Statistical Parsing Algorithms for Lexicalized Tree Adjoining Grammars

Anoop Sarkar

Advisor: Prof. Aravind Joshi
Overview

- Introduction

- Results
  - Determining if a probabilistic TAG is well-defined.
  - Computing inside probabilities: a statistical parser for TAGs.
  - Computing prefix probabilities.
  - Training a parser by combining labeled and unlabeled data.

- Proposed Work
Statistical Parsing

Pierre Vinken will join the board as a non-executive director
Language Modeling

**Input:** Pierre Vinken will join the . . .

**Word Prediction:**

```
S
  NP    VP
    Pierre Vinken will VP
      V NP
        join the board
        ? bored
```
Tree Adjoining Grammars

- The notion of predicate-argument structure is captured elegantly.
- Locality and independence assumptions.
- Simple and well-defined probability model.
- Parsing can be treated in two steps:
  1. Classification: structured labels (elementary trees) are assigned to each word in the sentence.
  2. Attachment: the elementary trees are connected to each other to form the parse.
Bilexical CFG with ‘features’ (Collins 1999)

NP{store} → [ NP{+H} SBAR{+gap} ]
SBAR{+gap} → [ WHNP S{+H}{+C}{+gap} ]
S{+gap} → [ NP{+C} SBAR{+H}{+gap} ]
VP{+gap} → [ VB{+H} TRACE{+C} NP ]
Tree Adjoining Grammars: Different Modeling of Bilexical Dependencies

The store bought last week IBM.
Probabilistic TAGs: Substitution

\[
\sum_{t'} P(t, \eta \rightarrow t') = 1
\]
\[ \mathcal{P}(t, \eta \rightarrow NA) + \sum_{t'} \mathcal{P}(t, \eta \rightarrow t') = 1 \]
Tree Adjoining Grammars

- Simpler model for parsing.
  Performance (Chiang 2000): 86.9% LR 86.6% LP ($\leq$ 40 words)

- Parsing can be treated in two steps:
  1. Classification: structured labels (elementary trees) are assigned to each word in the sentence.
  2. Attachment: Apply substitution or adjunction to combine the elementary trees to form the parse.

- Produces more than the phrase structure of each sentence.
  A more embellished parse in which phenomena such as predicate-argument structure, subcategorization and movement are given a probabilistic treatment.
Parsing as Classification and Attachment

- Assigning structured labels to each word results in an ‘almost parse’ (Srinivas 1997)
  - A probabilistic treatment of classification: SuperTagging
  - A heuristic treatment of attachment: Lightweight Dependency Analyzer

- This work: a probabilistic treatment of both classification and attachment

- Extension to a more unsupervised approach
  (combining labeled and unlabeled data)
Theory and Practice of Probabilistic TAGs

- Applications of probabilistic grammars involve one or more of the following tasks, quoted from (Jelinek and Lafferty 1991):
  - What is the probability that a given string $x$ is generated by a grammar? A probabilistic grammar is well-defined if:
    \[
    \sum_{n=1}^{\infty} \sum_{w_1w_2\ldots w_n \in \mathcal{V}} \mathcal{P}(s \rightarrow w_1w_2\ldots w_n) = 1
    \]
  - What is the single most likely parse (or derivation) for $x$?
  - What is the probability that $x$ occurs as a prefix of some string generated by the grammar?
  - How should the parameters (e.g., rule probabilities) be chosen?
Results: Overview

- A probabilistic grammar is well-defined if:

\[
\sum_{n=1}^{\infty} \sum_{w_1 w_2 \ldots w_n \in \mathcal{V}} P(s \rightarrow w_1 w_2 \ldots w_n) = 1
\]

- What is the single most likely parse (or derivation) for \(x\)?

- What is the probability that \(x\) occurs as a prefix of some string generated by the grammar?

- How should the parameters (e.g., rule probabilities) be chosen?
Consistent Probabilistic TAGs

- A probabilistic grammar is well-defined if:

\[
\sum_{n=1}^{\infty} \sum_{w_1 w_2 \ldots w_n \in V} P(s \rightarrow w_1 w_2 \ldots w_n) = 1
\]

- Is it enough to have the following conditions?

\textit{Substitution:} \quad \sum_{t'} P(t, \eta \rightarrow t') = 1

\textit{Adjunction:} \quad P(t, \eta \rightarrow NA) + \sum_{t'} P(t, \eta \rightarrow t') = 1

(a \textit{proper} Probabilistic TAG)
Consistent Probabilistic TAGs

- In the PCFG: $(p = 0.99): S \rightarrow SS$ and $(1-p): S \rightarrow a$

- Let $x_h$ be the total probability of all parses with height $h$.

- $x_h + 1 = 1 - p + p \cdot x_h^2$

- When $h \rightarrow \infty$: $x = 1 - p + p \cdot x^2$

- $x = (1/p) - 1$ since $x_h$ is increasing.

- If $p > 1/2$ then all parses cumulatively get probability $x < 1$. Thus, the model is deficient.
Consistent Probabilistic TAGs

\[
P(S_r \rightarrow \beta_1) = .5 \quad P(S_r \rightarrow \beta_2) = .5
\]
\[
P(S_1 \rightarrow \beta_1) = .01 \quad P(S_1 \rightarrow \beta_2) = .98
\]
\[
P(S_2 \rightarrow \beta_1) = .98 \quad P(S_2 \rightarrow \beta_2) = .01
\]
\[
P(S_3 \rightarrow \beta_1) = .01 \quad P(S_3 \rightarrow \beta_2) = .98
\]
\[
P(S_4 \rightarrow \beta_1) = .98 \quad P(S_4 \rightarrow \beta_2) = .01
\]
\[
P(S_3 \rightarrow NA) = .01 \quad P(S_4 \rightarrow NA) = .01
\]
Consistent Probabilistic TAGs

- First Result: An algorithm to decide whether a Probabilistic TAG is consistent
  - Describe TAG derivations as Markov branching processes.
    (described in proposal document)
  - Also can be shown by reducing Prob TAG grammar to a degenerate PCFG. (as suggested by Steve Abney)
  - Useful when we discuss the algorithm for computing prefix probabilities.
Consistent Probabilistic TAGs

\[
\begin{align*}
\alpha & \rightarrow S_r \\
\beta_1 & \rightarrow S_1 S_2 \\
S_1 & \rightarrow \beta_1 | \beta_2 | \epsilon \\
S_2 & \rightarrow \beta_1 | \beta_2 | \epsilon \\
S_r & \rightarrow \beta_1 | \beta_2 | \epsilon \\
\beta_2 & \rightarrow S_3 S_4 \\
S_3 & \rightarrow \beta_1 | \beta_2 | \epsilon \\
S_4 & \rightarrow \beta_1 | \beta_2 | \epsilon
\end{align*}
\]
Consistent Probabilistic TAGs

\[
\begin{align*}
\mathcal{P}(\alpha \rightarrow S_r) & = 1.0 \\
\mathcal{P}(S_r \rightarrow \beta_1) & = 0.5 \\
\mathcal{P}(S_r \rightarrow \beta_2) & = 0.5 \\
\mathcal{P}(S_r \rightarrow \epsilon) & = 0 \\
\mathcal{P}(\beta_1 \rightarrow S_1 S_2) & = 1.0 \\
\mathcal{P}(\beta_2 \rightarrow S_3 S_4) & = 1.0 \\
\mathcal{P}(S_1 \rightarrow \beta_1) & = 0.01 \\
\mathcal{P}(S_1 \rightarrow \beta_2) & = 0.98 \\
\mathcal{P}(S_1 \rightarrow \epsilon) & = 0.01 \\
\mathcal{P}(S_2 \rightarrow \beta_1) & = 0.98 \\
\mathcal{P}(S_2 \rightarrow \beta_2) & = 0.01 \\
\mathcal{P}(S_2 \rightarrow \epsilon) & = 0.01
\end{align*}
\]
Consistent Probabilistic TAGs

- Apply the PCFG result (Booth and Thompson 1973) on this grammar to check for consistency.

- Compute the expectation matrix $\mathcal{M}$ which contains expected values of observing a tree $t$ when rewriting a node $\eta$.

- Check that the spectral radius $\rho(\mathcal{M}) < 1$. $\rho(\mathcal{M})$ is the modulus of the largest eigenvalue of $\mathcal{M}$. 
Consistent Probabilistic TAGs

\[
M = \begin{bmatrix}
S_1 & S_2 & S_3 & S_4 & S_r & \alpha & \beta_1 & \beta_2 \\
S_1 & 0 & 0 & 0 & 0 & 0 & 0.01 & 0.98 \\
S_2 & 0 & 0 & 0 & 0 & 0 & 0.98 & 0.01 \\
S_3 & 0 & 0 & 0 & 0 & 0 & 0.01 & 0.98 \\
S_4 & 0 & 0 & 0 & 0 & 0 & 0.98 & 0.01 \\
S_r & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\
\alpha & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\
\beta_1 & 1.0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\
\beta_2 & 0 & 0 & 1.0 & 1.0 & 0 & 0 & 0
\end{bmatrix}
\]

- \( \rho(M) = 1.4071 \). The input Prob TAG is correctly tagged as inconsistent.
- A condition for consistency and an algorithm for detecting deficiency.
- [http://www.cis.upenn.edu/~anoop/distrib/consist/](http://www.cis.upenn.edu/~anoop/distrib/consist/)
A probabilistic grammar is well-defined if:

\[
\sum_{n=1}^{\infty} \sum_{w_1 w_2 \ldots w_n \in \mathcal{V}} \mathcal{P}(s \rightarrow w_1 w_2 \ldots w_n) = 1
\]

What is the single most likely parse (or derivation) for \( x \)?

What is the probability that \( x \) occurs as a prefix of some string generated by the grammar?

How should the parameters (e.g., rule probabilities) be chosen?
Head corner chart parser for TAGs. Based on (van Noord 1990) head corner traversal

Less average time/space complexity in practice compared to CKY for TAGs.

Tree classification step reduces parsing time dramatically.

ftp://ftp.cis.upenn.edu/pub/xtag/lem/
Results: Overview

- A probabilistic grammar is well-defined if:

$$\sum_{n=1}^{\infty} \sum_{w_1,w_2 \ldots w_n \in \mathcal{V}} P(s \rightarrow w_1w_2 \ldots w_n) = 1$$

- What is the single most likely parse (or derivation) for $x$?

- What is the probability that $x$ occurs as a prefix of some string generated by the grammar?

- How should the parameters (e.g., rule probabilities) be chosen?
Prefix Probabilities: (with M.J. Nederhof and G. Satta)

- What is the probability that $x$ occurs as a prefix of some string generated by the grammar?

- Language model: given a string $a_1, \ldots, a_{i-1}, a_i$ can be any word in the vocabulary $\Sigma$, what is $P(a_i \mid a_1, \ldots, a_{i-1})$?

- Standard techniques use trigram models:

$$P(a_i \mid a_{i-2}, a_{i-1})$$

- A stochastic grammar can be used by computing the prefix probability:

$$\sum_{w \in \Sigma^*} P(a_1, \ldots, a_i w)$$
Let prefix $\equiv abd$
Prefix Probabilities for CFGs (Jelinek and Lafferty 1991)
\[ \mathcal{P}([N_{\beta_1}, i, j, f_1, f_2]) = \sum_{\beta_2, f_1', f_2'} \mathcal{P}_{\text{outer}}([N_{\beta_1}, i, j, f_1, f_2], [\beta_2, f_1', f_2']) \times \mathcal{P}_{\text{split}}([R_{\beta_2}, i, j, f_1', f_2']) \]
Prefix Probabilities

- Derivations are a combination of two kinds of subderivations:
  
  1. potentially unbounded subderivations, independent of input
  2. bounded subderivations, depend on input symbols

- Problem: how to partition derivations uniquely into subderivations.

- Without unique partitions, algorithm will return incorrect probabilities.
Prefix Probabilities: Some Details

\[ \sum_w \mathcal{P}(a_1 \ldots a_nw) = \sum_{t \in \mathcal{T}} \mathcal{P}([t, 0, n, -, -]) \]

\[ \mathcal{P}([N, i, j, -, -]) = \]
\[ \mathcal{P}(N \rightarrow NA) \times \mathcal{P}([cdn(N), i, j, -, -]) + \]
\[ \sum_{k,l} \mathcal{P}([cdn(N), k, l, -, -]) \times \sum_t \mathcal{P}(N \rightarrow t) \times \mathcal{P}([t, i, j, k, l]) \]

\[ \mathcal{P}([\alpha N, i, j, -, -]) = \sum_k \mathcal{P}([\alpha, i, k, -, -]) \times \mathcal{P}([N, k, j, -, -]) \]
Prefix Probabilities: Some Details

\[ \sum_w \mathcal{P}([\alpha N, i, j, -, -]) = \sum_k \mathcal{P}([\alpha, i, k, -, -]) \times \mathcal{P}([N, k, j, -, -]) \]

\[ \mathcal{P}_{outer}([\alpha N, i, j, -, -], [t, f_1', f_2']) = \]
\[ \mathcal{P}_{outer}([\alpha, i, j, -, -]) \times \mathcal{P}([N, j, j, -, -]) + \]
\[ \mathcal{P}([\alpha, i, i, -, -]) \times \mathcal{P}_{outer}([N, i, j, -, -], [t, f_1', f_2']) \]
Prefix Probabilities: Some Details

\[ \sum_w P([\alpha N, i, j, -, -]) = \sum_k P([\alpha, i, k, -, -]) \times P([N, k, j, -, -]) \]

\[ P_{\text{split}}([\alpha N, i, j, -, -], [t, f'_1, f'_2]) = \]
\[ (\sum_k P([\alpha, i, k, -, -]) \times P([N, k, j, -, -])) + \]
\[ P_{\text{split}}([\alpha, i, j, -, -]) \times P([N, j, j, -, -]) + \]
\[ P([\alpha, i, i, -, -]) \times P_{\text{split}}([N, i, j, -, -], [t, f'_1, f'_2]) \]
Prefix Probabilities: Offline Computation

\[ M = \begin{bmatrix}
S_1 & S_2 & S_3 & S_4 & S_r & \alpha & \beta_1 & \beta_2 \\
S_1 & 0 & 0 & 0 & 0 & 0 & 0 & .01 & .98 \\
S_2 & 0 & 0 & 0 & 0 & 0 & 0 & .98 & .01 \\
S_3 & 0 & 0 & 0 & 0 & 0 & 0 & .01 & .98 \\
S_4 & 0 & 0 & 0 & 0 & 0 & 0 & .98 & .01 \\
S_r & 0 & 0 & 0 & 0 & 0 & 0 & .5 & .5 \\
\alpha & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\
\beta_1 & 1.0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\beta_2 & 0 & 0 & 1.0 & 1.0 & 0 & 0 & 0 & 0 
\end{bmatrix} \]
**Prefix Probabilities: Offline Computation**

- How to compute $P_{outer}(N_{\beta_1}, i, j, f_1, f_2, \beta_2, f'_1, f'_2)$?

- $Q = M + M^2 + M^3 + \ldots$

- $Q = M[\mathcal{I} - M]^{-1}$

- Compute $P([\alpha, i, i, -, -])$ in a similar way.
A probabilistic grammar is well-defined if:

\[
\sum_{n=1}^{\infty} \sum_{w_1 w_2 \ldots w_n \in \mathcal{V}} P(s \rightarrow w_1 w_2 \ldots w_n) = 1
\]

What is the single most likely parse (or derivation) for \( x \)?

What is the probability that \( x \) occurs as a prefix of some string generated by the grammar?

How should the parameters (e.g., rule probabilities) be chosen?
Training a Statistical Parser

- How should the parameters (e.g., rule probabilities) be chosen?

- Several alternatives:
  - Supervised training from a Treebank (Chiang 2000)
  - Parsing as Classification. Explore new machine learning techniques.
Open Issues in Lexicalized, Corpus-based Language Processing

- Adapting to new domains: training on one domain, testing (using) on another.

- Achieving higher performance when using limited amounts of annotated data.

- Separating structural (robust) aspects of the problem from lexical (sparse) ones.
  Explained in more detail later . . .
Statistical Parsing: Supervised vs. Unsupervised Methods

- “Stone soup” approaches to unsupervised learning of parsers cannot handle structurally rich parses found in the Penn Treebank. (Lafferty et al. 1992; Della Pietra et al. 1994; de Marcken 1995)

- A feasible technique: Combining Labeled and Unlabeled Data
  - Active Learning: Bet on which examples are the hardest. (and annotate them) (Hwa 2000)
  - Co-Training: Bet on which examples can be handled with high confidence. (use as labeled data)
Case Study in Unsupervised Methods: POS Tagging

- POS Tagging: finding categories for words

- ... the stocks $[\text{rose}] / V$ ... vs. ... a $[\text{rose}] / N$ bouquet ...

- Tag dictionary: $[\text{rose}]$: $N$, $V$
  and nothing else
Case Study: Unsupervised POS Tagging

- (Cutting et al. 1992) The Xerox Tagger: used HMMs with hand-built tag dictionaries. High performance: 96% on Brown

- (Merialdo 1994; Elworthy 1994) used varying amounts of labeled data as seed information for training HMMs.
  Conclusion: HMMs do not effectively combine labeled and unlabeled data

- (Brill 1997) aggressively used tag dictionaries taken from labeled data to train an unsupervised POS tagger.
  Performance: 95% on WSJ. Approach does not easily extend to parsing: no notion of tag dictionary.
Co-Training (Blum and Mitchell 1998; Yarowsky 1995)

- Pick two (or more) “views” of a classification problem.

- Build separate models for each of these “views” and train each model on a small set of labeled data.

- Sample an unlabeled data set and to find examples that the models agree upon the most. Exploit the mutual constraints between the models.

- Agreement can be computed as a simple product or in a more complex fashion. (Collins and Singer 1999; Goldman and Zhou 2000)

- Bet that these examples are good as training examples and iterate.
Pierre Vinken will join the board as a non-executive director
Recursion in Parse Trees

- Usual decomposition of parse trees:

  S(join) → NP(Vinken) VP(join)

  NP(Vinken) → Pierre Vinken

  VP(join) → will VP(join)

  VP(join) → join NP(board) PP(as)

  ...
Parsing as Tree Classification and Attachment: (Srinivas 1997; Xia 2000)

Model H1: $P(T_i \mid T_{i-2}T_{i-1}) \times P(w_i \mid T_i)$
Parsing as Tree Classification and Attachment

Model H2: $\mathcal{P}(\text{TOP} = w, T) \times \prod_i \mathcal{P}(w_i, T_i \mid \eta, w, T)$
The Co-Training Algorithm

1. Input: labeled and unlabeled

2. Update cache
   - If unlabeled is empty; exit
   - Randomly select sentences from unlabeled and refill cache

3. Train models H1 and H2 using labeled

4. Apply H1 to cache

5. Apply H2 to output of Step 4

6. Pick best $n$ given overall score combining H1 and H2

7. Remove best $n$ from cache and add to labeled

8. $n = 2n$; Go to Step 2
Preliminary Experiment

- *labeled* was set to Sections 02-06 of the Penn Treebank WSJ (9625 sentences)

- *unlabeled* was 30137 sentences (Section 07-21 of the Treebank stripped of all annotations).

- A TAG dictionary of all lexicalized trees from *labeled* and *unlabeled*. Similar to the approach of (Brill 1997)
  Novel trees were treated as unknown tree tokens

- The *cache* size was 3000 sentences.
Preliminary Experiment

- Test set: Section 0 (development test set)

- Baseline Model was trained only on the labeled set:
  Labeled Bracketing Precision = 67.43% Recall = 64.93%

- After 12 iterations of Co-Training:
  Labeled Bracketing Precision = 81.2% Recall = 78.94%

- NEW!: Evaluation of an unsupervised approach is directly comparable to other supervised parsers.
Summary

- Methods that combine labeled and unlabeled data provide a promising new direction towards unsupervised learning.

- Co-Training, previously used for classifiers with 2/3 labels, was extended to the complex problem of statistical parsing.

- Parsing treated as providing structured (tree) labels with attachments computed between these labels.

- Evaluation of an unsupervised method for parsing directly comparable with supervised approaches.
Proposed Work

- Evaluation of the Prefix Probability Parser.
- Further Evaluation of Co-Training.
- Learning Tag Dictionaries for Parsing.
- Integrate lexical knowledge acquisition into the Co-Training method.
Proposed Work: Evaluation of Prefix Probability Parser

- Modify existing parser to work from left to right.
- Note that we compute possible future contexts as well as histories. (Unlike (Chelba and Jelinek 1998; Johnson and Roark 1999))
- Is it better to parse word graphs/lattices rather than parse left to right?
- Compare perplexity with a backed-off trigram model. Combining labeled and unlabeled data useful here.
- Compare word-error rate.
Proposed Work: Further Evaluation of Co-Training

- Current Work: Improve parser (better smoothing); Better combination of the models.

- Experiment with using a larger labeled (1M words) and unlabeled set (23M words).

- Experiment with smaller corpora, across domains and using Tree-banks in other languages.

- Conjecture: Active Learning and Co-Training can be combined into a single framework.
Proposed Work: Learning Tag Dictionaries for Parsing

- Use machine learning techniques for learning the tag dictionary.
  - Subcategorization frame learning. (with D. Zeman)
    * Learnt 137 (lexicalized) subcat frames for Czech PDT.
    * Identified subcat frames in unseen data: 88% P, 74% R.
  - Learning Verb Classes. Pred-Argument structure (with W. Tripasai)
    * Classify V into $\text{NP}_0 \text{V} \text{NP}_1; \text{NP}_1 \text{V}; \text{NP}_0 \text{V}$
    * Using Decision Trees: Error Rate = 33.4%

- Integrate lexical knowledge acquisition into the Co-Training method.
Summary

- Results (Algorithms for . . .)
  - Determining if a probabilistic TAG is well-defined.
  - Computing inside probabilities: a statistical parser for TAGs.
  - Computing prefix probabilities.
  - Training a parser by combining labeled and unlabeled data.

- Proposed Work
  - Evaluation of the Prefix Probability Parser.
  - Further Evaluation of Co-Training.
  - Learning Tag Dictionaries for Parsing.
  - Integrate lexical knowledge acquisition into the Co-Training method.