Ling/CSE 472: Intro to Computational Linguistics

September 30, 2004

Chapter 1

Introduction, overview

Overview

- What is computational linguistics
- Who’s here
- Syllabus
- Showing a computer who’s boss
- Preview: Regular expressions

What is Computational Linguistics?

- Everything you'd need to know to build the language interface part of HAL. What would that be?
- Processing language on computers:
  - For practical applications
  - For linguistic research

Practical applications (1/2)

- Speech recognition
- Speech synthesis
- Machine translation
- Information retrieval
- Natural language interfaces to computers
- Dialogue systems (e.g., airline flight status)

Practical applications (2/2)

- Computer-assisted language learning (CALL)
- Grammar checkers
- Spell checkers
- OCR (optical character recognition)
- Hand-writing recognition
- Augmentative and assistive communication
- …

Linguistic research

- Searching large corpora for patterns, examples
- Creating structured databases of linguistic information for typological research (e.g., Autotyp)
- Modeling human linguistic competence and performance
  - Computational psycholinguistics
  - Grammar engineering (Matrix)
- Software to facilitate language documentation (Elan, FIELD, GOLD, Montage)

Statistical v. symbolic methods

- A hot topic in the field these days
- Statistical methods involve *training a stochastic model* on a body of data so it can predict the most probable outcome/category/etc for new data.
- Symbolic methods involve *knowledge engineering*, or hand-coding linguistic knowledge and then applying that knowledge to tasks.
- Statistical methods provide *robustness*.
- Symbolic methods provide *precision*.
- Statistical and symbolic methods can be combined.

What is Computational Linguistics?

- Everything you'd need to know to build the language interface part of HAL. What would that be?
- Processing language on computers:
  - For practical applications
  - For linguistic research

Goals of this course

- Familiarity with computational linguistic resources and how they are applied in research in computational linguistics and other subfields.
- A rough sense of the state of the art (what can we do with language on computers anyway?)
- Ability to conceptualize problems from the perspective of computational linguistics.

Syllabus

- On the web page: courses/ling472
- Slides will be posted (often before lecture)
- Requirements:
  - 6 Homework assignments: 45%
  - Midterm: 20%
  - Final (paper, project, or exam): 30%
  - Class participation (incl. EPost): 5%
- Lab meetings (Fridays)

Who's here

- A good class to work together – everyone brings different skills
- This is fabulous/brilliant/extra!
- I'm going to bring a lot to this class because...

Letting the computer know who's boss (1/2)

- Computer 'literacy' is really a combination of experience and attitude.
- Experience gives you the answers to many questions and a sense of what the possible space of answers is.
- The important attitude boils down to confidence in one's ability to find the answer to a new question.
- There are always new questions because:
  - the technology is always developing
  - there is too much for any one person to know it all

Letting the computer know who's boss (2/2)

- Keep in mind:
  - It's always obvious once you know the answer.
  - All pieces of software were designed by some person or people with some functionality in mind.
- Places to look for answers:
  - on-line documentation (man, info, help)
  - product websites (esp. discussion forums)
  - Google: websites, and especially newsgroups
  - off-line documentation (i.e., books)
- Work together!

Administrivia

- Office hours
- WebQ on course website

Preview: Regular Expressions

- Formal languages
- Regular languages/regular expressions/FSA
- Formal definition of regular languages
- Regular expressions and search
- Perl regular expression syntax

Formal languages

- From the point of view of formal languages theory, a *language* is a set of strings defined over some alphabet
- The *Chomsky hierarchy* is a description of classes of languages.
- Languages from a single level in the hierarchy can be described in terms of the same formal devices.
- *Regular languages* can be described by *regular expressions* and by *finite-state automata*.
- Regular languages $\langle$ context-free languages $\langle$ context-sensitive languages $\langle$ all languages

Three views on the same object

- Regular language: a set of strings
- Regular expression: an expression from a certain formal language which describes a regular language
- Finite-state automaton: a simple computing machine which accepts or generates a regular language

Formal definition of regular languages: Symbols

- ϵ is the *empty string*
- ϕ is the *empty set*
- Σ is an alphabet (set of symbols)

Formal definition of regular languages

- The class of regular languages over Σ is formally defined as:
  - ϕ is a regular language
  - $\forall a \in \Sigma \cup \epsilon$, $\{a\}$ is a regular language.
  - If L_1 and L_2 are regular languages, then so are:

  (a) $L_1 \cdot L_2 = \{xy \mid x \in L_1, y \in L_2\}$ (concatenation)

  (b) $L_1 \cup L_2$ (union or disjunction)

  (c) L_1^* (Kleene closure)

(Jurafsky & Martin 2000:49)

Examples

- `abc`
- `a|bc`
- `(a|b)c`
- `a*b`
- `[^a]*th[aeiou]+[a-z]*`

Regular expressions are useful for search

- Return all lines/documents containing a string of the specified language.
- Allows one to search for something that is conceptually one thing but linguistically or orthographically varied, e.g.: [Uu]niversit[y|ies]
- (Also allows searches for two things at the same time, e.g.: cat|dog)
- NB: Matching a string containing a substring described by a regular expression $\neq$ being in the language described by the regular expression.

Regular expression syntax

- Concatenations, disjunction, Kleene * (cf. formal def)
- Plus some syntactic sugar, i.e., other operations definable in terms of the above.
- Similar across Perl, Unix (including grep) and MS products.
- We'll be using Perl (and some grep).

Perl regular expression syntax (1/2)

- Concatenation = concatenation

  `/abc/`, `/ab c/`

- Disjunction: |, []

  `/[Aa][Bb][Cc]/`, `/cat|dog/`, `/kitt(y|ies)/`

- Kleene * (and + and ? and ranges)

  `/aa*/`, `/aa?a*/`, `/a+/`, `/a1,/`, `a1,10`

- Ranges, negation and wildcard:

  `/[A-Z][0-9][a-z][d-l][^jk][^t-z]./`,
  `/\d/`

Perl regular expression syntax

- Escape character:

  `/\[\.\|\{^/`

- Anchors: beginning and end of lines

  `/^Once upon a time/,/The end\.$/`

- Operator precedence hierarchy:

  parenthesis $\rangle\rangle$ counters $\rangle\rangle$ sequences and anchors $\rangle\rangle$ disjunction

Regular expression summary

- Encode regular languages (sets of strings)
- Are useful for search
- Syntax: concatenation, disjunction, Kleene *, plus sugar
- Useful for search
- More on Friday and Tuesday

Overview

- What is computational linguistics
- Who’s here
- Syllabus
- Showing a computer who’s boss
- Preview: Regular expressions

Reminders

- Pick up your Husky card
- Fill out the WebQ