School of Computing and Information Systems
The University of Melbourne
COMP90042
WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2018)

Sample solutions for discussion exercises: Week 6

Discussion

1. What differentiates **probabilistic parsing** from **chart parsing**? Why is this important? How does this affect the algorithms used for parsing?

   - **Parsing** in general is the process of identifying the structure(s) of a sentence, according to a grammar of the language.
   - In general, the search space is too large to do this efficiently, so we use a dynamic programming method to keep track of partial solutions. The data structure we use for this is a **chart**, where entries in the chart correspond to partial parses (licensed structures) for various spans (sequences of tokens) within the sentence. **Probabilistic parsing** adds real-valued weights (probability) to each production in the grammar, such that parse trees can be assigned a score, namely the product of the probabilities of the productions in the tree. This is important as it allows for discrimination between likely and unlikely parses, rather than just provide a list of all parses, as in standard chart parsing. This affects the algorithms as they need to track the maximum probability analysis for each span, rather than the set of grammar symbols. However the parsing algorithms are very closely related.

2. What is a **probabilistic grammar** and what problem does it attempt to solve?

   - A **probabilistic grammar**, as described above, adds real-valued weights (probability) to each production in the grammar. This attempts to provide a "language model", that is, describe the likely sentences in a language, which is facilitated by their grammatical structure.

3. A hidden Markov model assigns each word in a sentence with a tag, e.g.,

   Donald/NNP has/VBZ small/JJ hands/NNS

   The probability of the sequence is based on the tag-word pairs, and the pairs of adjacent tags. Show how this process can be framed as a CFG, and how the various probabilities (e.g., observation, transition, and initial state) can be assigned to productions. What are the similarities and differences between CYK parsing with this grammar, and the HMM's Viterbi algorithm for finding the best scoring state sequence?

   - The first step is to write the probability assigned by the HMM to the tagged sentence.
   - This can be drawn as a chain with the tag-tag transitions as the "spine", with each observation as a leaf. The probabilities of initial / transitions / observations can be attached to each edge. Overall this gives a right-branching tree.

- If we treat this tree as a "parse tree", then each clique from the tree (parent and direct children) can be treated as a CFG rule, parent → left-child right-child. E.g., NNP → Donald VBZ.
- The score for this rule would be $A_{\text{NNP,VBZ}} \times O_{\text{NNP,Donald}}$. Note that we could use different grammar symbols, such that each production maps to a single HMM component. E.g., split the above rule to NNP' → NNP VBZ ($A_{\text{NNP,VBZ}}$); and NNP → Donald ($O_{\text{NNP,Donald}}$); where the "prime" version of the tag is a newly introduced grammar symbol.
- CYK parsing under this grammar will do the same as Viterbi in the HMM, but will waste effort assigning analysis to all word spans in the sentence, while Viterbi effectively only considers spans that start at the first word of the sentence.

4. 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
```

5. 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 Earley'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.