Study of Probabilistic Parsing in Syntactic Analysis

Dr. Girish Katkar
Head of Department
Department of Computer Science
Arts, Commerce & Science College Koradi Dist:- Nagpur
Girishkatkar2007@rediffmail.com

Krishna Karoo
Research Scholar
Department of Electronics & Computer Science
R.T.M. Nagpur University, Nagpur
Krishna.karoo@gmail.com

Abstract:-Statistical parser, like statistical tagging requires a corpus of hand –parsed text. There are such corpora available, the most notably being the Penn-tree bank. The Penn-tree bank is large corpus of articles from the Wall Street Journal that have been tagged with Penn tree-Bank tags and then parsed accordingly to a simple set of phrase structure rules conforming to Chomsky Government and binding syntax[1].

Keywords*:-Probabilistic parsing, Penn-tree bank.*

*****

I. INTRODUCTION

1.1 Probabilistic CFGs

The probabilistic model assigning probabilities to parse trees, getting the probabilities for the model. Parsing with probabilities -Slight modification to dynamic programming approach, Task is to find the max probability tree for an input[2].

1.2Probability Model

The grammer G=(V,T,P,S) and attached certain rules to the grammar.The expansions for a given non-terminal sum to 1[3]

VP -> Verb	.55
VP-> Verb NP	.40
VP-> Verb NP NP	.05

A derivation (tree) consists of the set of grammar rules that are in the tree. The probability of a derivation (tree) is just the product of the probabilities of the rules in the derivation. The probability of a word sequence (sentence) is the probability of its tree in the unambiguous case. If the grammar have more than one parse tree[4].It's the sum of the probabilities of the trees in the ambiguous case.[5]

1.3 Getting the Probabilities

From an annotated database (a Treebank) which is already existing database corpus. Treebank are annotated manually English: Penn Treebank (about 1.6 million words).Chinese: Chinese Treebank (about 500K words).Learned from a raw corpus. The probabilities are taken using HMM with two or three combination of words and statistical methods [6]

1.4 Learning from a Treebank

The easiest and the most accurate way to learn probabilities is ,Get a large collection of parsed sentences .Collect counts for each non-terminal rule expansion in the collection[7]. If you don't have a treebank (and can't get one) then take a large collection of text and parse it using a grammar, In the case of syntactically ambiguous sentences collect all the possible parses. Prorate the rule statistics gathered for rules in the ambiguous case by their probability Proceed as you did with a treebank.

1.5 Assumptions

Assuming that there is a grammar to be used to parse with. Considering the existence of a large robust dictionary with parts of speech .Assume that the ability to parse (i.e. a parser) Given all that… we can parse probabilistically Alternatively: we can build the dictionary with parts of speech and the grammar based on the Treebank corpus.[8]

II. TYPICAL APPROACH

There are different approaches in statistical analysis to reduce the ambiguity most of them are Bottom-up dynamic programming approach [9], assign probabilities to constituents as they are completed and placed in the table and Use the max probability for each constituent going up.

2.1 Use the max probability for each constituent :-

There are certain grammar rules that are used to find the probabilistic parsing some of the rules are

S->0NPiVPj

The probability of the S is…

P(S->NP VP)*P(NP)*P(VP)

2.2 Max

P(NP) is known. if there are multiple NPs for the span of text in question .Take the max of finding values.Does not mean that other kinds of constituents for the same span are ignored (i.e. they might be in the solution)[9]

2.3 Problems

The probability model we're using is just based on the rules in the derivation. Doesn't use the words in any real way. Doesn't take into account where in the derivation a rule is used.

2.4 Solution

Add lexical dependencies to the schem.Infiltrate the influence of particular words into the probabilities in the derivation I.e. condition on actual words All the words? No, only the right ones.

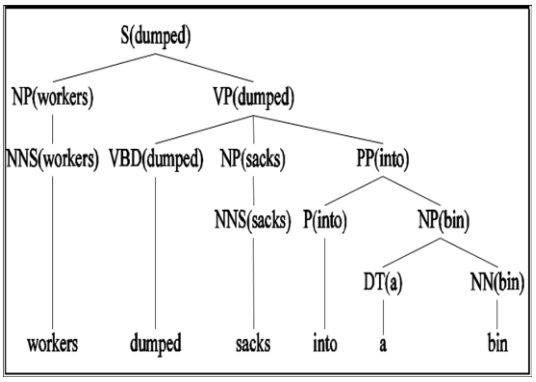
Heads

There are certain notations that have used to do that going to make use of the notion of the head of a phrase. There certain rules are. The head of an NP is its noun, The head of a VP is its verb, The head of a PP is its preposition ,A way of achieving some generalization ,Replace phrases with their heads, Finding basic phrases (NP, VP, PP) and their heads, Shallow parsing

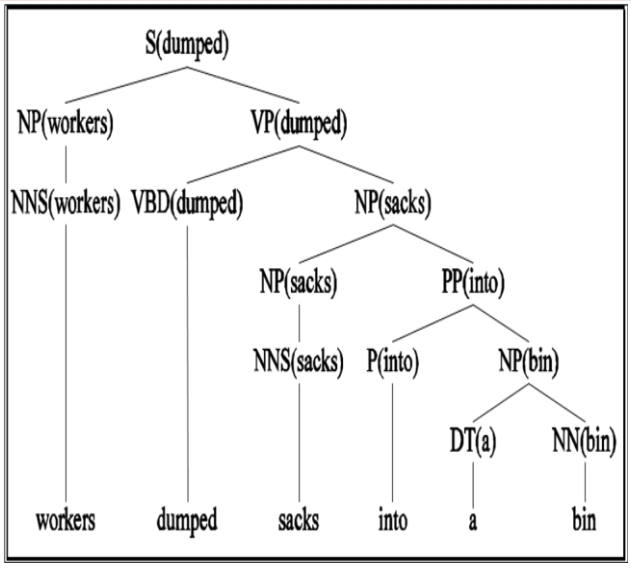
Using certain grammar rules the tree which is constructed which don't have ambiguity is as follows

+ Heuristics

2.5 Example (right)



2.6 Example (wrong)



Using certain productions like VP -> V NP PP ,P(r|VP) ,That's the count of this rule divided by the number of VPs in a treebank .Now we consider the production rule like VP(dumped)-> V(dumped) NP(sacks)PP(in),P(r|VP ^ dumped is the verb ^ sacks is the head of the NP ^ in is the head of the PP) ,Not likely to have significant counts in any treebank

III. DECLARE INDEPENDENCE

When stuck, exploit independence and collect the statistics that can focus on capturing two things subcategorizing the verbs. Particular verbs have affinities for particular VPs Objects affinities for their predicates .Some objects fit better with some predicates than others[10]

3.1 Subcategorization

Condition particular VP rules on their head, the rule is

Rule: VP -> V NP PP P(r|VP)

Becomes

P(r | VP ^ dumped)

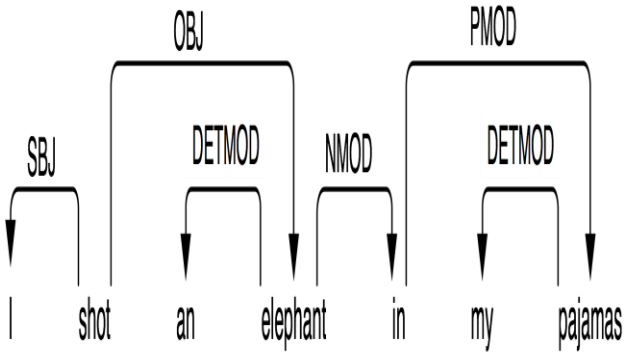
The count may be, How many times was this rule used with dump, divided by the number of VPs that dump appears in total.

3.2 Preferences

Consider the verb phrase 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[11].

3.3 Dependency grammars

- Model **binary** dependencies between words
- For instance:
  - I eat
  - eat apples
- Find the set of dependencies that best fits the input words (I.e. with no contradictions)



3.4 Evaluating parsers:-There are certain parsers which studies in Probabilistic estimation including

1)Precision-# of correct dependencies from the parse / total # of dependencies in the parse

2)Recall-# of correct dependencies from the parse / total # of dependencies in the treebank (gold standard)

3)Cross-brackets # of crossing brackets in the parse and the treebank

- E.g. (A (B C)) and ((A B) C) has one crossing bracket

IV. PROBABILISTIC PARSING WITH CONTEXT FREE GRAMMARS USING DYNAMIC PROGRAMMING

We need a method that fills a table with partial results that

a. Does not do (avoidable) repeated work
b. Does not fall prey to left-recursion
c. Solves an exponential problem in (approximately) polynomial time

4.1 Earley Parsing

Fills a table in a single sweep over the input words Table is length N+1; N is number of words Table entries represent Completed constituents and their locations
In-progress constituents Predicted constituents

4.2 States

The table-entries are called states and are represented with dotted-rules.

S -> · VP	A VP is predicted
NP -> Det · Nominal	An NP is in progress
VP -> V NP ·	A VP has been found

4.2.1 States/Locations

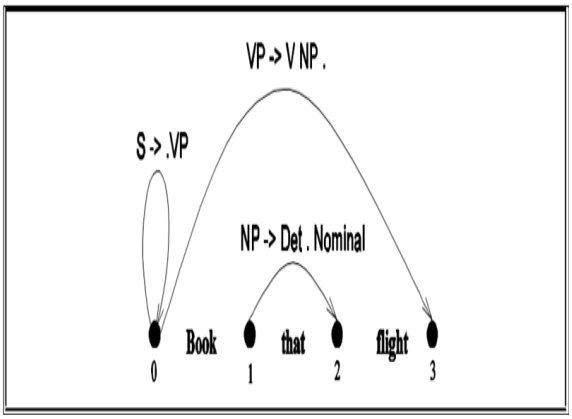
It would be nice to know where these things are in the input so…
S -> · VP [0,0] Predictor A VP is predicted at the start of the sentence

NP -> Det · Nominal[1,2]Scanner

An NP is in progress;
The Det goes from 1 to 2

VP -> V NP · [0,3] Completer
A VP has been found tarting at 0 and ending at 3

4.3 Graphical representation of Production rules in statistical Parsing:-



V. EARLEY PARSING

As with most dynamic programming approaches, the answer is found by looking in the table in the right place.In this case, there should be an S state in the final column that spans from 0 to n+1 and is complete.

- If that's the case you're done.
  - S -> α · [0,n+1]
  - So sweep through the table from 0 to n+1…
  - Predictor: New predicted states are created by states in current chart
  - Scanner: New incomplete states are created by advancing existing states as new constituents are discovered

- Completer: New complete states are created in the same way[11].
- More specifically…
- 1. Predict all the states
- 2. Read a word
- Extend states based on matches
- Add new predictions
- Go to 2
- 3. Look at N+1 to see if you have a winner
- 5.1 Earley parsing and Left Recursion
- So Earley solves the left-recursion problem without having to alter the grammar or artificially limit the search
- Never place a state into the chart that's already there
- Copy states before advancing them with productions

  S -> NP VP

  NP -> NP PP

  - The first rule predicts

    S -> · NP VP [0,0] that adds

    NP -> · NP PP [0,0]

    stops there since adding any subsequent prediction would be fruitless
  - When a state gets advanced make a copy and leave the original alone
    - Say we have

      NP -> · NP PP [0,0]
    - We find an NP from 0 to 2 so we create

      NP -> NP · PP [0,2]
    - But we leave the original state

Example

Book that flight

We should find… an S from 0 to 3 that is a completed state…

Chart[0]

$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
$NP \rightarrow \bullet Det\ NOMINAL$	[0,0]	Predictor
$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
$S \rightarrow \bullet VP$	[0,0]	Predictor
$VP \rightarrow \bullet Verb$	[0,0]	Predictor
$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor

Chart[1]

$Verb \rightarrow book \bullet$	[0,1]	Scanner
$VP \rightarrow Verb \bullet$	[0,1]	Completer
$S \rightarrow VP \bullet$	[0,1]	Completer
$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
$NP \rightarrow \bullet Det\ NOMINAL$	[1,1]	Predictor
$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor

Chart[2]

$Det \rightarrow that \bullet$	[1,2]	Scanner
$NP \rightarrow Det \bullet NOMINAL$	[1,2]	Completer
$NOMINAL \rightarrow \bullet Noun$	[2,2]	Predictor
$NOMINAL \rightarrow \bullet Noun\ NOMINAL$	[2,2]	Predictor

CONCLUSION:-

N-grams are more precise than (N-1)grams But also, N-grams are more sparse than (N-1) grams attempt N-grams and back-off to (N-1) if counts are not available E.g. attempt prediction using 4-grams, and back-off to trigrams (or bigrams, or unigrams) if counts are not available. In this paper I tried to explain different type of parsing methods ,parsing techniques and grammar rules which are used in probabilistic parsing

REFERENCES

[1] Akhmatova, Elena. 2005. Textual entailment resolution via atomic propositions. In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment.

[2] Bos, Johan and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment.

[3] Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2005.The PASCAL Recognising Textual Entailment Challenge.In Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment.

[4] de Marneffe, Marie-Catherine, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D.Manning. 2006. Learning to distinguish valid textual entailments.In Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment.

[5] Fowler, Abraham, Bob Hauser, Daniel Hodges, Ian Niles,Adrian Novischi, and Jens Stephan. 2005. Applying COGEX to recognize textual entailment. In Proceedings of the PASCAL ChallengesWorkshop on Recognising Textual Entailment.

[6] Harabagiu, Sanda and Andrew Hickl. 2006. Using scenario knowledge in automatic question answering. In Proceedings of the Workshop on Task-Focused Summarization and Question Answering, pages 32–39, Sydney.

[7] Harmeling, Stefan. 2007. An extensible probabilistic transformation-based approach to the Third Recognizing Textual Entailment Challenge. In ACL-07 Workshop on Textual Entailment and Paraphrasing, Prague.

[8] Hickl, Andrew, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizingtextual entailment with LCC's GROUNDHOG system. In Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment.

[9] Jiang, Jay J. and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings of the International Conference on Research in Computational Linguistics.

[10] Jijkoun, Valentin and Maarten de Rijke. 2005. Recognizing textual entailment using lexical similarity. In Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment, pages 73–76.

[11] Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In Proceedings of ACL-03, Sapporo.Lakoff, George. 1970. Linguistics and natural logic. Synthese,22:151–271.

[12] Levy, Roger and Galen Andrew. 2006. Tregex and Tsurgeon:tools for querying and manipulating tree data structures.In Proceedings of LREC-06, Genoa.

[13] MacCartney, Bill and Christopher D. Manning. 2007. Natural logic for textual inference. In ACL-`07 Workshop on Textual Entailment and Paraphrasing, Prague.MacCartney, Bill, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning.2006. Learning to recognize features of valid textual entailments. In Proceedings of NAACL-06, New York.

[14] Marsi, Erwin and Emiel Krahmer. 2005. Classification of semantic relations by humans and machines. In ACL-05 Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor.

[15] Nairn, Rowan, Cleo Condoravdi, and Lauri Karttunen. 2006.Computing relative polarity for textual inference. In Proceedings of ICoS-5 (Inference in Computational Semantics),Buxton, UK.

[16] Romano, Lorenza, Milen Kouylekov, Idan Szpektor, Ido Dagan,and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In Proceedings of EACL 2006.

[17] S´anchez Valencia, Victor. 1991. Studies on Natural Logic and Categorial Grammar. Ph.D. thesis, University of Amsterdam.

[18] van Benthem, Johan. 1986. Essays in logical semantics. Reidel,Dordrecht.van der Sandt, Rob A. 1992. Presupposition projection as anaphora resolution. Journal of Semantics, 9(4):333–377