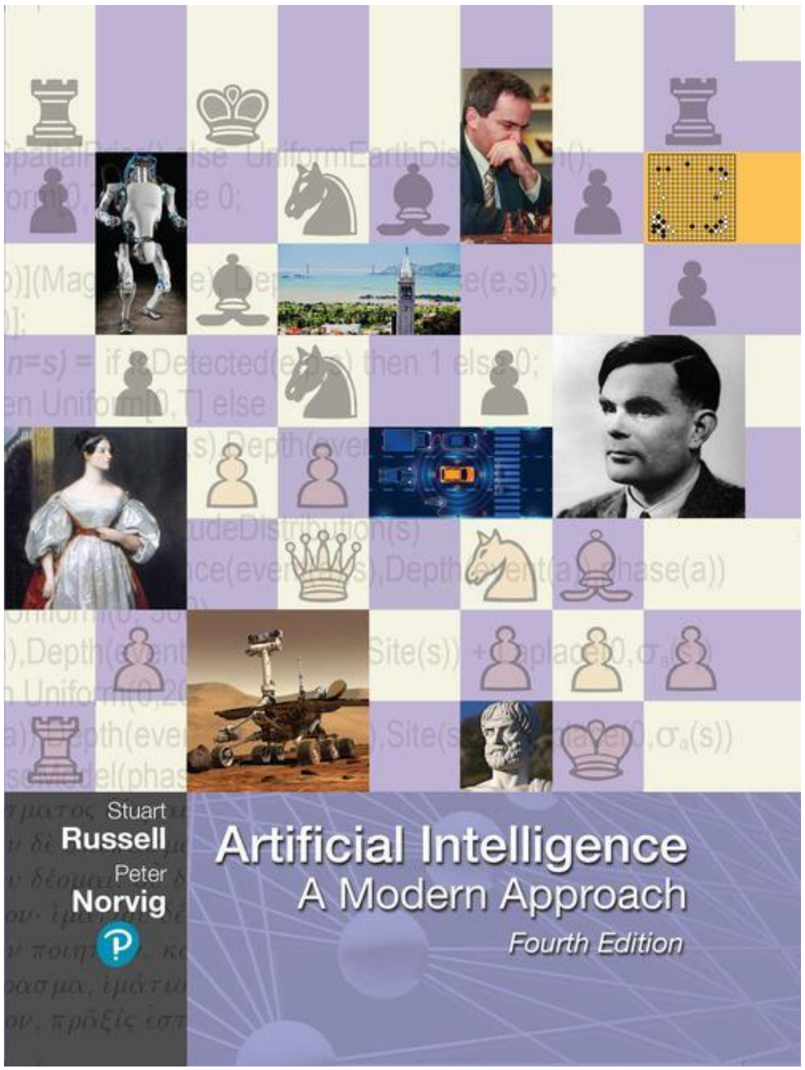
Artificial Intelligence: A Modern Approach

Fourth Edition



Chapter 23

Natural Language Processing

Outline

- Language Models
- Grammar
- Parsing
- Augmented Grammars
- Complications of Real Natural Language
- Natural Language Tasks

Language Models

Language Models

- Language judgments vary from person to person and time to time
- Natural language is ambiguous and vague ("He saw her duck.")
- The mapping from symbols to objects is not formally defined ("Richard")

- **Language model**: a probability distribution describing the likelihood of any string. [English is not defined as is Python 3.10]

The bag-of-words model

- The application of naive Bayes to strings of words
- generative model that describes a process for generating a sentence
- *Class* can be business, sports, weather, programming etc.

Given a sentence consisting of the words $w_1, w_2, \ldots w_N$

$$\mathbf{P}(Class \mid w_{1:N}) = \alpha\, \mathbf{P}(Class) \prod_j \mathbf{P}(w_j \mid Class)\,.$$

- Each category has a bag full of words
- To generate text, first select one of the bags and discard the others
- into that bag and pull out a word at random [This is a terrible model]
- the process of dividing a text into a sequence of words is called tokenization

Pearson

Language Models

N-gram word models

("first quarter earnings report" vs "fourth quarter touchdown passes")
making a word dependent on **all** previous words in a sentence

$$P(w_{1:N}) = \prod_{j=1}^{N} P(w_j \mid w_{1:j-1}).$$

- captures all possible interactions between words, but it is not practical
  - a vocabulary of 100,000 words and a sentence length of 40, this model would have 10^{200} parameters to estimate.
  - Compromise with a Markov chain model
  - n-gram model: a sequence of written symbols of length n is called an n-gram, with special cases "unigram" for 1-gram, "bigram" for 2-gram, and "trigram" for 3-gram
  - In an n-gram model, the probability of each word is dependent only on the $n-1$ previous words;

$$P(w_{1:N}) = \prod_{j=1}^{N} P(w_j \mid w_{j-n+1:j-1}).$$

Language Models

Other n-gram models

- character-level model
    - Probability of each character is determined by the $n - 1$ previous characters
    - for dealing with unknown words. [dextroamphetamine, Plattsburg]
    - for languages that tend to run words together [Geschwindigkeitsbegrenzung]
    - well suited for the task of language identification
- skip-gram model
    - count words that are near each but skip a word or more between them [je ne comprends pas => je comprends + ne pas]

Language Models

Smoothing n-gram models

chance that we will be asked to evaluate a text containing an unknown or out-of-vocabulary word: one that never appeared in the training corpus

- One approach to handle: modify the training corpus by replacing infrequent words with a special symbol, traditionally <UNK>.

The **problem** is that some low-probability n-grams appear in the training corpus, while other equally low-probability n-grams happen to not appear at all.

- apply **smoothing** to all the similar n-grams—reserving some probability mass of the model for **never seen n-grams**, to reduce the variance of the model
  - estimate the probability of rare events by Pierre-Simon Laplace
  - Another option is backoff model,
    - Start by estimating n-gram counts
    - If low or zero count of sequence back off to $(n-1)$-grams.

- **Linear interpolation smoothing** is a backoff model that combines trigram, bigram, unigram models by linear interpolation

$$\hat{P}(c_i|c_{i-2:i-1}) = \lambda_3 P(c_i|c_{i-2:i-1}) + \lambda_2 P(c_i|c_{i-1}) + \lambda_1 P(c_i),$$

where $\lambda_3 + \lambda_2 + \lambda_1 = 1$

Language Models

Word representations

[a fulvous cat => fulvous is an adjective]

The *n*-gram model misses this generalization because it is an *atomic* model

- each word is an atom, distinct from every other word, with no internal structure

Dictionary: structured word model.

- **Wordnet** :
  - open-source, hand-curated dictionary in machine readable format
  - help separate the nouns from the verbs,
  - get the basic categories

Language Models

Part-of-speech (POS) tagging

- way to categorize words (lexical category/tag)
- POS allows language models to capture generalizations such as "adjectives generally come before nouns in English" [not in French]
- useful first step in many other NLP tasks, such as question answering or translation [speech: *record* Noun vs *record* Verb]

Hidden Markov model (HMM)
is a common (**generative**) model for POS tagging

- HMM is a generative model that says that the way to produce language is to start in one state, such as IN, the state for prepositions. Two choices:
  - What word should be emitted
  - What state should come next

- A weakness of HMM models is that everything we know about language has to be expressed in terms of the transition and sensor models.

logistic regression weights in the logistic regression model correspond to how predictive each feature is for each category;

- the weight values are learned by gradient descent
- **a discriminative model -** Cannot generate random sentences

Language Models

A set of POS tagging features used by a logistic model might include:
verb (VBP), past tense verb (VBD), noun (NN), present tense verb (VBP)

“ I walk to school” walk is a verb (bottom left row)

$w_{i-1} =$ “I”	$w_{i+1} =$ “for”
$w_{i-1} =$ “you”	$c_{i-1} = \text{IN}$
w_i ends with “ous”	w_i contains a hyphen
w_i ends with “ly”	w_i contains a digit
w_i starts with “un”	w_i is all uppercase
$w_{i-2} =$ “to” and $c_{i-1} = \text{VB}$	w_{i-2} has attribute PRESENT
$w_{i-1} =$ “I” and $w_{i+1} =$ “to”	w_{i-2} has attribute PAST

Language Models

Tag	Word	Description	Tag	Word	Description
CC	*and*	Coordinating conjunction	PRP$	*your*	Possessive pronoun
CD	*three*	Cardinal number	RB	*quickly*	Adverb
DT	*the*	Determiner	RBR	*quicker*	Adverb, comparative
EX	*there*	Existential there	RBS	*quickest*	Adverb, superlative
FW	*per se*	Foreign word	RP	*off*	Particle
IN	*of*	Preposition	SYM	+	Symbol
JJ	*purple*	Adjective	TO	*to*	to
JJR	*better*	Adjective, comparative	UH	*eureka*	Interjection
JJS	*best*	Adjective, superlative	VB	*talk*	Verb, base form
LS	*1*	List item marker	VBD	*talked*	Verb, past tense
MD	*should*	Modal	VBG	*talking*	Verb, gerund
NN	*kitten*	Noun, singular or mass	VBN	*talked*	Verb, past participle
NNS	*kittens*	Noun, plural	VBP	*talk*	Verb, non-3rd-sing
NNP	*Ali*	Proper noun, singular	VBZ	*talks*	Verb, 3rd-sing
NNPS	*Fords*	Proper noun, plural	WDT	*which*	Wh-determiner
PDT	*all*	Predeterminer	WP	*who*	Wh-pronoun
POS	*’s*	Possessive ending	WP$	*whose*	Possessive wh-pronoun
PRP	*you*	Personal pronoun	WRB	*where*	Wh-adverb
$	$	Dollar sign	#	#	Pound sign
“	‘	Left quote	”	’	Right quote
(	[	Left parenthesis	)	]	Right parenthesis
,	,	Comma	.	!	Sentence end
:	;	Mid-sentence punctuation			

Part-of-speech tags (with an example word for each tag) for the Penn Treebank corpus (Marcus *et al.*, 1993). Here “3rd-sing” is an abbreviation for “third person singular present tense.”

Language Models

To get a feeling for what word models can do, we built unigram, bigram, and trigram models over the words in this book and then randomly sampled sequences of words from the models. The results are

Unigram: logical are as are confusion a may right tries agent goal the was
Bigram: systems are very similar computational approach would be represented
Trigram: planning and scheduling are integrated the success of naive Bayes model is
n = 4: taking advantage of the structure of Bayesian networks and developed various languages for writing "templates" with logical variables, from which large networks could be constructed automatically for each problem instance.

Language Models

Added the King James Bible to the 4-gram model yielding these random samples:

• Prove that any 3-SAT problem can be reduced to simpler ones using the laws of thy God.
• Masters, give unto your servants that which is true iff both P and Q in any model m by a simple experiment: put your hand unto, ye and your households for it is pleasant.
• Many will intreat the LORD your God, Saying, No; but we will ignore this issue for now; Chapters 7 and 8 suggest methods for compactly representing very large belief states.
• And it came to pass, as if it had no successors.
• The direct utility estimation is just an instance of the general or algorithm in which new function symbols are constructed "on the fly." For example, the first child of the Holy Ghost.

Grammar

- A **grammar** is a set of rules that defines the tree structure of allowable phrases [e.g., BNF is a context free grammar for computer languages]

- A **language** is the set of sentences that follow those rules.

- **Syntactic categories** such as noun phrase or verb phrase help to constrain the probable words at each point within a sentence

- The **phrase structure** provides a framework for the meaning or semantics of the sentence

Probabilistic context-free grammar (PCFG)

- A probabilistic grammar assigns a probability to each string
- “context-free” means that any rule can be used in any context

PCFG grammar **applied to Wumpus World**

- will define a for a tiny fragment of English that is suitable for communication between agents exploring the Wumpus world language
- **Grammar rule**

$$
\begin{array}{lcll}
Adjs & \rightarrow & Adjective & [0.80] \\
 & \mid & Adjective\ Adjs & [0.20]
\end{array}
$$

Grammar

S	→	*NP VP*	[0.90]	I + feel a breeze
	|	*S Conj S*	[0.10]	I feel a breeze + and + It stinks
NP	→	*Pronoun*	[0.25]	I
	|	*Name*	[0.10]	Ali
	|	*Noun*	[0.10]	pits
	|	*Article Noun*	[0.25]	the + wumpus
	|	*Article Adjs Noun*	[0.05]	the + smelly dead + wumpus
	|	*Digit Digit*	[0.05]	3 4
	|	*NP PP*	[0.10]	the wumpus + in 1 3
	|	*NP RelClause*	[0.05]	the wumpus + that is smelly
	|	*NP Conj NP*	[0.05]	the wumpus + and + I
VP	→	*Verb*	[0.40]	stinks
	|	*VP NP*	[0.35]	feel + a breeze
	|	*VP Adjective*	[0.05]	smells + dead
	|	*VP PP*	[0.10]	is + in 1 3
	|	*VP Adverb*	[0.10]	go + ahead
Adjs	→	*Adjective*	[0.80]	smelly
	|	*Adjective Adjs*	[0.20]	smelly + dead
PP	→	*Prep NP*	[1.00]	to + the east
RelClause	→	*RelPro VP*	[1.00]	that + is smelly

The grammar for $\mathcal{E}_0$, with example phrases for each rule. The syntactic cate-gories are sentence (*S*), noun phrase (*NP*), verb phrase (*VP*), list of adjectives (*Adjs*), prepo-sitional phrase (*PP*), and relative clause (*RelClause*).

Grammar

Noun	$\rightarrow$	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | ...
Verb	$\rightarrow$	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | ...
Adjective	$\rightarrow$	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] ...
Adverb	$\rightarrow$	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | ...
Pronoun	$\rightarrow$	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | ...
RelPro	$\rightarrow$	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] | ...
Name	$\rightarrow$	**Ali** [0.01] | **Bo** [0.01] | **Boston** [0.01] | ...
Article	$\rightarrow$	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | ...
Prep	$\rightarrow$	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | ...
Conj	$\rightarrow$	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] | ...
Digit	$\rightarrow$	**0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | ...

The lexicon for $\mathscr{E}_0$. *RelPro* is short for relative pronoun, *Prep* for preposition, and *Conj* for conjunction. The sum of the probabilities for each category is 1.

Parsing

- **Parsing** is the process of analyzing a string of words to uncover its phrase structure, according to the rules of a grammar.
- Search for a valid parse tree whose leaves are the words of the string
- can start with the *S* symbol and search top down, or we can start with the words and search bottom up.
- **Inefficiency**: If the algorithm guesses wrong, it will have to backtrack all the way to the first word and reanalyze the whole sentence under the other interpretation.
- **dynamic programming**: every time we analyze a substring, store the results so we won't have to reanalyze it later.
- **Chart parser:** records result in a data structure known as a chart

Parsing

List of items	Rule
S	
NP VP	$S \to NP\ VP$
NP VP Adjective	$VP \to VP\ Adjective$
NP Verb Adjective	$VP \to Verb$
NP Verb **dead**	*Adjective* $\to$ **dead**
NP **is dead**	*Verb* $\to$ **is**
Article Noun **is dead**	$NP \to Article\ Noun$
Article **wumpus is dead**	*Noun* $\to$ **wumpus**
the wumpus is dead	*Article* $\to$ **the**

Parsing the string "The wumpus is dead" as a sentence, according to the gram-mar $\mathcal{E}_0$. Viewed as a top-down parse, we start with *S*, and on each step match one nontermi-nal *X* with a rule of the form $(X \to Y \ldots)$ and replace *X* in the list of items with $Y \ldots$; for example replacing *S* with the sequence *NP VP*. Viewed as a bottom-up parse, we start with the words "the wumpus is dead", and on each step match a string of tokens such as $(Y \ldots)$ against a rule of the form $(X \to Y \ldots)$ and replace the tokens with *X*; for example replacing "the" with *Article* or *Article Noun* with *NP*.

Parsing

The CYK algorithm for parsing. [Cocke, Younger and Kasami]

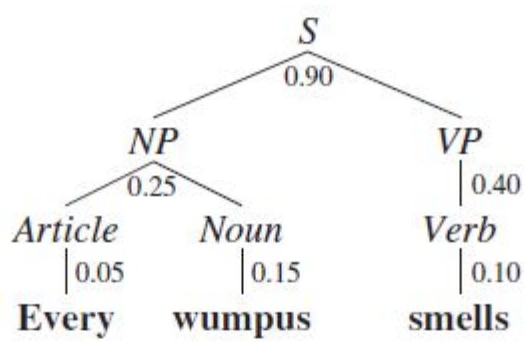
```
function CYK-PARSE(words, grammar) returns a table of parse trees
  inputs: words, a list of words
          grammar, a structure with LEXICALRULES and GRAMMARRULES
  T ← a table        // T[X, i, k] is most probable X tree spanning words_{i:k}
  P ← a table, initially all 0      // P[X, i, k] is probability of tree T[X, i, k]
  // Insert lexical categories for each word.
  for i = 1 to LEN(words) do
    for each (X, p) in grammar.LEXICALRULES(words_i) do
      P[X, i, i] ← p
      T[X, i, i] ← TREE(X, words_i)
  // Construct X_{i:k} from Y_{i:j} + Z_{j+1:k}, shortest spans first.
  for each (i, j, k) in SUBSPANS(LEN(words)) do
    for each (X, Y, Z, p) in grammar.GRAMMARRULES do
      PYZ ← P[Y, i, j] × P[Z, j+1, k] × p
      if PYZ > P[X, i, k] do
        P[X, i, k] ← PYZ
        T[X, i, k] ← TREE(X, T[Y, i, j], T[Z, j + 1, k])
  return T

function SUBSPANS(N) yields (i, j, k) tuples
  for length = 2 to N do
    for i = 1 to N + 1 − length do
      k ← i + length − 1
      for j = i to k − 1 do
        yield (i, j, k)
```

Parsing

CYK is $O(n^2m)$ but A* is faster for simpler grammars.
Using *A** search

- don't have to search entire state space
- guaranteed that the first parse found will be the most probable (assuming an admissible heuristic).
- faster than CYK, but (depending on the details of the grammar) still slower than O(n).
- Parse tree for the sentence "Every wumpus smells" and probabilities:

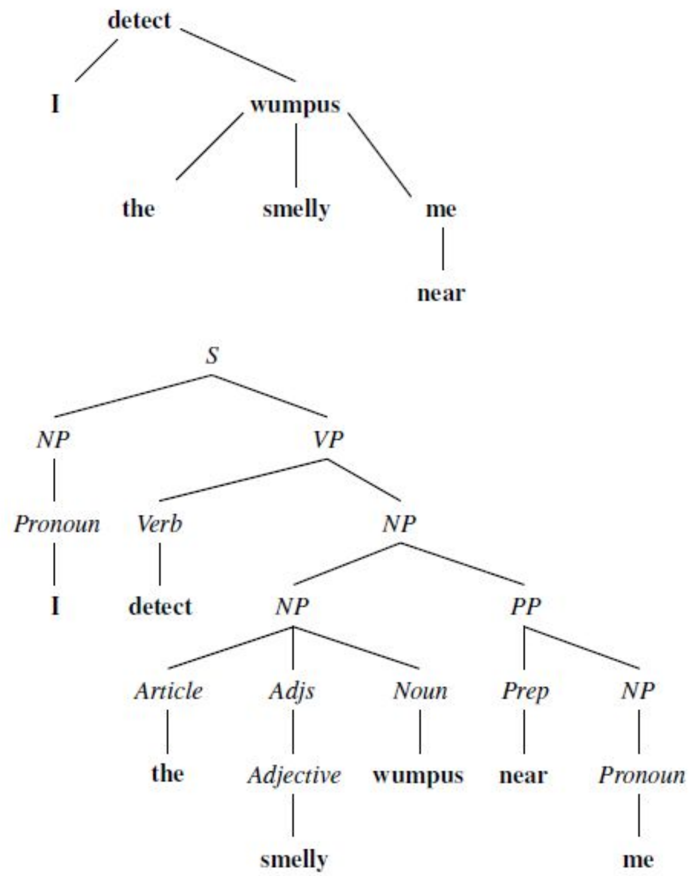


Parsing

Using beam search

- consider only the b most probable alternative parses
- Not guaranteed to find the parse with highest probability
- However parser can operate in $O(n)$ time and still finds the best parse most of the time
- beam search parser with $b = 1$ is called a **deterministic parser**.
- Example: shift-reduce parsing
  - go through the sentence word by word
  - choosing at each point whether to shift the word onto a stack of constituents,
  - or to reduce the top constituent(s) on the stack according to a grammar rule

Parsing



A dependency-style parse (top) and the corresponding phrase structure parse (bottom) for the sentence *I detect the smelly wumpus near me.*

Parsing

Dependency parsing

dependency grammar:

- assumes that syntactic structure is formed by binary relations between lexical items, without a need for syntactic constituents

- phrase structure tree is annotated with the head of each phrase
  - recover dependency tree

- Convert dependency tree to phrase structure with arbitrary categories

Parsing

Learning a parser from examples

- Given treebank, create a PCFG just by counting the number of times each node-type appears in a tree (with the usual caveats about smoothing low counts)

- Annotated tree [from Penn Treebank] for

"Her eyes were glazed as if she didn't hear or even see him"

```
[ [S [NP-2 Her eyes]
     [VP were
         [VP glazed
             [NP *-2]
             [SBAR-ADV as if
                   [S [NP she]
                      [VP did n't
                         [VP [VP hear [NP *-1]]
                             or
                             [VP [ADVP even] see [NP *-1]]
                             [NP-1 him]]]]]]]]
  .]
```

Parsing

Learning a parser from examples

- If there are 1000 S nodes of which 600 are of this form, then we create the rule:

$$S \rightarrow NP\ VP\ [0.6]$$

- Penn Treebank has over 10,000 different node types

- “the good and the bad” is parsed as a single noun phrase

$$NP \rightarrow Article\ Noun\ Conjunction\ Article\ Noun\ .$$

Other approaches:

- **unsupervised parsing**: learn new grammar (or improve an existing grammar) using a corpus of sentences without trees
- **curriculum learning**: start with the easy part of the curriculum—short unambiguous 2-word sentences
- **semisupervised parsing**: start with a small number of trees as data to build an initial grammar, then add a large number of unparsed sentences to improve the grammar. [e.g. use HTML tags]

Augmented Grammars

Pronoun category

- "I" : can be subject of a sentence, singular
- "me" : cannot be subject of a sentence, plural
- *Pronoun* that has been augmented with features like "subjective case, first person singular" is called a **subcategory**

- **Lexicalized PCFG:** a type of augmented grammar that allows us to assign probabilities based on properties of the words in a phrase other than just the syntactic categories

$$\begin{array}{lll}
VP(v) & \rightarrow Verb(v)\ NP(n) & [P_1(v,n)] \\
VP(v) & \rightarrow Verb(v) & [P_2(v)] \\
NP(n) & \rightarrow Article(a)\ Adjs(j)\ Noun(n) & [P_3(n,a)] \\
NP(n) & \rightarrow NP(n)\ Conjunction(c)\ NP(m) & [P_4(n,c,m)] \\
Verb(\textbf{ate}) & \rightarrow \textbf{ate} & [0.002] \\
Noun(\textbf{banana}) & \rightarrow \textbf{banana} & [0.0007]
\end{array}$$

- The notation $VP(v)$ denotes a phrase with category *VP* whose head word is *v*.
- Here $P_1(v, n)$ means the probability of a *VP* headed by *v* joining with an *NP* headed by *n* to form a *VP*.

Augmented Grammars

encode facts completely in the probability entries, for example:

- $P_1(v, she)$ made a very small number, for all verbs v.

$$S(v) \rightarrow NP(Sbj, pn, n)\ VP(pn, v)\ [P_5(n, v)]$$

NP is followed by a *VP* they can form an *S*, but only if the *NP* has the subjective (*Sbj*) case and the person and number (*pn*) of the *NP* and *VP* are identical.

$$
\begin{aligned}
S(v) &\rightarrow NP(Sbj,pn,n)\ VP(pn,v) \mid \ldots \\
NP(c,pn,n) &\rightarrow Pronoun(c,pn,n) \mid Noun(c,pn,n) \mid \ldots \\
VP(pn,v) &\rightarrow Verb(pn,v)\ NP(Obj,pn,n) \mid \ldots \\
PP(head) &\rightarrow Prep(head)\ NP(Obj,pn,h) \\
Pronoun(Sbj,1S,\mathbf{I}) &\rightarrow \mathbf{I} \\
Pronoun(Sbj,1P,\mathbf{we}) &\rightarrow \mathbf{we} \\
Pronoun(Obj,1S,\mathbf{me}) &\rightarrow \mathbf{me} \\
Pronoun(Obj,3P,\mathbf{them}) &\rightarrow \mathbf{them} \\
\\
Verb(3S,\mathbf{see}) &\rightarrow \mathbf{see}
\end{aligned}
$$

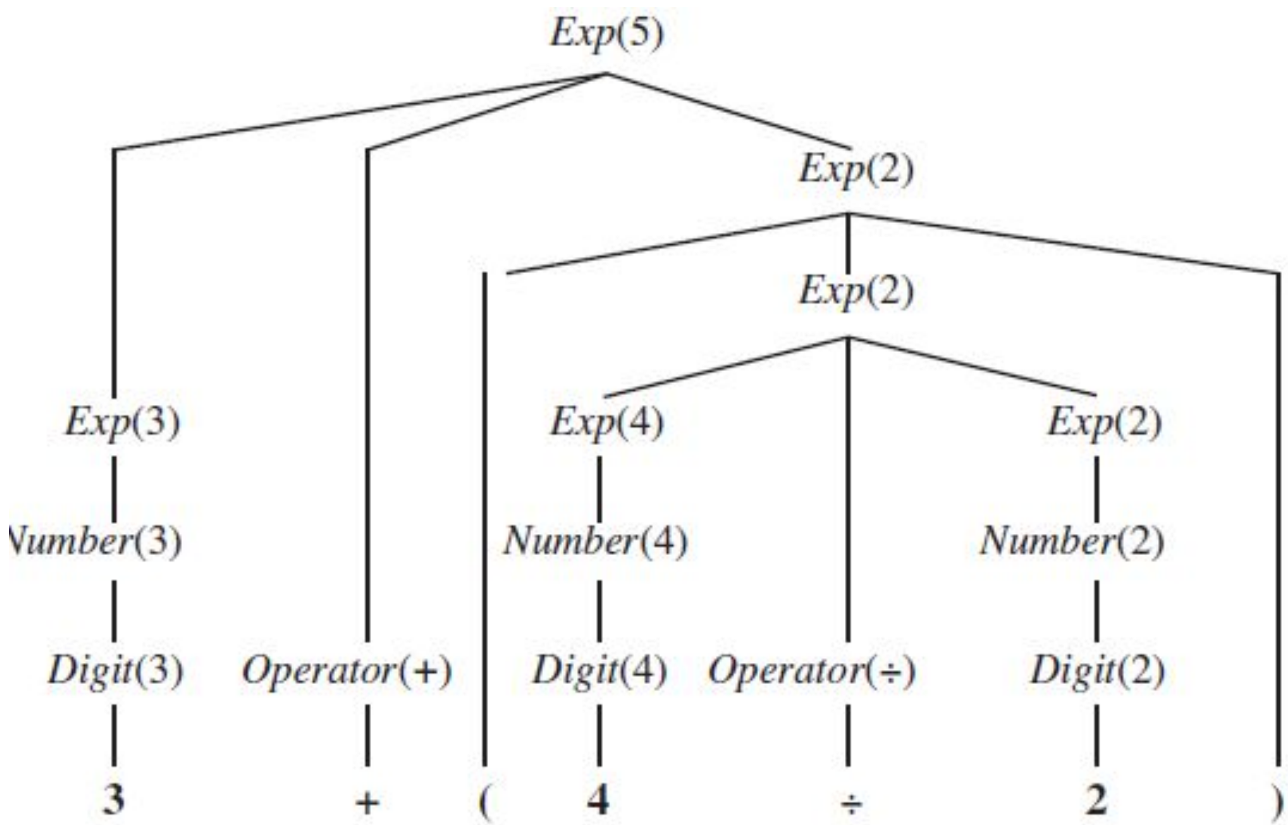
Part of an augmented grammar that handles case agreement, subject–verb agreement,and head words.

Augmented Grammars

$$
\begin{aligned}
&Exp(op(x_1,x_2)) \rightarrow Exp(x_1)\ Operator(op)\ Exp(x_2)\\
&Exp(x) \rightarrow (\ Exp(x)\ )\\
&Exp(x) \rightarrow Number(x)\\
&Number(x) \rightarrow Digit(x)\\
&Number(10 \times x_1 + x_2) \rightarrow Number(x_1)\ Digit(x_2)\\
&Operator(+) \rightarrow \mathbf{+}\\
&Operator(-) \rightarrow \mathbf{-}\\
&Operator(\times) \rightarrow \times\\
&Operator(\div) \rightarrow \div\\
&Digit(0) \rightarrow \mathbf{0}\\
&Digit(1) \rightarrow \mathbf{1}\\
&\ldots
\end{aligned}
$$

Grammar for arithmetic expressions, augmented with semantics. Each variable x_i represents the semantics of a constituent.

Augmented Grammars



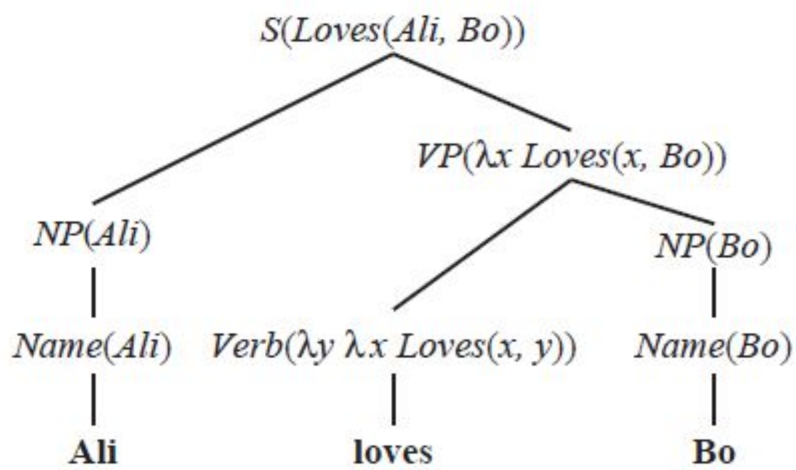
Parse tree with semantic interpretations for the string “3 + (4 ÷ 2)”.

Augmented Grammars

$$S(pred(n)) \rightarrow NP(n)\ VP(pred)$$
$$VP(pred(n)) \rightarrow Verb(pred)\ NP(n)$$
$$NP(n) \rightarrow Name(n)$$

$$Name(Ali) \rightarrow \textbf{Ali}$$
$$Name(Bo) \rightarrow \textbf{Bo}$$
$$Verb(\lambda y\ \lambda x\ Loves(x,y)) \rightarrow \textbf{loves}$$

(a)



(b)

(a) A grammar that can derive a parse tree and semantic interpretation for "Ali loves Bo" (and three other sentences). Each category is augmented with a single argument representing the semantics.

(b) A parse tree with semantic interpretations for the string "Ali loves Bo."

Augmented Grammars

Learning semantic grammars

Sentence: What states border Texas?
Logical Form: $\lambda x.state(x) \wedge \lambda x.borders(x, Texas)$

large collection of pairs like this and a little bit of hand-coded knowledge for each new domain, the system generates plausible lexical entries

much easier to gather examples of question/answer pairs:

Question: What states border Texas?
Answer: Louisiana, Arkansas, Oklahoma, New Mexico.
Question: How many times would Rhode Island fit into California?
Answer: 135

Complications of Real Natural Language

Quantification: “Every agent feels a breeze.”

- Standard approach: define not an actual logical semantic sentence, but rather **a quasi-logical form** that is then turned into a logical sentence by algorithms outside of the parsing process
- have preference rules for choosing one quantifier scope over another

Pragmatics:

- resolving the meaning of indexicals, which are phrases that refer directly to the current situation
- Example sentence: “I am in Boston today,” both “I” and “today” are indexicals. The word “I” would be represented by Speaker, a fluent that refers to different objects at different times
- interpreting the **speaker’s intent**
- The speaker’s utterance is considered a speech act, and it is up to the hearer to decipher what type of action it is (question, a statement, a promise, a warning, a command, etc.) “go to 2 2”

Complications of Real Natural Language

Long-distance dependencies:

- Standard approach: define not an actual logical semantic sentence, but rather **a quasi-logical form** that is then turned into a logical sentence by algorithms outside of the parsing process
- have preference rules for choosing one quantifier scope over another

Time and tense:

- English uses verb tenses (past, present, and future) to indicate the relative time of an event.
- One good choice to represent the time of events is the event calculus notation

$$\text{Ali loves Bo: } E_1 \in Loves(Ali, Bo) \land During(Now, Extent(E_1))$$
$$\text{Ali loved Bo: } E_2 \in Loves(Ali, Bo) \land After(Now, Extent(E_2))\,.$$

$$Verb(\lambda y\ \lambda x\ e \in Loves(x, y) \land During(Now, e)) \ \rightarrow\ \textbf{loves}$$
$$Verb(\lambda y\ \lambda x\ e \in Loves(x, y) \land After(Now, e)) \ \rightarrow\ \textbf{loved}\,.$$

Pearson

Complications of Real Natural Language

Ambiguity:

- Squad helps dog bite victim.
- Police begin campaign to run down jaywalkers.
- Helicopter powered by human flies.
- Once-sagging cloth diaper industry saved by full dumps.
- Include your children when baking cookies.
- Portable toilet bombed; police have nothing to go on.
- Milk drinkers are turning to powder.
- Two sisters reunited after 18 years in checkout counter.

Pearson

Complications of Real Natural Language

Ambiguity:

- tend to think of ambiguity as a failure in communication
- when a listener is consciously aware of an ambiguity in an utterance, it means that the utterance is unclear or confusing.
- **Lexical ambiguity** is when a word has more than one meaning
  - “back” can be an adverb (go back),
  - an adjective (back door),
  - a noun (the back of the room),
  - a verb (back a candidate), or
  - a proper noun (a river in Nunavut, Canada).

Syntactic ambiguity

refers to a phrase that has multiple parses: “I smelled a wumpus in 2,2” has two parses: one where the prepositional phrase “in 2,2” modifies the noun and one where it modifies the verb.

- Leads to **semantic ambiguity,**

Complications of Real Natural Language

Disambiguation is the process of recovering the most probable intended meaning of an utterance.

disambiguation requires combination of four models:

- **The world model:** the likelihood that a proposition occurs in the world.
- **The mental model:** the likelihood that the speaker forms the intention of communicating a certain fact to the hearer.
- **The language model:** the likelihood that a certain string of words will be chosen, given that the speaker has the intention of communicating a certain fact.
- **The acoustic model:** for spoken communication, the likelihood that a particular sequence of sounds will be generated, given that the speaker has chosen a given string of words.

Natural Language Tasks

Speech recognition is the task of transforming spoken sound into text.

- We can then perform further tasks on the resulting text
- Current systems have a word error rate of about 3% to 5%
- The challenge: to respond appropriately even when there are errors on individual words.

Text-to-speech synthesis is the inverse process—going from text to sound

Machine translation transforms text in one language to another.

Information extraction is the process of acquiring knowledge by skimming a text and looking for occurrences of particular classes of objects and for relationships among them.

Information retrieval is the task of finding documents that are relevant and important for a given query

Question Answering is a different task, in which the query really is a question

Summary

- Probabilistic language models based on n-grams recover a surprising amount of information about a language
- Word embeddings can give a richer representation of words and their similarities.
- To capture the hierarchical structure of language, phrase structure grammars (and in particular, context-free grammars) are useful
- Sentences in a context-free language can be parsed in $O(n^3)$ time by a **chart parser** such as the **CYK algorithm**, which requires grammar rules to be in **Chomsky Normal Form**.
- A treebank can be a resource for learning a PCFG grammar with parameters.
- Semantic interpretation can also be handled by an augmented grammar.