School of Computing and Information Systems
The University of Melbourne
COMP90042
WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2017)

Workshop exercises: Week 5

Discussion

1. Using typical dependency types, construct (by hand) a dependency parse for the following sentence: *Yesterday, I shot an elephant in my pyjamas.* Check your work against the output of the online GUI for the Stanford Parser (`http://nlp.stanford.edu:8080/parser/index.jsp`).
2. In what ways is (transition–based, probabilistic) dependency parsing similar to (probabilistic) Earley parsing? In what ways is it different?
3. Give illustrative examples that show the difference between:
   (a) **Synonyms** and **hypernyms**
   (b) **Hyponyms** and **meronyms**
4. One possible step of text normalisation (tokenisation) is conflating synonyms as a single representation. Give a couple of reasons why this doesn't usually happen.
5. Using some Wordnet visualisation tool, for example, `http://wordnetweb.princeton.edu/perl/webwn` and the Wu & Palmer definition of **word similarity**, check whether the word *information* more similar to the word *retrieval* or the word *science* (choose the sense which minimises the distance). Does this mesh with your intuition?
6. What is **word sense disambiguation**?
   (a) The **Yarowsky** method from the lectures uses two heuristics — what do they mean and why are they significant? Can you find counter–examples?

Programming

1. NLTK doesn't have much dependency parsing support — there is a little MaltParser interface (`http://maltparser.org/`), but it can be unreliable.

   One popular dependency parsing platform is the Stanford Parser (`https://nlp.stanford.edu/software/lex-parser.shtml`) — the entire package is a somewhat large and requires Java. There are some Python bindings, however. (For example, `http://projects.csail.mit.edu/spatial/Stanford_Parser`)

   (a) Parse the (tokenised) sentences in `nltk.corpus.treebank_raw.sents()`
   (b) What proportion of the resulting dependency trees are non–projective? Why do you suppose this is?
2. Consider the iPython notebook `lexical_semantics`. Repeat the exercise about word similarity from the Discussion problems above; confirm that you get the same answer. Now try the Lin similarity — do you get the same result? Why or why not?

Catch-up

- What is a **head**? A **dependency**? How do dependencies differ from **constituents**?
- What are the major differences between the (tree) structure produced by parsing using a context–free grammar, and that produced by parsing using a dependency grammar?
- What are some common dependency arc labels, and what grammatical notions do they encode?
- What do we mean when we say "a word means [something]"?
- What is a **synonym**? How is it related to meaning?
- What is **information**, with respect to **entropy**? How might we calculate the information of a word in a corpus? How might we calculate the information of a (textual) message with respect to the information in a corpus? (There are many different ways!)
- What is **WordNet**? What is a **synset**?

Get ahead

- Revise how to train a PCFG parser.

  Read up on how to use the Stanford parser as a "vanilla" PCFG parser — compare its output on some selected sentences, when using training sets of different sizes (for example, slices of the Penn Treebank).
- Choose an individual word and consider its different synsets in Wordnet.
  - Find a number of instances of that word in a corpus. Assign each token to its sense. Is one sense more frequent than the other senses?
- Build a system which attempts to use the **Lesk** strategy of WSD based on the Wordnet bindings in NLTK. Choose some word(s) in some sentence(s) and observe its output — does the correct sense get returned? Why or why not?