Introduction to Computational Linguistics

Chunking and Partial Parsing

Week 5, Lecture 2

October 21, 2004

Today

- Motivation
- What are chunks?
- Chunking in CASS
- Chunking in NLTK

Partial Parsing / Chunking

Assign a *partial structure* to a sentence.

- Don't try to deal with all of language
- Don't attempt to resolve all semantically significant decisions
- Use deterministic grammars for easy-to-parse pieces, and other methods for other pieces, depending on task.
  - "easy to parse" — avoid ambiguity, avoid recursion
- Partial parsing is usually:
  - easier to implement
  - more robust
  - faster

Chunking

Goal: Divide a sentence into a sequence of CHUNKS.

- Abney (1994): [when I read] [a sentence], [I read it] [a chunk] [at a time]
- Chunks are non-overlapping regions of text
  [walk] [straight past] [the lake]
- (Usually) each chunk contains a **head**, with the possible addition of some preceding function words and modifiers
  [**walk**] [straight **past**] [the **lake**]
- Chunks are non-recursive:
  - A chunk cannot contain another chunk of the same category

Chunking, cont.

- Chunks are non-exhaustive
  - Some words in a sentence may not be grouped into a chunk
    [take] [the second road] that [is] on [the left hand side]
- Chunks are typically subsequences of constituents (they don't cross constituent boundaries)
- *noun groups* — everything in NP up to and including the head noun
- *verb groups* — everything in VP (including auxiliaries) up to and including the head verb

Chunk Grammars

Approach adopted in CASS (Abney)

- Recognition carried out by a cascade of FSMs – output of one is the input to another

  Level 0: tagged words

  Level 1: all sequences at level 0 that match a given pattern are replaced by appropriate label
  - e.g., date expressions replace by `Date`

  Level n**:** do something with output of Level $n-1$
- Strings that don't match a pattern just passed up unchanged

CASS RegEx Grammar

Automata defined by a ‘regular expression grammar’

```
:chunks
 nx -> DT? NN+
 vx -> VBZ | VBD | BE VBG
:phrases
 vp -> vx nx*
 pp -> IN nx
:clause
 c -> pp* nx pp* vp pp*
```


CASS Example

```
take/VBP the/DT road/NN on/IN the/DT left/NN

[vx take/VBP] [nx the/DT road/NN] on/IN [nx the/DT left/NN]

[vx take/VBP] [nx the/DT road/NN] [pp on/IN [nx the/DT left/NN]]

[c [vx take/VBP] [nx the/DT road/NN] [pp on/IN [nx the/DT left/NN]]
```

Chunk Parsing in NLTK

- Regular expression matching

```
rule1 = Chunkrule('<DT|NN>+',
     'Chunk sequences of NN and DT')

RegexpChunkparser([rule],
                  SUBTOKENS='WORDS',
                  chunk_node='NP',
                  top_node='S')
```


BIO Notation for Chunks

Instead of using bracketing, as in

```
take [nx the second road] on [nx the left]
```

we **tag** words according to where they are in a chunk:

```
take/O
  the/B-NP second/I-NP road/I-NP
on/O
  the/B-NP left/I-NP
```

where `B-NP` is 'Begin noun chunk', `I-NP` is 'Inside noun chunk' and `O` is 'Outside any chunk'.

- Used in CoNNL shared tasks
- Allows off-the-shelf statistical taggers to be used

Reading

- Steven Abney. Parsing By Chunks. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), Principle-Based Parsing. Kluwer Academic Publishers, Dordrecht. 1991.
- Steven Abney. Partial Parsing via Finite-State Cascades. J. of Natural Language Engineering, 2(4): 337-344. 1996.
- Abney's publications:
  `http://www.vinartus.net/spa/publications.html`
- Jurafsky and Martin, Section 10.5
- NLTK Chunk Parsing Tutorial

Final Remarks

- Since it is relatively easy to identify chunks, widely used as a stage in larger NLP tasks:
  - Information Extraction
  - Question Answering
  - Extracting subcatgorization frames
  - Providing features for machine learning, e.g., for building Named Entity recognizers.
  - Assignment 2
- **No lectures or lab sessions next week**!
- Week 7: Full parsing