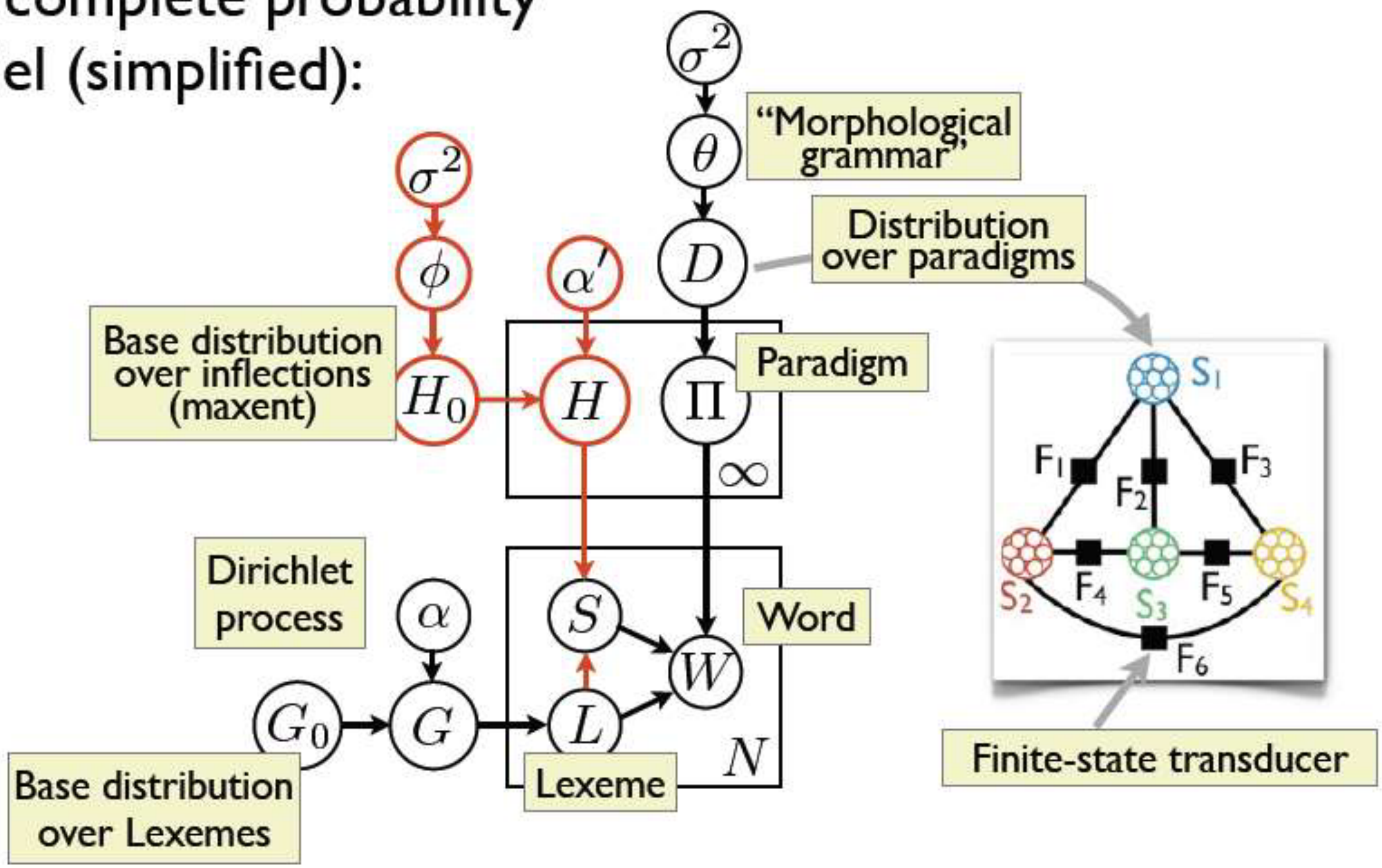
Deep Learning of **Recursive** Structure: Grammar Induction

Jason Eisner
Johns Hopkins University
ICLR 2013

With Henry Pao. Thanks also to Darcey Riley, Matt Gormley, Michael Tontchev.

Representation learning?

The complete probability model (simplified):



Representation learning?

Representation learning!

very deep

Representation learning!

When I saw deep belief networks in 2006, I wondered:

> "Could greedily trained overcomplete representations help in grammar induction?"

In this talk, I'll explain why this might help and suggest a possible architecture.

You’ll see the usual ideas …

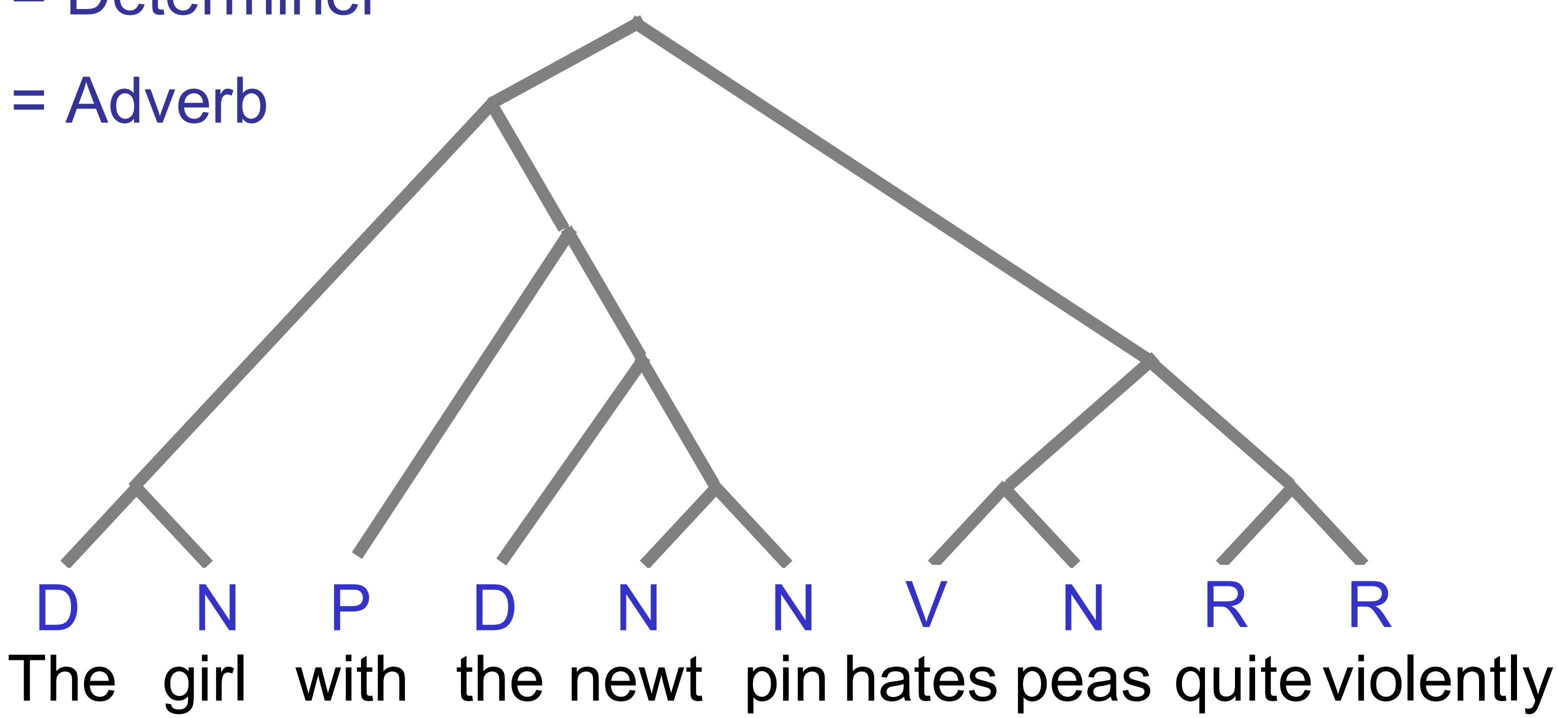
- autoencoders, bottlenecks
- convolutional parameters
- sum-product networks
- stacked training
- feature dictionaries
- word embeddings
- gradient dilution
- supervised fine-tuning

… but they'll look different

- autoencoders, bottlenecks
- convolutional parameters
- sum-product networks
- stacked training
- feature dictionaries
- word embeddings
- gradient dilution
- supervised fine-tuning

Tree structure

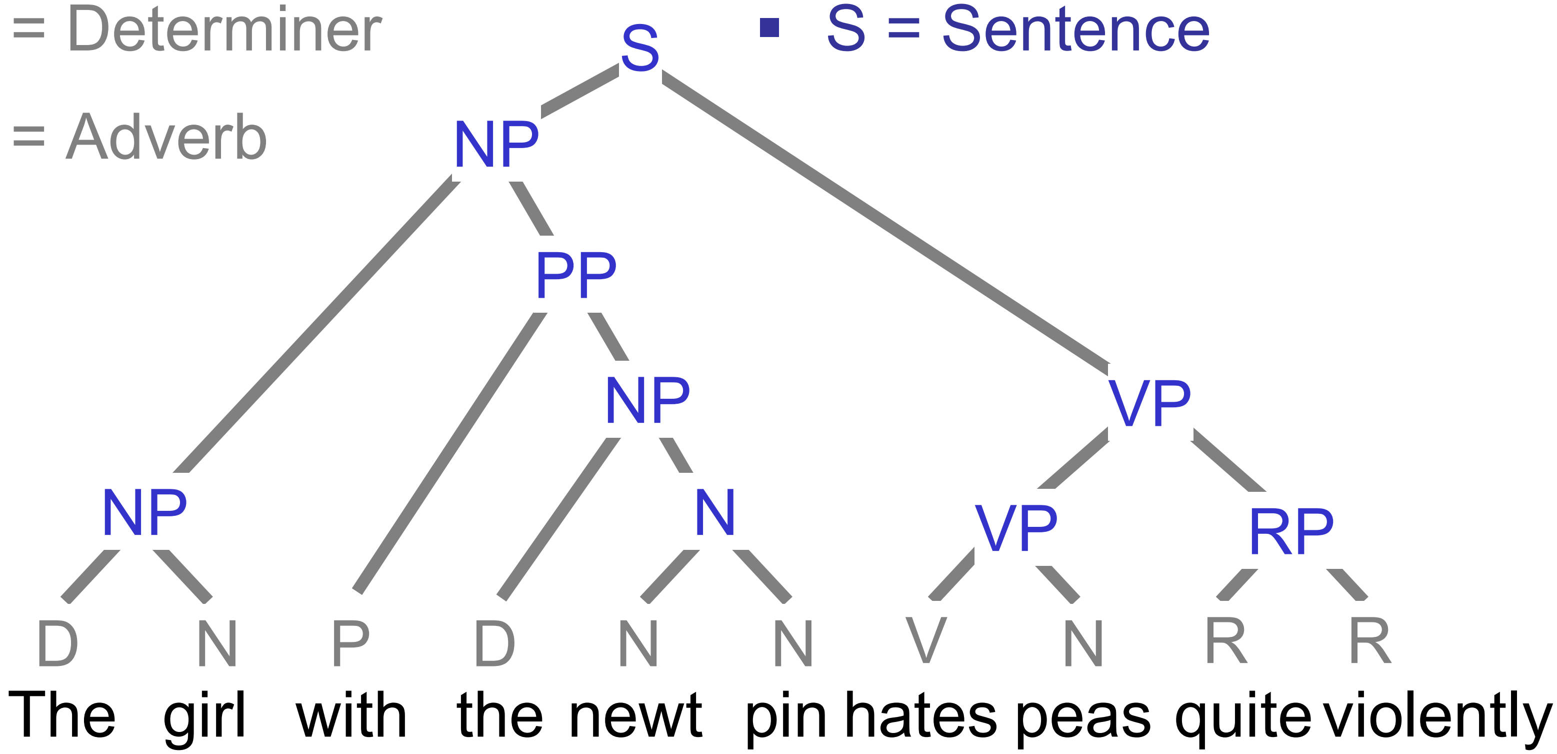
- N = Noun
- V = Verb
- P = Preposition
- D = Determiner
- R = Adverb



Tree structure

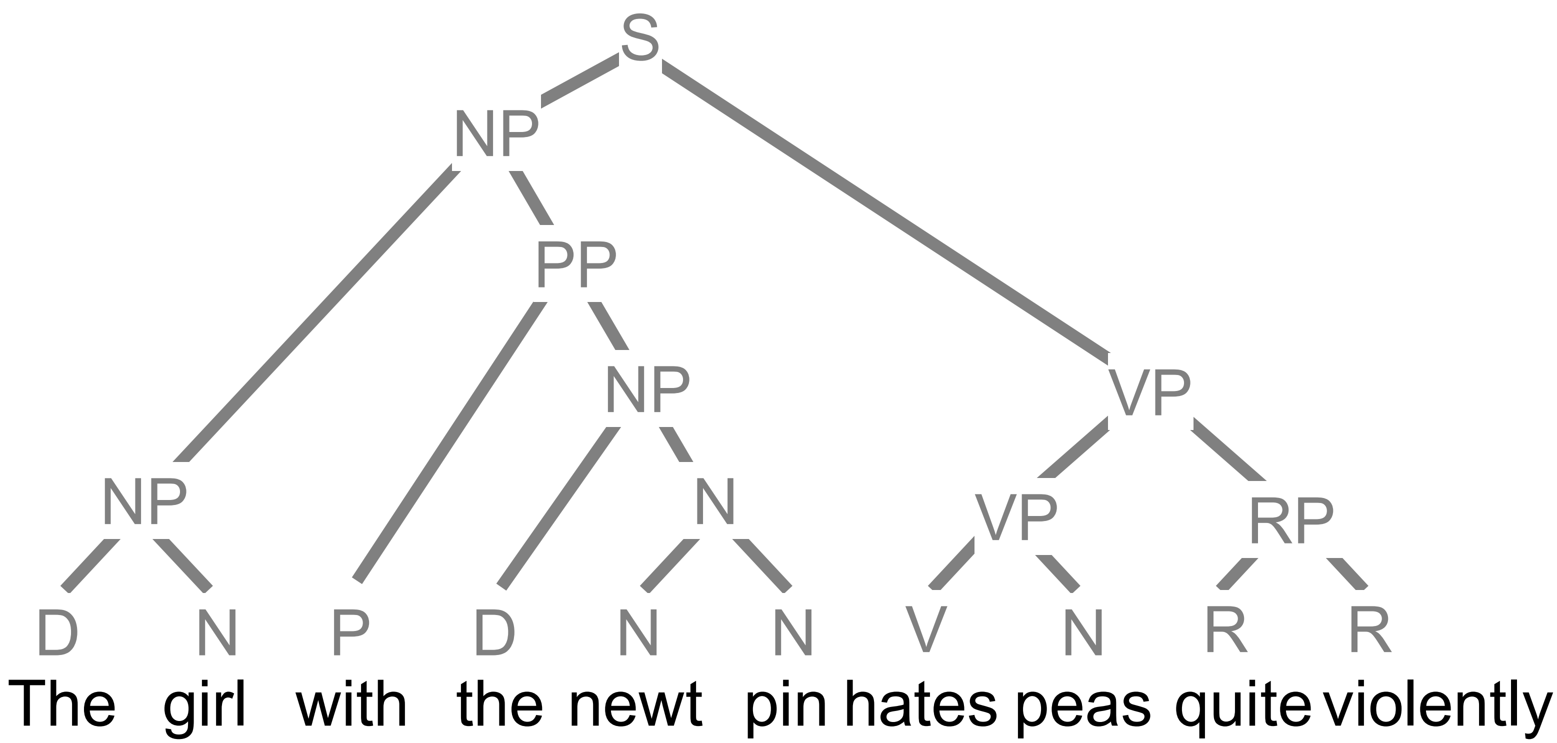
- N = Noun
- V = Verb
- P = Preposition
- D = Determiner
- R = Adverb

- NP = Noun phrase
- VP = Verb phrase
- PP = Prepositional phrase
- S = Sentence



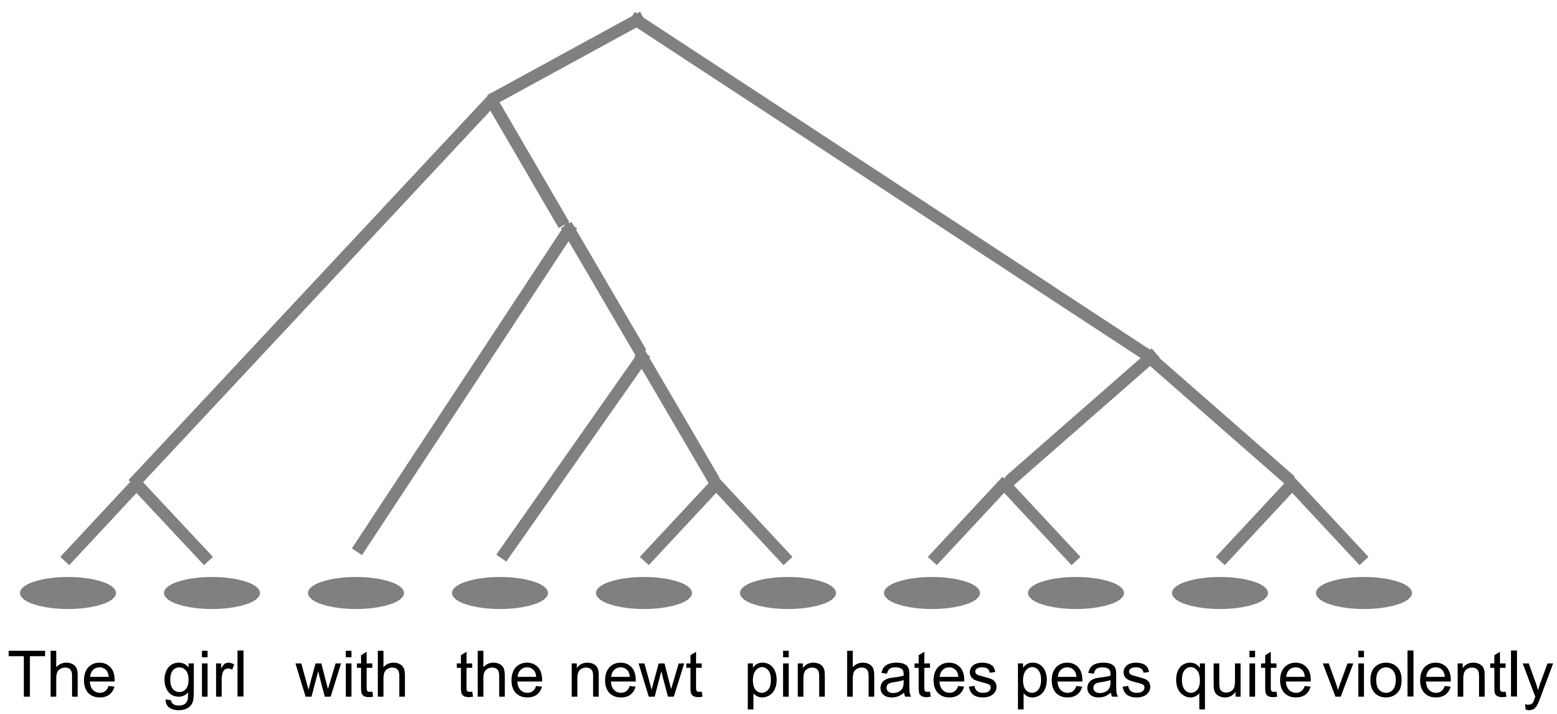
Parsing

- Finding these phrases is analogous to object detection.
  - Humans can do it.
  - The phrases help with tasks: translation, question answering, etc.
- We can do okay with a **supervised** model.



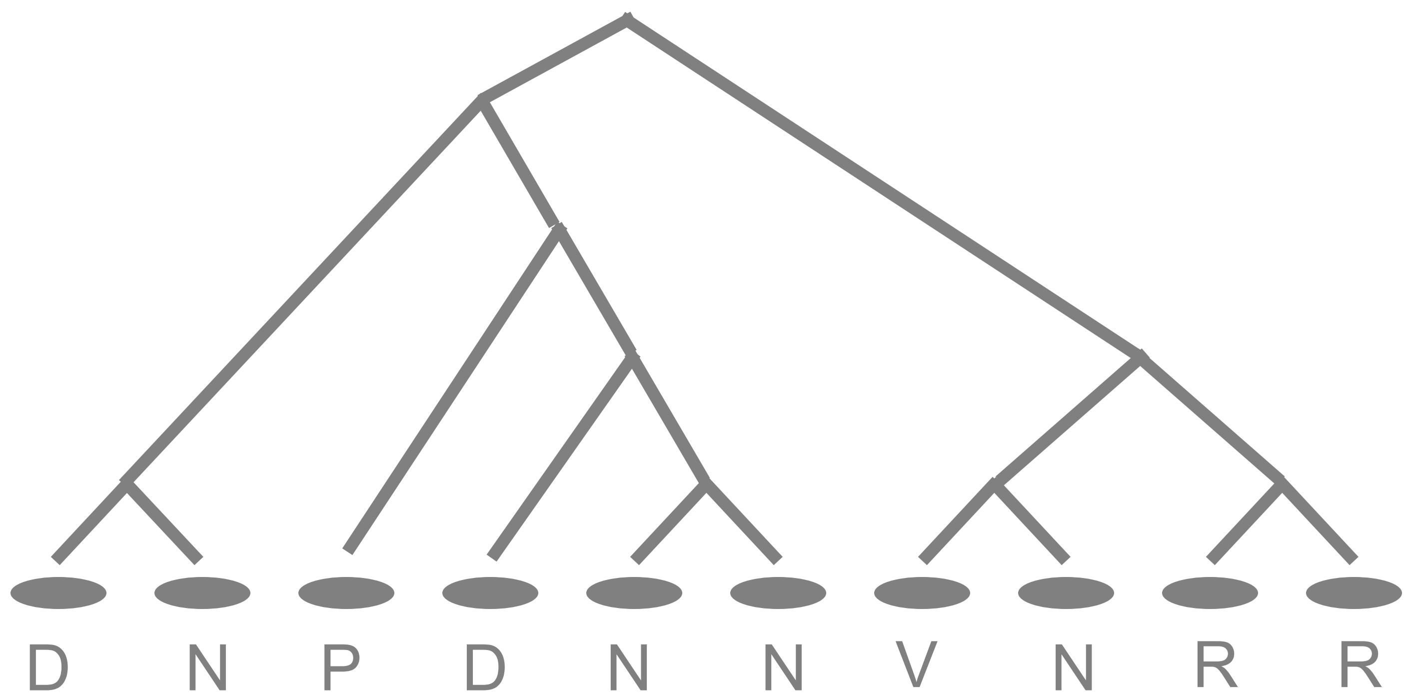
Grammar Induction

- Grammar induction is the **unsupervised** case.
- Given a corpus of sentences, can we find reasonable trees?
- Most people cheat: Our input is part-of-speech sequences.



Grammar Induction

- Grammar induction is the **unsupervised** case.
- Given a corpus of sentences, can we find reasonable trees?
- Most people cheat: Our input is part-of-speech sequences.
  - Less work to do: start with helpful low-dim word representations.



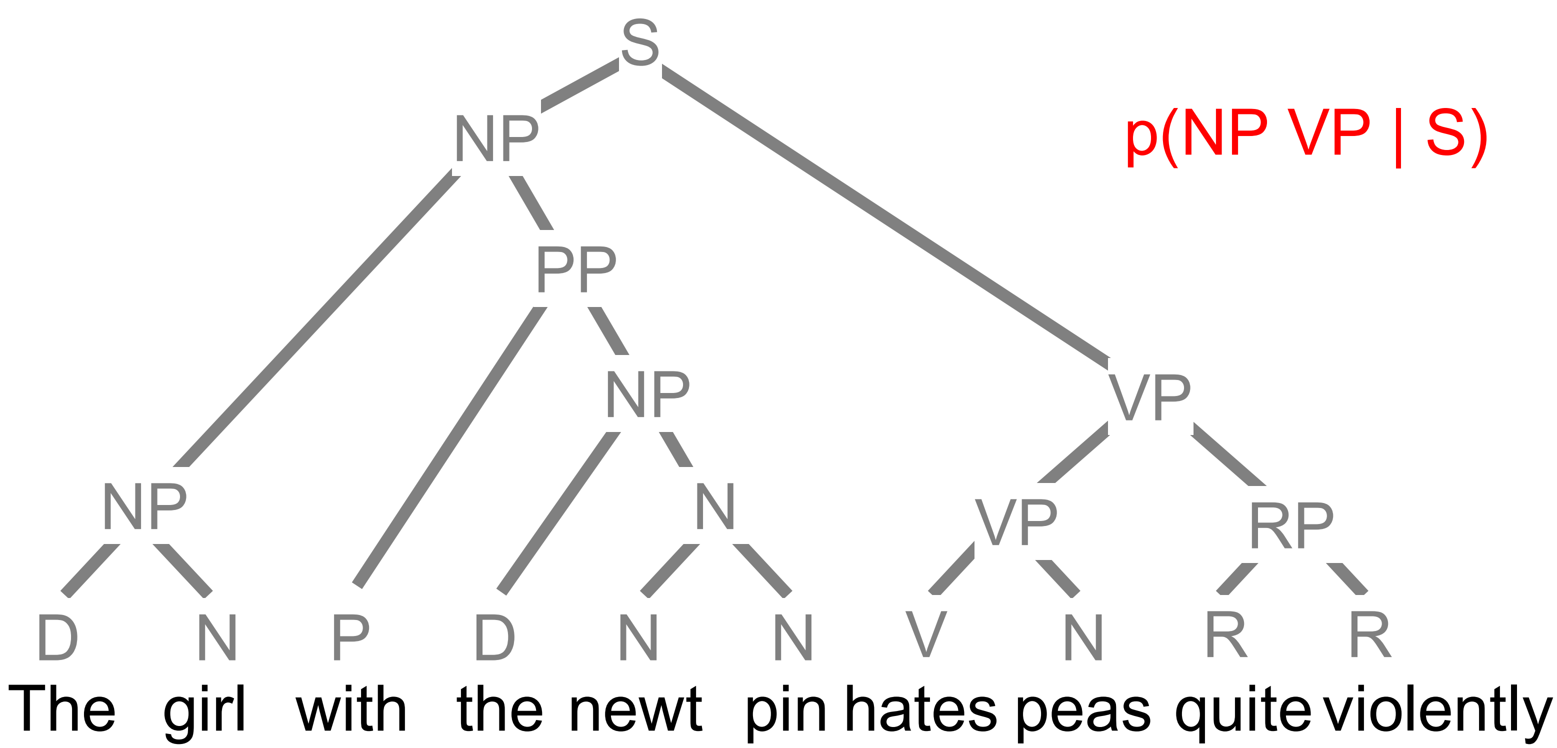
Grammar Induction

- Even then, current methods don't "really" work.
- Measure **directed dependency accuracy.**
  - What fraction of the words correctly identify which (single) other word they're modifying?
  - Currently 1/3 to 2/3, depending on language.
  - (English is around 1/2.)
  - And that's only on sentences of length 10 …
- We need more magic.
  - *Caveat: We're measuring agreement with linguists' conventions. To the extent that those conventions are arbitrary, a bit of supervised fine-tuning might help (Smith & Eisner 2009). Or we could evaluate on a downstream task.*

One of the two best language-learning devices I helped build

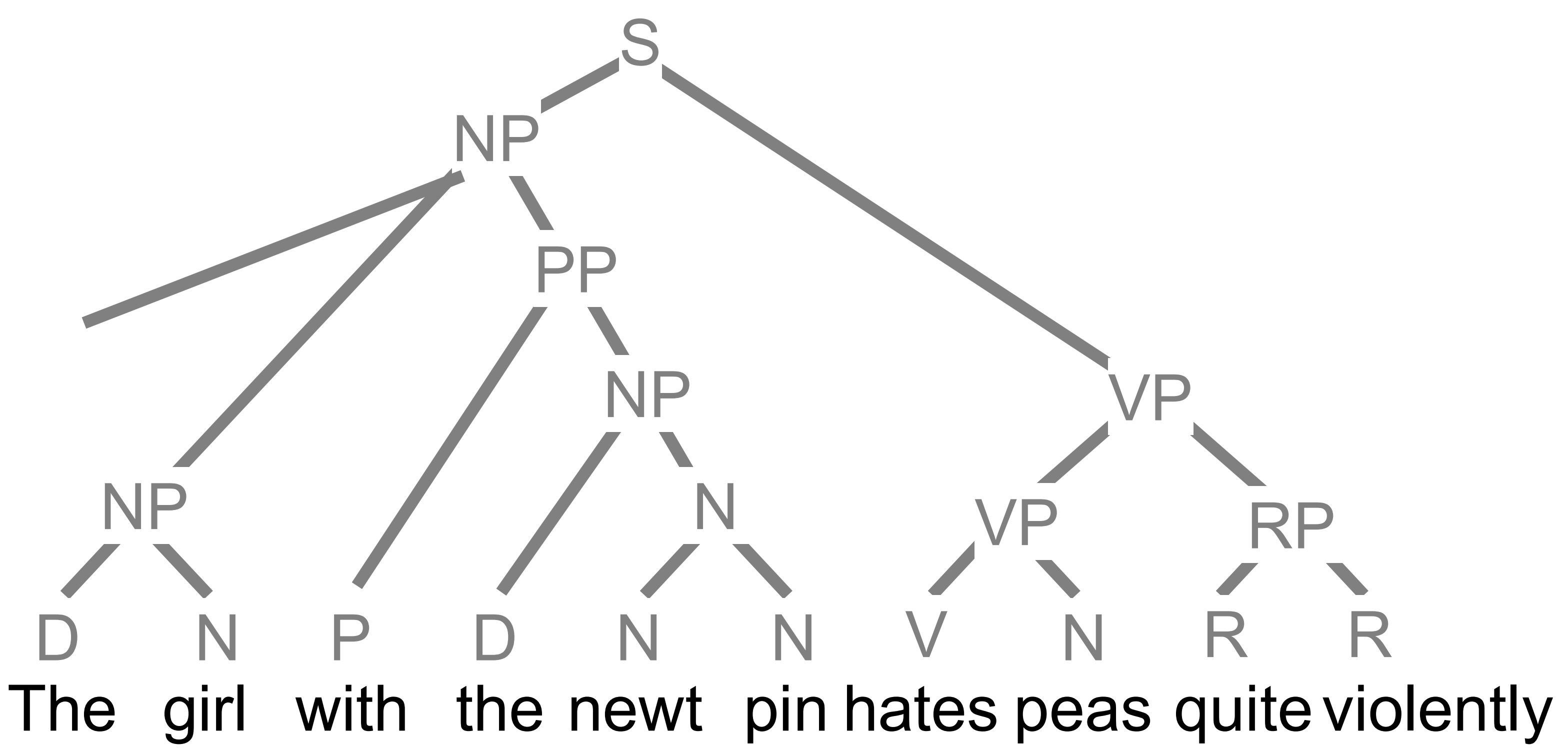
Generative Story: PCFG

- Given a set of symbols (phrase types)
- Start with S at the root
- Each symbol randomly generates 2 child symbols, or 1 word
- **Our job (maybe): Learn these probabilities**



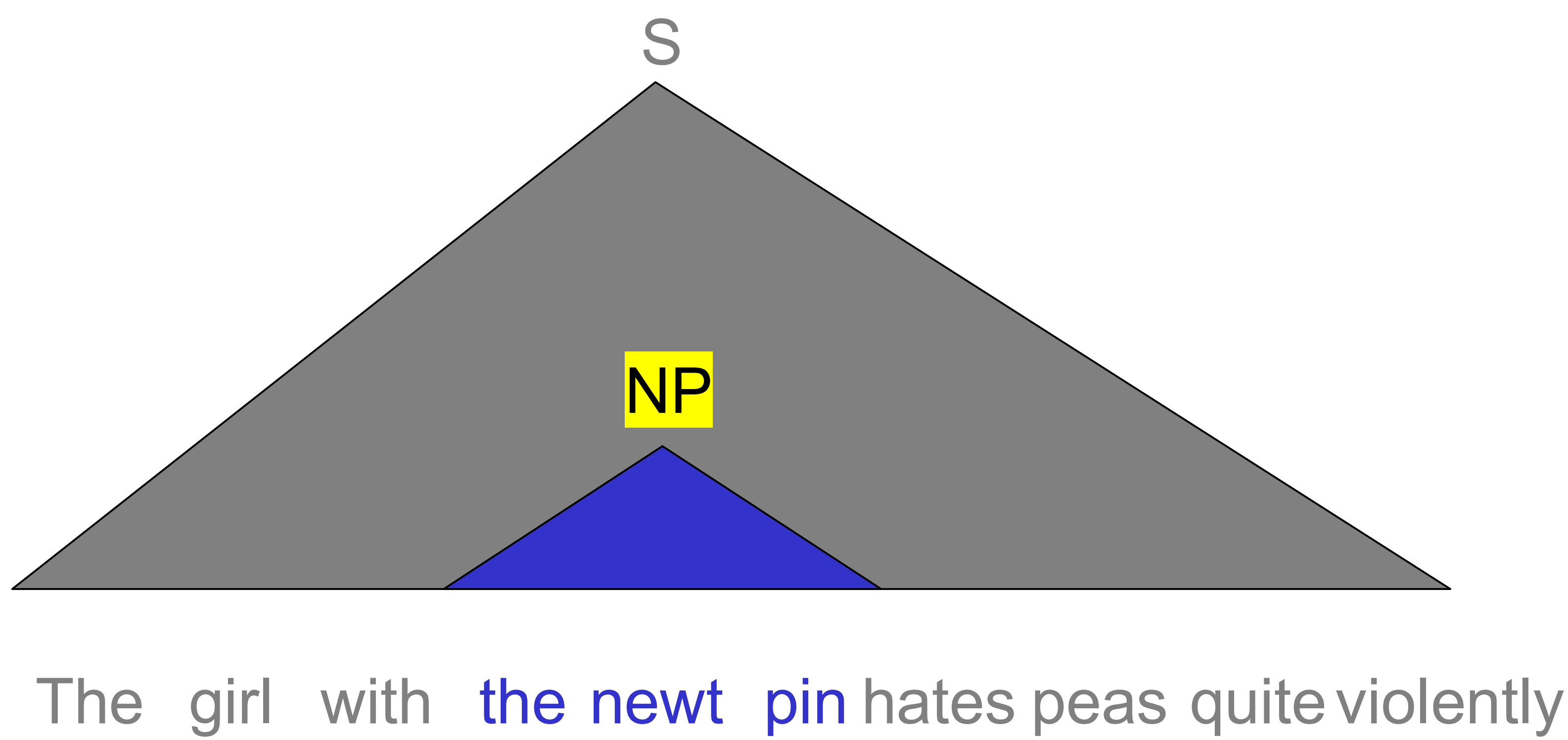
Context-Freeness of Model

- In a PCFG, the string generated under NP doesn't depend on the context of the NP.
- All NPs are interchangeable.



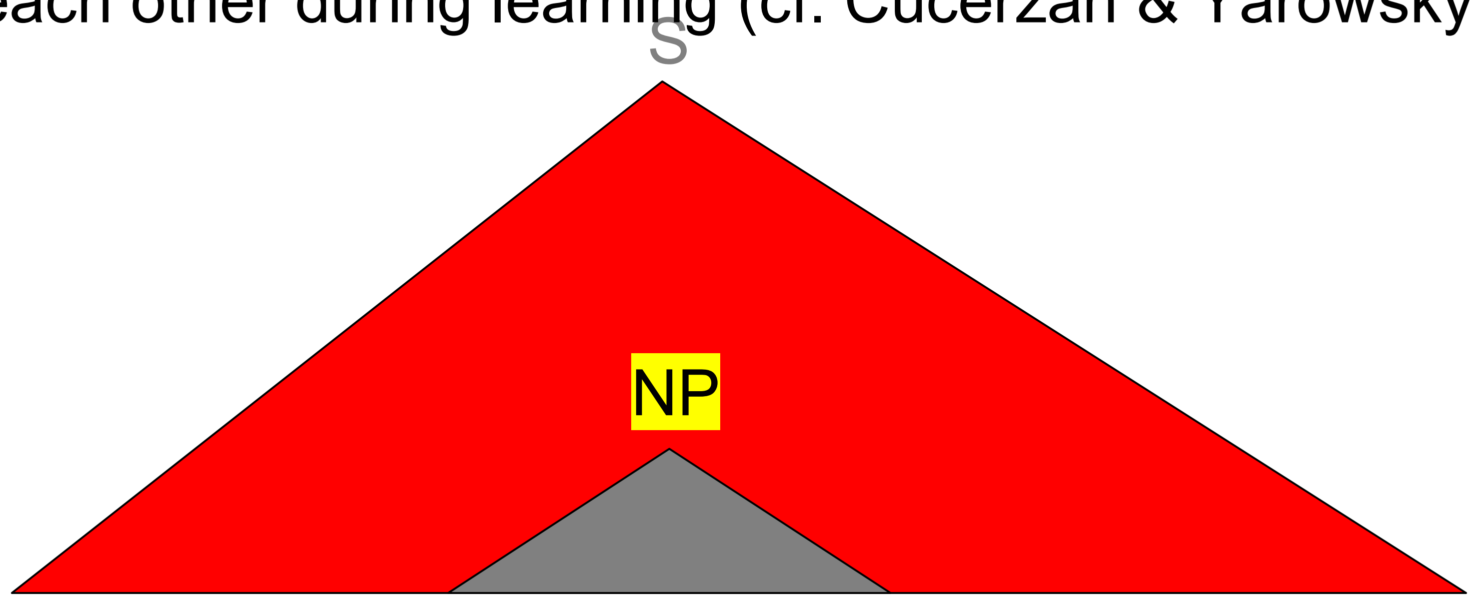
Inside vs. Outside

- This NP is good because the “inside” string looks like a NP



Inside vs. Outside

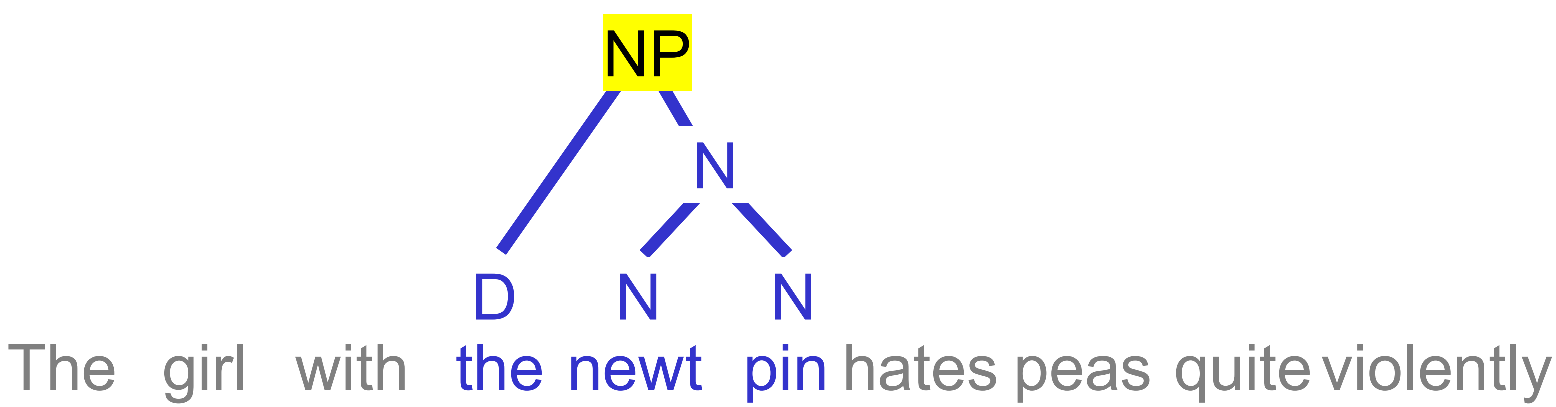
- This NP is good because the "inside" string looks like a NP
- **and** because the "outside" context looks like it expects a NP.
- These work together in global inference, and could help train each other during learning (cf. Cucerzan & Yarowsky 2002).



The girl with the newt pin hates peas quite violently

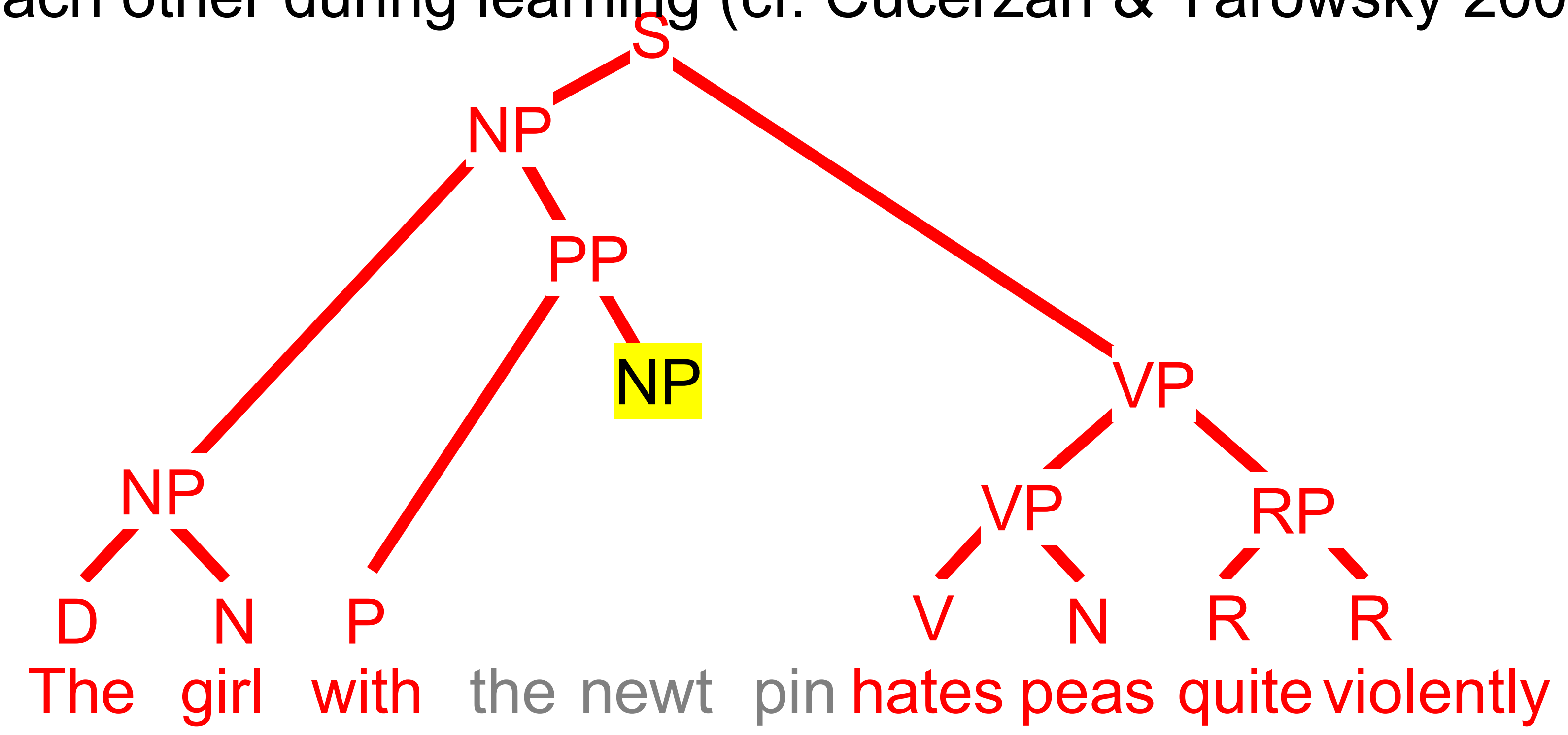
Inside vs. Outside

- This NP is good because the "inside" string looks like a NP
- **and** because the "outside" context looks like it expects a NP.
- These work together in global inference, and could help train each other during learning (cf. Cucerzan & Yarowsky 2002).



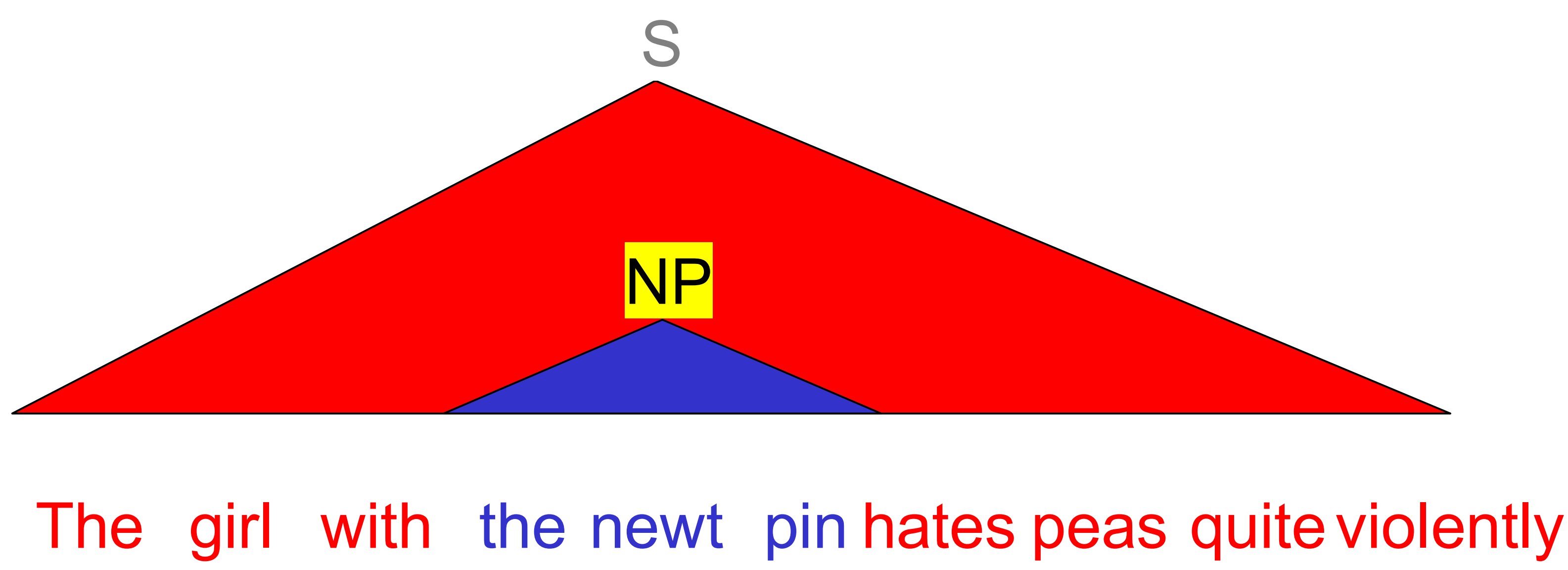
Inside vs. Outside

- This NP is good because the "inside" string looks like a NP
- **and** because the "outside" context looks like it expects a NP.
- These work together in global inference, and could help train each other during learning (cf. Cucerzan & Yarowsky 2002).



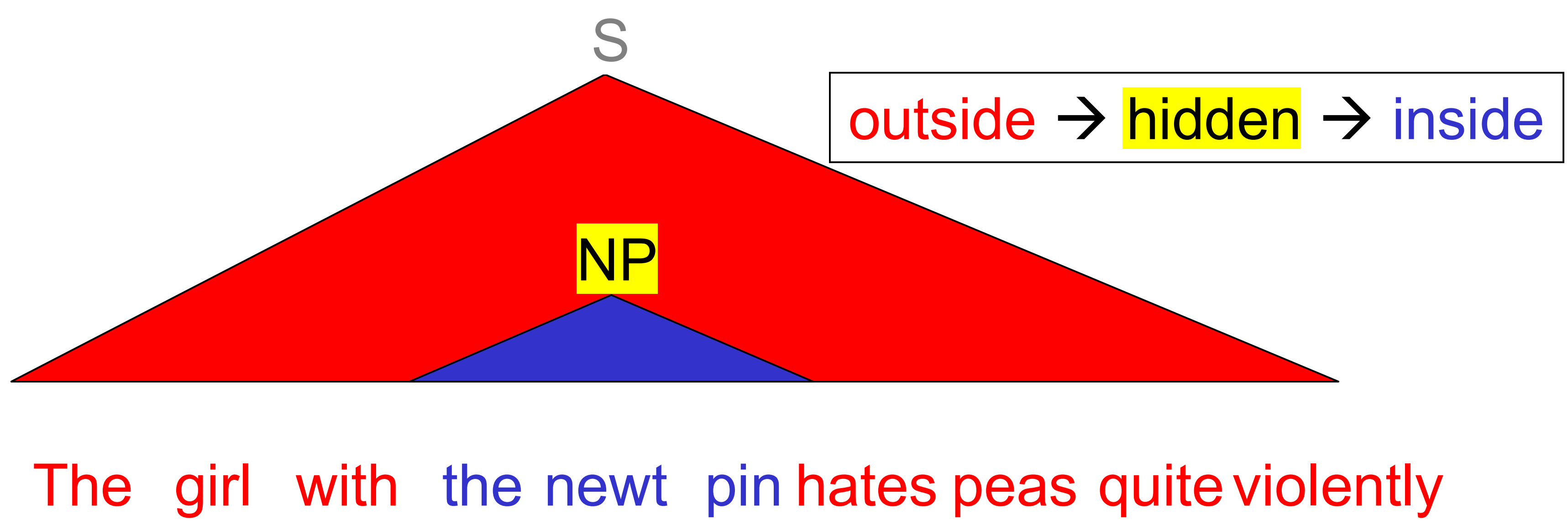
First Idea: Bottleneck

- Inside & outside strings are conditionally independent given the nonterminal symbol.
- Could build a network that maps outside → hidden → inside and use the hidden representation as the symbol.
  - If PCFG assumption is right, a 1-of-k hidden layer would be enough.



First Idea: Bottleneck

- We can’t easily do this between unbounded strings.
  - We need to abstract out *features* of the input and the output.
- Possible strategy is a bit like Alan Yuille’s talk yesterday …

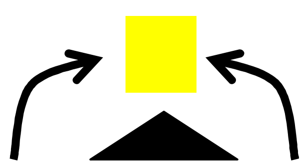


First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille’s talk yesterday …
- First learn representations for 1-word phrases given surrounding words

“level 0”

outside → hidden → inside



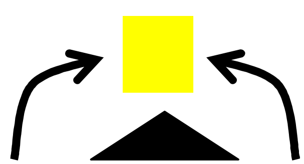
The girl with the newt pin hates peas quite violently

First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille's talk yesterday …
- First learn representations for 1-word phrases given surrounding words

"level 0"

outside → hidden → inside



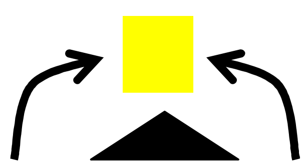
The girl with the newt pin hates peas quite violently

First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille's talk yesterday …
- First learn representations for 1-word phrases given surrounding words

"level 0"

outside → hidden → inside



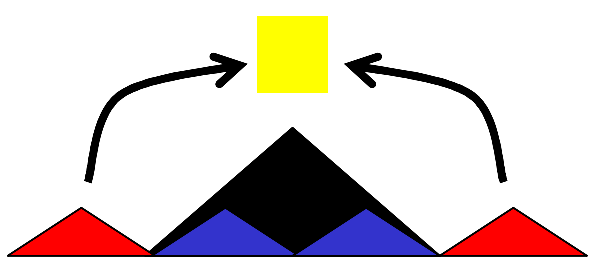
The girl with the newt pin hates peas quite violently

First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille’s talk yesterday …
- First learn representations for 1-word phrases given surrounding words
- Now learn representations for pairs of adjacent phrases given surrounding phrases
  (using only phrases for which we’ve already learned representations)

“level 1”

outside → hidden → inside



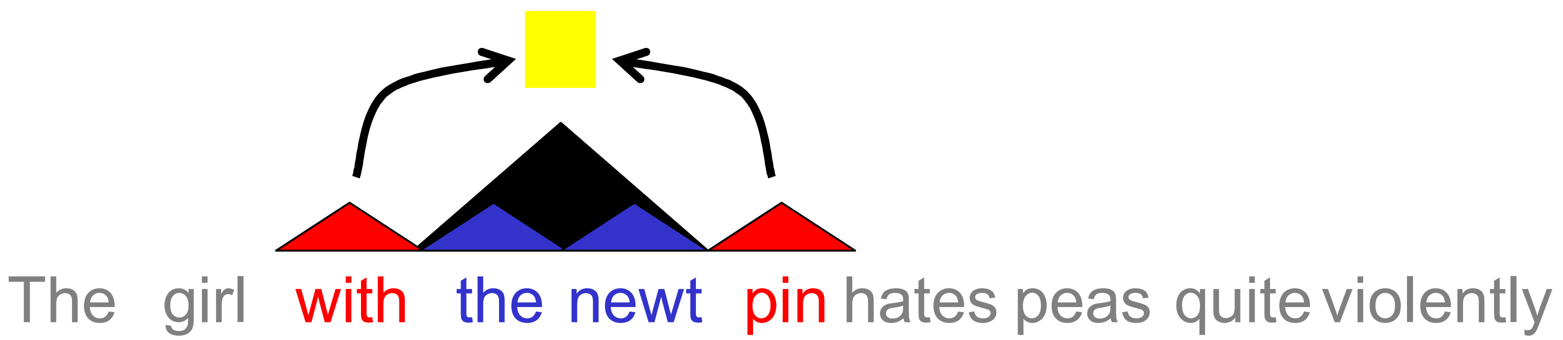
The girl with the newt pin hates peas quite violently

First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille's talk yesterday …
- First learn representations for 1-word phrases given surrounding words
- Now learn representations for pairs of adjacent phrases given surrounding phrases
  (using only phrases for which we've already learned representations)

"level 1"

outside → hidden → inside

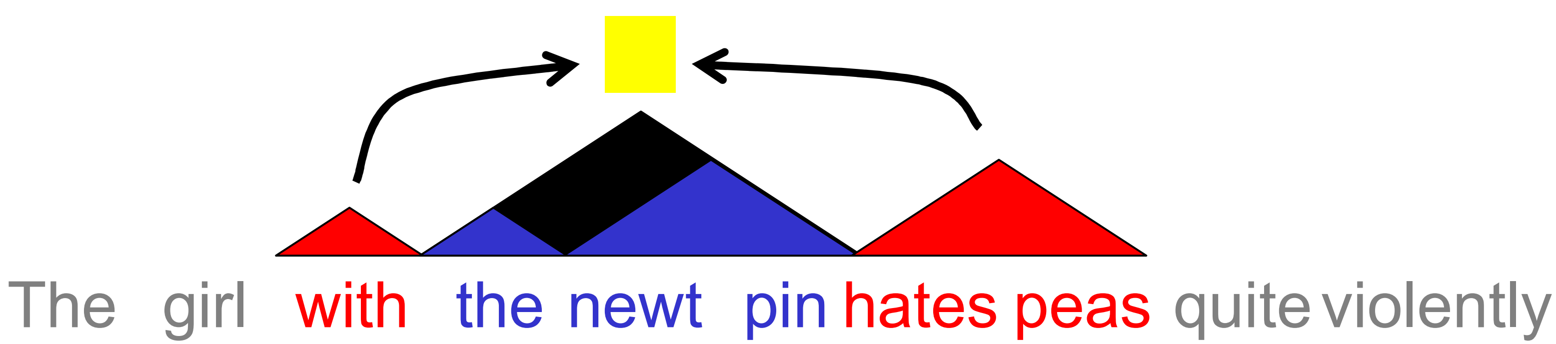


First Idea: Bottleneck

- Possible strategy is a bit like Alan Yuille’s talk yesterday …
- First learn representations for 1-word phrases given surrounding words
- Now learn representations for pairs of adjacent phrases given surrounding phrases
  (using only phrases for which we’ve already learned representations)

“level 2”

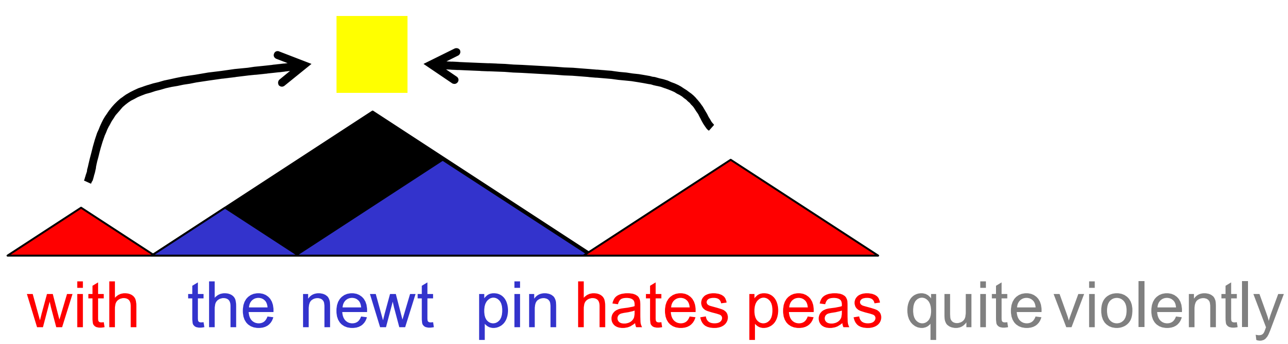
outside → hidden → inside



Problems with Bottleneck Idea

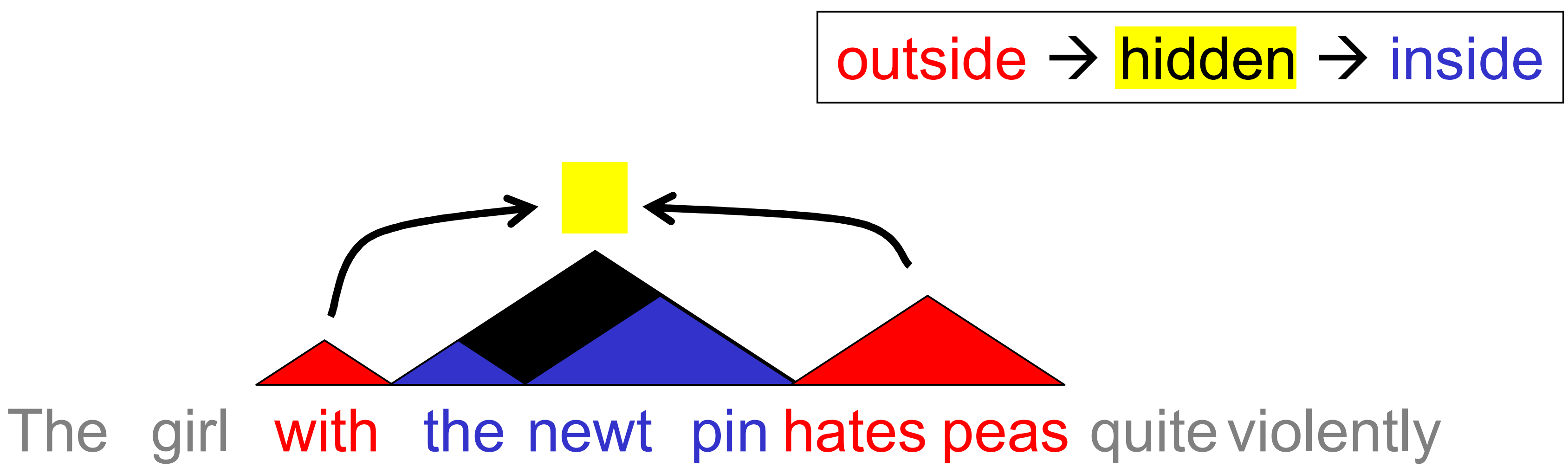
1. Relationship between inside and outside isn't linear (CCA not good enough)
   - It's not a neural net either.
   - It's actually a PCFG – we "know" the structure!
     - Note: A PCFG = a sum-product network (Poon & Domingos 2011)

outside → hidden → inside



Problems with Bottleneck Idea

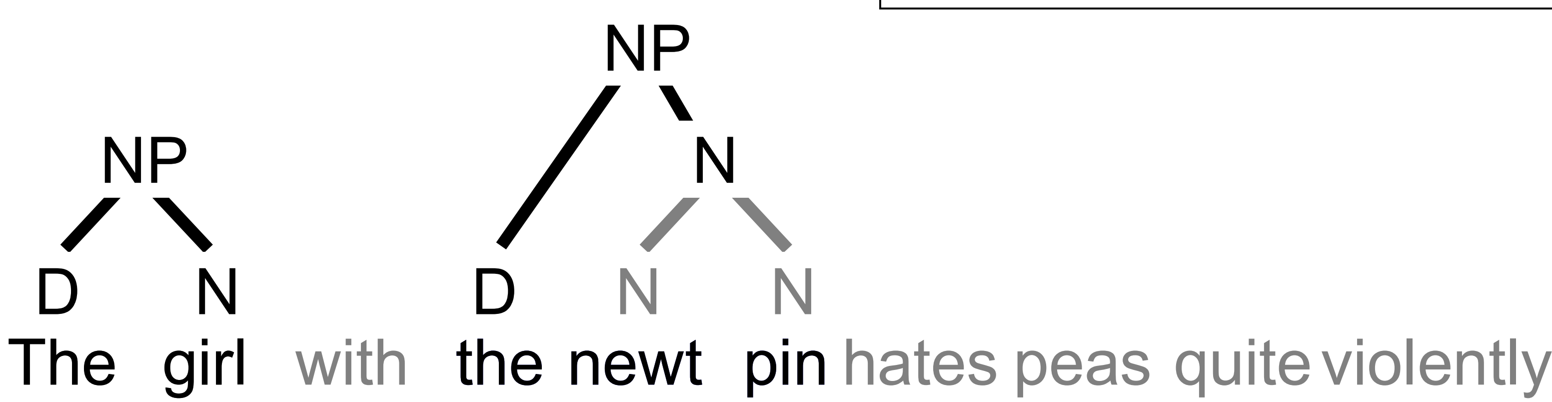
2. We also learn representations for non-constituents like “newt pin hates.”
   - Maybe that’s good: if we let 1000 flowers bloom, at least we won’t miss the good stuff.
   - But how do we put the pieces back together?
     - (Maybe there are ways: cf. Socher et al. 2011.)



Problems with Bottleneck Idea

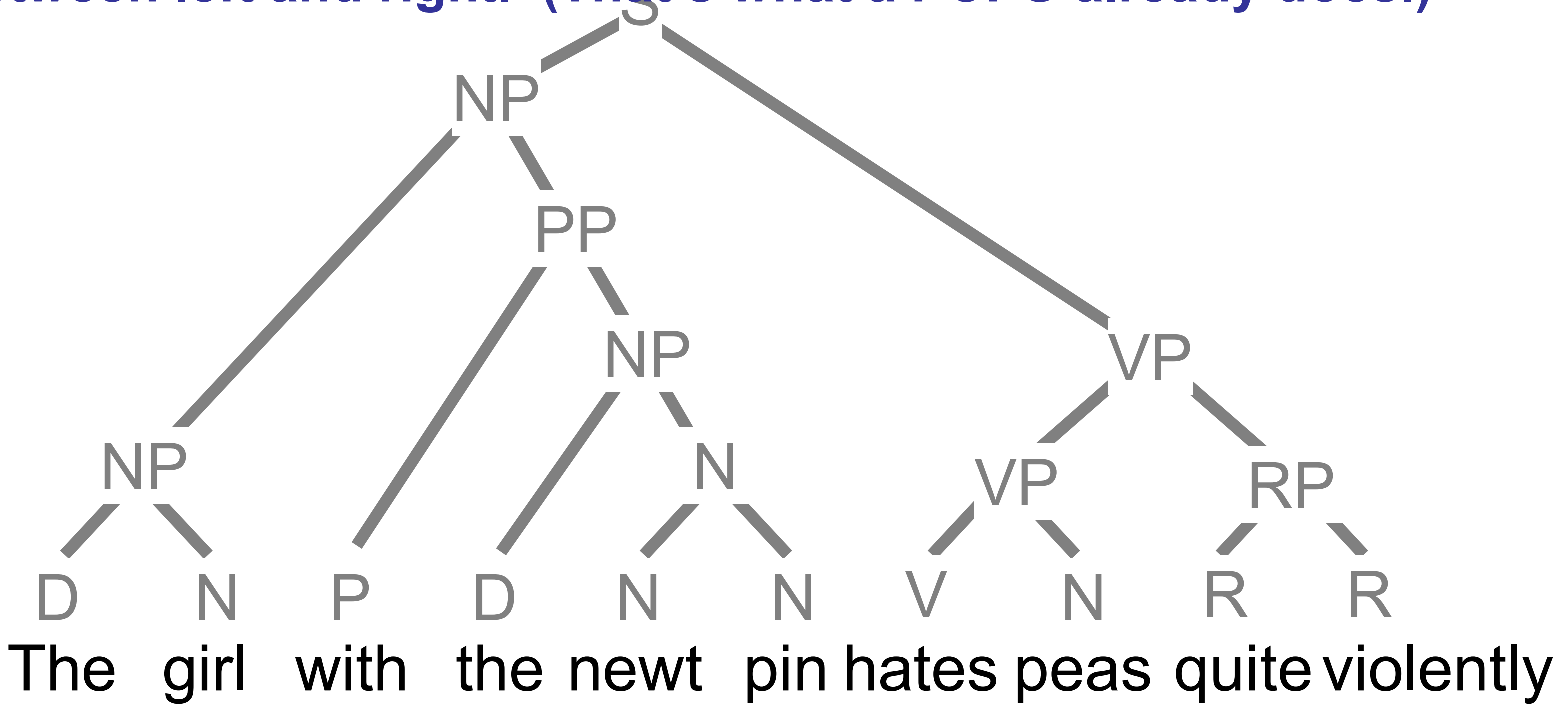
3. "The girl" was learned at level 2, but "the newt pin" was learned separately at level 3.
   - These levels learn separate parameters.
   - So different representations, even though both have the same top-level structure and are interchangeable.
   - Oops!

outside → hidden → inside



Convolutional Parameter Tying

- **Conclusion: Depth of network doesn't map onto to depth of tree.**
- **Low levels of tree are just as important as high levels for evaluation and supervised tasks.**
- **We want to share parameters between low and high, not merely between left and right. (That's what a PCFG already does.)**



Second Idea: Global Training

- Just use a fixed PCFG throughout (one that allows anything to rewrite as anything).
- Adjust its parameters to optimize likelihood.
- Fundamental tool: inside-outside algorithm.
  - Baker 1979
  - This is an ingredient in almost anything!

Inside-Outside Algorithm

- Under a PCFG, with T=tree, S=sentence, $p(T|S) = (1/Z(S)) \exp(\phi \cdot \text{features}(T,S))$
- The inside algorithm computes Z by dynamic programming in $O(n^3)$ time, provided that the features are rule-local.
- The outside algorithm is just backprop to compute ∇Z in $O(n^3)$ time.
- Because the model has exponential form, $\nabla Z / Z = \nabla \log Z$ gives the expected features of the tree given the sentence.
  - Use this to get the expected count of each rule [at each position].
  - Can use that for EM (or just do gradient ascent).

Inside-Outside Algorithm

- Seems like this should work!
- But it doesn’t.
  - (Lari & Young 1990; Merialdo 1994 had trouble even for the special case of HMMs)
- Space is riddled with local maxima, nearly all of them bad.
- Algorithms quickly discover superficial phrases like “of the,” and then never let go.

Things People Have Tried …

1. Modify the objective function to make it easier to optimize.
   - Klein & Manning 2002: constituent-context model
   - Spitkovsky et al. 2012: dependency-and-boundary models
   - Gimpel & Smith 2012: convex objective
   - (and others)

Things People Have Tried …

2. More effective search, usually via search bias
   - Klein & Manning 2002: initializers
   - Smith & Eisner 2004: deterministic annealing
   - Spitkovsky et al. 2009, 2010, 2012: “baby steps,” fragments
   - Spitkovsky et al. 2011: lateen EM
   - Gormley & Eisner 2013: global branch-and-bound

Things People Have Tried …

3. Incorporate linguistic knowledge into objective

   - Headden et al. 2009: richer generative model
   - Naseem et al. 2010, Druck et al. 2009:
     constrain to be consistent with "universal" grammar
     (see also Marecek and Zabokrtsky 2011)
   - Gillenwater et al. 2010: posterior regularization for sparsity (see also Ravi & Knight 2009)
   - Cohen & Smith 2010: hierarchical prior on parameters
   - Spitkovsky et al. 2010, 2011, 2012: pay attention to punctuation, capitalization, hypertext markup
   - Pate & Goldwater 2013: pay attention to acoustics

Things People Have Tried …

4. Multi-task learning or co-training
   - Klein & Manning 2002: constituent-context model
   - Berg-Kirkpatrick & Klein 2010: phylogenetic grammar induction
   - Cohen et al. 2001: multilingual grammar induction
5. Change the objective function to mitigate model misspecification
   - Smith & Eisner 2005: contrastive estimation
   - Asks "*Why* is likelihood poorly correlated with parse accuracy?"
6. Spectral methods
   - But so far, these assume the tree structure is known

Things People Have Tried …

- Summary: A pile of tricks that we hope can help solve the intractable problems that humans solve. (See Cohen 2011, Hsu & Liang 2012.)
- Just like in deep learning!

Third Idea: Bottom-Up

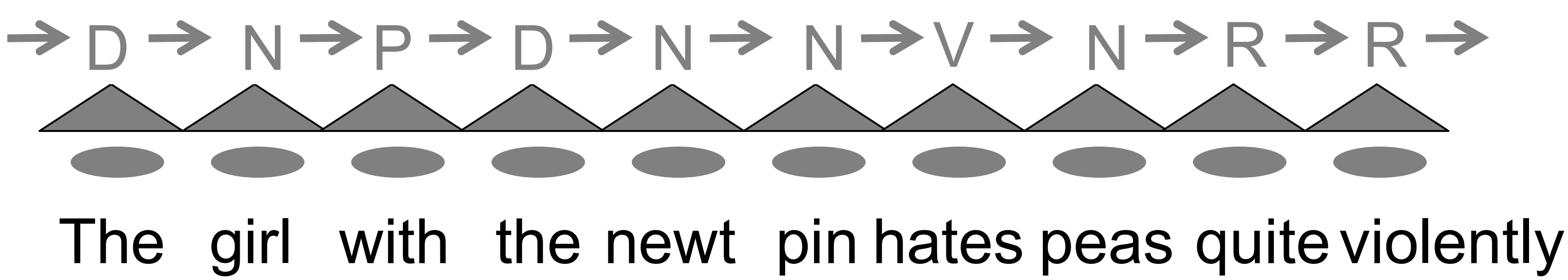
- Keep the PCFG maximum likelihood objective, but impose a search bias.

- Let’s again work upward from the bottom, but now get a global solution at each step by parsing.
  - This idea is a variant of one of the “structural annealing” techniques of Smith & Eisner 2006.

(Smith & Eisner, 2006)

Third Idea: Bottom-Up

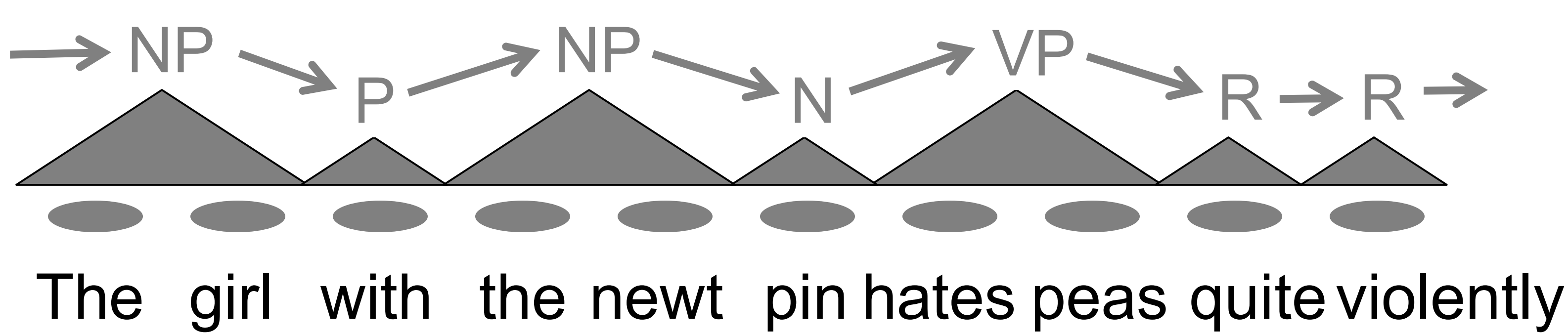
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

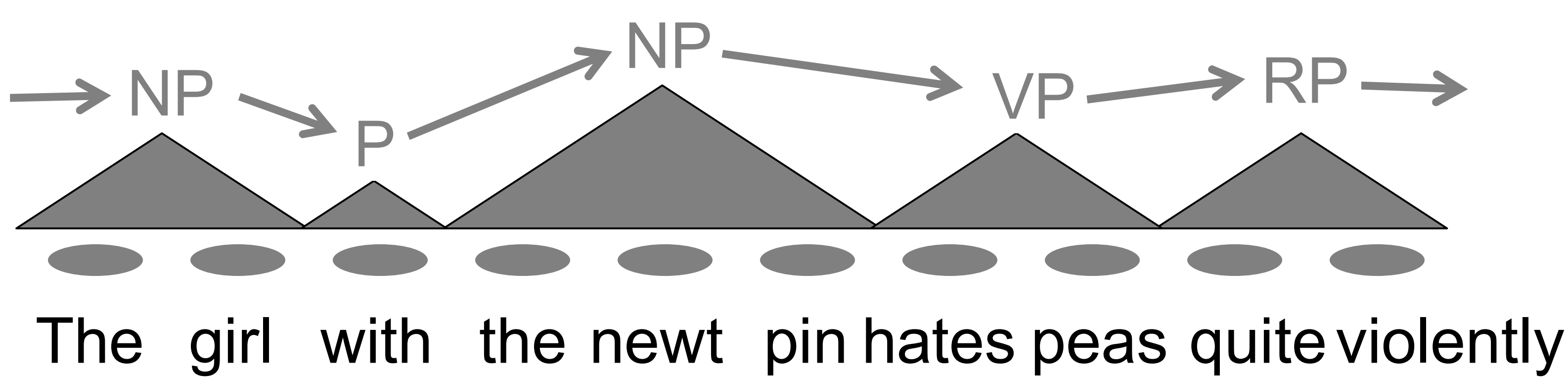
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

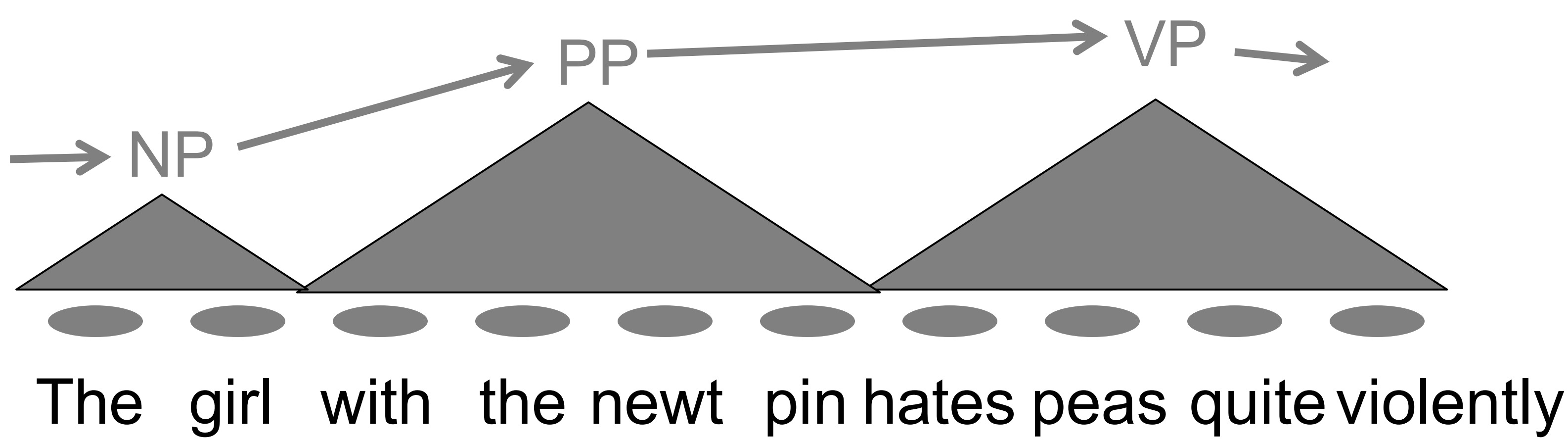
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

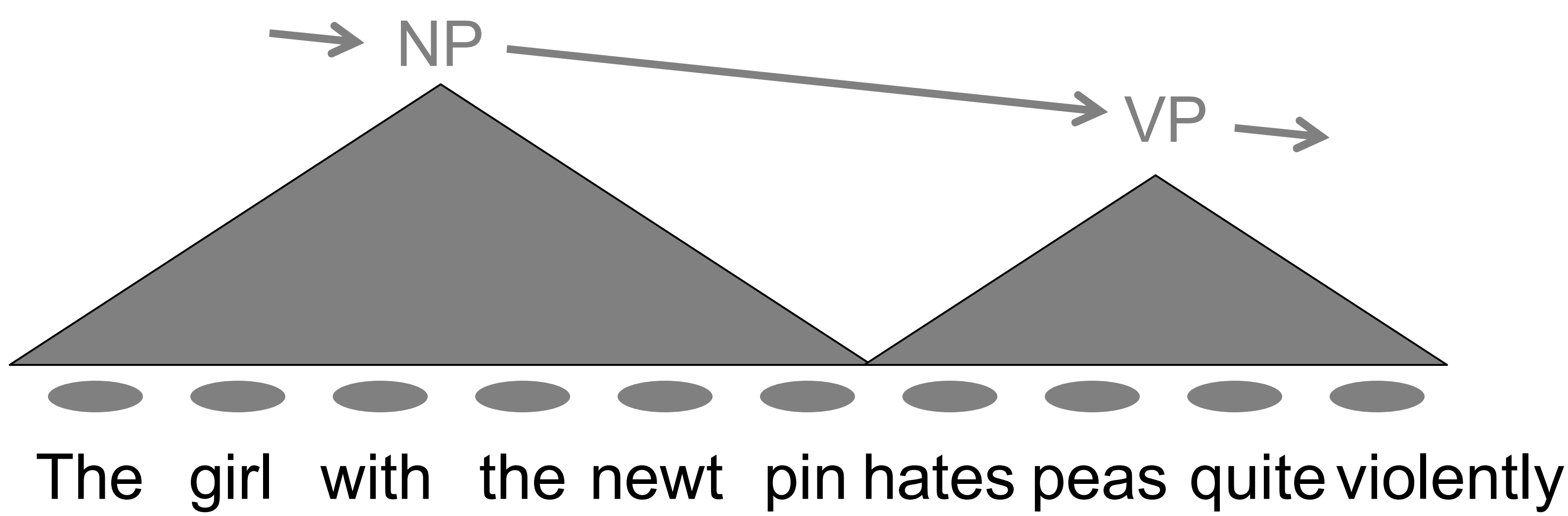
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

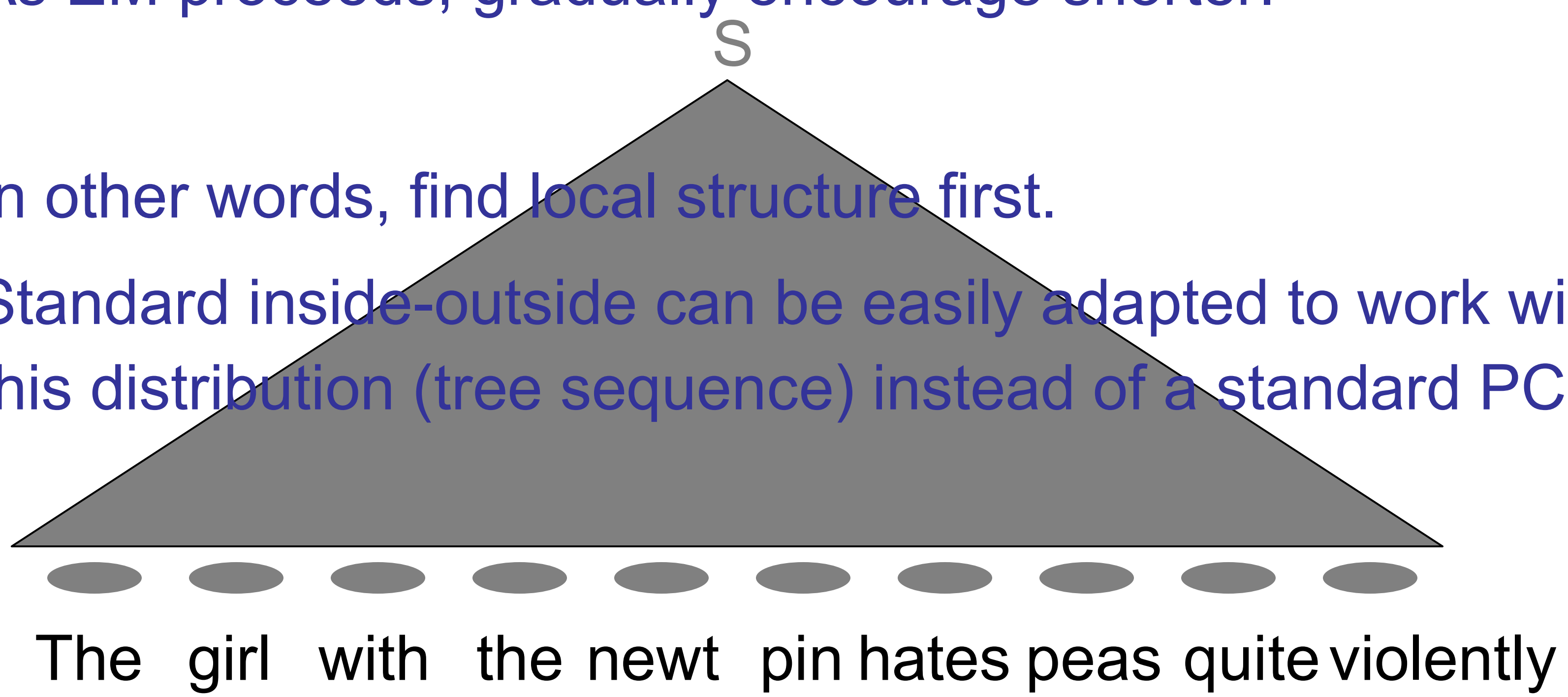
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

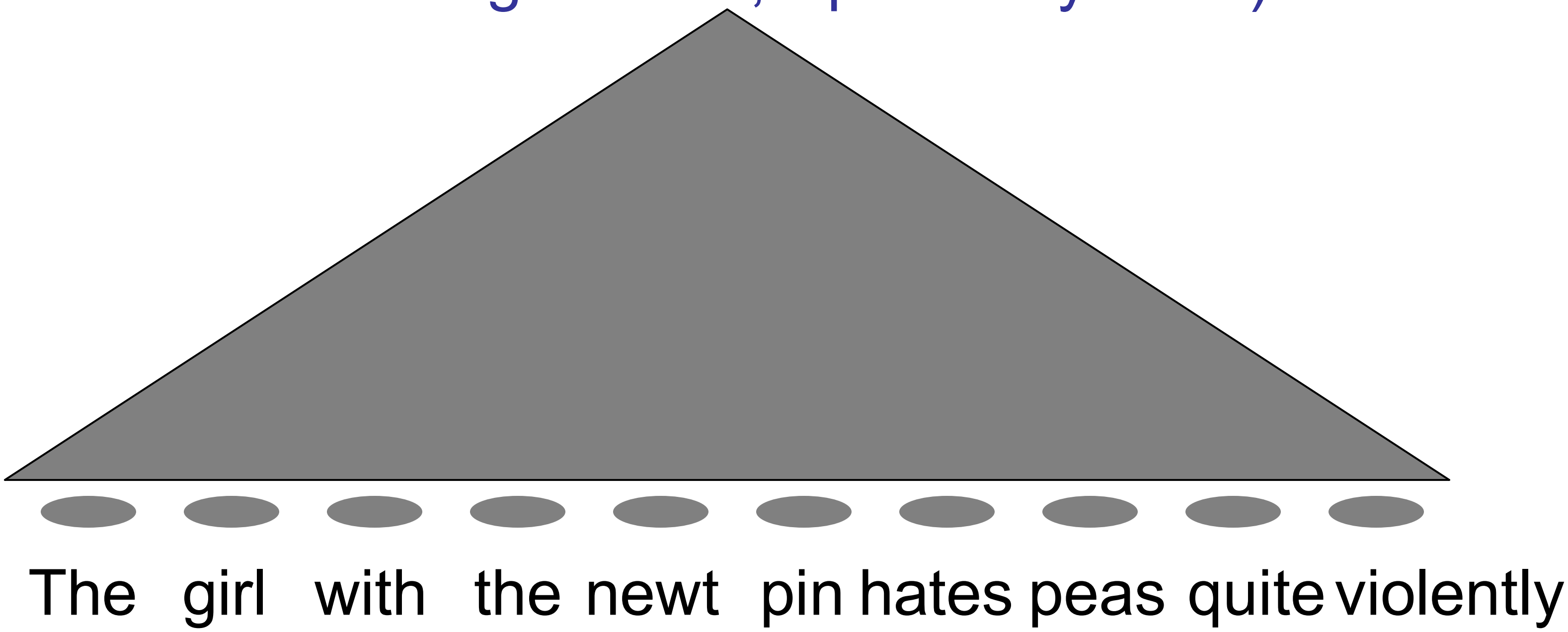
- Instead of *one* tree, cover sentence with a *sequence* of trees.
- Explain the root sequence with a bigram (Markov) model.
- Start by encouraging long sequences.
- As EM proceeds, gradually encourage shorter.
- In other words, find local structure first.
- Standard inside-outside can be easily adapted to work with this distribution (tree sequence) instead of a standard PCFG.



(Smith & Eisner, 2006)

Third Idea: Bottom-Up

- In other words, find local structure first.
- Sort of like layer-wise training, but now, once you learn a rule, you can use it at all levels.
- Can anneal the distribution to avoid gradient dilution (cf. Gens & Domingos 2012, Spitkovsky 2012)

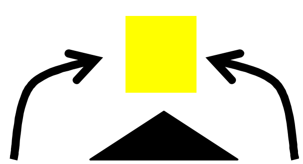


Fourth Idea: Context

- **Firth 1957: "You shall know a word by the company it keeps."**
- **Our previous bottleneck model was circular, because it predicts *each* word from its neighbors, which are also words.**
- **But you can think of this as a "restricted" auto-encoder where the sentence is used to generate itself.**
- **And it's reminiscent of successful work on word embeddings (see review in Turian et al. 2010, and later work e.g. by Dhillon et al.)**

"level 0"

outside → hidden → inside



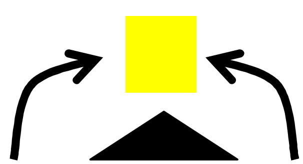
The girl with the newt pin hates peas quite violently

Fourth Idea: Context

- Firth 1957: “You shall know a ~~word~~ phrase by the company it keeps.”
- Brill & Marcus 1992: If tag X appears in the same word contexts as tag sequence Y Z, then maybe Y Z is a phrase of the same type as X.
  - So add rule X → Y Z.
  - ProperNoun → Determiner Noun (Mary vs. the girl)
  - Noun → Adjective Noun (girl vs. tall girl) (recursive!)

“level 0”

outside → hidden → inside

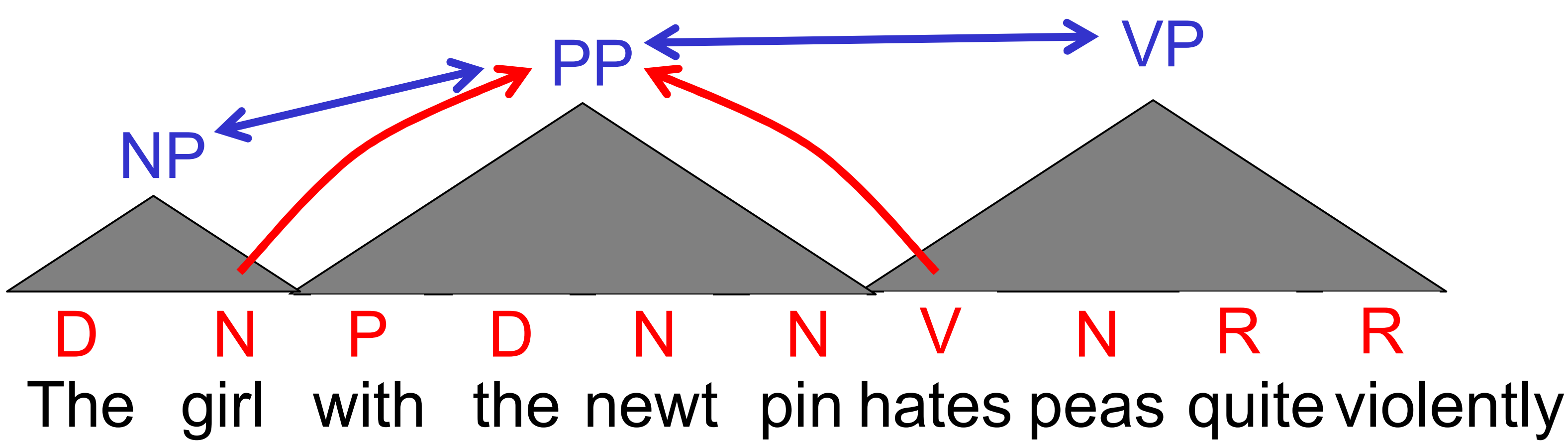


The girl with the newt pin hates peas quite violently

Fourth Idea: Context

- Back to bottom-up approach. Modify the root sequence model so that the sequence is now conditioned on context.
  - Globally normalized log-linear model of root sequence

    $p({}_0NP_2\ {}_2PP_6\ {}_6VP_{10} \mid \text{red stuff})$ [like a semi-Markov CRF]
  - The features of this example sequence make it probable
    - Happy root bigrams **NP PP** and **PP VP**
    - The **PP** covers positions 2-6, so is happily surrounded by **N, V**



Fourth Idea: Context

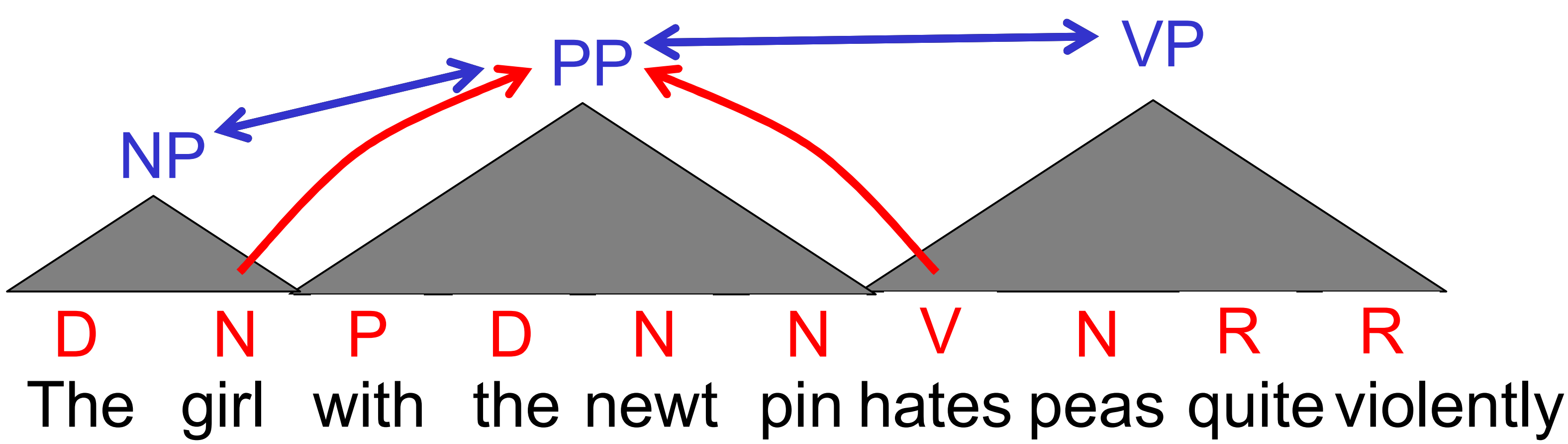
- Full likelihood of sentence sums over all root sequences, with probabilities from log-linear parameters θ.

$p({}_0w_{10})$ = a sum over many explanations like

$$p_\theta({}_0NP_2\ {}_2PP_6\ {}_6VP_{10} \mid \text{red stuff})$$

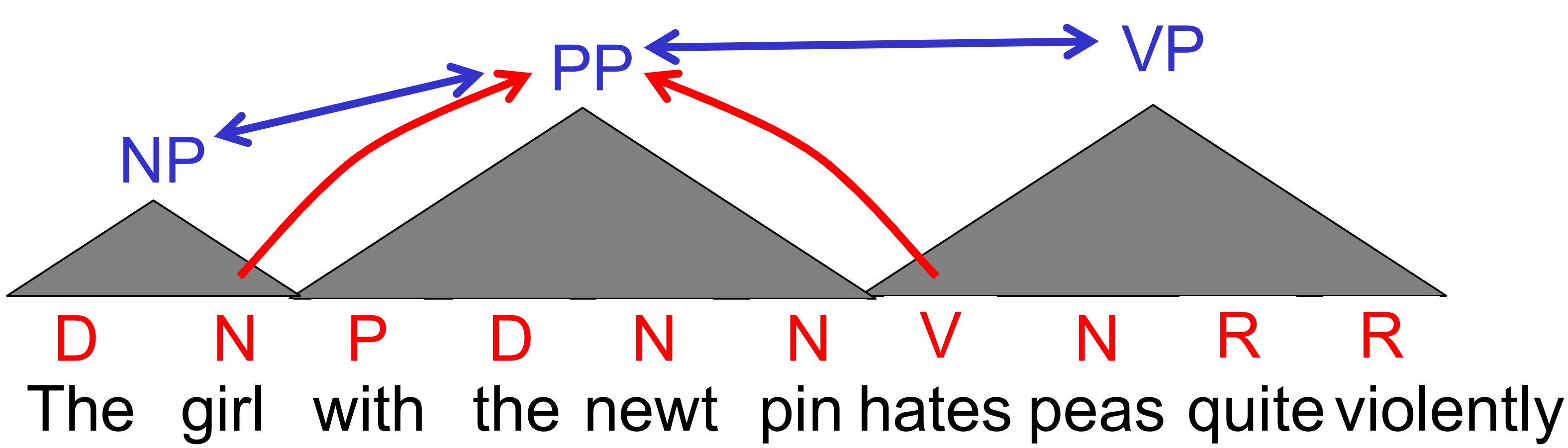
$$\cdot\, p_G({}_0w_2 \mid NP) \cdot p_G({}_2w_6 \mid PP) \cdot p_G({}_6w_{10} \mid VP)$$

where p_G denotes the PCFG and sums over many trees.



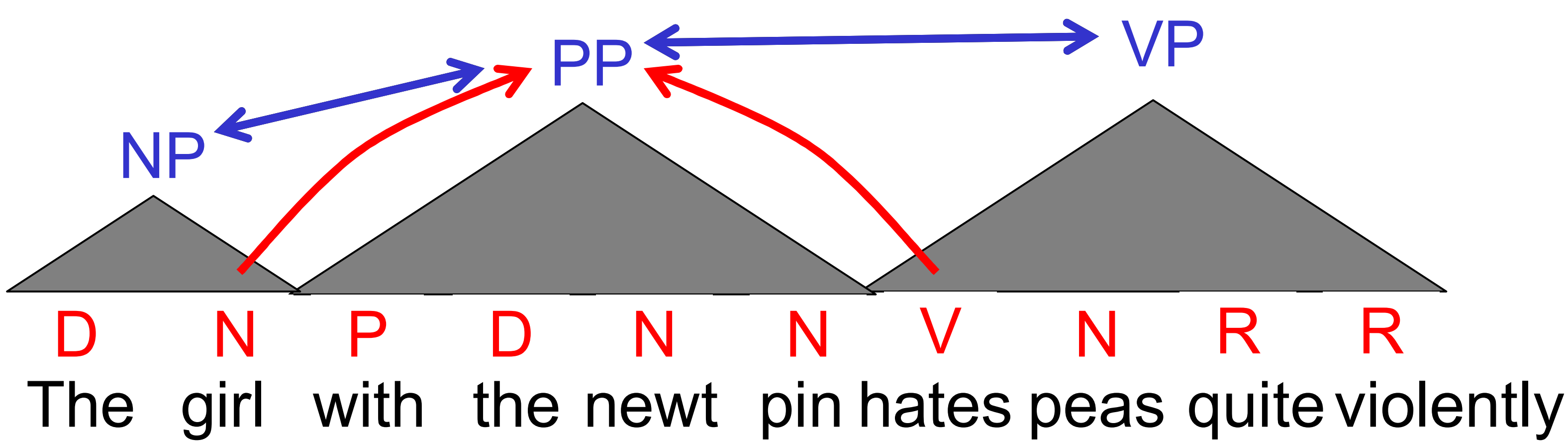
Fourth Idea: Context

- We jointly learn the model θ of the root sequence and the model G of each tree given its root. Training is still tractable!
  - Gradient is a difference of 2 expected feature vectors.
  - One sums over explanations of *this* sentence.
  - Other sums over explanations of *any* sentence (given red stuff).
  - Tractable because $p_G(\cdot \mid PP) = 1$ because G is a PCFG, and red stuff allows us to sum over all root sequences by dynamic programming.



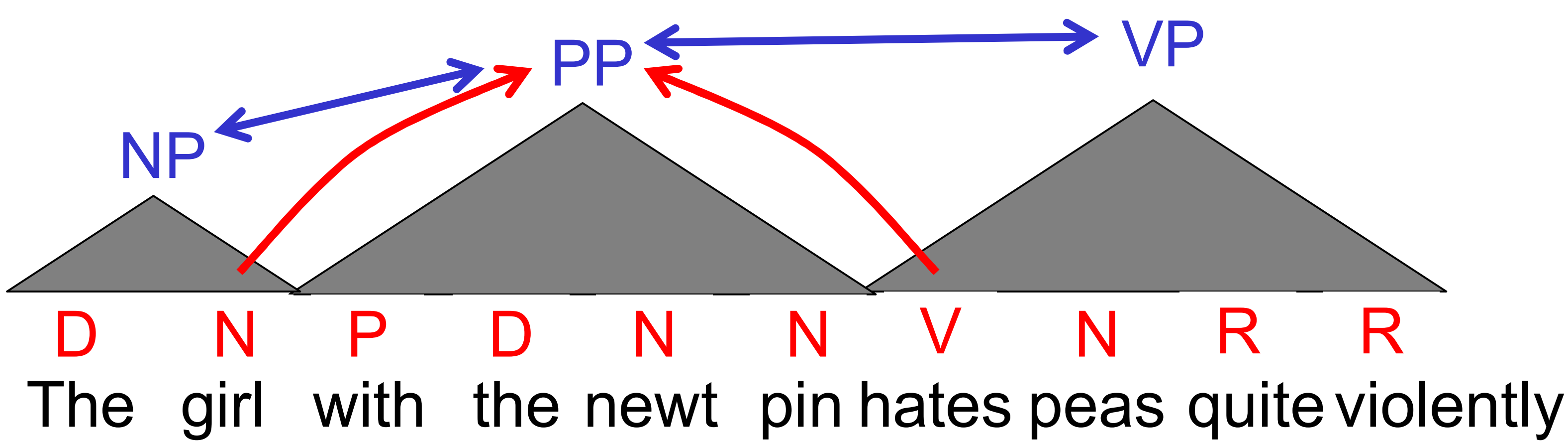
Fourth Idea: Context

- We jointly learn the model θ of the root sequence <u>and</u> the model G of each tree given its root. Training is still tractable!
  - Gradient is a difference of 2 expected feature vectors.
  - One sums over explanations of *this* sentence.
  - Other sums over explanations of *any* sentence (given red stuff).
  - Tractable because $p_G(\cdot \mid PP) = 1$ because G is a PCFG, and red stuff allows us to sum over all root sequences by dynamic programming.



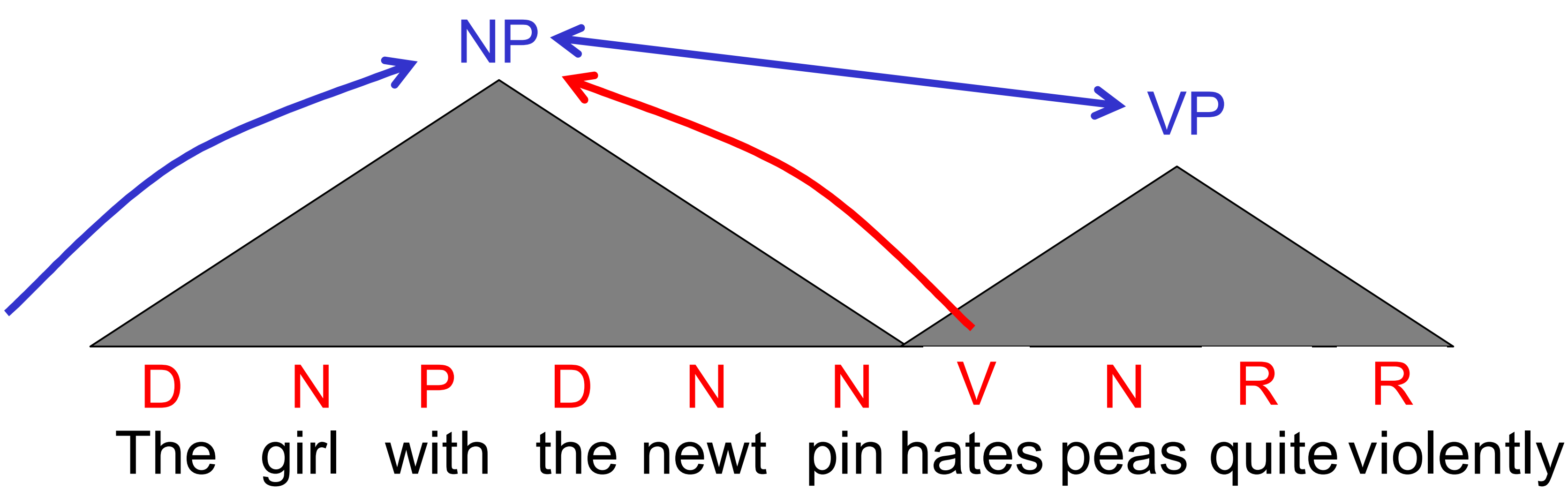
Fourth Idea: Context

- It's still a circular model: each phrase must be generated after the surrounding phrases that provide its context.
- But we still have the auto-encoder excuse: red stuff is like an ambient field that favors certain root sequences.
- And since a root only looks at context outside itself, this context goes away as we anneal toward a single tree. At the end of the day, we have a pure CFG!



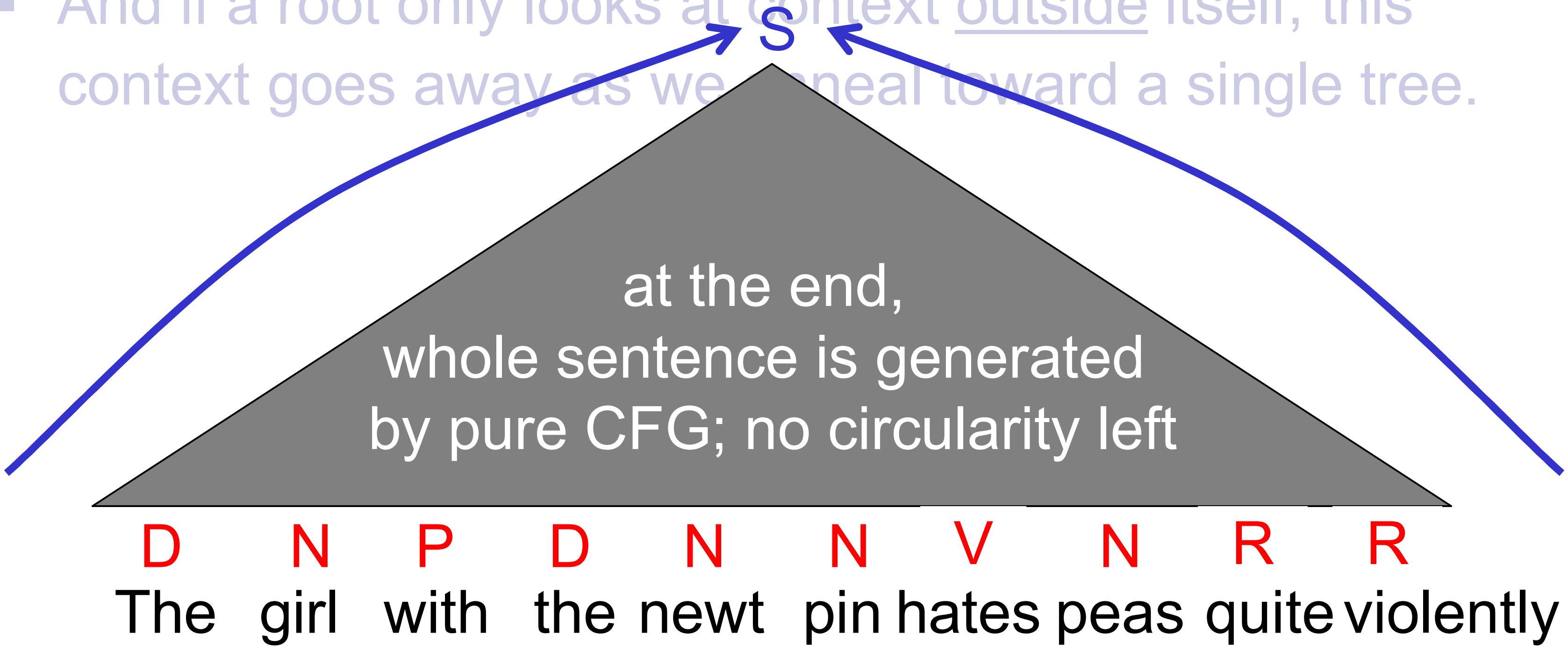
Fourth Idea: Context

- It's still a circular model: each phrase must be generated after the surrounding phrases that provide its context.
- But we still have the auto-encoder excuse: red stuff is like an ambient field that favors certain root sequences.
- And if a root only looks at context <u>outside</u> itself, this context goes away as we anneal toward a single tree.



Fourth Idea: Context

- It's still a circular model: each phrase must be generated after the surrounding phrases that provide its context.
- But we still have the auto-encoder excuse: red stuff is like an ambient field that favors certain root sequences.
- And if a root only looks at context outside itself, this context goes away as we anneal toward a single tree.



Fifth Idea: Stacking

- One thing that the above methods have in common is that they're all **local search** methods. Could prematurely commit to local optima.

- Also, as in bottleneck idea, we'd like to use better and better features as we proceed in training. Why limit ourselves to the contextual *tags* when we could use *phrases*?

- So let's put the "deep" back into "deep learning"!

Fifth Idea: Stacking

- Put the "deep" back into "deep learning"!
- Run the learner several times. On each run, take at least one snapshot of <u>that</u> learned grammar.
- These snapshots give additional context features!
  - More "red stuff" to sharpen our root sequence model.
  - We don't know if there's an NP immediately to the left: but we know that grammar #2 thought there was an 80% posterior chance of one, and that's a feature.

- (cf. Christodoulopoulos et al. 2011, who iterate tag induction and dependency grammar induction)

Sixth Idea: Vector-Valued Nonterminals

- Linguists know we need richer symbols!
- And so do we: PCFG generates badly.
- All of the foregoing could be applied to a PCFG-like formalism where the symbols are vectors and the grammar is a CRF that models p(2 children | 1 parent).
- But we haven't implemented it yet.
  - Requires variational approximation.

Summary

- Deep learning in grammar induction doesn’t correspond to the depth in the tree.
  - Convolution is “two-dimensional.”
- It might correspond to iterated learning.
- Context is important, at least during learning.
- But at the end of the day, we need the tree structure to be largely responsible for the words.
  - That’s the only reason we’ll learn a good tree structure.
  - Annealing away the effect of context is one solution.