Parsing:
Lexicalized Statistical Parsing
Evaluation of Parsing
Available Parsers

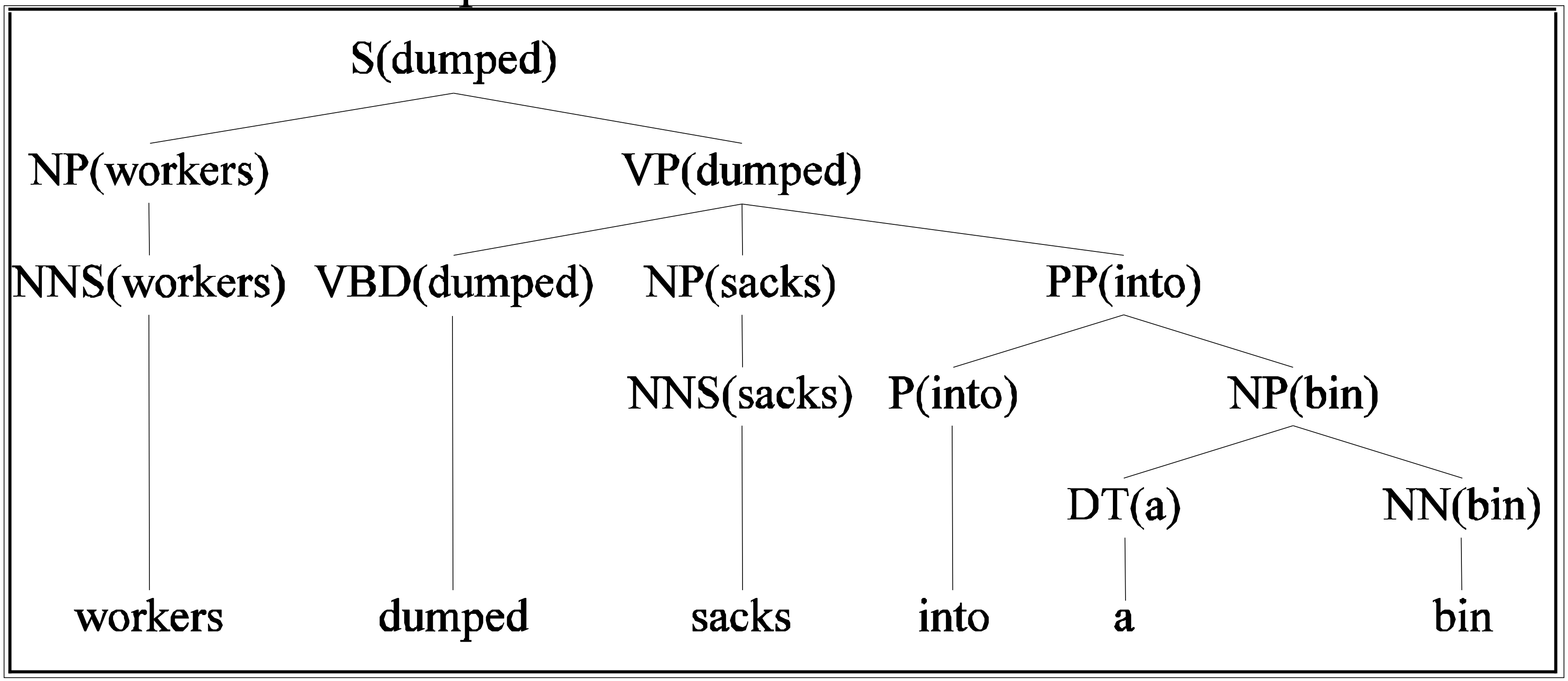
Lexicalized Statistical Parsing

- Add lexical dependencies to the scheme of probabilities
  - Integrate the preferences of particular words into the probabilities in the derivation
  - i.e. Condition the rule probabilities on the actual words
- To do that we're going to make use of the notion of the head of a phrase
  - The head of an NP is its noun
  - The head of a VP is its verb
  - The head of a PP is its preposition

  (It's really more complicated than that but this will do.)
- Main parsing breakthrough idea of the 1990's
- Expand the set of phrase types with phrase type/word
  - In practice, we learn probabilities to automatically detect head words

Example (right)

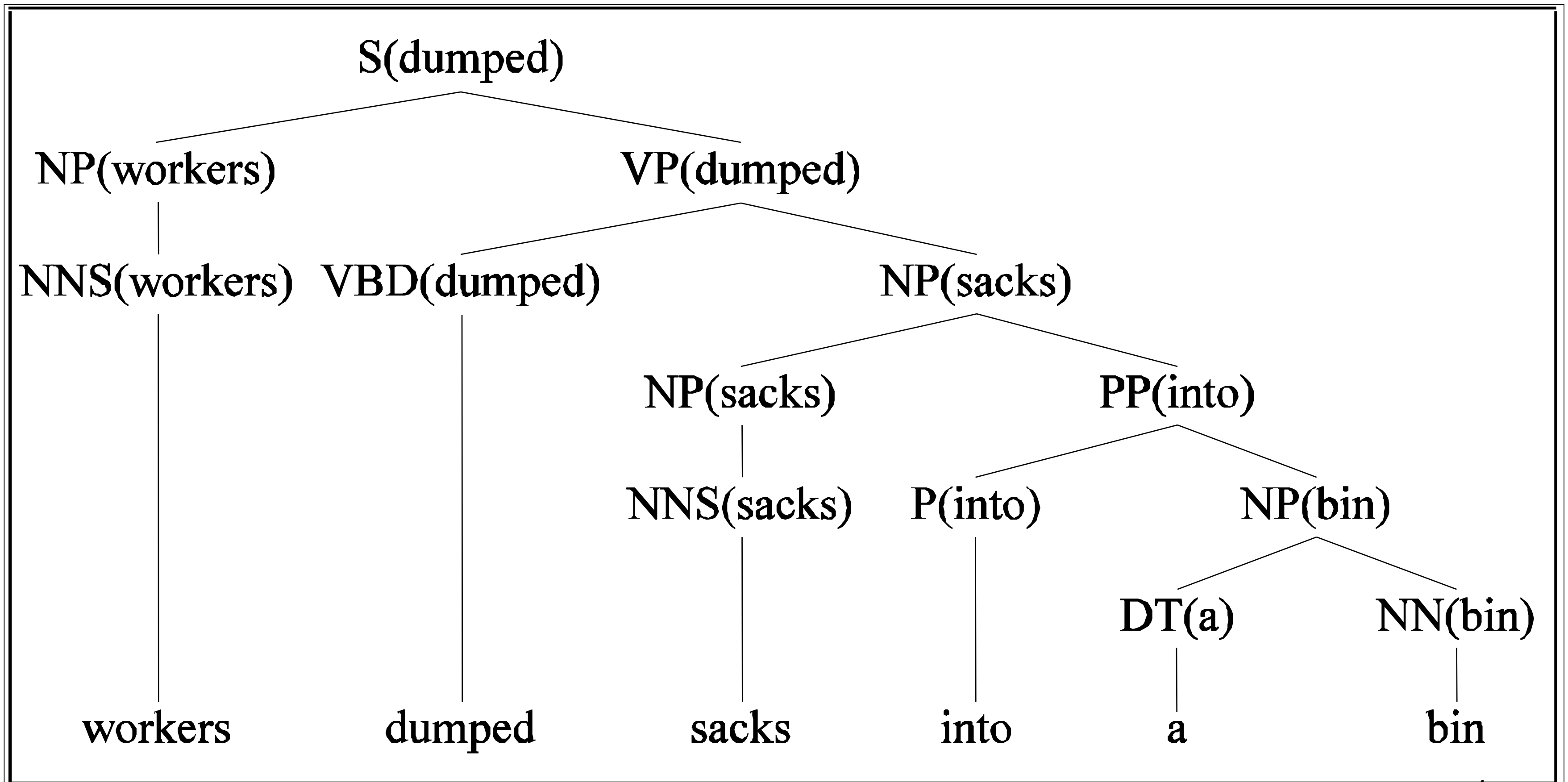
- Should we attach the prepositional phrase with head “into” to the verb “dumped”?



- In this tree, each phrase type, such as NP or VP, is also shown with its attached head word.

Example (wrong)

- Or should we attach the prepositional phrase with head "into" to the noun "sacks"?



Preferences

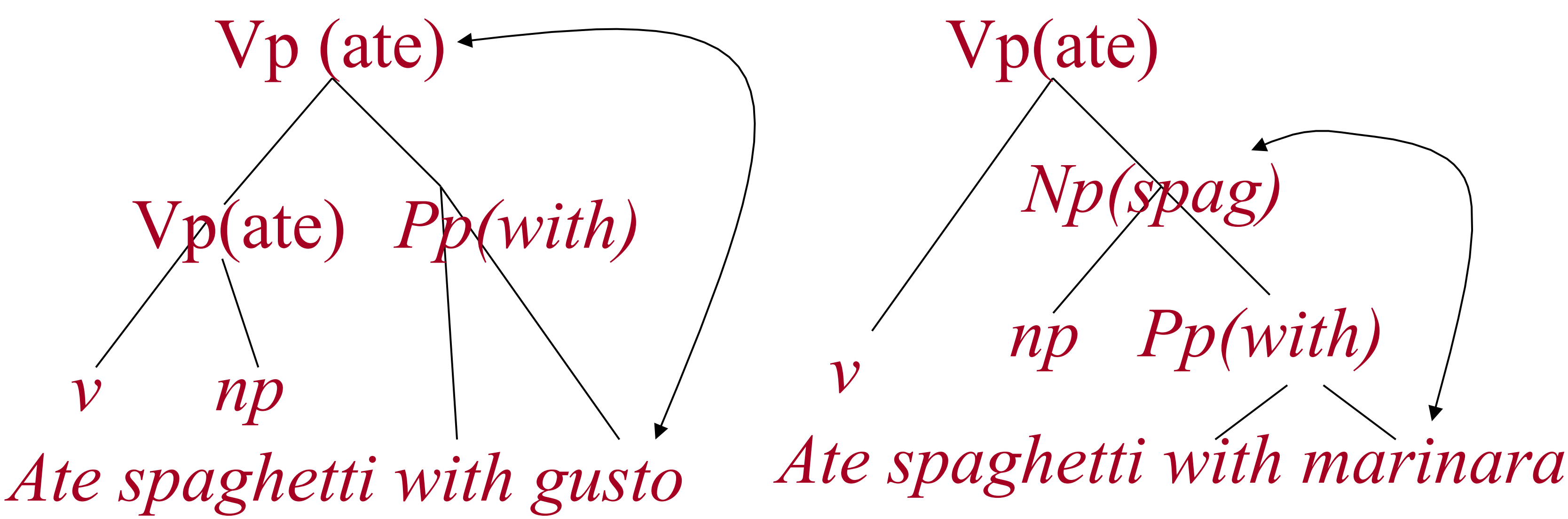
- The issue here is the attachment of the PP. So the affinities we care about are the ones between dumped and into vs. sacks and into.
  - So count the places where dumped is the head of a constituent that has a PP child with into as its head and normalize
  - Vs. the situation where sacks is a constituent with into as the head of a PP child.
- In general, collect statistics on preferences (aka affinities)
  - Use verb subcategorization
    - Particular verbs have affinities for particular VPs
  - Objects affinities for their verbs, mostly their parents and grandparents
    - Some objects fit better with some verbs than others

Preference example

- Consider the VPs
  - Ate spaghetti with gusto
  - Ate spaghetti with marinara
- The affinity of gusto for eat is much larger than its affinity for spaghetti
- On the other hand, the affinity of marinara for spaghetti is much higher than its affinity for ate

Preference Example (2)

- Note the relationship here is more distant and doesn't involve a headword since gusto and marinara aren't the heads of the PPs.



Note

- Jim Martin: “In case someone hasn’t pointed this out yet, this lexicalization stuff is a thinly veiled attempt to incorporate semantics into the syntactic parsing process…
  - Duhh..,. Picking the right parse requires the use of semantics.”

Last Points

- Statistical parsers are getting quite good, but it's still quite challenging to expect them to come up with the correct parse given only statistics from syntactic and lexical information.
- But if our statistical parser comes up with the top-N parses, then it is quite likely that the correct parse is among them.
- Lots of current work on
  - Re-ranking to make the top-N list even better.

- There are also grammar-driven parsers that are competitive with the statistical parsers, notably the CCG (Combinatory Categorial Grammar) parsers

Evaluation

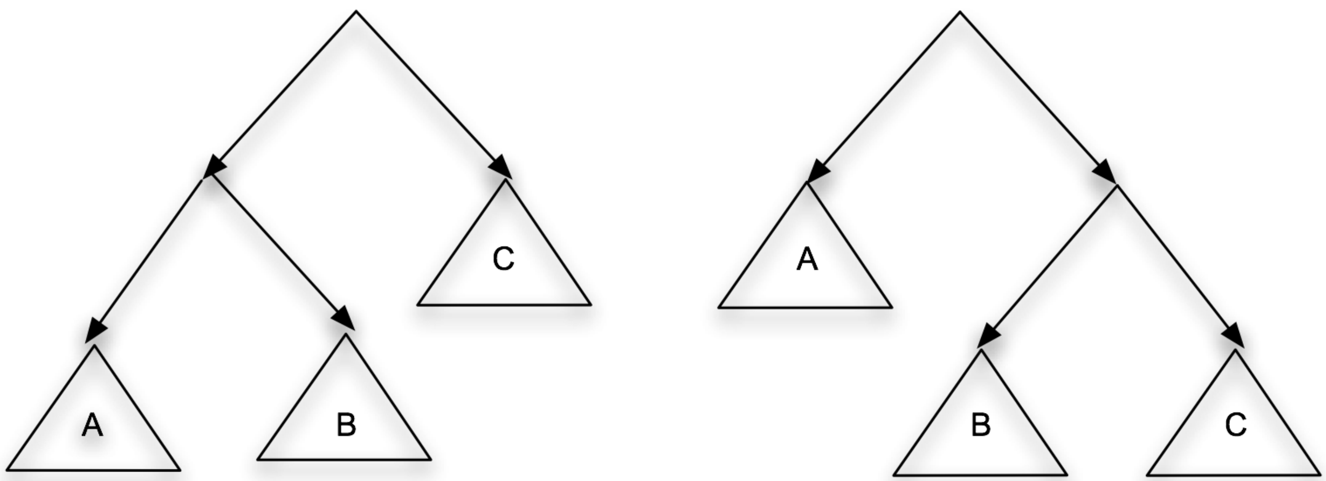
- Given that it is difficult/ambiguous to produce the entire correct tree, look at how much of content of the trees are correctly produced
  - Evaluation measures based on the correct number of constituents (or sub-trees) in the system compared to the reference (gold standard)
- Precision
  - What fraction of the sub-trees in our parse matched corresponding sub-trees in the reference answer
    - How much of what we're producing is right?
    - Reduce number of false positives
- Recall
  - What fraction of the sub-trees in the reference answer did we actually get?
    - How much of what we should have gotten did we get?
    - Reduce number of false negatives
- F-measure combines precision and recall to give an overall score.

Evaluation

- An additional evaluation measure that is often reported is that of Crossing Brackets errors, in which the subtrees are equal, but they are put together in a different order.

Parser hypothesis

Reference answer



((A B) C)

(A (B C))

Available Parsers

- Among the family of lexicalized statistical parsers are the original Collins parser (Michael Collins 1996, 1999) and the Charniak parser (1997)
  - both are publicly available and widely used, for non-commercial purposes.
- The Charniak series of parsers is still under development, by Eugene Charniak and his group; it produces N-best parse trees.
  - Its evaluation is on the Penn Treebank at about 91% F measure.
- Another top performing parser, originally by Dan Klein and Christopher Manning, is available from the Stanford NLP group
  - combines "separate PCFG phrase structure and lexical dependency experts".
  - Demo at: http://nlp.stanford.edu:8080/parser/

Available Parsers

- The CCG parsers are available from their open source page
  - http://groups.inf.ed.ac.uk/ccg/software.html
- Parsers are also available through the OpenNLP project, with the OpenNLP API:
  - http://opennlp.sourceforge.net/

Dependency Parsing

- Dependency parsing has some resemblance to lexicalized statistical parsing because of the importance of the lexical entities (words) to capturing the syntactic structure
- But dependency parsing produces a simpler representation of the structure.
  - Can be easier to use in some semantic applications
- Parsing algorithms are similar to constituent parses
  - Statistics for dependency relations learned from Penn Treebank
  - Used bottom-up parser to find best parse(s)
  - Some additional mechanism used to find non-projective parses