PROBABILISTIC CONTEXT FREE GRAMMARS AND STATISTICAL PARSING

Saurabh Prasad

Center of Advanced Vehicular Systems
Mississippi State University
prasad @cavs.msstate.edu

ABSTRACT

Analyzing the structure of language is critical to many applications, e.g., understanding the meaning of a sentence, grammatical correction of written documents, large vocabulary continuous speech recognition etc. Parsing algorithms based on context free grammars provide speech researchers and linguists with tools to explore the structure of spoken and written language. Parsing paradigms based on context free grammars fail to resolve problems like inefficient reparsing of trees, left-recursive rules and ambiguity in context. Probabilistic natural language processing methods aim at resolving issues like ambiguity by assigning probabilities to various parts of the representation to determine the likelihood of a word sequence, interpretation and sentence structure. This paper reviews the current state of probabilistic context free grammars and statistical parsing techniques and summarizes various algorithms employed by a statistical parser.

1. INTRODUCTION

Grammar specifies the permissible structures of a language. Formal language theory provides linguists and speech researchers with useful tools to determine and exploit the generality, selectivity, and understandability of the grammar. Knowledge of this structure has many useful applications. As an example, compare the search space of a large vocabulary that has no defined grammar structure, with that of a large vocabulary having a well-defined grammar. This limited search space [1] provided by a well-structured grammar to a speech decoder illustrates one among the many practical uses of formal language theory.

Chomsky's formal language theory [2,3] defines grammar as a 4-tuple G = (V, T, P, S) where V and T are (finite) sets of terminals and non-terminals respectively, P is a set of production rules and S is a non-terminal, called the start symbol. Context Free Grammars (CFGs) have been described as being useful structures, powerful enough to specify most of the structure in spoken language. CFGs have been widely used in natural language processing due to their efficient parsing ability and a powerful representation of the structure of language.

In fact, CFGs serve as a bridge, connecting the class of stochastic language modeling techniques (e.g., N-Grams) with tools from formal grammar theory, provided it is possible to find a probabilistic model in the production rules of the grammar. Augmenting a regular CFG with these probabilities allows for it being used to capture the actual structure of spoken language, minimizing syntactic ambiguity. These Probabilistic Context Free Grammars (PCFGs) use this probabilistic model to more effectively distinguish between ambiguous choices, especially when the number of production rules is large. Further, in a speech recognition setup, PCFGs play an important role [4] in combining low level word models with higher level language models. They also provide a sound basis for ranking and pruning of parses.

Section 2 provides a background of parsing using regular CFGs and motivates the need for statistical parsing. Section 3 describes probabilistic context free grammars, and explains the various algorithms employed in a statistical parsing paradigm. Section 4 concludes the paper with benefits and weaknesses of statistical parsing techniques, and provides a parallel between statistical parsers and Hidden Markov Models.

2. CONTEXT FREE GRAMMARS AND PARSING

The language generated by a grammar G is context free if the production rule is of the general form

$$A \rightarrow \alpha \tag{1}$$

where the general form of a rewrite rule from a general (even context sensitive) grammar is given by

$$\phi A \varphi \rightarrow \phi \alpha \varphi \tag{2}$$

Here, A is a non-terminal, ϕ and φ are left and right contexts respectively and α is the replacement string. In either case, a production rule replaces a single non-terminal on the left side by a string of terminal/non-terminal symbols on the right side. Equation (1) depicts the independence of the production rule and the context, while equation (2) depicts a context sensitive production rule, dependent on the left and right contexts.

A typical parsing algorithm searches through various combinations of the grammatical (production) rules to find a

combination that generates a tree describing the structure of the input sentence accurately. Figure 1 depicts a parse tree generating the syntactic structure of the sentence 'I prefer coffee'. Given a lexicon, it is possible to construct input strings (of terminals) that have a constituent syntax satisfied by the grammar. Hence, in a typical application, a parser would take an input string (with unknown semantic structure) and search from among various possible trees to find an optimal parse tree - one that defines the structure of the input accurately.

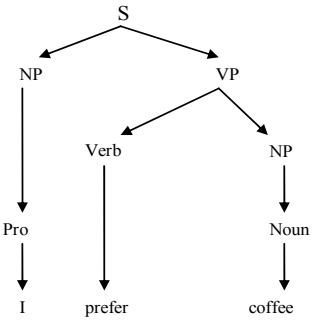

Figure 1: Illustrating the parse tree of the sentence 'I Prefer Coffee'

Standard goal-directed and data-directed parsers fail to resolve problems like inefficient reparsing of trees, left-recursive rules and ambiguity in context. The Earley algorithm [5] exploits the advantage of dynamic programming to solve these problems to some extent. However, huge grammars are typically riddled with ambiguity - e.g., hundreds of successful parse trees could be generated for a sentence having 20 words [5]. To make matters worse, making the grammar comprehensive would amplify this problem. Regular CFGs are not predictive, and hence fail to resolve ambiguity from a human perception point of view. Probabilistic natural language processing methods estimate grammar parameters by collecting counts from texts or structured analysis of texts. They aim at resolving issues like ambiguity by assigning probabilities to various parts of the representation to determine the likelihood of a word sequence, interpretation and sentence structure.

3. PROBABILISTIC CONTEXT FREE GRAMMARS AND STATISTICAL PARSING

Probabilistic Context Free Grammars are a natural extension of CFGs. A PCFG augments each production rule in the CFG with a probability. Hence, a PCFG is a 5-tuple G = (V, T, P, S, D) where V. T, P and S are defined previously. D is a mapping of each production rule in the grammar to a probability. This can be described as

$$A \rightarrow \alpha \quad [p]$$

or as, (3)

$$P(A \rightarrow \alpha \mid A)$$

This probability assignment of various syntactic constructions is backed by psycholinguistic research that correlates the likelihoods of various production rules to comprehension and reading difficulty [7].

One of the advantages of this probability mapping is the ability of a PCFG to assign a probability to every possible parse tree T of a sentence S. This helps in effective disambiguation. Figure 2 illustrates a sample ambiguity faced by parsing algorithms. The probability of a possible parse tree T is defined [5] to be the product of probabilities of every rule r used to expand every node n in the tree:

$$P(T,S) = \prod_{n \in T} P(r(n)) \quad (4)$$

Further, since P(S|T) is 1, the probability of a parse tree is:

$$P(T) = \frac{P(T,S)}{P(S \mid T)} \quad (5)$$
$$= P(T,S)$$

Hence, a typical disambiguation algorithm to select the best tree for a sentence S uses the intuition of picking the parse tree with the highest probability. To motivate on the appropriateness of this choice, consider the following analysis – we wish to find the most likely tree given S:

$$\hat{T}(S) = \arg\max_{T \in \tau(S)} P(T/S)$$
$$= \arg\max_{T \in \tau(S)} \frac{P(T,S)}{P(S)} \quad (6)$$
$$= \arg\max_{T \in \tau(S)} P(T)$$

where $\tau(S)$ represents the possible parse trees given S.

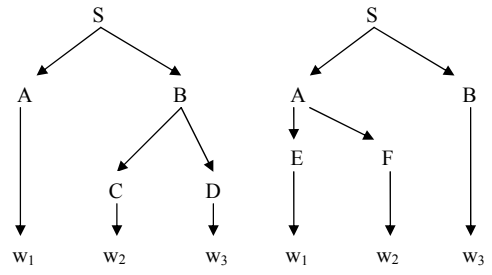

Figure 2: Illustrating two ambiguous parse trees for a string ''w1w2w3'

Hence, a statistical parsing paradigm based on PCFGs employs equation (6) to generate the most likely parse of S. A statistical parser must address the following issues:

- **Alignment problem**: Finding the optimal parse of the input sequence based on available parameterized PCFGs (Solution - CYK algorithm)
- **Scoring problem**: Finding the probability of a sequence given a parameterized PCFG (Solution - Inside algorithm)
- **Training problem**: Given a set of sequences, parameterizing the PCFG to best represent the training data (Solution - EM algorithm)

3.1 The Inside Algorithm

The inside algorithm computes probabilities $\alpha(i, j, A_k)$ of a sub-tree rooted at a non-terminal A_k for a subset of the input (observed) string sequence $w_i, \cdots, w_j$. Hence, this algorithm provides the likelihood that the substring $w_i, \cdots, w_j$ of the complete string sequence $\boldsymbol{w}$ is generated by the non-terminal A_k. Conceptually, this algorithm is similar to the forward algorithm in HMMs, which in the context of parse trees would generate the probability of everything above a certain node. Assuming that we are dealing with grammars in the Chomsky's normal form (i.e., allowing production rules of the form $A \rightarrow \alpha$ and $A \rightarrow BC$) the probability of A_k producing $w_i, \cdots, w_j$ by the production $A_k \rightarrow XY$ is given by [6]

$$\alpha(i, j, A_k) = \sum_X \sum_Y \sum_p \alpha(i, p, X)\alpha(k+1, j, Y)P(A_k \rightarrow XY) \quad (7)$$

In the above equation, we can sum up the individual probabilities of generating $w_i, \cdots, w_p$ (by the Non-Terminal X) and $w_p, \cdots, w_j$ (by the Non-Terminal Y) because these correspond to disjoint events. When A_k is a leaf, then i=j and alpha corresponds to $P(A_k \rightarrow XY)$.

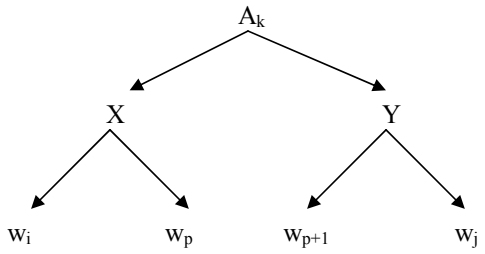

Figure 3: Illustrating the use of the inside algorithm for computing P(A_k -->'$w_i \ldots w_j$' | G)

3.2 The Outside Algorithm

The outside algorithm [6] computes the probability $\beta(i, j, k)$ of a complete tree rooted at the start symbol S for the complete string sequence $\boldsymbol{w}$, excluding all valid parse sub-trees for the sub-sequence $w_i, \cdots, w_j$ rooted at a non-terminal A_k.

$$\beta(i, j, A_k) = P(w_{1(i-1)}, A_k, w_{(j+1)n} \mid G) \quad (8)$$

This is similar conceptually to the backward probability for HMMs, which in the context of parse trees would represent the probability of everything below a certain node. Figure 4 illustrates the outside algorithm. Using arguments similar to those for deriving the inside probability, the outside probability is expressed as

$$\beta(i, j, A_k) = \sum_X \sum_Y \sum_{q>i} \alpha(i+1, q, X)\beta(i, q, Y)P(A_k \rightarrow XY) \quad (9)$$

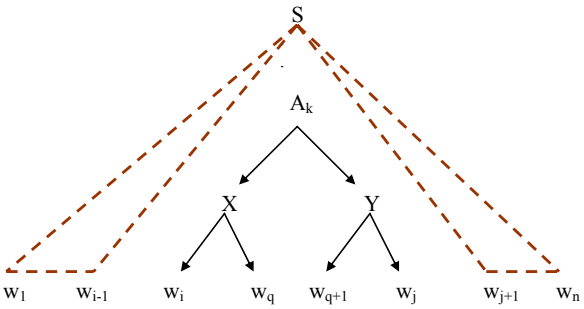

Figure 4 : Illustrating the use of the outside algorithm for computing P($w_{1(i-1)}$ A_k $w_{(j+1)n}$)

Note that since the production $A_k \Rightarrow^* w_i \cdots w_j$ can result from two possible intermediate productions $A_k \rightarrow XY$ and $A_k \rightarrow YX$, the sum in equation (9) must be computed over both possibilities. Another interesting point worth a mention here is that inside probabilities involve 'bottom-up' computation, while outside probabilities use 'top-down' computation.

3.3 Estimating PCFG parameters from Inside-Outside Probabilities

We can use the inside-outside probabilities to parameterize PCFGs using the well known Expectation-Maximization (EM) algorithm. Before heading towards using the EM algorithm for learning PCFG parameters, let us consider an easier approach – using a training corpus containing previously parsed sentences (referred to as a Treebank [5,

8]. An estimate of the probability of each production rule in the grammar can be made by the ratio

$$P(A \to \alpha) = \frac{Count(A \to \alpha)}{\sum_{\gamma} Count(A \to \gamma)} = \frac{Count(A \to \alpha)}{Count(A)} \quad (10)$$

When a Treebank is unavailable, we can estimate the counts in equation (10) by using the inside and outside probabilities. The EM estimation equations (11) and (12) for the probabilities of the production rules of a PCFG (in the Chomsky's normal form) are used to recursively update PCFG probabilities [6]. After appropriate initialization, a recursion through equations (11) and (12) will produce accurate production rule probabilities. The recursion can be terminated when the change in estimated probabilities is small.

$$P^{new}(A \to XY) = \frac{Count(A \to XY)}{Count(A)}$$

$$= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j-1} \beta(i, j, A)\, \alpha(k+1, j, z)\, P(A \to XY)}{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha(i, j, A)\beta(i, j, A)} \quad (11)$$

and

$$P^{new}(A \to \alpha) = \frac{Count(A \to \alpha)}{Count(A)}$$

$$= \frac{\sum_{i=1}^{n} \beta(i, i, A)\, P(A \to \alpha)}{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha(i, j, A)\beta(i, j, A)} \quad (12)$$

PCFG parameter estimation through the inside-outside algorithm is a slow process, and like any other gradient-descent or EM algorithm based learning technique, this algorithm can produce unwanted results if the recursion hits a local maxima [6]. Developing a PCFG in the absence of a Treebank is still an active research area.

3.4 The CYK (alignment) Algorithm

The Cocke-Younger-Kasami (CYK) algorithm finds the optimal parse given a parameterized PCFG, a set of possible (ambiguous) parses and the input string. The CYK algorithm is essentially a bottom-up parser using a dynamic programming structure similar to the one used in the Earley algorithm [5].

A dynamic programming variable, $\gamma(i,j,A_k)$, is defined as the likelihood of the most likely parse of $w_i, \cdots, w_j$ rooted at the non-terminal A_k. This is computed using the most likely transition out of A_k. Upon termination, $\gamma(1,n,S)$ will provide the most likely parse of $\boldsymbol{w}$ satisfying the grammar G. When dealing with log-likelihoods, the algorithm is as follows [6]:

- Initialization:
  For i = 1 : Num_terminals
    For A = 1 : Num_Non_terminals
      $\gamma(i, i, A) = \log\{P(A \to w_i)\}$
      $\Gamma(i, i, A) = (0, 0, 0)$
- Recursion:
  For i = 1 : n-1
    For j = 1 : i+1
      For A = 1 : num_non_terminals
      $\gamma(i, j, A) = \max_{y,z} \max_{k=i \ldots j-1} \{$
      $\gamma(i, k, y) + \gamma(k+1, j, z) + \log[P(A \to YZ)]\}$

      $\Gamma(i, j, A) = \max_{y,z} \max_{k=i \ldots j-1} \{$
      $\Gamma(i, k, y) + \Gamma(k+1, j, z) + \log[P(A \to YZ)]\}$
- Termination:
  $\log P[\boldsymbol{w} \mid \hat{\pi}, G] = \gamma(1, n, 1)$

After passing the entire word sequence through the CYK parser, γ will hold the log-likelihoods of all productions in the 'most-likely' parse $\hat{\pi}$, and Γ will hold the necessary information required to trace-back and reconstruct the most probable parse in a bottom-up fashion.

4. CONCLUSION

Section 3 summarizes a statistical parsing paradigm based on probabilistic context free grammars. As was indicated before, probability assignments of production rules allow for choosing the most probable parse among ambiguous parse trees for a given input string. The CYK algorithm provides for a convenient bottom-up decoding (alignment) of the input string sequence to the appropriate parse. This is equivalent to the use of the Viterbi algorithm in HMMs for finding the optimal alignment between the input stream and the possible HMMs.

The probabilities of a PCFG can be learned from either an annotated corpus (Treebank) or by using the inside-outside algorithm. The inside-outside algorithm described in section 3.3 uses the EM algorithm to compute the probabilities that result in the best parse. This is equivalent to the forward-backward algorithm used for learning the parameters $\{A,B,\pi\}$ of an HMM [9].

Though PCFGs provide a natural framework for modeling probabilistic dependencies in a parsing operation, they do have a few drawbacks. PCFGs assume invariance of

the production rule probabilities with respect to the position in the tree (HMMs similarly assume time-invariance of the emission probabilities of symbols from each state). Further, as indicated by equation (4), PCFGs assume that the probabilities of production rules are independent of the context. This problem is resolved by lexicalized PCFGs [7], which use head driven grammars to condition production rule probabilities on the corresponding lexical head. As pointed out previously, computing the PCFG probabilities efficiently and accurately from an un-annotated corpus is still an active research problem.

5. REFERENCES

[1] Neeraj Deshmukh, Aravind Ganapathiraju, Joseph Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, September 1999.

[2] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, *Spoken Language Processing*, Prentice Hall, 2001.

[3] J. Picone, "Fundamentals Of Speech Recognition," http://www.cavs.msstate.edu/hse/ies/publications/courses/ece_8463/
Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, USA.

[4] Andreas Stolcke, "An Efficient Probabilistic Context Free Parsing Algorithm that Computes Prefix Probabilities," *International Computer Science Institute*, November 1993.

[5] Daniel Jurasfky, James H. Martin, *Speech and Language Processing*, Prentice Hall, 2000.

[6] Terry Speed, "From hidden Markov models to stochastic CFGs",
http://www.stat.berkeley.edu/users/terry//Classes/s246.2002/Week9/week9b.pdf, Department of Statistics, University of California, Berkley, California, USA, 2002.

[7] Jeremy Gillmor Kahn, "Moving Beyond the Lexical Layer in Parsing Conversational Speech," *MA Thesis*, Department of Linguistics, University of Washington, Washington, USA, 2005.

[8] The Penn Treebank Project, "Descriptions and samples of annotated corpora,"
http://www.cis.upenn.edu/~treebank/
Linguistic Data Consortium, University of Pennsylvania, Pennsylvania, USA, 1999.

[9] LR Rabiner, "A tutorial on HMM and selected applications in speech recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286, Feb. 1989