CSC290 Communication Skills for Computer Scientists

Lisa Zhang

Lecture 6; Feb 11, 2019

Announcements

- **Code Commit** Due Sunday 9pm
- You don't have to finish the entire game by this weekend, just one commit!

Project Code Commit

Information

- Due **Feb 17, 9pm**
- Submit the github link to **one** of your commits
- You do **not** need to finish your game this week!
- Project completion deadline is the repository submission deadline March 20

Project Repository Setup

- Have one team member create a public github repository
  - https://help.github.com/articles/create-a-repo/
- Add every team member as a collaborator to the repository
  - https://help.github.com/articles/inviting-collaborators-to-a-personal-repository/

Project Commit

- Each team member will submit one substantial commit
  - Show both their participation in the project
  - Show the use of good coding practices.
- Commits should be decently sized, but not too large since each commit should be atomic
  - The commit should add or significantly modify a game functionality
  - Cannot just be a simple typo fix

Project Commit Grading

Graded out of 4 (like the blog posts) as follows:

- **1 point** for a non-trivial, atomic commit.
  - (If a commit is deemed trivial, you will receive a grade of 0/4 for this portion of the group assignment.)
- **1 point** for a commit message that follows the guidelines discussed in class.
- **2 point** for code that follows the guidelines discussed in class.

Project Commit Submission

Submit a link to a github commit page. Your link should look something like this:

`https://github.com/numpy/numpy/commit/5c0c3ebb3fb4b861ed624`

Questions?

Technical Writing

What is technical writing?

Any writing that communicates technical material.

Examples:

- User manual
- Documentation
- Help files
- Bug report
- Instructions

How is technical writing different from normal writing?

How is technical writing different from normal writing?

- Dense, difficult material that needs many sign posts and transition words to keep readers on track.
- Clarity and precision is most important.
- Often involves symbols (mathematical symbols, function names, etc) that need to be differentiate from words.
  - Symbols increase clarity, but makes the writing more dense.
  - $\forall x \in y, x \leq 3$

Guidelines for technical writing

- We'll talk about a few guidelines for technical writing.
- The most important is to **keep the reader in mind**.
  - What is the background of the reader?
  - What do the readers know so far?
  - How will they be reading your material? Will they read it cover-to-cover, or will they be skipping sections?
  - **What do they want to do after they read your writing?**

Remember the SMCR model of communication!

Worksheet

- Two versions of a paragraph from an introduction to Pygame

Before we start:

- **What does the audience want to do after they read the tutorial?**
- What can we assume about the audience, what they know, and what they don't know?
- Will the reader be skipping sections?

Worksheet

- Two versions of a paragraph from an introduction to Pygame

Before we start:

- **What does the audience want to do after they read the tutorial?**
- What can we assume about the audience, what they know, and what they don't know?
- Will the reader be skipping sections?

Work with a partner.

Separate symbols in different formulas

Symbols in different formulas should be separated by words.

- **Bad**: Consider S_q, $q < p$.
- **Good**: Consider S_q, where $q < p$.

Separate symbols in different formulas

Symbols in different formulas should be separated by words.

- **Bad**: Consider S_q, $q < p$.
- **Good**: Consider S_q, where $q < p$.

Example:

- **Bad**: After calling the function `next`, `finish` should be called.

Separate symbols in different formulas

Symbols in different formulas should be separated by words.

- **Bad**: Consider S_q, $q < p$.
- **Good**: Consider S_q, where $q < p$.

Example:

- **Bad**: After calling the function `next`, `finish` should be called.
- **Good**: After calling the function `next`, call the function `finish`.

Avoid symbols at beginning of sentences

Avoid beginning a sentence with a symbol.

- **Bad**: `print` is a Python builtin function.
- **Good**: The function `print` is a Python builtin function.
- **Good**: In Python, `print` is a builtin function.

Avoid symbols at beginning of sentences

Avoid beginning a sentence with a symbol.

- **Bad**: `print` is a Python builtin function.
- **Good**: The function `print` is a Python builtin function.
- **Good**: In Python, `print` is a builtin function.

Example:

- **Bad**: 96 students attended lecture today.

Avoid symbols at beginning of sentences

Avoid beginning a sentence with a symbol.

- **Bad**: `print` is a Python builtin function.
- **Good**: The function `print` is a Python builtin function.
- **Good**: In Python, `print` is a builtin function.

Example:

- **Bad**: 96 students attended lecture today.
- **Good**: Today, 96 students attended lecture.

Symbols at beginning of sentences (continued)

Example:

- **Bad**: $x^n - a$ has n distinct zeros.

Symbols at beginning of sentences (continued)

Example:

- **Bad**: $x^n - a$ has n distinct zeros.
- **Good**: The polynomial $x^n - a$ has n distinct zeros.

Instead of starting a sentence with `<symbol>`, annotate its **type**.

Annotating **types** of pronouns and symbols will make your text easier to follow.

Avoid passive voice

In Computer Science, use "we" to avoid passive voice.

- **Bad**: The following result can be proved by contradiction.
- **Good**: We prove the following result by contradiction.

Avoid passive voice

In Computer Science, use "we" to avoid passive voice.

- **Bad**: The following result can be proved by contradiction.
- **Good**: We prove the following result by contradiction.

Example:

- **Bad**: The design principles should be followed whenever code is written.

Avoid passive voice

In Computer Science, use "we" to avoid passive voice.

- **Bad**: The following result can be proved by contradiction.
- **Good**: We prove the following result by contradiction.

Example:

- **Bad**: The design principles should be followed whenever code is written.
- **Good**: We should follow the design principles whenever we write code.

Left-to-right

Readers read left-to-right, your sentences should be understandable.

- **Bad**: We prove that `<grunt>` and `<snort>` implies `<blah>`.

Why?

Left-to-right

Readers read left-to-right, your sentences should be understandable.

- **Bad**: We prove that `<grunt>` and `<snort>` implies `<blah>`.

Why?

- We prove that `<grunt>`
  - *oh! okay!*

Left-to-right

Readers read left-to-right, your sentences should be understandable.

- **Bad**: We prove that `<grunt>` and `<snort>` implies `<blah>`.

Why?

- We prove that `<grunt>`
  - *oh! okay!*
- . . . and `<snort>`
  - *oh, so we're proving two things!*

Left-to-right

Readers read left-to-right, your sentences should be understandable.

- **Bad**: We prove that `<grunt>` and `<snort>` implies `<blah>`.

Why?

- We prove that `<grunt>`
  - *oh! okay!*
- . . . and `<snort>`
  - *oh, so we're proving two things!*
- . . . implies `<blah>`.
  - *wait, let me go back and read again. . .*

Left-to-right

Readers read left-to-right, your sentences should be understandable.

- **Bad**: We prove that `<grunt>` and `<snort>` implies `<blah>`.

Why?

- We prove that `<grunt>`
  - *oh! okay!*
- . . . and `<snort>`
  - *oh, so we're proving two things!*
- . . . implies `<blah>`.
  - *wait, let me go back and read again. . .*
- **Good**: We prove that the two conditions `<grunt>` and `<snort>` imply the result `<blah>`.

Left-to-right . . .

- **Bad**: We want to prove that `<grunt>` and `<snort>` are not true.

Left-to-right . . .

- **Bad**: We want to prove that `<grunt>` and `<snort>` are not true.
- **Good**: We want to disprove the claims `<grunt>` and `<snort>`

Again, annotating the **types** of things can help

- the *claim* `<grunt>`
- the *function* `print`
- the *polynomial* $x^n - a$

More on annotating types

- **Bad**: Some say that waiting until the last minute helps them work quickly, but *this* has pitfalls.

More on annotating types

- **Bad**: Some say that waiting until the last minute helps them work quickly, but *this* has pitfalls.
- **Good**: Some say that waiting until the last minute helps them work quickly, but *this approach* has pitfalls.

Avoid Jargon

Whether a terminology is appropriate or jargon depends on the *audience*. We need technical terminology to describe specific ideas. Ask yourself whether that specificity is worth trading off readability.

Example:

- **Bad**: The patient is being given positive-pressure ventilatory support.

Avoid Jargon

Whether a terminology is appropriate or jargon depends on the *audience*. We need technical terminology to describe specific ideas. Ask yourself whether that specificity is worth trading off readability.

Example:

- **Bad**: The patient is being given positive-pressure ventilatory support.
- **Good**: The patient is on a respirator.

Example:

- **Bad**: This program requires too many CPU cycles to complete.

Avoid Jargon

Whether a terminology is appropriate or jargon depends on the *audience*. We need technical terminology to describe specific ideas. Ask yourself whether that specificity is worth trading off readability.

Example:

- **Bad**: The patient is being given positive-pressure ventilatory support.
- **Good**: The patient is on a respirator.

Example:

- **Bad**: This program requires too many CPU cycles to complete.
- **Good**: This program is too slow.

Define the unfamiliar

If you are defining a new term, *highlight* its first occurence, and define them right away.

Choose singular over plural noun

Singular nouns are more clear:

– **Bad**: Lexical analyzers translate regular expressions into nondeterministic finite automata.

Choose singular over plural noun

Singular nouns are more clear:

– **Bad**: Lexical analyzers translate regular expressions into nondeterministic finite automata.

– **Good**: A lexical analyzer translates each regular expression into a nondeterministic finite automaton.

How will you know if a single lexical analyzer translates one expression or many?

What is wrong with the following sentence?

Alice told Mary that her code does not compile.

What is wrong with the following sentence?

Alice told Mary that her code does not compile.

Alice told Mary that **her** code does not compile.

What is wrong with the following sentence?

Alice told Mary that her code does not compile.

Alice told Mary that **her** code does not compile.

- Alice told Mary, “Your code does not compile.”
- Alice told Mary, “My code does not compile.”

What is wrong with the following sentence?

It says that ambiguous references are bad.

What is wrong with the following sentence?

It says that ambiguous references are bad.

It says that ambiguous references are bad.

What is wrong with the following sentence?

It says that ambiguous references are bad.

It says that ambiguous references are bad.

- The slides say that ambiguous references are bad.
- The notes say that ambiguous references are bad.
- The instructions say that ambiguous references are bad.

What is wrong with the following sentence?

It says that ambiguous references are bad.

It says that ambiguous references are bad.

- The slides say that ambiguous references are bad.
- The notes say that ambiguous references are bad.
- The instructions say that ambiguous references are bad.

What about

- It says in the instructions that ambiguous references are bad.

What is wrong with the following sentence?

It says that ambiguous references are bad.

It says that ambiguous references are bad.

- The slides say that ambiguous references are bad.
- The notes say that ambiguous references are bad.
- The instructions say that ambiguous references are bad.

What about

- It says in the instructions that ambiguous references are bad.

Not as preferable, because the reader has to wait for the subject of the sentence.

Structure

Each paragraph should have a clear purpose.

Give plenty of examples for new definitions.

Use lists, tables, and other structures when appropriate.

How to begin

Your opening paragraph should be your best paragraph. Its first sentence should be your best sentence.

Avoid starting sentences of the form "An x is y."

- **Bad**: An important method for internal sorting is quicksort.
- **Good**: Quicksort is an important method for internal sorting, because . . .

Example:

- **Bad**: A commonly used data structure is the priority queue.

How to begin

Your opening paragraph should be your best paragraph. Its first sentence should be your best sentence.

Avoid starting sentences of the form "An x is y."

- **Bad**: An important method for internal sorting is quicksort.
- **Good**: Quicksort is an important method for internal sorting, because . . .

Example:

- **Bad**: A commonly used data structure is the priority queue.
- **Good**: Priority queues are significant components of the data structures needed for many applications.

Highlight Important Information

We made a bunch of changes: The registration chapter has been split in two, between adding registration and then associating users with objects. The chapter was giant before so this makes it more managable. Screenshots of the admin have been updated to reflect the new style. The few minor typos have been fixed. Updated the version of django-registration-redux that we use to 1.3. Last but not least, the Introduction has been updated.

Alternative

We made a bunch of changes:

- **The registration chapter has been split in two**, between adding registration and then associating users with objects. The chapter was giant before so this makes it more managable.
- **Screenshots of the admin have been updated** to reflect the new style.
- **The few minor typos** have been fixed.
- **Updated the version** of django-registration-redux that we use to 1.3.
- **The Introduction has been updated.**

Worksheet Page 2

1. Separate symbols in different formulas using words.
2. Avoid beginning a sentence with a symbol or number.
3. Avoid passive voice.
4. Start with the most important information.
5. Write your sentence to be understandable left-to-right.
6. Avoid vague pronouns, and “type annotate” when possible.
7. Avoid jargon.
8. Choose singular over plural noun.
9. Use paragraphs effectively, and start each paragraph with a topic sentence.
10. Highlight important information.

References

Much of the lecture content is from Donald Knuth's "Mathematical Writing" notes.

`http://jmlr.csail.mit.edu/reviewing-papers/knuth_mathematic`