CSC290 Communication Skills for Computer Scientists

Lisa Zhang

Lecture 7; Oct 28, 2019

Announcements

More Design Review Presentation this week!

Attendance will be taken at the *beginning* of the tutorial. If you are late, you will be marked as absent. (Folks wandering in late have been distacting last week)

The feedback is meant to help you with your final presentation, and other presentations that you will deliver in your career.

Code Commit

- Due last night
- Key insight:
  - Writing good commit messages is a process
  - Writing good code is a process
  - One you share something with others, it is difficult to go back and correct
  - **Fix things before you push your code!**

Critical Review

- Still slower than expected due to benchmarking.
- Critical Review Edits deadline will be pushed back by at least one week.

Mid-course Survey

Thank you for your responses!

Here's some of what you said, and how I plan to make the course better for you:

Mid-course Survey

Thank you for your responses!

Here's some of what you said, and how I plan to make the course better for you:

- "I feel like [the course is] going well, although it does seem like there's a lot happening at once. . ."
  - Agreed. The deadlines were tighted this term because I moved the *Critical review* deadline to week 4 to get another tutorial in to help you out.
  - The deadlines in the rest of the course should be more spread out, assuming you *start early*

Mid-course Survey: Activity

This year, I am trying to incorporate activities that use the features of the room. How effective do you think are the classroom activities?

- Responses were mixed
- "LESS WORKSHEETS! MORE ACTIVITY!"
- "If the table is actually willing to participate, quite effective"
- Lecture conciseness depends on paritcipation

Mid-course Survey

What do you want to see more of?

- "More on how to get hired. . . "
  - Last two weeks of lecture! Plus, come see me during office hours!
- "I still have a lot of questions on what networking is and how to do it properly"
  - Please come to office hours! I love having these conversations to help you!
- "summary of information we need to know"
  - This is interesting, and something I'm trying to do more of. I posted a list of midterm topics.

Mid-course Survey

There are a lot of other ideas!

- "Keep some kind of engagement during the week"
- "Our pod is at the periphery of the class. Sometimes we raise our hands to answer when no one else does but our attempts go unseen."
- "Can the blog and critical review deadlines be pushed to 11:59pm instead of 9pm"
- ...

Thank you!

Midterm

The midterm is next Monday!

The test is divided into two stages:

- 60 min: Individual test
  - LEC0101: 3:10pm to 4:10pm
  - LEC0102: 5:10pm to 6:10pm
- 30 min: Group test
  - LEC0101: 4:25pm to 4:55pm
  - LEC0102: 6:25pm to 6:55pm

Accessibility midterms

If you write your midterms with accessibility, then:

- Write your individual test with accessibility
  - Start *early* if your accomodations grants you extra time
- Return to this room to do the group test by 4:25pm or 6:25pm

What's on the test?

The list of topics posted at
https://www.cs.toronto.edu/~lczhang/290/midterm.html

- Materials from weeks 1-7
- Required Readings from weeks 1-7
- Coding in Python

Sample midterms are posted on the course website.

Midterm - what to expect

- Wait outside the room while test is being set up
- You will need a **pencil** and eraser for the multiple choice questions
- You can use either a pen or pencil for the short answer questions
- Please bring your T-card for attendance
- Bags, books, coats must be **under your chair**
- Cellphones OFF and in your bags
- No student can leave until end of test time

Group test

The group test is so that you get immediate feedback on your answers from your peers. We're using the midterm as an opportunity to learn to collaborate!

- You may select your own group **of 4 or 5.**
- You **must** write with a group. If you don't have a group, you'll be grouped with others on test day.
- Groups will get ONE copy of test/answer sheet on which you must work together.

Group Midterm Strategies

- Talk to your group members ahead of time about how to handle disagreements
- Assign roles:
  - Who will be writing answers on behalf of group?
  - Who will lead the discussions?
  - etc.
- Define discussion strategies, to ensure everyone's input is heard

Technical Writing

Review: What is technical writing?

Any writing that communicates technical material.

Examples:

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

Review: 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.
  - $\forall x \in y, x \leq 3$

Review: 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!

Activity

- 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 your group!

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?

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.**

Activity 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`