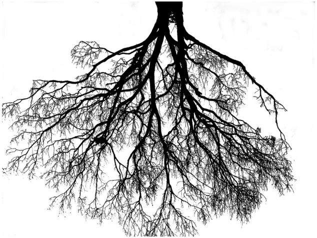
Probabilistic Context-free Grammars and Other Syntactic Language Models



Jon Dehdari

January 4, 2016

Kelsey and other Grammers

- A **grammar** here is another word for a language model
- They consist of four sets $G = \langle \Sigma, N, S, P \rangle$

terminals – word types; lowest nodes in syntax trees
Examples: *dog, the, eats*

non-terminals – phrasal types; middle nodes in syntax trees
Examples: *VP, DET, NP*

start symbol – "S"; the top node in syntax trees

Kelsey and other Grammers

- A **grammar** here is another word for a language model
- They consist of four sets $G = \langle \Sigma, N, S, P \rangle$

terminals – word types; lowest nodes in syntax trees

Examples: *dog, the, eats*

non-terminals – phrasal types; middle nodes in syntax trees

Examples: *VP, DET, NP*

start symbol – "S"; the top node in syntax trees

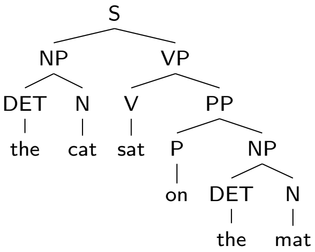
production rules – recursive symbol substitutions

Examples:

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow DET\ N \\
NP &\rightarrow ADJ\ N \\
VP &\rightarrow V\ NP \\
VP &\rightarrow V \\
N &\rightarrow dog \\
N &\rightarrow cat \\
V &\rightarrow barks \\
DET &\rightarrow the
\end{aligned}
$$

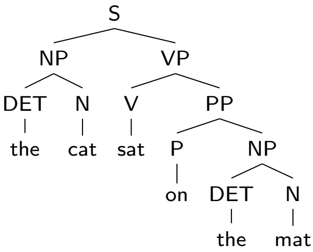
Visualization

- Sentences are often visualized using **derivation trees**, also known as **parse trees** or **syntax trees**
- Example:



Visualization

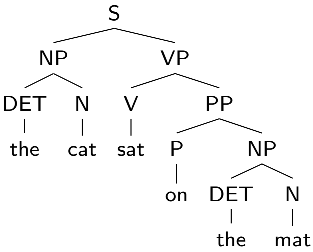
- Sentences are often visualized using **derivation trees**, also known as **parse trees** or **syntax trees**
- Example:



$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & DET\ N \\
DET & \rightarrow & the \\
N & \rightarrow & cat \\
VP & \rightarrow & V\ PP \\
V & \rightarrow & sat \\
PP & \rightarrow & P\ NP \\
N & \rightarrow & mat
\end{array}
$$

Visualization

- Sentences are often visualized using **derivation trees**, also known as **parse trees** or **syntax trees**
- Example:

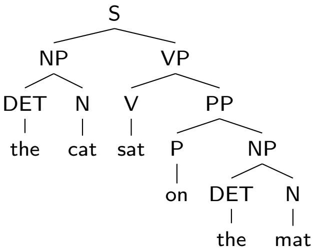


$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow DET\ N \\
DET &\rightarrow the \\
N &\rightarrow cat \\
VP &\rightarrow V\ PP \\
V &\rightarrow sat \\
PP &\rightarrow P\ NP \\
N &\rightarrow mat
\end{aligned}
$$

- Originally these trees were **mere visualizations** of how you could generate a grammatical sentence, given a grammar

Visualization

- Sentences are often visualized using **derivation trees**, also known as **parse trees** or **syntax trees**
- Example:

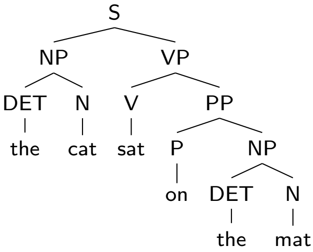


$$
\begin{aligned}
S &\rightarrow NP\ VP\\
NP &\rightarrow DET\ N\\
DET &\rightarrow the\\
N &\rightarrow cat\\
VP &\rightarrow V\ PP\\
V &\rightarrow sat\\
PP &\rightarrow P\ NP\\
N &\rightarrow mat
\end{aligned}
$$

- Originally these trees were **mere visualizations** of how you could generate a grammatical sentence, given a grammar
- Then people started to think of these trees as the actual **structure** of a sentence

Visualization

- Sentences are often visualized using **derivation trees**, also known as **parse trees** or **syntax trees**
- Example:



$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & DET\ N \\
DET & \rightarrow & the \\
N & \rightarrow & cat \\
VP & \rightarrow & V\ PP \\
V & \rightarrow & sat \\
PP & \rightarrow & P\ NP \\
N & \rightarrow & mat
\end{array}
$$

- Originally these trees were **mere visualizations** of how you could generate a grammatical sentence, given a grammar
- Then people started to think of these trees as the actual **structure** of a sentence
- Confusion ensued

Context-free Grammars

- A **context-free grammar** (CFG) is a generative model that can generate context-free languages, which are somewhere in the middle of the formal language hierarchy
- Many, but not all, phenomena in natural languages can be generated by CFGs

Context-free Grammars

- A **context-free grammar** (CFG) is a generative model that can generate context-free languages, which are somewhere in the middle of the formal language hierarchy
- Many, but not all, phenomena in natural languages can be generated by CFGs
- Context-free production rules have the general form of a non-termal rewriting to a sequence (string) of terminals and/or non-terminals ($A \rightarrow \alpha$)

Context-free Grammars

- A **context-free grammar** (CFG) is a generative model that can generate context-free languages, which are somewhere in the middle of the formal language hierarchy
- Many, but not all, phenomena in natural languages can be generated by CFGs
- Context-free production rules have the general form of a non-termal rewriting to a sequence (string) of terminals and/or non-terminals $(A \to \alpha)$
- CFGs can generate and recognize **center embedding**, but not more complex word order phenomena, so effectively CFG parse trees have **no crossing lines**

Context-free Grammars

- A **context-free grammar** (CFG) is a generative model that can generate context-free languages, which are somewhere in the middle of the formal language hierarchy
- Many, but not all, phenomena in natural languages can be generated by CFGs
- Context-free production rules have the general form of a non-termal rewriting to a sequence (string) of terminals and/or non-terminals ($A \rightarrow \alpha$)
- CFGs can generate and recognize **center embedding**, but not more complex word order phenomena, so effectively CFG parse trees have **no crossing lines**
- Non-projective dependency grammars are more or less equivalent to CFGs (they have the same weak generative capacity)

Treebanks

- It's a lot of work to define a language model by hand (including context-free grammars), so another way is to annotate treebanks
- Example: (S (NP (DET the) (N cat))(VP (V sat)(PP (P on)(NP (DET the) (N mat)))))

Treebanks

- It's a lot of work to define a language model by hand (including context-free grammars), so another way is to annotate treebanks
- Example: (S (NP (DET the) (N cat))(VP (V sat)(PP (P on)(NP (DET the) (N mat)))))
- There are treebanks for about 10–20 languages, the Penn Treebank being the most well-known for English

Treebanks

- It's a lot of work to define a language model by hand (including context-free grammars), so another way is to annotate treebanks
- Example: (S (NP (DET the) (N cat))(VP (V sat)(PP (P on)(NP (DET the) (N mat)))))
- There are treebanks for about 10–20 languages, the Penn Treebank being the most well-known for English
- Treebanks can be annotated with various grammatical annotations, like **constituency / phrase-structure** (as above), **dependency grammar** (as we saw last class), categorial grammar, HPSG, etc.
- Most of these annotation styles can be approximately mapped to other styles

Treebanks

- It's a lot of work to define a language model by hand (including context-free grammars), so another way is to annotate treebanks
- Example: (S (NP (DET the) (N cat))(VP (V sat)(PP (P on)(NP (DET the) (N mat)))))
- There are treebanks for about 10–20 languages, the Penn Treebank being the most well-known for English
- Treebanks can be annotated with various grammatical annotations, like **constituency / phrase-structure** (as above), **dependency grammar** (as we saw last class), categorial grammar, HPSG, etc.
- Most of these annotation styles can be approximately mapped to other styles
- Here is a link to a list of syntactic treebanks

PCFGs

- We can induce a **probabilistic context-free grammar** (PCFG) from the treebank
- With multiple annotated sentences, we can get probabilities for production rules. Example:

1.0	S	$\rightarrow NP\ VP$
0.6	NP	$\rightarrow DET\ N$
0.4	NP	$\rightarrow ADJ\ N$
0.7	VP	$\rightarrow V\ NP$
0.3	VP	$\rightarrow V$
0.8	N	$\rightarrow dog$
0.2	N	$\rightarrow cat$
1.0	V	$\rightarrow barks$
1.0	DET	$\rightarrow the$

PCFGs

- We can induce a **probabilistic context-free grammar** (PCFG) from the treebank
- With multiple annotated sentences, we can get probabilities for production rules. Example:

1.0	S	$\rightarrow NP\ VP$
0.6	NP	$\rightarrow DET\ N$
0.4	NP	$\rightarrow ADJ\ N$
0.7	VP	$\rightarrow V\ NP$
0.3	VP	$\rightarrow V$
0.8	N	$\rightarrow$ *dog*
0.2	N	$\rightarrow$ *cat*
1.0	V	$\rightarrow$ *barks*
1.0	DET	$\rightarrow$ *the*

- Notice that the probabilities for each left-hand side must sum to one (unity)

Parameter Estimation

- So how do we get these probabilities?
- If we have a treebank, we can start with just counting how often productions occur (maximum likelihood estimation)

Parameter Estimation

- So how do we get these probabilities?
- If we have a treebank, we can start with just counting how often productions occur (maximum likelihood estimation)
- If we don't have a treebank, we can still use unannotated text, and apply the **inside-outside algorithm**
- The inside–outside algorithm is just the expectation–maximization (EM) algorithm applied to trees
- We start by randomly initializing probabilities to all possible rule productions, then use EM to search for good rule probabilities that maximize the likelihood of the training set

Inside-Outside Algorithm

- The inside–outside algorithm uses inside- and outside-probabilities:
  - Inside probability: $\beta_j(p,q) = P(w_{pq} | N^j_{pq}, G)$
  - Outside probability: $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} | G)$

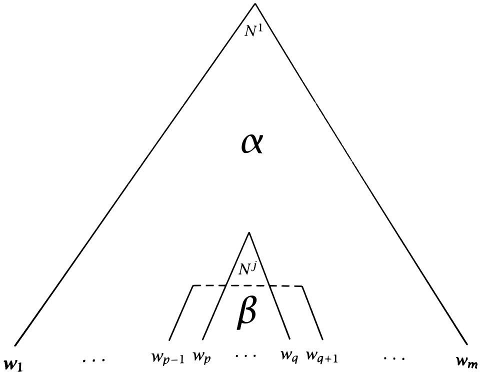


Image courtesy of Manning & Schütze (1999), figure 11.3.

String Probabilities

- We use the inside probability of the entire sentence to get the probability of that sentence:

$$P(w_{1\,m}|G) = P(N^1 \overset{*}{\Rightarrow} w_{1\,m}|G) = \beta(1, m)$$

- Inside probabilities are calculated recursively & compositionally for each rule production
- We can do this because rule productions in context-free grammars are, well, context-free!

String Probabilities

- We use the inside probability of the entire sentence to get the probability of that sentence:

$$P(w_{1\,m}|G) = P(N^1 \stackrel{*}{\Rightarrow} w_{1\,m}|G) = \beta(1, m)$$

- Inside probabilities are calculated recursively & compositionally for each rule production
- We can do this because rule productions in context-free grammars are, well, context-free!
- Probabilities of ambiguous parses at a given non-terminal are summed, since either parse could have produced the final substring

PCFGs *vs.* *n*-gram Language Models (Lexicalized Probabilistic Regular Grammars)

- PCFGs can better handle long-distance dependencies like subject-verb agreement and filler-gap dependencies
- PCFGs usually give worse perplexity than *n*-gram LMs. Why?

PCFGs *vs.* n-gram Language Models (Lexicalized Probabilistic Regular Grammars)

- PCFGs can better handle long-distance dependencies like subject-verb agreement and filler-gap dependencies
- PCFGs usually give worse perplexity than n-gram LMs. Why? Mostly because PCFGs are unlexicalized – they use pre-terminals (word classes / POS tags). Thus they fail to account for local co-occurrences like multiword expressions and proper names.

PCFGs *vs.* n-gram Language Models (Lexicalized Probabilistic Regular Grammars)

- PCFGs can better handle long-distance dependencies like subject-verb agreement and filler-gap dependencies
- PCFGs usually give worse perplexity than n-gram LMs. Why? Mostly because PCFGs are unlexicalized – they use pre-terminals (word classes / POS tags). Thus they fail to account for local co-occurrences like multiword expressions and proper names.
- PCFGs take longer to train
- PCFGs need manually-annotated treebanks to give decent results
- PCFG parsers (eg. CKY) are usually not incremental