Language modeling with tree substitution grammars

Matt Post
Daniel Gildea

NIPS workshop on Grammar Induction, Representation of Language, and Language Learning

December 11, 2009
1. certain learned TSGs (a) have lower perplexity and (b) are roughly the same size as the standard Treebank CFG
Standard CFG

Nonterminals rewrite as sequence of child nonterminals
Standard CFG

Nonterminals rewrite as sequence of child nonterminals

parse tree from training data
Standard CFG

Nonterminals rewrite as sequence of child nonterminals

TSG

Nonterminals rewrite as sequence of child subtrees, each of arbitrary size

parse tree from training data
Standard CFG

Nonterminals rewrite as sequence of child nonterminals

TSG

Nonterminals rewrite as sequence of child subtrees, each of arbitrary size

parse tree from training data
how do we learn a TSG?
how do we learn a TSG?

**DOP**

use *all* of the subtrees in the training data

(Bod, 2001)
how do we learn a TSG?

**DOP**

use *all* of the subtrees in the training data

(Bod, 2001)

**EM**

guess the derivations and count
Overview: Post & Gildea (2009)

(see also Cohn et al. (2009), Tenenbaum et al. (2009))

overfitting
use a Dirichlet Process prior that discourages large subtrees

\[ g_X \sim DP(\alpha, G_X) \]
\[ G_X(t) = \Pr_s(|t|; p_s) \prod_{r \in t} \Pr_{\text{MLE}}(r) \]
Overview: Post & Gildea (2009)

(see also Cohn et al. (2009), Tenenbaum et al. (2009))

**Overfitting**
use a Dirichlet Process prior that discourages large subtrees

\[ g_X \sim DP(\alpha, G_X) \]
\[ G_X(t) = \Pr_\$ (|t|; p_\$) \prod_{r \in t} \Pr_{MLE}(r) \]

**Space efficiency**
only maintain counts of subtrees from the set of existing derivations in the training data

Collapsed Gibbs sampling
(Goldwater et al., 2009)
Treebank initialization

spinal initialization
Treebank initialization

spinal initialization
Overview: Post & Gildea (2009)

- used: DOP
- used: sampled
- grammar: DOP
- grammar: sampled
Overview: Post & Gildea (2009)

- Used: DOP
- Used: Sampled
- Grammar: DOP
- Grammar: Sampled

Graph: Count vs. Rule Size (by token)
Overview: Post & Gildea (2009)

- used: DOP
- used: sampled
- grammar: DOP
- grammar: sampled

Graph showing the count of rules by size (in tokens) for different grammars and usage scenarios.
Overview: Post & Gildea (2009)

The graph shows the count of rules of different sizes (by token) for two grammars: DOP and sampled. The x-axis represents the rule size (by token), and the y-axis represents the count. The graph includes two lines:
- Green line: used: DOP
- Brown line: used: sampled
- Red line: grammar: DOP
- Grey line: grammar: sampled
Experiments

Treebank grammar

All rules have a depth of one
Treebank grammar
All rules have a depth of one

“spinal” grammar
TSG subtrees induced by maximally projecting each word
Treebank grammar

All rules have a depth of one

“spinal” grammar

TSG subtrees induced by maximally projecting each word
Treebank grammar

All rules have a depth of one

“spinal” grammar

TSG subtrees induced by maximally projecting each word

sampled grammar

TSG subtrees induced with a collapsed Gibbs sampler and Dirichlet Process prior
Okanohara and Tsujii (2007)

[banks investment Big]_{NP} refused to step up to [plate the]_{NP} to support [traders floor beleaguered the]_{NP} by buying [[of stock]_{PP} [blocks big]_{NP}]_{NP}, traders say.
Perplexity on pseudo-negative text

![Graph showing perplexity versus number of rules (thousands).]

- Treebank
- spinal+Treebank
- sampled+Treebank
- sampled
Flattening

TSG derivation in training corpus

internal nodes removed
2.
sampled TSGs lead to perplexity improvements with a bilexical parser, suggesting they are improving Treebank structure
CFG parsing
CFG parsing

1. Replace each nonterminal with its children in a single act
   \( \text{Pr}(\text{rhs} | \text{P}) \)
CFG parsing

1. Replace each nonterminal with its children in a single act
   \[ \text{Pr}(\text{rhs}| P) \]
2. Recurse
Bilexical parsing (Collins Model 1)

S
Bilexical parsing (Collins Model 1)

\[ S^{(VBZ/has)} \]

1. Generate the head word and tag
   \[ \text{Pr}(h, t \mid P) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ Pr(h,t \mid P) \]
2. Generate the head child
   \[ Pr(H \mid P,h,t) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ \text{Pr}(h,t \mid P) \]
2. Generate the head child
   \[ \text{Pr}(H \mid P,h,t) \]
3. Generate the sibling head labels and tags
   \[ \text{Pr}(C,c_t \mid P,h,t,H) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ \Pr(h,t | P) \]

2. Generate the head child
   \[ \Pr(H | P,h,t) \]

3. Generate the sibling head labels and tags
   \[ \Pr(C,c_t | P,h,t,H) \]

\[ S_{(VBZ/has)} \]
\[ \rightarrow \]
\[ VP_{(VBZ/has)} \].
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ Pr(h,t | P) \]
2. Generate the head child
   \[ Pr(H | P,h,t) \]
3. Generate the sibling head labels and tags
   \[ Pr(C,c_t | P,h,t,H) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ \text{Pr}(h,t \mid P) \]
2. Generate the head child
   \[ \text{Pr}(H \mid P,h,t) \]
3. Generate the sibling head labels and tags
   \[ \text{Pr}(C,c_t \mid P,h,t,H) \]
4. Generate the sibling head word
   \[ \text{Pr}(c_w \mid P,h,t,H,c_h,c_t) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \( \text{Pr}(h,t \mid P) \)
2. Generate the head child
   \( \text{Pr}(H \mid P,h,t) \)
3. Generate the sibling head labels and tags
   \( \text{Pr}(C,c_t \mid P,h,t,H) \)
4. Generate the sibling head word
   \( \text{Pr}(c_w \mid P,h,t,H,c_h,c_t) \)
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ \Pr(h,t \mid P) \]
2. Generate the head child
   \[ \Pr(H \mid P,h,t) \]
3. Generate the sibling head labels and tags
   \[ \Pr(C,c_t \mid P,h,t,H) \]
4. Generate the sibling head word
   \[ \Pr(c_w \mid P,h,t,H,c_h,c_t) \]
Bilexical parsing (Collins Model 1)

1. Generate the head word and tag
   \[ \text{Pr}(h,t \mid P) \]
2. Generate the head child
   \[ \text{Pr}(H \mid P,h,t) \]
3. Generate the sibling head labels and tags
   \[ \text{Pr}(C,c_t \mid P,h,t,H) \]
4. Generate the sibling head word
   \[ \text{Pr}(c_w \mid P,h,t,H,c_h,c_t) \]
5. Recurse
Raising

internal nodes removed

preterminals reintroduced
Conflict

NP (NN[fever]/fever)  VBZ[has] (VBZ/has)  VP (VBN/cooled) . (./.)
Conflict

NP (NN[fever]/fever)  VBZ[has] (VBZ/has)  VP (VBN/cooled) . (./.)

three-level interpolation
Conflict

NP (NN[fever]/fever)  VBZ[has] (VBZ/has)  VP (VBN/cooled)  

S (VBZ/has)

P(NP,NN[fever] | S,VBZ[has],VBZ,has,←)

three-level interpolation
Conflict

three-level interpolation

\[
P(NP, NN[fever] | S, VBZ[has], VBZ, \leftarrow) \\
P(NP, NN[fever] | S, VBZ[has], VBZ, \leftarrow)
\]
Conflict

three-level interpolation

\[ P(NP,NN[fever] \mid S,VBZ[has],VBZ,\text{has}, \leftarrow) \]

\[ P(NP,NN[fever] \mid S,VBZ[has],VBZ, \leftarrow) \]

\[ P(NP,NN[fever] \mid S,VBZ[has],\text{has}, \leftarrow) \]
TSG

Subtrees can extend down to the leaves of the parse tree

parse tree from training data
TSG

Subtrees can extend down to the leaves of the parse tree

Detached TSG

Words are detached from TSG subtrees at the preterminal before flattening and raising

parse tree from training data
Perplexities

The graph shows perplexities for different methods:
- Treebank
- spinal(d)
- sampled(d)
- sampled(d) + Treebank
- sampled(d) + spinal(d)

The x-axis represents grammatical perplexities, and the y-axis represents ungrammatical perplexities. The trend indicates that the combination of sampled(d) and Treebank methods performs better, as indicated by the lower perplexity values.
QUESTIONS
References


State-split grammars

![Graph showing perplexity vs. number of rules (thousands) for different grammars including Treebank, parent annotated, spinal+Treebank, sampled+Treebank, sampled, DOP+Treebank, and Berkeley. The graph indicates a trend where better performance is associated with a decrease in perplexity as the number of rules increases.](image-url)