Natural Language Processing (NLP): Overview & Tools

L715/B659

Dept. of Linguistics, Indiana University
Fall 2014
Natural Language Processing

Natural Language Processing (NLP): “The goal of this . . . field is to get computers to perform useful tasks involving human language” (Jurafsky & Martin 2009, p. 1)

Applications include:

▶ conversational agents / dialogue systems
▶ machine translation
▶ question answering
▶ ...

We will focus on natural language understanding (NLU): obtaining linguistic information (meaning) from input (text)
What do we need NLP for?

▶ One hand: we intend to do NLP, i.e., automatically analyze natural language for the purposes of providing meaning (of a sort) from a text
▶ Other hand: use NLP tools to pre-process data, i.e., provide sentence-level grammatical information:
  ▶ Segment sentences
  ▶ Tokenize words
  ▶ Part-of-speech tag words
  ▶ Syntactically (and semantically?) parse sentences
  ▶ Provide semantic word senses
  ▶ Provide named entities
  ▶ Provide language models

This kind of (pre-)processing is the focus for today
Where we’re going

We are going to focus on:

➢ what the general tasks are & what the uses are
➢ what kinds of information they generally rely on
➢ what tools are available

We’ll look at POS tagging, parsing, word sense assignment, named entity recognition, & semantic role labeling

➢ We’ll focus on English, but try to note general applicability

Many taggers/parsers have *pre-built* models; others can be *trained* on annotated data

➢ For now, we’ll focus on pre-built models
Wikis with useful technology information

Places you can get your own information:

- Our very own IU CL wiki, which includes some people’s experiences with various tools
  - http://cl.indiana.edu/wiki
  - Always feel free to add your own experiences to help the next person who wants to use that tool
- ACL wiki & resources
  - ACL software registry: http://registry.dfki.de/
General NLP packages

- Stanford NLP: http://nlp.stanford.edu/software/ (see esp. the CoreNLP package)
- ClearNLP: http://www.clearnlp.com
- FreeLing: http://nlp.lsi.upc.edu/freeling/
- LingPipe: http://alias-i.com/lingpipe/
- OpenNLP: http://opennlp.apache.org/index.html
- Natural Language Toolkit (NLTK): http://www.nltk.org/
- Illinois tools: http://cogcomp.cs.illinois.edu/page/software
- DKPro: https://www.ukp.tu-darmstadt.de/research/current-projects/dkpro/
  - Also includes a text classification tool built on top of weka
Topic #1: POS Tagging

**Idea:** assign a part-of-speech to every word in a text

- (Supervised) Taggers work by:
  - looking up a set of appropriate tags for a word in a dictionary
  - using local context to disambiguate from among the set
- Sequence modeling (HMMs, CRFs) are thus popular

Some examples illustrating the utility of local context:

- *for the man*: noun or verb?
- *we will man*: noun or verb?
- *I can put*: verb base form or past?
- *re-cap real quick*: adjective or adverb?

Bigram or trigram tagging is quite popular

- Take L545/L645 if you want to know more
Motivation for POS tags

What are POS tags good for in our intended downstream applications?

- First step towards knowing the meaning, e.g., for word senses (e.g., *leaves*)
- Help identify function words & content words (e.g., for stylometry)
- POS sequences (*n*-grams) may be indicative of style
  - POS *n*-grams approximate syntax

Note that POS tags are generally very fast to obtain & are generally accurate (for English, on well-formed data)
Challenges for POS tagging

General challenges:

- Ambiguity
  - e.g., *still* as noun, verb, adverb, adjective, ...
- Unknown words
  - Programs use things like suffix tries to guess at the possible POS tags for unknown words

These challenges are exacerbated in the following areas:

- Morphologically-rich languages
- Data which is not well-edited (e.g., web data)
POS taggers

- TnT: http://www.coli.uni-saarland.de/~thorsten/tnt/
  - Trainable; models for German & English
- TreeTagger: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/
  - Trainable; models for English, German, Italian, Dutch, Spanish, Bulgarian, Russian, & French; unix, mac, PC
- Qtag: http://www.english.bham.ac.uk/staff/omason/software/qtag.html
  - Trainable; models for German & English
  - Has a variety of NLP modules
- OpenNLP: http://opennlp.sourceforge.net/
  - Models for English, German, Spanish, & Thai; Has a variety of NLP modules
POS taggers (2)

- ACOPOST: http://acopost.sourceforge.net/
  - Trainable; integrates different technologies
- Stanford tagger:
  http://nlp.stanford.edu/software/tagger.shtml
  - Trainable; models for English, Arabic, Chinese, & German
- CRFTagger: http://crftagger.sourceforge.net/
  - English
- Can also use SVMTool
  (http://www.lsi.upc.edu/~nlp/SVMTool/) or CRF++
  (http://crfpp.sourceforge.net/) for tagging sequential data, or fnTBL for classification tasks
  (http://www.cs.jhu.edu/~rflorian/fntbl/index.html)
Specialized POS taggers

Twitter tagger:
- CMU Ark: http://www.ark.cs.cmu.edu/TweetNLP/
- GATE: https://gate.ac.uk/wiki/twitter-postagger.html
  (also available to plug into Stanford tagger)

Biomedical tagger:
- GENIA tagger:
  http://www.nactem.ac.uk/tsujii/GENIA/tagger/
- cTAKES (clinical Text Analysis and Knowledge Extraction System):
  https://ctakes.apache.org/index.html
Topic #2: Parsing

Parsers attempt to build a tree, based on some grammar

▶ Efficiency based on many things, including the manner in which the tree is built
▶ They often disambiguate by using probabilities of rules

Again, take L545/L645 for more details
Constituencies & Dependencies

Rough idea of the difference:

Constituency:

```
S
/   \
|    |
NP   VP
/   /
|   |
DT NN VBD NN
```

Dependency:

```
vroot
```

```
|    |
DET NN VBD NN
|    |
the dragon breathed fire
|    |
the DET SUBJ OBJ
|    |
the dragon breathed fire

Rough idea of the difference:

Constituency:

```
S
/   \
|    |
NP   VP
/   /
|   |
DT NN VBD NN
```

Dependency:

```
vroot
```

```
|    |
DET NN VBD NN
|    |
the dragon breathed fire
|    |
the DET SUBJ OBJ
|    |
the dragon breathed fire
```
Constituency parsing

Goal is to obtain phrases

- Structured prediction: dealing with embedded / recursive structures
- Parsing can be slow, but tends to be fairly accurate
  - POS tags obtained while parsing more accurate than with a standalone POS tagger

Usefulness for downstream applications:

- Identifying sequences, e.g., named entities
- Identifying complexity, e.g., depth of embedding
- Identifying particular types of constructions, e.g., relative clauses
Challenges in parsing

In addition to things like lexical ambiguity & unknown words, additional challenges include:

▶ Structural ambiguity: e.g., *They saw the man in the park with a telescope*

▶ Garden paths: e.g., *The horse raced past the barn fell*

Again, out-of-domain data poses a challenge

▶ Note that for morphologically-rich languages, parsing is underdeveloped and that some of the work is in the morphology
Dependency parsing

Dependency parsing is the task of assigning dependency (grammatical) relations to a sentence

- Provides quick access to semantic relations ("who did what to whom")
- Can be done on top of constituency parsing or on its own
  - Formally, dependency parsing is simpler: assign a single head & relation for every word (single-head constraint)

Useful applications:
- Pretty close to the same set as with constituencies ...
Constituency Parsers

- LoPar: http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LoPar.html
  - Trainable; models for English & German
- BitPar: http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/BitPar.html
  - Trainable; models for English & German
- Charniak & Johnson parser: http://www.cs.brown.edu/people/ec/#software
  - Trainable; mainly used for English
Constituency Parsers (2)

- Collins/Bikel parser:
  http://people.csail.mit.edu/mcollins/code.html
  http://www.cis.upenn.edu/~dbikel/software.html
  - Trainable on English, Chinese, and Arabic; designed for Penn Treebank-style annotation

- Stanford parser:
  - Trainable; models for English, German, Chinese, & Arabic; dependencies also available

- Berkeley parser:
  http://code.google.com/p/berkeleyparser/
  - Trainable; models for English, German, and Chinese
Dependency parsers

Recent parsers, which generally include other NLP tools:
- Mate Parser: https://code.google.com/p/mate-tools/
- TurboParser: http://www.ark.cs.cmu.edu/TurboParser/
- ZPar: http://sourceforge.net/projects/zpar/

Classic dependency parsers:
- MaltParser:
  http://w3.msi.vxu.se/~nivre/research/MaltParser.html
  - Trainable; models for Swedish, English, & Chinese
- MSTParser: http://sourceforge.net/projects/mstparser
  - Trainable; has some models for English & Portuguese
- Link Grammar parser:
  http://www.abisource.com/projects/link-grammar/
  - English only

CCG parsers: http://groups.inf.ed.ac.uk/ccg/software.html
- Primarily for English, although can be trained on German CCGbank
Topic #3: Semantics

**Semantics** is the study of meaning in language

We’ll break it down into:

- Lexical semantics: word meaning
- Compositional semantics: sentence meaning

and look at technology for both
Semantic class assignment
Word sense disambiguation

Word sense disambiguation (WSD): for a given word, determine its semantic class

- **bank.01**: They robbed a **bank** and took the cash.
- **bank.02**: They swam awhile and then rested on the **bank**.

Lexical resources define the senses, e.g.

- **WordNet**: http://wordnet.princeton.edu
- **BabelNet**: http://babelnet.org
WSD software

- GWSD: Unsupervised Graph-based Word Sense Disambiguation
  http://web.eecs.umich.edu/~mihalcea/downloads.html
- SenseLearner: All-Words Word Sense Disambiguation Tool:
  http://web.eecs.umich.edu/~mihalcea/downloads.html
- KYOTO UKB graph-based WSD:
  http://ixa2.si.ehu.es/ukb/
- pyWSD: Python Implementation of Simple WSD algorithms:
  https://github.com/alvations/pywsd
- Various packages from Ted Pedersen, including Senseval systems:
  http://www.d.umn.edu/~tpederse/code.html
Semantic class assignment

Named entity recognition

Named entity recognition (NER): classify elements (words, phrases) into pre-defined entity classes

▶ Common categories include: PER(son), ORG(анизация), LOC(ация), etc.
▶ May have hierarchical categories

Techniques often rely on phrase chunking & may involve using a gazetteer (external list of entities)

▶ From the list of general NLP tools above, Stanford, UIUC, & OpenNLP have NER modules
Semantic role labeling

**Idea:** The words of a sentence combine to form a meaning

- Hypothesis: the syntax and semantics can be built up in a corresponding fashion

**Semantic role labeling** is the task of assigning semantic roles to arguments in a sentence

e.g., for *John loves Mary*:

- *(to)* love is the predicate
- *John* is the agent (ARG0)
- *Mary* is the patient (ARG1)
Semantic role labelers

- Clear: http://www.clearlp.com
- SENNA: http://ml.nec-labs.com/senna/
- UIUC: http://cogcomp.cs.illinois.edu/page/software_view/SRL
- SEMAFOR: https://code.google.com/p/semafor-semantic-parser/
- SwiRL: http://www.surdeanu.info/mihai/swirl/
- Shalmaneser: http://www.coli.uni-saarland.de/projects/salsa/shal/
- MATE: https://code.google.com/p/mate-tools/
- Turbo: http://www.ark.cs.cmu.edu/TurboParser/
Language models store lots of text in $n$-gram form, using it to assign probabilities to new sequences of text

- Tend to be fast & surprisingly accurate

Some packages:

- MIT Language Modeling Toolkit: https://code.google.com/p/mitlm/