Computational Linguistics (Syllabus)

Course	Info
Course#	LIN 335
Time	MW 11:00-12:20am
Location	SBS S-218
Website	many
Instructor	Thomas Graf
Email	`lin335@thomasgraf.net`
Office hours	M 2:00-3:30, W 2:00-2:30, F 9:30-10:30
Office	SBS N-249
TA	Magda Markowska
Email	`magdalena.markowska@stonybrook.edu`

1 Bulletin description

An introduction to computational linguistics for students with previous programming experience. This course explores the models, algorithms, and techniques that dominate modern-day language technology, and it evaluates them from a linguistically informed perspective. Topics include corpus-based methods, finite-state approaches, machine learning, and model evaluation techniques. Great emphasis is put on discussing the limitations of existing techniques and how they might benefit from linguistic insights. Students will also hone their programming skills and develop familiarity with state-of-the-art software packages for computational linguistics. Formerly offered as LIN 220; not for credit in addition to LIN 220.

SBC: STEM+

Courses meeting the STEM+ requirement will typically have prerequisites from the Master Quantitative Problem Solving (QPS), Study the Natural World (SNW), or Understand Technology (TECH) areas. Such courses will require students to cover material in the STEM area at a deeper level than the introductory Versatility courses.

2 Teaching goals

- expand existing programming skills
- master essential concepts and techniques in computational linguistics
- critically evaluate computational models from a linguistically informed perspective
- translate abstract computational models into fully functional source code
- develop learning autonomy and the ability to deepen your programming knowledge through self-study
- Students must use the skills expected from their Versatility courses to study and practice them in greater depth, with further study applied to the area in which they are certified.

3 Prerequisites

3.1 Formal prerequisites

C or better in at least one of the following courses:

- LIN 120
- CSE 110
- CSE 114
- ISE 108

Students who do not meet this requirement may be allowed to enroll with permission of the instructor.

3.2 What's required?

This course assumes that you have already mastered essential programming concepts, including

- variables
- basic data types (strings, integers, arrays/lists)
- control flow (`if-else`)
- loops (`while`, `for`)
- custom functions

Even though the course uses Python as its programming language, you do not need to have prior experience with that specific language. The basic programming techniques are virtually identical across all popular languages: Python, Java, C, Perl, Ruby, Haskell, and so on.

3.3 What's NOT required?

You do **not** need any of the following:

1. Experience with intermediate or advanced programming techniques, e.g. hash maps, recursive functions, or object oriented programming
2. Programming experience with Python; the first two weeks of the course include a review of all the Python fundamentals you need to succeed
3. Knowledge of advanced mathematics, e.g. calculus or linear algebra
4. A general linguistics background; students who have taken LIN 101 or some other introductory course will recognize many concepts, but we discuss them from a very different perspective.
5. Having taken LIN 120; while this course continues LIN 120, it is also suitable for students who have previously taken a programming class on campus.

3.4 Resources

You need to have access to a computer, ideally a laptop you can bring to class.

The modern tech world is full of platforms, toolkits, frameworks, all kinds of stuff. Whereas LIN 120 provides a very sanitized environment so you can focus on the concepts, this course exposes you more to the mad jungle that is modern software development, data science, and computational linguistics. Hence there isn't a single resource you will be drawing from. Throughout the semester, you will be using

- Google Colab
- Github
- a local git install
- a local Python install and text editor
- a local diff tool
- the Linux command line
- vim (so that you can learn how to close it)

Each one of those will be introduced at specific points during the semester. We will start with Colab to keep things simple and add on complexity from there.

Also, we will use Brightspace for sending out announcements and sharing some optional materials with you (e.g. additional readings).

4 Grading

This course can only be taken for 3 credits. Student grades are determined by the following components:

1. **Assignments (60%)**
   There will be approximately 7 assignments throughout the semester. A "first pass" is due after a week, and the final solution a week after that. The grading specifics vary by assignment, but largely follow the same template: the assignment consists of multiple components, each component is P/F, and your overall grade on the assignment is determined by the number of components with a P. For the final grade, we drop the lowest

assignment and take the mean of the remaining 6.

2. **Class participation (20%)**
   Class participation is highly encouraged and can take various forms in-person and online.
3. **Pre/Post Assessment (20%)**
   Hand in the non-graded pre-assessment and non-graded post-assessment for 10% each.

There are no midterms, final exams, or final projects. On the flip side, there's no options for extra credit either.

4.1 Grading example

- Try (1 point): Do the assignment yourself; P/F depending on whether a reasonable attempt was made
- Revise (1 point): after handing in the assignment, use Chat GPT, Github Copilot, some other AI tool, or discussion with your peers to fix your prior solution and hand it in a week later; new solution graded P/F depending on whether it works correctly and the coding style isn't egregiously atrocious
- Document & Test (1 point): P/F depending on whether the fixed up solution also contains appropriate docstrings and passes pep8 (and later on, type annotations that pass a test with mypy, or possibly even unit tests or meta tests with hypothesis)
- Reflect (1 point): P/F depending on whether you also hand in a diff of the two versions, with comments on what changes were made, and why

The point range is 0-4 and maps directly to the grade letter range F-A.

- If you want a C, give it a first try and then revise with help.
- If you want a B, do everything for a C and then either reflect on the changes made or add extensive documentation.
- If you want an A, do all components.

4.2 Class participation examples

- asking questions in class
- participating during in-class discussions
- pointing out problems with the lecture notes (e.g. typos, confusing wording, broken code)
- posting in the course's Help and Support forum (this includes both asking questions of your own and answering fellow students' questions)
- suggesting topics for lectures
- posting links to relevant online material (e.g. tutorials or news paper articles)
- and much more

Note that while class participation is required, being physically present is not. If you don't think that coming to class has enough of a learning payoff, feel free to stay home. We're all adults here. But then make sure you participate via other means.

4.3 Academic integrity

- The assignments are deliberately designed so that cheating is counter-productive.
- On your first try, do the work on your own. Don't discuss it with others, don't look up solutions online, see how far you can get yourself. Making mistakes here is good, it'll make it easier for you to improve things in the second round.
- ChatGPT is a powerful tool, but use it widely. You don't learn how to run a 5k by driving the distance in your car. The more you rely on those tools, the more you deprive yourself of the opportunity to actually learn.

5 Schedule by week

The weekly schedule consists of two lectures. There is no recitation. Monday lectures focus on theory and general discussion, whereas Wednesday lectures cover the programming side of things. That said, the distinction won't always be that clear-cut.

The approximate schedule for topics and assignments is as follows. The core of the course spans the first 10 weeks.

Week	Topic
1	syllabus & big picture; Python recap
2	overview of computational linguistics; Python recap
3	n-grams, word prediction; unit tests, zip, counters
4	probabilities, data science
5	corpora, model evaluation; regular expressions
6	spellcheckers, edit distance; dynamic programming
7	tagging, stemming, lemmatization; nltk
8	Spring break
9	finite-state morphology; object oriented programming
10	syntactic analysis; spacy

The remaining weeks are earmarked for guest lectures, safety buffers, and some general skills like the Linux command line that don't directly tie into other topics but are essential for data science.

6 Policies

6.1 Contacting me

- Emails should be sent to `lin335@thomasgraf.net`
- Reply time < 24h in simple cases, possibly more if meddling with bureaucracy is involved.
- If you want to come to my office hours and anticipate a longer meeting, please email me so that we can set apart enough time and avoid collisions with other students.

6.2 Student Accessibility Support Center (SASC) Statement

If you have a physical, psychological, medical or learning disability that may impact your course work, please contact Student Accessibility Support Center, ECC (Educational Communications Center) Building, Room 128, (631)632-6748. They will determine with you what accommodations, if any, are necessary and appropriate. All information and documentation is confidential.

Students who require assistance during emergency evacuation are encouraged to discuss their needs with their professors and Student Accessibility Support Center. For procedures and information go to the following website: http://www.stonybrook.edu/ehs/fire/disabilities.

6.3 Academic Integrity

Each student must pursue his or her academic goals honestly and be personally accountable for all submitted work. Representing another person's work as your own is always wrong. Faculty is required to report any suspected instances of academic dishonesty to the Academic Judiciary. Faculty in the Health Sciences Center (School of Health Technology & Management, Nursing, Social Welfare, Dental Medicine) and School of Medicine are required to follow their school-specific procedures. For more comprehensive information on academic integrity, including categories of academic dishonesty please refer to the academic judiciary website at http://www.stonybrook.edu/commcms/academic_integrity/index.html.

6.4 Critical Incident Management

Stony Brook University expects students to respect the rights, privileges, and property of other people. Faculty are required to report to the Office of University Community Standards any disruptive behavior that interrupts their ability to teach, compromises the safety of the learning environment, or inhibits students' ability to learn. Faculty in the HSC Schools and the School of Medicine are required to follow their school-specific procedures.