Lecture 0

Syllabus

Course: Computational Linguistics 2
Course#: Lin637
Time: TR 10:00–11:20am
Location: Psychology A 144
Course Website: lin637.thomasgraf.net

Name: Thomas Graf
Email: lin637@thomasgraf.net
Office hours: Tue 11:30–2:30pm
Office: SBS N249
Personal Website: thomasgraf.net

1 Course Outline

1.1 Contents

This course serves a specific purpose in our program (see Fig. 1 on page III): it acts as the bridge from introductory courses in linguistics (Syntax 1, Phonology 1, Phonetics) and computational methods (Statistics, Mathematical Methods in Linguistics, Computational Linguistics 1) to advanced courses and seminars in computational/mathematical linguistics. In contrast to the NLP courses offered by the department of computer science, our courses focus on studying the properties of natural language from a computationally informed perspective. The question is not how computers can solve linguistic tasks, but how language can be conceptualized as a computational problem. This emphasis is also reflected in the selection of topics for this course.

- **What this course is not about**
  - Computer-assisted research methods in linguistics
  - Programming
  - Software development for natural language tasks
- **What is not covered but benefits from what is covered**
  - Speech recognition
  - OCR
  - Text generation
  - Parsing
  - Semantic analysis
  - Machine translation

- **List of topics**
  - *Phonology*
    - The role of formalization
    - String languages
    - Subregular hierarchy
    - Regular languages
    - Generative capacity of phonology
    - String transductions
    - 2-level morphology
    - Equivalence of SPE and OT
  - *Syntax*
    - Tree languages
    - Syntax is more complex than phonology
    - Mildly context-sensitive formalisms (TAG, MGs)
    - Tree transductions
    - Regular representations of MCS formalisms
    - Reinterpreting the T-model

A rough outline of the course progression is given in Tab. 1 on page III. Many of the topics we cover draw from very specialized areas of formal language theory that even most mathematicians and computer scientists do not know about, e.g. the correspondence between finite-state machines and monadic second-order logic, or the logical characterization of tree transductions. So this course might be of value to you even if you do not particularly care about natural language. Make no mistake, though, we'll talk a lot about language and linguistics — this is not a math class.

1.2 Prerequisites

The only official prerequisite is Computational Linguistics 1 (Lin 537) or comparable programming skills in Python. Python will be used to illustrate some formal concepts, and some of the homeworks will require you to implement an algorithm or procedure in Python.

It is also helpful to have some basic familiarity with linguistics (phonemes, phrase structure rules, syntactic trees) and mathematics (sets, functions, relations, and first-order logic as covered in Semantics 1, for instance). You can take an online survey to identify weaknesses, and several introductory readings on these topics are available on the course website.

Survey URL: https://testmoz.com/432409
Password: CompLing2

2 Teaching Goals

- **Practical Skills**
  - conceptualize a problem in mathematical terms

Wk	*Classes*	*Formal*	*Linguistics*
1	Jan 27, 29	What is computation?	Marr's Three Levels
2	Feb 3, 5	Formalizing phonology	Why formalize?
3	Feb 10, 12	Strictly local languages	Local dependencies
4	Feb 17, 19	Subregular hierarchy	How powerful is phonology?
5	Feb 24, 26	Regular languages	Abstractness
6	(DGFS)		
7	Mar 10, 12	String transductions	SPE-OT equivalence
8	(Spring Break)		
9	Mar 24, 26	Weak Generative Capacity	Phonology < Syntax
10	Mar 31, Apr 2	Tree languages	Headedness, feature percolation
11	Apr 7, 9	Local tree languages	GPSG
12	(GLOW)		
13	Apr 21, 23	Recognizable tree languages	GB
14	Apr 28, 30	TAG and MGs	Minimalist syntax
15	May 5, 7	Tree transductions	Reinterpreting the T-model

Table 1: Tentative course outline

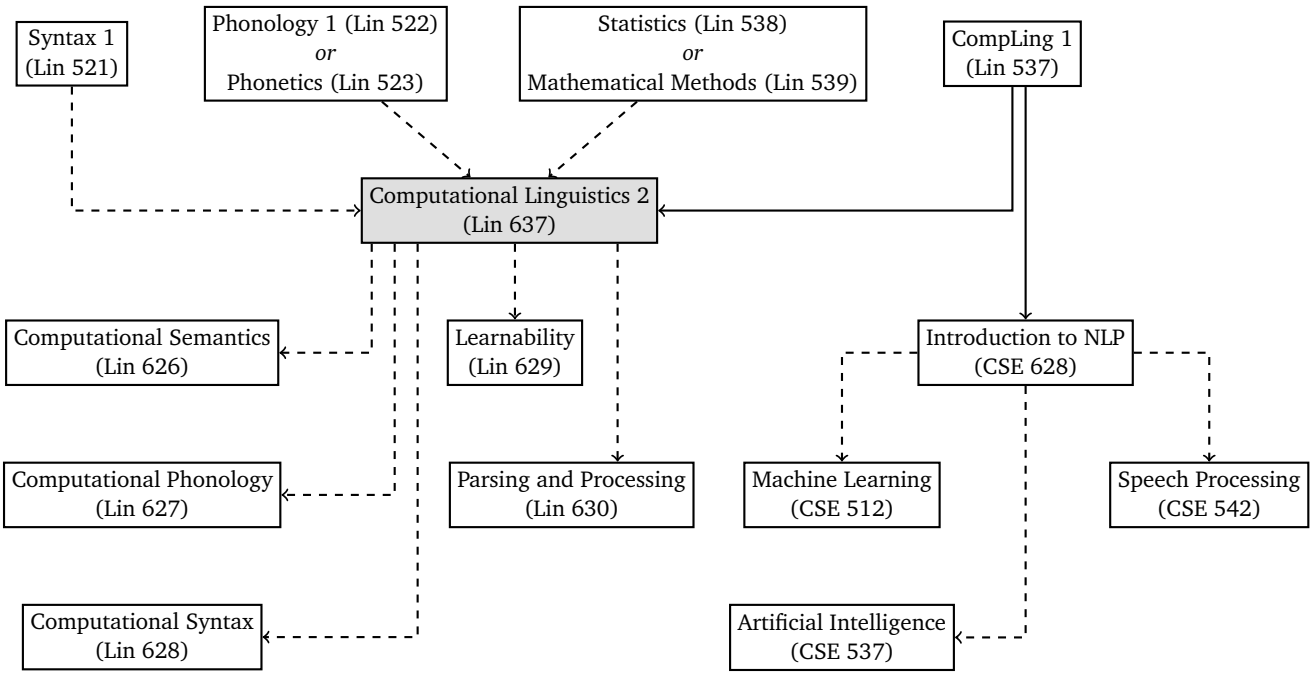

Figure 1: Computational Linguistics 2 in the curriculum (dashed lines indicate recommendations rather than prerequisites)

  - optimize your programs through the use of adequate algorithms and data structures (dynamic programming techniques, hash tables, etc.)
  - a more abstract and theoretically informed perspective on current tools and techniques in NLP
  - an understanding for how linguistic insights can be invoked to simplify NLP tasks

- **Research Skills**
  - assess linguistic phenomena from a computational perspective
  - evaluate linguists' claims about computational efficiency
  - basic overview of current research in theoretical computational linguistics
  - use computational concepts to identify new empirical generalizations
  - bring linguistic data to bear on computational claims
  - mathematically informed understanding of linguistic theories

3 Grading

- **Homework**
  - weekly exercises, programming assignments, or critical evaluations of assigned readings
  - a random sample of homeworks will be collected and graded
  - solutions will be made available online after the due date
  - Collaboration on homework problems is encouraged as long as you write up the solutions by yourself, using your own words, examples, notation, and code.

- **Readings**
  - about two readings per week
  - you have to collectively write a summary for each reading in the course wiki

- **Survey Squibs**
  - By the end of the course, the wiki should contain survey articles on a number of topics not covered in this course (or not covered at the same depth).
  - You have to pick a topic and write the corresponding survey article.
  - These articles should be succinct and simple enough that they are comprehensible to a researcher with little exposure to computational linguistics, yet at the same time include enough technical detail that the claims can be verified by somebody with the appropriate background. (Why this weird requirement? Because that's the recipe for writing a computational paper that can be published in a linguistics journal!)

- **Workload per Credits**

- *3 credits*: homework, readings, squib
- *2 credits*: homework, readings
- *1 credit*: readings
- *0 credits*: none, but I highly recommend that you at least read the assigned papers as they will be important for following the lectures

4 Policies

4.1 Contacting me

- Emails should be sent to lin637@thomasgraf.net to make sure they go to my high priority inbox. Disregarding this policy means late replies and is a sure-fire way to get on my bad side.
- 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.

4.2 Special Needs

If you have any special needs that might impact your class performance (learning disabilities, impaired sight or hearing, etc.), please come to my office hours or contact me via mail so we can make suitable arrangements.

5 Online Component

This class will use some online tools to facilitate homework collaboration and submission, student discussions, and dynamic lecture evaluation. This is the first time I'm trying something like this for one of my courses, so expect a few bumps in the road.

- **Homework submission**
  How it works: Homeworks will distributed via a github repository. You can fork this repo and upload your own code, or checkout other students' forks to see how they dealt with the problem. In order to submit a homework you upload your solution to your fork and issue a pull request. After the due date, I'll upload my solution to the repository.

  Why we do it: This setup mimics the modern workflow in collaborative development projects. Git is one of the best-known version control systems, and github is the biggest online service for hosting git repositories. Familiarity with version control systems is an essential job requirement for computational linguists, and it is also very helpful for academic work. See this discussion on Stackflow for some ideas how git can be used in conjunction with Latex: http://stackoverflow.com/questions/6188780/git-latex-workflow

  What you'll need: A github account (the free tier is enough) and a way of uploading your code to a github repository. Linux users can install git via the

command line, whereas Windows and Mac users should download and install the github app, which comes with a nice GUI.

- **Homework Feedback and Discussion**
  How it works: Every github repository comes with an issue (= ticket) tracker. If you have a question or wish to discuss a topic, you can open a ticket on the main repository (note: tickets support markdown). I will also use tickets to leave comments on your homeworks.

  Why we do it: Once again this is an essential part of modern software development. And it is also a lot more convenient than anything Blackboard has to offer.

  What you'll need: Not much beyond the ability to navigate the github repos.

- **Reading summaries**
  How it works: Every week I create an empty page on the course wiki for each assigned reading. Your job is to expand these articles into useful summaries. The default syntax for the wiki entries is markdown, a very simple *lingua franca* for writing plain text documents that can easily be converted into a variety of file formats (doc, pdf, html, and so on).

  Why we do it: Wikis are frequently used for software documentation nowadays, so you have to know how to work with one. You will also get to see the advantages of markdown, in particular for short documents that do not need a lot of fancy typesetting (this can even include your own website!). Hopefully this will move you away from formats like doc, which

  – are proprietary, and
  – lack backwards compatibility, and
  – are tedious to index and search, and
  – fail to separate the content of a document from its presentation, thereby distracting you with unnecessary details.

  Also, learning markdown takes a lot less time than relearning the MS Office user interface with each new version. If you find that markdown is too simple for your own purposes, you can also switch to a more expressive superset like pandoc.

  What you'll need: Nothing except the ability to read (and ideally write in) markdown. The github editor makes this fairly easy with its built-in help, formatting buttons and dynamic preview. You can also checkout this interactive markdown tutorial, which should take you about 15 minutes (yes, markdown is that easy!): http://markdowntutorial.com/

- **Class announcements**
  How it works: Normal announcements (readings, due dates) are put on the course website, time-critical ones (i.e. class cancellations) are emailed out via Blackboard.

  What you'll need: If you're not officially enrolled in the course, send me a message so I can add you to Blackboard.

- **Handouts**
  How it works: Handouts are made available online Monday and Wednesday

before 3pm. You can look at them on your laptop/tablet or make a hardcopy before class. I will not bring handouts to class unless I could not upload them on time the day before.

Why we do it: The department copier has a nasty habit of breaking down at the most inopportune times, and even when it is able to print other things like the stapler do not work. And most students throw away the handout anyways if they also have the PDF.