

Kevin Knight's "Bayesian Inference with Tears"

Yanjun Ma

Centre for Next Generation Localisation,
Dublin City University,
Dublin 9, Ireland

Outline

Bayesian Inference

A toy example: tree substitution grammar

Example of success: Part-of-Speech tagging

Outline

Bayesian Inference

A toy example: tree substitution grammar

Example of success: Part-of-Speech tagging

Bayesian approach

Valuing prior knowledge (A Priori)

Bayesian approach

Valuing prior knowledge (A Priori)

As opposed to Frequentist approach

- “Frequent” is “Probable”

Bayesian approach

Valuing prior knowledge (A Priori)

As opposed to Frequentist approach

- “Frequent” is “Probable”

Bayes' theorem

$$\underbrace{P(H|D)}_{\text{posterior}} = \frac{\overbrace{P(D|H)}^{\text{likelihood}} \overbrace{P(H)}^{\text{prior}}}{P(D)}$$

- H : a specific hypothesis
- D : the observed data

Inference

Statistical inference

Inference

Statistical inference

- As opposed to descriptive inference

Inference

Statistical inference

- As opposed to descriptive inference
- Concerns a statistical model and data

Inference

Statistical inference

- As opposed to descriptive inference
- Concerns a statistical model and data
- The model makes up some “story” about how the data was generated (“generative story”, “modelling”)

Inference

Statistical inference

- As opposed to descriptive inference
- Concerns a statistical model and data
- The model makes up some “story” about how the data was generated (“generative story”, “modelling”)
- The conclusion of inference is normally an estimate, i.e. particular values that best approximates some parameters of interest in the model

Inference

Statistical inference

- As opposed to descriptive inference
- Concerns a statistical model and data
- The model makes up some “story” about how the data was generated (“generative story”, “modelling”)
- The conclusion of inference is normally an estimate, i.e. particular values that best approximates some parameters of interest in the model
- The specific inference techniques depend on the specific model (e.g. EM)

Inference

Statistical inference

- As opposed to descriptive inference
- Concerns a statistical model and data
- The model makes up some “story” about how the data was generated (“generative story”, “modelling”)
- The conclusion of inference is normally an estimate, i.e. particular values that best approximates some parameters of interest in the model
- The specific inference techniques depend on the specific model (e.g. EM)

Modelling: how is the story made up?

Modelling: how is the story made up?

- Fully parametric model: “a short story where the end is already known”

Modelling: how is the story made up?

- Fully parametric model: “a short story where the end is already known”

The probability distributions describing the data-generation process are assumed to be fully described by a family of probability distributions involving only a finite number of unknown parameters.

Modelling: how is the story made up?

- Fully parametric model: “a short story where the end is already known”

The probability distributions describing the data-generation process are assumed to be fully described by a family of probability distributions involving only a finite number of unknown parameters.

- Non-parametric model: “a serial story which runs depending on the mood of the author”

Modelling: how is the story made up?

- Fully parametric model: “a short story where the end is already known”

The probability distributions describing the data-generation process are assumed to be fully described by a family of probability distributions involving only a finite number of unknown parameters.

- Non-parametric model: “a serial story which runs depending on the mood of the author”

The model structure is not specified a priori but is instead determined from data. The term nonparametric is not meant to imply that such models completely lack parameters but that the number and nature of the parameters are flexible and not fixed in advance.

This Talk

Non-parametric Bayesian Inference

- Non-parametric \Rightarrow Parameters are not fixed in advance
- Bayesian \Rightarrow Prior is important
- Kevin Knight \Rightarrow Making ideas crystal clear with minimal maths

This Talk

Non-parametric Bayesian Inference

- Non-parametric \Rightarrow Parameters are not fixed in advance
- Bayesian \Rightarrow Prior is important
- Kevin Knight \Rightarrow Making ideas crystal clear with minimal maths

Unsupervised learning

- Supervised learning has been considered “no fun” in the world of Bayesian learning

This Talk

Non-parametric Bayesian Inference

- Non-parametric \Rightarrow Parameters are not fixed in advance
- Bayesian \Rightarrow Prior is important
- Kevin Knight \Rightarrow Making ideas crystal clear with minimal maths

Unsupervised learning

- Supervised learning has been considered “no fun” in the world of Bayesian learning

Most of the following contents are adapted from (Knight 2009)

- Kevin Knight. Bayesian Inference with Tears. *Tutorial, ISI/University of Southern California*, 2009.

Outline

Bayesian Inference

A toy example: tree substitution grammar

Example of success: Part-of-Speech tagging

Chinese Restaurant Process

The restaurant has an infinite number of round tables, each of which accommodates an infinite number of customers. When a new customer walks in, she can either i) start a new table and put up rules for this table, or ii) randomly picks an already-seated table and obey the rules of this table. Note that as a table collects more customers, it becomes an even more attractive target.

Tree substitution grammar (TSG)

CFG v.s TSG

- CFG is “context-free” ($NP \rightarrow DT\ NNS$)
- TSG takes more context into account using multilevel rules- “fragments” ($NP \rightarrow DT(\text{the})\ NNS$)

Tree substitution grammar (TSG)

CFG v.s TSG

- CFG is “context-free” ($NP \rightarrow DT\ NNS$)
- TSG takes more context into account using multilevel rules- “fragments” ($NP \rightarrow DT(\text{the})\ NNS$)

Practical issues

- Infeasible to enumerate all possible fragments from a treebank
- “Nobody can even tell how many TSG rules are implicit in the Treebank”
- Impractical to directly read *all* rules off the Treebank

Tree substitution grammar (TSG)

CFG v.s TSG

- CFG is “context-free” ($NP \rightarrow DT\ NNS$)
- TSG takes more context into account using multilevel rules- “fragments” ($NP \rightarrow DT(\text{the})\ NNS$)

Practical issues

- Infeasible to enumerate all possible fragments from a treebank
- “Nobody can even tell how many TSG rules are implicit in the Treebank”
- Impractical to directly read *all* rules off the Treebank

Modelling issues

- Parameter estimation (rule probabilities) is accused of bias and inconsistency
- Finding best derivation is tractable, best parse is not

Algorithm 1 How the Treebank came to be (Old v.s New)

- 1: **repeat**
 - 2: set $root(rule) = S$
 - 3: pick a rule according to probability $P(rule|root(rule))$ and expand the tree with this rule
 - 4: Recursively pick rules to expand newly-introduced nodes, until the leaves of the tree are all words
 - 5: **until** Treebank is generated

 - 1: **repeat**
 - 2: set $root(rule) = S$
 - 3: pick a rule according to probability $P(rule|root(rule) + \text{counts of rules used so far})$ and expand the tree with this rule
 - 4: Recursively pick rules to expand newly-introduced nodes, until the leaves of the tree are all words
 - 5: **until** Treebank is generated
-

Tree substitution grammar (TSG)

Respect the history: cache model

Tree substitution grammar (TSG)

Respect the history: cache model

$$P(\text{rule} | \text{root}(\text{rule}) + \text{counts of rules used so far}) =$$

Tree substitution grammar (TSG)

Respect the history: cache model

$$P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) = \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})}$$

Tree substitution grammar (TSG)

Respect the history: cache model

$$P(\text{rule} | \text{root}(\text{rule}) + \text{counts of rules used so far}) = \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})}$$

How to pick the first rule: base distribution p_0

Tree substitution grammar (TSG)

Base distribution

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

1. with probability δ , quit

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

1. with probability δ , quit
2. with probability $1 - \delta$:

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

1. with probability δ , quit
2. with probability $1 - \delta$:
 - pick a number of children from a Poisson distribution (mean= λ)

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

1. with probability δ , quit
2. with probability $1 - \delta$:
 - pick a number of children from a Poisson distribution (mean= λ)
 - pick the identities of those children from a uniform distribution over symbols

Tree substitution grammar (TSG)

Base distribution

Generate-TSG-Rule(x)

1. with probability δ , quit
2. with probability $1 - \delta$:
 - pick a number of children from a Poisson distribution (mean= λ)
 - pick the identities of those children from a uniform distribution over symbols
 - call Generate-TSG-Rule(x) on each child x , to expand it further

Combining cache model with base distribution

$$P(\textit{rule}|\textit{root}(\textit{rule}) + \text{counts of rules used so far}) =$$

Combining cache model with base distribution

$$P(\textit{rule}|\textit{root}(\textit{rule}) + \text{counts of rules used so far}) = \\ \beta \times p_0(\textit{rule}|\textit{root}(\textit{rule})) +$$

Combining cache model with base distribution

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\beta \times p_0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \beta) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

Combining cache model with base distribution

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\beta \times p_0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \beta) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- β is often termed as “hyperparameter”

Combining cache model with base distribution

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\beta \times p_0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \beta) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- β is often termed as “hyperparameter”
- “hyperparameter” and base distribution are the two parameters of Dirichlet Process

Combining cache model with base distribution

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\beta \times p_0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \beta) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- β is often termed as “hyperparameter”
- “hyperparameter” and base distribution are the two parameters of Dirichlet Process
- Recall Chinese Restaurant Process ...

Bias towards cache model: “rich get richer”

Bias towards cache model: “rich get richer”

$$P(\textit{rule}|\textit{root}(\textit{rule}) + \text{counts of rules used so far}) =$$

Bias towards cache model: “rich get richer”

$$P(\textit{rule}|\textit{root}(\textit{rule}) + \text{counts of rules used so far}) = \\ \frac{\alpha}{\alpha+H} \times p0(\textit{rule}|\textit{root}(\textit{rule})) \quad +$$

Bias towards cache model: “rich get richer”

$$\begin{aligned}
 P(\text{rule} | \text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\frac{\alpha}{\alpha + H} \times p0(\text{rule} | \text{root}(\text{rule})) && + \\
 (1 - \frac{\alpha}{\alpha + H}) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

Bias towards cache model: “rich get richer”

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\frac{\alpha}{\alpha+H} \times p0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \frac{\alpha}{\alpha+H}) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- Replacing β with $\frac{\alpha}{\alpha+H}$

Bias towards cache model: “rich get richer”

$$\begin{aligned}
 P(\text{rule}|\text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\frac{\alpha}{\alpha+H} \times p0(\text{rule}|\text{root}(\text{rule})) && + \\
 (1 - \frac{\alpha}{\alpha+H}) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- Replacing β with $\frac{\alpha}{\alpha+H}$
- H (history) is the number of times non-terminal $\text{root}(\text{rule})$ has been expanded in the cache

Bias towards cache model: “rich get richer”

$$\begin{aligned}
 P(\text{rule} | \text{root}(\text{rule}) + \text{counts of rules used so far}) &= \\
 &\frac{\alpha}{\alpha + H} \times p_0(\text{rule} | \text{root}(\text{rule})) && + \\
 (1 - \frac{\alpha}{\alpha + H}) \frac{\text{count}(\text{rule in cache})}{\sum_{r \in \text{rules with same root as rule}} \text{count}(r \text{ in cache})} &
 \end{aligned}$$

- Replacing β with $\frac{\alpha}{\alpha + H}$
- H (history) is the number of times non-terminal $\text{root}(\text{rule})$ has been expanded in the cache
- α is often termed as “concentration parameter”

I drew a box around this formula

$$P(\textit{rule}|\textit{root}(\textit{rule})) = \frac{\alpha \times p_0(\textit{rule}|\textit{root}(\textit{rule})) + \textit{count}(\textit{rule in cache})}{\alpha + \underbrace{\textit{count}(\textit{root}(\textit{rule}) \textit{expanded in cache})}_H}$$

I drew a box around this formula

$$P(\textit{rule}|\textit{root}(\textit{rule})) = \frac{\alpha \times p_0(\textit{rule}|\textit{root}(\textit{rule})) + \textit{count}(\textit{rule in cache})}{\alpha + \underbrace{\textit{count}(\textit{root}(\textit{rule}) \textit{expanded in cache})}_H}$$

Some reflection

I drew a box around this formula

$$P(\textit{rule}|\textit{root}(\textit{rule})) = \frac{\alpha \times p_0(\textit{rule}|\textit{root}(\textit{rule})) + \textit{count}(\textit{rule} \textit{ in cache})}{\alpha + \underbrace{\textit{count}(\textit{root}(\textit{rule}) \textit{ expanded in cache})}_H}$$

Some reflection

- The base distribution can incorporate prior knowledge

I drew a box around this formula

$$P(\textit{rule}|\textit{root}(\textit{rule})) = \frac{\alpha \times p_0(\textit{rule}|\textit{root}(\textit{rule})) + \textit{count}(\textit{rule in cache})}{\alpha + \underbrace{\textit{count}(\textit{root}(\textit{rule}) \textit{expanded in cache})}_H}$$

Some reflection

- The base distribution can incorporate prior knowledge
- The distribution is not pre-determined as the cache model is dynamically updating

I drew a box around this formula

$$P(\text{rule}|\text{root}(\text{rule})) = \frac{\alpha \times p_0(\text{rule}|\text{root}(\text{rule})) + \text{count}(\text{rule in cache})}{\alpha + \underbrace{\text{count}(\text{root}(\text{rule}) \text{ expanded in cache})}_H}$$

Some reflection

- The base distribution can incorporate prior knowledge
- The distribution is not pre-determined as the cache model is dynamically updating
- However, most crucially, how to find a “good” base distribution using the data on hand?

Outline

Bayesian Inference

A toy example: tree substitution grammar

Example of success: Part-of-Speech tagging

Part-of-Speech (POS) tagging

Part-of-Speech (POS) tagging

HMM model

Part-of-Speech (POS) tagging

HMM model

$$p(w, t) = p(t)p(w|t) = \prod_{i=1}^n p(t_i|t_{i-1}) \prod_{i=1}^n p(w_i|t_i)$$

Part-of-Speech (POS) tagging

HMM model

$$p(w, t) = p(t)p(w|t) = \prod_{i=1}^n p(t_i|t_{i-1}) \prod_{i=1}^n p(w_i|t_i)$$

- Unsupervised setup (list of words associated with all legal POS tags)

Part-of-Speech (POS) tagging

HMM model

$$p(w, t) = p(t)p(w|t) = \prod_{i=1}^n p(t_i|t_{i-1}) \prod_{i=1}^n p(w_i|t_i)$$

- Unsupervised setup (list of words associated with all legal POS tags)
- Can be learned with EM (forward-backward algorithm)

Bayesian model for POS tagging

Emission and transition

Bayesian model for POS tagging

Emission and transition

$$p(w, t) = \prod_{i=1}^n \frac{\alpha_1 \times p_0(t_i|t_{i-1}) + \text{count}("t_{i-1}t_i" \text{ in cache})}{\alpha_1 + \text{count}("t_{i-1}" \text{ in cache})}$$

Bayesian model for POS tagging

Emission and transition

$$p(w, t) = \prod_{i=1}^n \frac{\alpha_1 \times p_0(t_i | t_{i-1}) + \text{count}("t_{i-1}t_i" \text{ in cache})}{\alpha_1 + \text{count}("t_{i-1}" \text{ in cache})} \\ \times \prod_{i=1}^n \frac{\alpha_2 \times p_0(w_i | t_i) + \text{count}("t_i w_i" \text{ in cache})}{\alpha_2 + \text{count}("t_i" \text{ in cache})}$$

Bayesian model for POS tagging

Emission and transition

$$\begin{aligned}
 p(w, t) &= \prod_{i=1}^n \frac{\alpha_1 \times p_0(t_i | t_{i-1}) + \text{count}("t_{i-1}t_i" \text{ in cache})}{\alpha_1 + \text{count}("t_{i-1}" \text{ in cache})} \\
 &\times \prod_{i=1}^n \frac{\alpha_2 \times p_0(w_i | t_i) + \text{count}("t_i w_i" \text{ in cache})}{\alpha_2 + \text{count}("t_i" \text{ in cache})}
 \end{aligned}$$

- Two cache models
- Base distributions for emission and transition

Improving parameter estimation for base distributions

Improving parameter estimation for base distributions

Fitting the training data

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation
4. Collect fractional counts off each derivation, weighted by $p(t|w)$

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation
4. Collect fractional counts off each derivation, weighted by $p(t|w)$
5. Normalise counts to get new values for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation
4. Collect fractional counts off each derivation, weighted by $p(t|w)$
5. Normalise counts to get new values for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$

However, step 2 is infeasible:

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation
4. Collect fractional counts off each derivation, weighted by $p(t|w)$
5. Normalise counts to get new values for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$

However, step 2 is infeasible:

- Cannot use forward-backward trick because the cache model is dynamically updating

Improving parameter estimation for base distributions

Fitting the training data

1. Uniform distribution for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$
2. For **every derivation** t of w (all possible tag sequences), compute $p(w, t)$
3. Normalise those values to get $p(t|w)$ for each derivation
4. Collect fractional counts off each derivation, weighted by $p(t|w)$
5. Normalise counts to get new values for $p(t_i|t_{i-1})$ and $p(w_i|t_i)$

However, step 2 is infeasible:

- Cannot use forward-backward trick because the cache model is dynamically updating
- Sampling!

Bayesian Inference: sampling

Bayesian Inference: sampling

Before entering a different world...

Bayesian Inference: sampling

Before entering a different world...

- enumerate \Rightarrow sample (verb)

Bayesian Inference: sampling

Before entering a different world...

- enumerate \Rightarrow sample (verb)
- all derivations \Rightarrow some derivations

Bayesian Inference: sampling

Before entering a different world...

- enumerate \Rightarrow sample (verb)
- all derivations \Rightarrow some derivations
- fractional counts \Rightarrow whole counts

Bayesian Inference: sampling

Before entering a different world...

- enumerate \Rightarrow sample (verb)
- all derivations \Rightarrow some derivations
- fractional counts \Rightarrow whole counts

How to do sampling?

Bayesian Inference: sampling

Before entering a different world...

- enumerate \Rightarrow sample (verb)
- all derivations \Rightarrow some derivations
- fractional counts \Rightarrow whole counts

How to do sampling?

- One method is Gibbs sampling (one example of Markov chain Monte Carlo methods)

Gibbs sampling

Gibbs sampling

1. Start with some initial sample (e.g. a random tagging of the corpus)

Gibbs sampling

1. Start with some initial sample (e.g. a random tagging of the corpus)
2. Make some small changes to the sample

Gibbs sampling

1. Start with some initial sample (e.g. a random tagging of the corpus)
2. Make some small changes to the sample
3. Collect counts off the new sample

Gibbs sampling

1. Start with some initial sample (e.g. a random tagging of the corpus)
2. Make some small changes to the sample
3. Collect counts off the new sample
4. Until tired, go to 2

Gibbs sampling

1. Start with some initial sample (e.g. a random tagging of the corpus)
2. Make some small changes to the sample
3. Collect counts off the new sample
4. Until tired, go to 2

Tagging accuracy for Bayesian model

- Bigram: 85% (81% using plain EM)

Comments

Comments

- The small change is user-defined and problem-specific.

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)
- Any idea to do TSG parsing with similar methods to POS tagging?

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)
- Any idea to do TSG parsing with similar methods to POS tagging?

Other applications

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)
- Any idea to do TSG parsing with similar methods to POS tagging?

Other applications

- Chinese word segmentation

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)
- Any idea to do TSG parsing with similar methods to POS tagging?

Other applications

- Chinese word segmentation
- DOP parsing

Comments

- The small change is user-defined and problem-specific.
- Any efficient methods to score neighboring derivations? (cf. Exchangeability)
- Any idea to do TSG parsing with similar methods to POS tagging?

Other applications

- Chinese word segmentation
- DOP parsing
- Synchronous context free grammar for Machine Translation (ITG)

