Cogs 520 - Advanced Research on Linguistics in Cognitive Science

Fall 2020, Department of Cognitive Science, METU

Preliminaries

Medium: `odtüclass` for course material, `github` for progressive assignments (a single repo, students in distinct folders), `googlegroups` for communication

Requirements: basic programming skills (preferably in one that's commonly utilized for nlp applications, e.g. python); basic linguistics knowledge (e.g. an introductory course taken before)

Bootstrapping:

- [**python**] interactive tutorials from codecademy (tutorial is for version 2.x, nearly all knowledge is directly transferable to the current version 3.x), learnpython, w3schools and geeksforgeeks; video lectures from mitopen and edx; guided projects from coursera (free for student memberships)
- [**git**] guides for git and github here, here and here
- [**linguistics and nlp**] introductory (i.e. first) chapters of *Introduction to theoretical linguistics* (John Lyons) and *Speech and language processing* (Daniel Jurafsky & James Martin); first three chapters of *Foundations of statistical NLP* by Christopher Manning & Hinrich Schütze

Resources:

- [**books**]
  - [**linguistics**] *Introduction to theoretical linguistics* by John Lyons
  - [**natural language processing (nlp)**] *Speech and language processing* by Daniel Jurafsky & James Martin; *NLP with python* by Steven Bird et al.; *Foundations of statistical NLP* by Christopher Manning & Hinrich Schütze; *NLP in action* by Hobson Lane et al.
  - [**machine learning (ml)**] *ML for text* by Charu Aggarwal; *ML with python cookbook* by Chris Albon
  - [**python**] *Learn python the hard way* by Zed Shaw
  - [**common lisp**] *ANSI common lisp* by Paul Graham
- [**lectures**]
  - [**common lisp**] algorithms for ai and nlp (pick a year, click 'slides' on the left)
- [**tutorials**]
  - [**python**] text processing from tutorialspoint, nanyang techonological uni. and geeksforgeeks
- [**documentation**]
  - [**python**] stable version 3.9.0

  - [**common lisp**] cliki: common lisp wiki; common lisp foundation; lisp-lang
- [**guides**]
  - [**regular expressions**] regxr
  - [**annotation**] PDTP v. 2.0 & v. 3.0; Penn Treebank II style bracketing guidelines, addendum and constituent tags
  - [**common lisp**] vencouver island uni.
- [**data**]
  - [**multi-language**] LDC: Linguistic Data Consortium; google n-gram; TalkBank; CHILDES; a list of multi-lingual semantic networks here
  - [**turkish**] METU Turkish Corpus (provided upon request); Turkish Discourse Bank (provided upon request); TUD: Türkçe Ulusal Derlemi; tscorpus; some other Turkish datasets are listed here
  - [**english**] CoRD: Corpus Resource Database; BNC: British National Corpus; Brown Corpus; Coca: Corpus of Contemporary American English; WordNet; FrameNet; PDTP: Penn Discourse Treebank; some other English corpora listed here
- [**libraries & tools**]
  - [**python**] nltk (natural language toolkit); numpy (scientific computing); scikit-learn (ml in python); pytorch (ml); pandas (data analysis); keras (deep learning); matplotlib (visualization); bokeh (visualization); networkx (network analysis); requests (fetching internet resources); urllib (fetching internet resources); beautifulsoup (html parsing)
  - [**common lisp**] cl-nlp: nlp for common lisp 1, 2 & 3; nlp in common lisp
  - [**nlp**] Standford NLP for English; ITU NLP Pipeline for Turkish (and a Python wrapper here); Zemberek NLP for Turkish (and two Python wrappers here and here); TRmorph for analyzing Turkish morphology
  - [**ml**] weka; tensorflow; openAI
  - [**visualization**] gephi (graph visualization and analysis); flourish; d3; observable; R graph gallery
- [**development environments**]
  - [**text editors**] visual studio code; atom; sublime text
  - [**python**] pycharm; spyder; anaconda
  - [**online editors**] jupyter; datalore; azure; kaggle; cocalc
- [**communities**]
  - [**programming**] stackoverflow
  - [**linguistics**] stackexchange linguistics
  - [**data science**] stackexchange datascience
- [**misc**]
  - pythex: check your python/re expressions if working

Program

Week 3 - Preprocessing

This week we will get familiar with how to approach any kind of text as data and common requirements for getting that data ready for further analysis, including (but not limited to) stripping, extraction, spelling and diacritic correction, elimination and normalization (case, number, proper names etc.). At this level the aim is to obtain a standardized chunk of information purged from both noise and irrelevant bits that will most efficiently yield itself to formal analysis, congruous with the problem in mind.

- [**sync.**] online tutorial, part I
  - [**programming language**] Python
  - [**input language**] Turkish
  - [**tasks**] (i) read from a text file, (ii) store data in intermediate structures, (iii) traverse chunks of input, (iv) utilize third party processes (e.g. ITU NLP pipeline) and (v) simplistic approaches (for the sake of demonstration) to certain preprocessing problems such as proper name recognition and stop word elimination, and (vi) finally write the processed data to an output file
- [**async.**] on language processing with python (ch. 1, Bird et al.); on preprocessing with python (ch. 3, Bird et al.)
- [**supplementary**] lecture slides; content included in the preliminaries, but not explicitly mentioned in the program
- [**exercises**] you can take a look at ch. 3.12 from Bird et al.
- [**assignments**] part I
  - [**programming language**] any
  - [**input language**] any
  - [**tasks**] code a basic preprocesser with a similar scope examplified in the online tutorial, which utilizes third party processes (e.g. Stanford NLP pipeline) when necessary

Week 4 - Tagging

After the language data is purged from noise (both as disturbance and redundancy) various layers of linguistic analysis might be necessary, such as part of speech tagging, morphological analysis and disambiguation, mapping syntactic dependencies, recognizing multi word expressions and/or annotating discourse relations.

- [**sync.**] online tutorial, part II
  - [**programming language**] Python
  - [**input language**] Turkish

  - [**tasks**] (i) analyze the output of part I morphologically and (ii) output the results where the associated linguistic analysis is encoded explicitly.

- [**async.**] on word level analysis (ch. 3, Jurafsky & Martin); on syntactic analysis, (ch. 12, Jurafsky & Martin); on tagged data (ch. 5.2, Bird et al. and ch. 4.3, Manning & Schütze)
- [**supplementary**] lecture slides; on in-depth parsing see ch.s 13 & 14 from Jurafsky & Martin; for parsing in python see ch. 8 from Bird et al.; content included in the preliminaries, but not explicitly mentioned in the program
- [**exercises**] you can take a look at ch. 5 from Bird et al.
- [**assignments**] part II
  - [**programming language**] any
  - [**input language**] same as part I
  - [**tasks**] (i) apply dependency parsing on the output of part II and (ii) output the results where dependency relations are encoded explicitly.

Week 5 - Structuring

Processing data (be it textual or otherwise) means structuring it according to a predefined agenda, and such a structure is usually manifold. In order to deal with a complicated data structure efficiently various aspects of parsing (e.g. delimiters, tokenization and segmentation) and data storage (e.g. flat file databases, data structures and database formats) becomes relevant.

- [**sync.**] online tutorial, part III
  - [**programming language**] Python
  - [**input language**] Turkish
  - [**tasks**] (i) read the output of part II into a dictionary and (ii) export the dictionary to a csv file while preserving the original structure of the data
- [**async.**] on tokenization and segmentation (ch. 4.2, Manning & Schütze); on storing structured data in python (ch. 5.3, Steven Bird et al.)
- [**supplementary**] lecture slides; content included in the preliminaries, but not explicitly mentioned in the program
- [**exercises**] TBA
- [**assignments**] part III
  - [**programming language**] any
  - [**input language**] English
  - [**tasks**] (i) read the output of part II into an appropriate data structure and (ii) export the dictionary to a json file while preserving the original structure of the data

Week 6 - Analysis

Testing hypotheses on structured data requires means of controlled access, formal analysis and representation of results. This week we will look at search by regular expressions, explore implicit statistical properties via frequencies, collocative relations, n-grams and moving windows and consider basic visualization techniques.

- [**sync.**] online tutorial, part IV
  - [**programming language**] Python
  - [**input language**] Turkish
  - [**tasks**] TBA
- [**async.**] on regular expressions, (ch. 2.1, Jurafsky & Martin); on collocations (ch. 5, Manning & Schütze); on n-gram models (ch.s 4.1-4.5, Jurafsky & Martin; ch. 6, Manning & Schütze)
- [**supplementary**] lecture slides; for data management in python see ch. 11 from Steven Bird et al.; content included in the preliminaries, but not explicitly mentioned in the program
- [**exercises**] you can take a look at ch. 6.6 from Manning & Schütze
- [**assignments**] part IV
  - [**programming language**] any
  - [**input language**] English
  - [**tasks**] TBA

Week 7 - Modeling

Now it is time moving on to the project proposals, defining central problems, engineering hypotheses and choosing appropriate data sources.

- [**sync.**] in-class discussion on project proposals
- [**async.**] accesing text corpora and lexical resources (ch. 2, Steven Bird et al.)
- [**supplementary**] *Markup systems and the furue of scholarly text processing* by Coombs et al. (1987); *An overview of empirical NLP* by Brill & Mooney (1997); *Data preprocessing and intelligent data analysis* by Famili et al. (1997); *Data preprocessing for supervised learning* by Kotsiantis et al. (2006); *Overview and semantic issues of text mining* by Stavrianou et al. (2007)
- [**exercises**] you can take a look at ch. 4.5. from Manning & Schütze
- [**assignments**] none

Other (optional)

When you need data in the wild one way of getting it is scraping from web.

- [**sync.**] online tutorial, part V
  - [**programming language**] Python
  - [**tasks**] scrape from X (TBA, with regular urls) where X is a well-structured and preferably small web page: (i) fetch html sources, (ii) parse them, (iii) download coexisting files (e.g. images) and (iv) generate a database from scratch
- [**async.**] TBA
- [**supplementary**] TBA
- [**exercises**] TBA
- [**assignments**] part V
  - [**programming language**] any
  - [**tasks**] code a periodic scraper that fetches from Wikipedia main-page everyday at 10:00 am

Notes

Semantics and advanced modeling approaches are left out of scope; but project specific guidance will be provided if necessary. For weekly tutorials, exercises and assignments additional resources will be announced if necessary. Chapter numbers are taken from Lyons ed. 1, Jurafsky & Martin ed. 2, Bird ed. 1, and Manning & Schütze ed. 1.