School of Computing and Information Systems
The University of Melbourne
COMP90042
WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2019)

Sample solutions for discussion exercises: Week 11

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`).

    - Dependency parses lend themselves to a flat representation, from which you can derive the tree if you wish:

ID	Token	Head	Relation
1	Yesterday	4	TMP
2	,	1	PUNCT
3	I	4	NSUBJ
4	shot	0	ROOT
5	an	6	DET
6	elephant	4	DOBJ
7	in	9	CASE
8	my	9	POSS
9	pyjamas	4	NMOD
10	.	4	PUNCT

2. In what ways is (transition–based, probabilistic) dependency parsing similar to (probabilistic) CYK parsing? In what ways is it different?

    - The connections are a little tenuous, but let's see what we can come up with:
        - Both methods are attempting to determine the structure of a sentence; both methods are attempting to disambiguate amongst the (perhaps many) possible structures licensed by the grammar by using a probabilistic grammar to determine the most probable structure.
        - Both methods process the tokens in the sentence one–by–one, left–to–right.
    - There are numerous differences (probably too many to enumerate here), for example:
        - Although POS tags are implicitly used in constructing the "oracle" (training), the depedency parser doesn't explicitly tag the sentence.
        - The transition–based dependency parser can potentially take into account other (non–local) relations in the sentence, whereas CYK's probabilities depend only on the (local) sub-tree.
        - CYK adds numerous fragments to the chart, which don't end up getting used in the final parse structure, whereas the transition–based dependency parser only adds edges that will be in the final structure.

3. What is **Discourse Segmentation**? What do the segments consist of, and what are some methods we can use to find them?

   - In Discourse Segmentation, we try to divide up a text into discrete, cohesive units based on sentences.
   - By interpretting the task as a boundary–finding problem, we can use rule–based or unsupervised methods to find sentences with little lexical overlap (suggesting a discourse boundary). We can also use supervised methods, by training a classifier around paragraph boundaries.

4. What is an **anaphor**?

   - From the lectures: an anaphor is a linguistic expression that refers back to one or more elements in the text (generally preceding the anaphor)
   - These tend to be pronouns (*he*, *she*) but can also be determiners (*which*, *the*, etc.).

   (a) What is **anaphora resolution** and why is it difficult?

   - This is the problem of working out which element (generally a noun or noun phrase, but sometimes a whole clause) a given anaphor is actually referring to.
   - For example:

     > Mary gave John a cat for **his** birthday. (i) **She** is generous. (ii) **He** was surprised. (iii) **He** is fluffy.

     his [birthday] obviously refers to John; (i) (presumably) refers to *Mary*; (ii) (presumably) refers to *John*; and (iii) (presumably) refers to *[the] cat*.

   (b) What are some useful heuristics (or features) to help resolve anaphora?

   - The most obvious (but inherent unreliable) heuristic is the **recency heuristic**: given multiple possible referents (that are consistent in meaning with the anaphor), the mostly intended one is the one most recently used in the text.
   - A better heuristic is that the most likely referent (consistent in meaning with the anaphor) is the focus of the discourse (the "center").
   - We can also build a supervised machine learning model, usually based around the semantic properties of the anaphor/nearby words and the sentence/discourse structure.