the entire paper. They should also be disjoint, and it is okay if they repeat themselves. The high-points of motivation, use-cases, approach, technical challenges, contributions, and results should all be touched on. Reading just your "introduction" should be a reasonable shortcut to reading the whole paper.

Miscellaneous

- Distill your most novel progress and significant achievements into a list of "contributions." This is what you have done, do not let the reader forget them! Reference the most significant of them in the abstract. Discuss them in the introduction and conclusion. Re-reference them when they come up elsewhere in the paper.
- Machine learning is not a magic black box. If you are using machine learning, you need to enumerate and discuss the features you are using, individually and explicitly. Intuitively, why do you expect a feature to work? What simple statistics from your data set support this? Similarly, machine learning has an established set of evaluation and output metrics we expect to see. Typically, cross-validation is used to produce precision-recall curves and AUC metrics. Those with numerical learning targets have error measures.

Towards a Successful Final Report

Revisit the Specification (discussed at length in the specification)

- *Standalone report*: Do not reference prior reports. Tell your complete story in this report. Assume someone has found it randomly on the Internet. Consider the detail this individual would find relevant (and how little they care are about your personal story).
- *Ethics*: A short section on the ethical ramifications of your project. This can include those issues you actually encountered in data collection (for example, if your project involved IRB approval). For most, it will probably focus on what could go wrong in a production system.

Results Section

- This is the area which will require the most expansion relative to the progress reports. Realize also that it is the last significant section in the report. Thus, an incomplete "results" section can leave a poor and lasting impression on the reader. Every year we see reports whose results have been conducted with obvious haste. We encourage teams to start this process early, even if it means sacrificing some system functionality.
- In addition to just presenting your results (hopefully with graphs/tables), try to contextualize them. Are these "good" results? How do similar systems perform? This may involve revisiting some of the related work.

Future Work

- In this section, speculate about how one could further develop your work. Try to concentrate on large technical challenges instead of small functionality. Do not let this turn into a "things we promised in our proposal but did not complete" type section.
- For those with space constraints, or those who see limited future directions, it may be okay to integrate this section into the beginning of "conclusions".