自然语言处理(Natural language processing)

- 语言模型
- 文法及句法分析

Slides based on the AIMA textbook, 4th edition.

Why NLP?

- 在很多情况下，人类使用语音与计算机进行交互是很方便的，而且在大多数情况下，使用自然语言要比使用一阶谓词演算等形式语言更加方便。
- 人类已经用自然语言记录了很多知识。仅维基百科就有3000万页事实知识，然而几乎没有任何一个这样的知识来源是用形式逻辑写成的。如果我们想让计算机系统知道很多知识，它最好能理解自然语言。

自然语言任务

- 语音识别（speech recognition）:将语音转换为文本
  - 当前语音识别系统的单词错误率大约为3%～5%
- 文本-语音合成（text-to-speech synthesis）:与语音识别相反的过程–将文本转换为语音
- 机器翻译（machine translation）：将文本从一种语言转换到另一种语言
- 信息提取（information extraction）：通过浏览文本查找文本中特定类别的对象及其关系
  - 典型的任务包括，从网页中提取地址实例，获取街道名、城市名、州名以及邮政编码等数据库字段

自然语言任务

- 信息检索（information retrieval）：查找与给定查询相关的重要文档
- 问答（question answering）：查询是一个问题，查询结果不是一个排好序的文档列表，而是一个实际答案
  - 如“Who founded the U.S. Coast Guard？”，“Alexander Hamilton.”

语言模型（language model）

- 形式语言，如一阶逻辑，是精确定义的：文法定义合法句的句法，语义规则定义其含义
- 然而，自然语言无法如此清晰地表示
- 我们将语言模型定义为描述任意字符串可能性的概率分布
  - 如“Do I dare disturb the universe?”作为一个英语字符串具有合理的概率，
  - 而“Universe dare the I disturb do?”是英语字符串的可能性极低。

语言模型

- 语言模型是各种自然语言任务的核心，例如
  - 通过语言模型，我们可以预测文本中接下来可能出现的单词，从而为电子邮件或短信息提供补全建议。
  - 我们可以计算出对文本进行哪些更改会使其具有更高的概率，从而提供拼写或文法更正建议。
- 自然语言是复杂的，因此任何语言模型至多是自然语言的一个近似.

词袋（bag-of-words）模型

- 句子分类的朴素贝叶斯模型
- 给定一个由N个单词组成的句子$w1, w2, \ldots, w_N$，记为$w_{1:N}$，
- $P(Class|w_{1:N}) = \alpha P(Class) \prod_j P(w_j|Class)$

词袋模型

- 它是一种生成模型，描述了句子生成的过程：
  - 想象对于每个类别(business、weather等)，都有一个装满单词的袋子(每个单词都写在袋子内的纸条上，一个单词越常见，重复的纸条就越多)。
  - 要生成一段文本，首先选择其中一个袋子。从那个袋子中随机抽出一个单词，这将是句子的第一个单词。然后将这个单词放回并抽取第二个单词。重复上述操作直到出现句末指示符(如句号)。
- 这一模型显然是错误的：它错误地假设每个单词都与其他单词无关，因此无法生成连贯的英语语句。

模型的学习

- 通过在文本语料库（corpus）上进行监督训练
- 文本的每个部分都标有它的类别
- 语料库通常由至少一百万字的文本和上万个不同词汇组成
- 例如，如果总共有3000条文本，其中300条被分类为business，那么可以估计$P(Class = business) \approx 0.1$.
- 如果在business类中，总共有100000个单词，而“stocks”一词出现了700次，那么可以估计$P(stocks|Class = business) \approx 0.007$.

其他模型

- 如逻辑回归、神经网络
- 特征是词汇表中的单词：“a”，“aardvark”,...,"zyzzyva"，
- 值是每个单词在文本中出现的次数(或者是一个布尔值，表示该单词是否在文本中出现)。
- 这导致特征向量是高维且稀疏的– 在语言模型中，可能有100000个单词，因此特征向量的维数为100000。但是对短文本来说，特征向量的绝大部分特征都是0。
- 为改进模型效果，可以进行特征选择，限制特征为单词中的一个子集
- 我们可以删除非常罕见的单词，和所有类别共有但对分类不起作用的单词(如“the”)

N元（N-gram）单词模型

- 引人一种新模型，其中每个单词都依赖于之前的单词
  $P(w_{1:N}) = \prod_{j=1}^{N} P(w_j | w_{1:j-1})$
- 但它并不实用：当词汇表中有100000个单词，句子长度为40时，模型需要估计10^{20}个参数
- 折中方法：在n元模型中，每个单词出现的概率仅依赖于前面的$n-1$个单词
- 适用于对报纸版面进行分类，垃圾邮件检测，情感分析，以及作者归属

其他n元模型

- 一种替代方法是字符级模型，其中每个字符的概率由之前的$n-1$个字符决定。
- 这种方法有助于处理未知单词，也有助于处理单词间不空格的语言
- 字符级模型适用于语言识别任务，即给定一个文本，确定它是用哪种语言写的
- 非常擅长某些特定的分类任务，如将单词识别为药品名，人名，或城市名

n元模型的平滑

- 像“of the”之类的高频n元在训练语料库中的计数值很高，因此它们的概率估计很可能是准确的
- 而低频n元由于计数值很低，容易受到随机噪声的干扰，方差较大。
- 此外，可能需要处理包含未知单词的文本。虽然未知单词从未在训练语料库中出现过，但将这类单词的概率设为0是错误的，因为整个句子的概率也将变为0。
- 对未知单词进行建模的一种方法是修改训练语料库，用特殊符号替换不常见单词，一般替换为<UNK>。
- 然后，我们照常对语料库中的n元计数，把<UNK>符号看作同其他任意单词一样。当测试集中出现未知单词时，我们查找<UNK>的概率。

词性(Part-of-speech, POS)标注

- 对单词进行分类的一种基本方法是依据它们的词性:名词、动词、形容词等。
- 词性允许语言模型捕提一般模式，例如，"英语中形容词通常在名词之前"
- 下图为Penn Treebank中使用的45个标签，Penn Treebank是一个由超过300万个单词的文本组成，并用词性标签标记的语料库。

From	the	start	,	it	took	a	person	with	great	qualities	to	succeed
IN	DT	NN	,	PRP	VBD	DT	NN	IN	JJ	NNS	TO	VB

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			

Figure 23.1 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.”

逻辑回归用于词性标注

- 在逻辑回归中，输入是特征值向量x。然后，我们将这些特征与预训练的权重向量w进行点积$w \cdot x$，然后将和转换为0~1之间的数字，这个数字可以解释为该输人是一个类别的正例的概率
- 逻辑回归模型中的权重对应于每个特征对每个类别的预测能力；权重值可以通过梯度下降法学习
- 对于词性标注，我们将建立45个不同的逻辑回归模型，每个词性使用一个模型
- 给定一个单词在特定上下文中的特征值，模型将给出该单词属于该类别的可能性

词性标注的特征

- 词性标注通常使用二元特征，对正在标注的单词w_i（可能还有其他相邻单词）及分配给前一个单词的词性c_{i-1}（可能还有更前面单词的词性）等信息进行编码
- 一组词性标注特征可能包括:

$w_{i-1} =$ "I"	$w_{i+1} =$ "for"
$w_{i-1} =$ "you"	$c_{i-1} =$ 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 $c_{i-1} =$ VB	w_{i-2} has attribute PRESENT
$w_{i-1} =$ "I" and $w_{i+1} =$ "to"	w_{i-2} has attribute PAST

- 例如，单词"walk " 可以是名词也可以是动词，但在 " I walk to school " 中，使用词性标注特征中最后一行左列的特征，我们可以将 " walk " 分类为动词

用逻辑回归处理序列

- 逻辑回归没有输入序列的概念:给定一个单独的特征向量（单个单词的相关信息），产生一个输出标注
- 可以强制逻辑回归模型通过贪心搜索来处理序列:首先为第一个单词选择最可能的标注，然后按照从左到右的顺序处理其余单词。在每一步，如下分配词性：
  $c_i = argmax_{c'} P(c'|w_{1:N}, c_{1:i-1})$
- 贪心搜索为每个单词做出确定性的词性选择，然后转到下一个单词;即使这一选择与句子后面的证据相矛盾，也不回溯。因此，算法速度很快,但准确性较低。
- 一个折中方案是束搜索，在每个时间步，只保留b个最可能的标注。

语言模型的比较

对本书中的单词建立了一元模型（即词袋）模型、二元、三元和四元模型，然后从4个模型中随机采样单词序列。

- $n = 1$: *logical are as are confusion a may right tries agent goal the was*
- $n = 2$: *systems are very similar computational approach would be represented*
- $n = 3$: *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*

可以清楚地看出，一元模型表现很差，而四元模型虽然并不完美，但效果要好得多。

N元模型的局限性

- 随着n的增加，n元模型将生成更流畅的语言，但它们倾向于逐字复制训练数据中的长段落，而不是生成新的文本。
- 具有更复杂的单词和上下文表示的语言模型可以在这方面做得更好。
- 后续将介绍文法如何对语语言模型进行改进，以及目前深度学习方法如何生成优秀的语言模型。

深度学习模型GPT-2

给定提示时可以生成与提示相关的流畅的英语样例. 例如,
提示: There is a limit to n-gram models – as n increases, they will produce language that is more fluent, but they tend to reproduce long passages from their training data verbatim, rather than generate novel text. Language models with more complex representations of words and context can do better.

生成: Numerous studies have shown that the best language models that can generate language are those that use neural networks, and in particular that use deep learning to model speech. I have used one of these models to generate hundreds of sentences in multiple languages, and it has the ability to generate almost perfect translations of my training sentences. It's a great model to have, and will save you a lot of time in the future when you need to translate between many languages.

文法

- 文法（grammar）是一组规则，定义了合法短语的树结构，而语言（language）是遵循这些规则的句子集。
- 自然语言与一阶逻辑形式语言并不完全一样：自然语言中不存在合法句和非法句之间的硬性界限，对于每个句子也没有单一的确定性的树结构。
- 但是，层次结构在自然语言中非常重要。
- 有许多语言模型基于分层句法结构的思想。

概率上下文无关文法

- 一种当前流行的模型是概率上下文无关文法（probabilistic context-free grammar, PCFG)。
- 概率文法为每个字符串分配一个概率，
- “上下文无关”意味着任一规则都可以在任何上下文中使用: 句子开头的名词短语的规则与句子后面的另一个名词短语的规则相同, 如果同一短语出现在两个不同位置，那么每次的概率都必须相同

$$\begin{array}{rcll} Adjs & \rightarrow & Adjective & [0.80] \\ & | & Adjective\ Adjs & [0.20] \end{array}$$

- 以上文法规则的含义：句法范畴Adjs （形容词列表）可以由单个Adjective（形容词）组成（概率为0.80), 也可以由Adjective 加上可以构成Adjs 的字符串组成（概率为0.20)。

我们将为英语的一个小片段$\mathcal{E}_0$定义一个PCFG 文法，
该片段适用于探索wumpus 世界的智能体之间的通信。

S	$\rightarrow$	*NP VP*	[0.90]	I + feel a breeze
	|	*S Conj S*	[0.10]	I feel a breeze + and + It stinks
NP	$\rightarrow$	*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	$\rightarrow$	*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	$\rightarrow$	*Adjective*	[0.80]	smelly
	|	*Adjective Adjs*	[0.20]	smelly + dead
PP	$\rightarrow$	*Prep NP*	[1.00]	to + the east
RelClause	$\rightarrow$	*RelPro VP*	[1.00]	that + is smelly

Figure 23.2 The grammar for $\mathcal{E}_0$, with example phrases for each rule. The syntactic categories are sentence (*S*), noun phrase (*NP*), verb phrase (*VP*), list of adjectives (*Adjs*), prepositional phrase (*PP*), and relative clause (*RelClause*).

$\mathcal{E}_0$的词典

词典（lexicon）：允许使用的单词的列表。

- 名词、名称、动词、形容词和副词被称为开放类（open class），列出所有单词是不可行的。每一类中不仅有成千上万的单词，而且还在不断增加新单词
- 代词、关系代词、介词、介词和连词被称为封闭类（closed class），它们的单词数量很少（十几个），而且要经过几个世纪才发生变化

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

Figure 23.3 The lexicon for $\mathcal{E}_0$. *RelPro* is short for relative pronoun, *Prep* for preposition, and *Conj* for conjunction. The sum of the probabilities for each category is 1.

句法分析（parsing）

- 根据文法规则分析一串单词以获得其短语结构的过程
- 可以将它看作对有效句法分析树的搜索
- 可以从S符号开始自顶向下搜索，在每一步中，用一个规则的右端替换左端
- 也可以从单词开始自底向上搜索，在每一步中，用一个规则的左端替换右端

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

图表句法分析器（chart parser）

- 单纯的自顶向下或自底向上的句法分析可能很低效
- 为提高效率，可以使用动态规划（dynamic programming）：每次分析子字符串时，存储分析结果，以免之后重复分析。
- 可以将结果记录在一种被称为图表（chart）的数据结构中
- CYK算法是一种自底向上的图表句法分析算法的概率版本，以其发明者姓氏的首字母命名

CYK算法

- 要求所有规则都必须为两种特定格式之一:
  $X \rightarrow word[p]$形式的词法规则，以及$X \rightarrow Y\ Z[p]$形式的句法规则，规则右边恰好有两个范畴。
- 这种文法格式称为乔姆斯基范式（Chomsky Normal Form)
- 任何上下文无关文法都可以自动转换为乔姆斯基范式

CYK算法

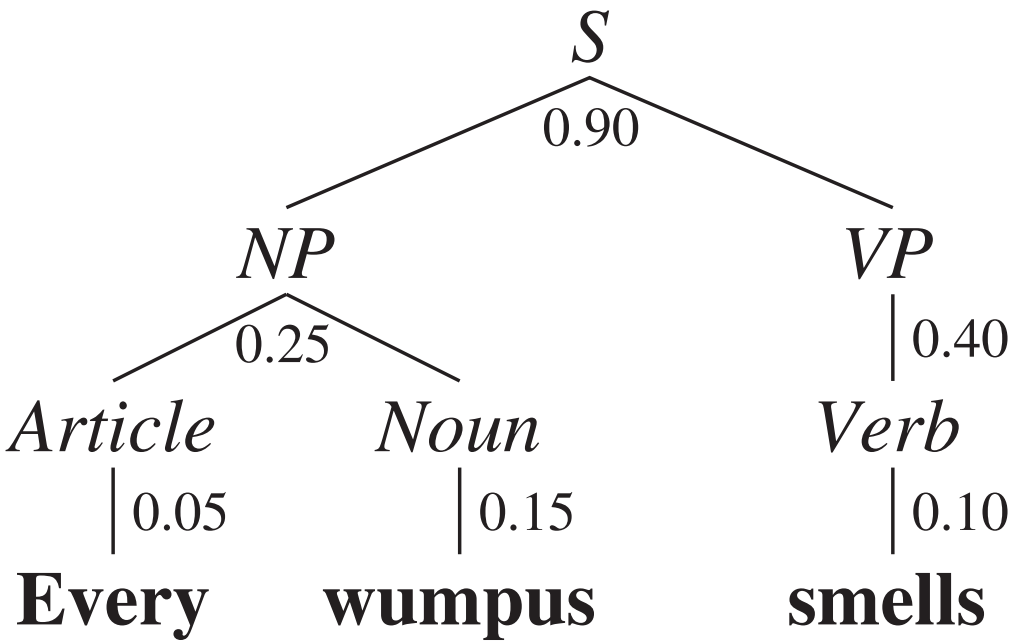
function CYK-PARSE(*words*, *grammar*) **returns** a table of parse trees
  inputs: *words*, a list of words
    grammar, a structure with LEXICALRULES and GRAMMARRULES
  $T \leftarrow$ a table // *T[X, i, k] is most probable X tree spanning* $words_{i:k}$
  $P \leftarrow$ a table, initially all 0 // *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**
      $P[X, i, i] \leftarrow$ p
      $T[X, i, i] \leftarrow$ 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**
      $PYZ \leftarrow P[Y, i, j] \times P[Z, j+1, k] \times p$
      if $PYZ > P[X, i, k]$ **do**
        $P[X, i, k] \leftarrow PYZ$
        $T[X, i, k] \leftarrow$ TREE(X, $T[Y, i, j]$, $T[Z, j+1, k]$)
  return T

表P 和表T 所需空间为$O(n^2m)$，所需时间为$O(n^3m)$，其中n 是句子中单词的数量，m 是文法中非终结符的数量。

CYK算法

- 给定一个单词序列，算法将找出该序列及其子序列的最可能的句法分析树
- 输出表$P[X, i, k]$表示的是单词串$words_{i:k}$构成范畴X的最可能树的概率
- 输出表$T[X, i, k]$表示范畴X涵盖位置i 到k的最可能树
- SUBSPANS 函数返回涵盖单词串$words_{i:k}$ 的所有元组(i, j, k)，其中$i \leq j < k$，按照$i : k$ 区间长度递增的顺序列出元组，
- LEXICALRULES(word)返回一组(X, p)对，其中$X \rightarrow word[p]$ 为一条规则
- GRAMMARRULEs 则返回一组(X, Y, Z, p)元组，其中$X \rightarrow Y\ Z[p]$ 为一条规则

句法分析树示例

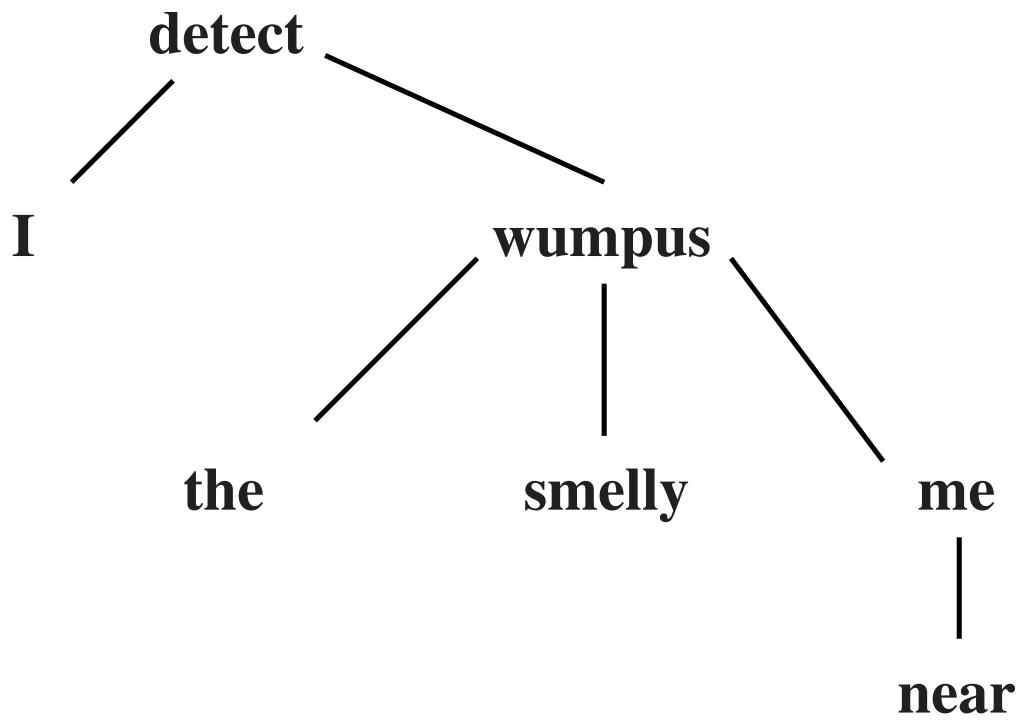


根据$\mathcal{E}_0$文法得到的句子“Every wumpus smells”树的每个内部节点均标有概率。整个树的概率为0.9x0.25x0.05x0.15x0.40x0.10=0.0000675。

依存分析(dependency parsing)

- 另一种广泛使用的句法，称为依存文法（dependency grammar ），它假定句法结构是由词汇项之间的二元关系形成的，不需要句法成分
- 从某种意义上说，依存文法和短语结构文法只是符号上的变体
- 因此，我们不会因为一种符号更强而选择这种符号；我们更愿意选择一种更自然的符号
- 通常，对词序几乎固定的语言（如英语）来说，短语结构树更加自然；对词序几乎无限制的语言（如拉丁语）来说，依存关系树更加自然，

依存分析举例



句子“I detect the smelly wumpus near me”的依存句法分析

从样例中学习句法分析器

- 为大规模的英语语句建立文法是一件费力且容易出错
- 因而直接学习文法规则（及其概率）比人工标注更好。
- 为了应用监督学习方法，我们需要句子的输入 / 输出对及其句法分析树。
- Penn Treebank 是最著名的此类数据的来源，由用句法分析树结构标记的十万多个句子组成。

标记树示例

```
[ [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]]]]]]]
  .]
```

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

一种文法现象：短语从树的一部分移动到另一部分。

这棵树将短语"hear or even see him"分析为由两个VP 成分组成，[VP hear [NP *-1]]和[VP [ADVP even] see [NP *-1]]，两者都含有一个空缺，*-1，表示在树中其他地方被标记为[NP-1 him]的NP。类似地，[NP *-2]表示[NP-2 Her eyes]

从样例中学习句法分析器

- 给定一个树库，可以通过计算每个节点类型在树中出现的次数来创建PCFG（通常需要注意平滑低计数值）
- 在以上例子中，有两个[S [NP ···][VP ···]] 形式的节点
- 我们将对这些节点以及语料库中其他所有根为S 的子树进行计数。
- 如果一共有1000个S 节点，其中600个为上述形式，那么可以创建规则：S $\rightarrow$ NP VP [0.6]

扩展文法

- 目前为止，我们已经处理了上下文无关文法。
- 但是，并不是每个NP 都可以以相同的概率出现在每个上下文中。“I ate a banana”是符合文法的，但是“Me ate a banana”不符合文法，而"I ate a bandanna"则不符合实际（bandanna 是色彩鲜艳的围巾）。
- 问题在于我们的文法侧重于词汇范畴，如代词，但尽管“I”和“me”都是代词，却只有“I”可以作为句子主语。同样，"banana"和"bandanna"都是名词，但前者更有可能成为"ate"的宾语。
- 此外，“I”是第一人称，“I”还是单数
- 像代词这样的范畴，用诸如“主格、第一人称单数”之类的特征进一步扩展，称为子范畴（subcategory ）

词汇化PCFG(lexicalized PCFG)

- PCFG 是一种扩展文法，允许我们根据短语中单词的属性而不仅仅是句法范畴来分配概率。
- 引人短语头（head）的概念– 短语中最重要的词
- 例如，"banana"是NP "a banana"的头，"ate"是VP "ate a banana"的头。
- 符号$VP(v)$表示范畴为VP 且头为v 的短语。

PCFG示例

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

通过保证$P_1(ate, banana) > P_1(ate, bandanna)$，我们可以让"ate a banana"的概率大于"ate a bandanna"的概率。

具有附加变量的PCFG示例

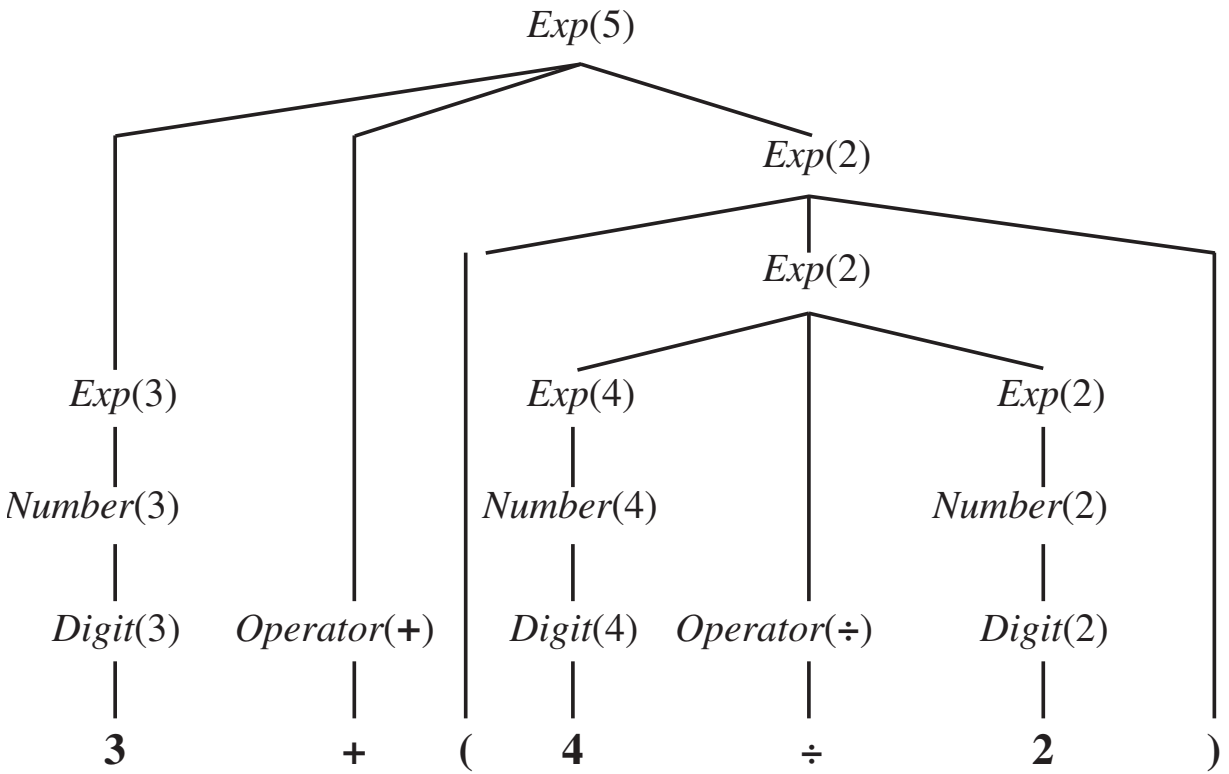
$$
\begin{aligned}
S(v) &\rightarrow NP(Sbj,pn,n)\ VP(pn,v) \mid \ldots \\
NP(c,pn,n) &\rightarrow Pronoun(c,pn,n) \mid Noun(c,pn,n) \mid \ldots \\
VP(pn,v) &\rightarrow Verb(pn,v)\ NP(Obj,pn,n) \mid \ldots \\
PP(head) &\rightarrow Prep(head)\ NP(Obj,pn,h) \\
Pronoun(Sbj,1S,\mathbf{I}) &\rightarrow \mathbf{I}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

- 保有格一致性、主语动词一致性和短语头单词的扩展文法的一部分，省略了概率
- $NP(c, pn, n)$用于表示一个名词短语，c表示格（主格或宾格），pn 表示人称和单复数（如第三人称单数），n 表示短语头。
- 大写字母开头的名称表示常量：Sbj 和Obj 表示主格和宾格；1S 表示第一人称单数；1P和3P分别表示第一人称复数和第三人称复数。小写名称是变量

如何在文法中添加语义?以算术表达式为例

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

每个规则都用一个描述短语语义解释的变量扩展



字符串"3+(4÷2)"的带语义解释的句法分析树

自然语言

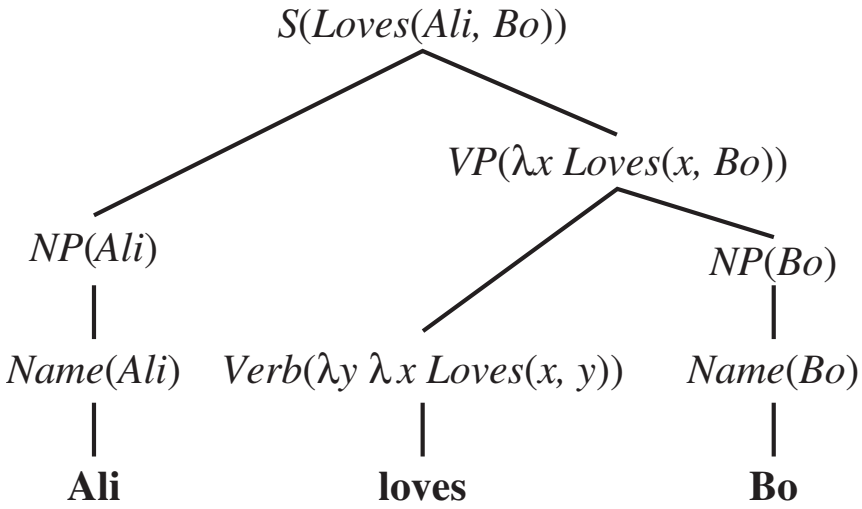
- 我们将使用一阶逻辑进行语义表示
- 用逻辑项Ali 表示NP "Ali"
- 使用λ记号, "loves Bo"表示为谓词$\lambda x Loves(x, Bo)$
- $(\lambda x Loves(x, Bo))(Ali) = Loves(Ali, Bo)$,
  称为λ函数应用的β归约。
- 动词"loves"表示为$\lambda y \lambda x Loves(x, y)$

举例

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

$Name(Ali) \rightarrow$ **Ali**
$Name(Bo) \rightarrow$ **Bo**
$Verb(\lambda y\ \lambda x\ Loves(x,y)) \rightarrow$ **loves**

(a)



(b)

学习语义文法

- Penn Treebank中不包括句子的语义表示，只包括句法树
- (Zettlemoyer and Collins, 2005)中系统从样例中学习问答系统的文法，样例由一个句子和该句子的语义形式组成
  - 句子：What states border Texas?
  - 逻辑形式：$\lambda x.state(x) \wedge borders(x, Texas)$
- 给定大量上述样例以及每个新领域的少量人工标注知识，系统就会生成合理的词汇条目，同时学习文法参数，使得系统可以将句子分析为语义表示
- 系统在两个由未参与学习的句子组成的不同测试集上达到了79%的准确率

学习语义文法

- 以上系统的一个局限性在于训练数据包括逻辑形式
- 创建这样的训练数据代价昂贵，需要具有专业知识的人工注释者
- 而收集问题-答案对样例则要容易得多
  - 问题：What states border Texas?
  - 答案：Louisiana, Arkansas, Oklahoma, New Mexico.
  - 问题：How many times would Rhode Island fit into California?
  - 答案：135
- 这样的问题-答案对在网络上非常常见
- 使用这种大型数据源，有可能构建出比使用标记有逻辑形式的小型数据库构建出的分析器性能更好的句法分析器