# 人工智能-第4次课程作业报告

授课教师：张宇 作者：蒋雨初-58121102

## 1 问题描述

### 1.1题目介绍

给定语言的词典和语言的语法，现输入字符串语句S = “the wumpus is dead”。 请采用短语结构句法解析语句S，并输出句法解析树的线性表示 “[*S*:[*NP*:[*Article*: **the**] [*Noun*: **wumpus**]] [*VP*:[*Verb*: **is**][*Adjective*: **dead**]]]”。

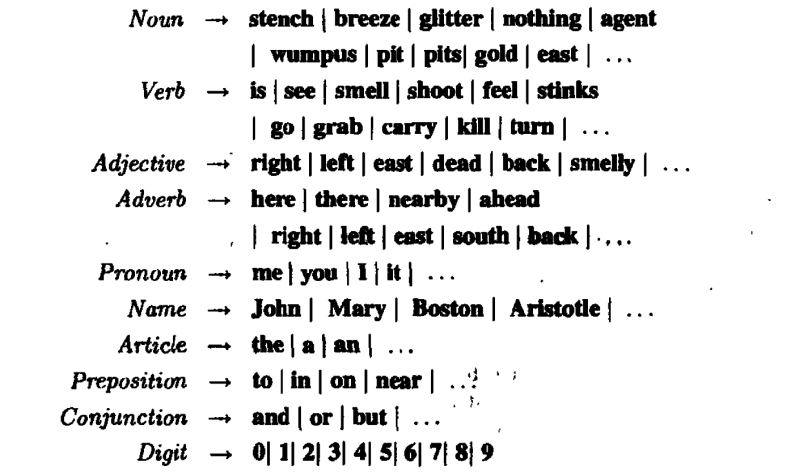


图1

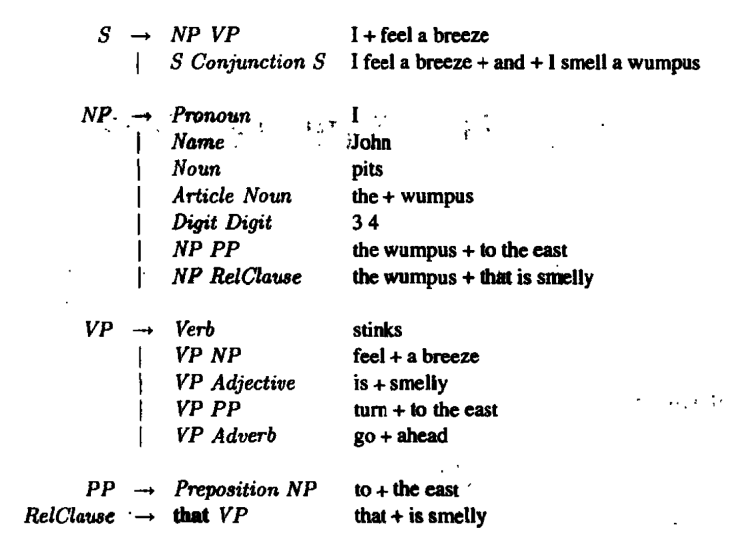


图2

### 1.2任务说明

Nlp类中保存两个重要的成员变量，分别为字典和语法。

字典的数据结构为：map< string, vector<LexRule> >，字典的键为词性，值为vector<LexRule>。

形式如下：

{

*"Noun"*: [

{*"pos"*: "Noun", *"lexicon"*: "stench", *"prob"*: 0.05},

{*"pos"*: "Noun", *"lexicon"*: "breeze", *"prob"*: 0.1}

]

}

最外层的“Noun”代表该键储存所有名词的词汇规则，LexRule有三个成员变量，分别为词性（pos），词汇（lexicon），此种规则出现的概率（prob）。

语法的数据结构为：map< string, vector<CFGRule> >，字典的键为语法规则的左边，值为vector<CFGRule>。

形式如下：

{

*"S"*: [

{*"lhs"*: "S", *"rhs"*: ["NP", "VP"], *"prob"*: 0.9},

{*"lhs"*: "S", *"rhs"*: ["S", "Conj", "S"], *"prob"*: 0.1}

]

}

最外层的“S”代表该键储存所有句子语法规则，CFGRule有三个成员变量，分别为规则左边（lhs），规则右边（rhs），此种规则出现的概率（prob）。上面的例子表示，一个句子有90%的概率会被拆解为名词性词组+动词性词组（NP+VP）的形式，有10%的概率会被拆解为句子+连接词+句子（S+Conj+S）的形式。

### 1.3实验环境

### 设备规格

设备名称 LAPTOP-TFMBQKQ8

处理器 AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

机带 RAM 16.0 GB (15.4 GB 可用)

设备 ID 6CC35513-8821-49C9-A60B-C44F31F302A7

产品 ID 00342-35891-56086-AAOEM

系统类型 64 位操作系统, 基于 x64 的处理器

笔和触控 没有可用于此显示器的笔或触控输入

### 系统规格

版本 Windows 11 家庭中文版

版本 22H2

安装日期 ‎2022/‎9/‎28

操作系统版本 22621.521

体验 Windows Feature Experience Pack 1000.22634.1000.0

### 开发环境

Microsoft Visual Studio Professional 2022

Version 17.2.4

VisualStudio.17.Release/17.2.4+32602.215

Microsoft .NET Framework

Version 4.8.09032

Installed Version: Professional

Visual C++ 2022 00483-00000-00004-AA929

Microsoft Visual C++ 2022

### 语言标准

C++20 or latest.

### 1.4评价标准

1个函数，8分，分数由解析树的正确率决定。由于该句子在给定句法下没有歧义，所以只有正确和错误两种结果。

## 2 实验方案

### 2.1 短语结构语法

第22章提到的n元语言模型是基于单词序列的，这些模型的最大问题是数据稀疏（data sparsity）—对于一个包含10个单词的词汇表，将有10个三元概率需要估计，所以，即使语料库有上万亿个单词，也不能提供可掌的评估，我们可以通过推广的方法来解决稀疏问题。举例来说。“black dog”比“dog black”出现得更频繁，相似的，我们推广成：英语中形容词倾向于出现在名词之前（相反，在法语中形容词倾向于出现在名词之后：“chien noir”出现得更频繁）。当然，总会有例外的情况，“galore”形容词，它跟在它所修饰的名词的后面。尽管有特例，词法范畴（lexical category，也称为词类，part of speech），如名词或形容词，是一种有效的推广方法。但是，如果我们将词类组合成句法范畴，如名词短语或动词短语，并将这些句法范畴构成句子的短语结构（phrase structure）树，每个嵌套短语都表示为一个范畴，模型就会更加有效。

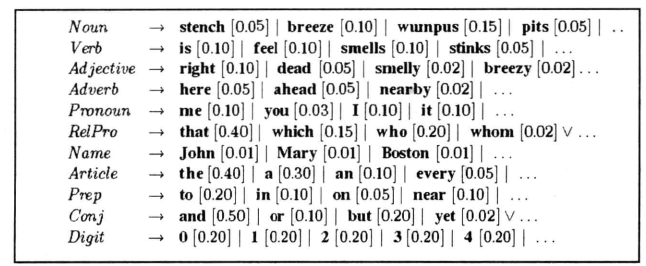
一直有很多基于短语结构思想的有竞争力的语言模型，我们将介绍一个比较流行的模型：概率上下文无关文法（probabilistic context-free grammar）或者PCFG，文法（grammar） 是一个规则的集合，它将语言定义为一个允许的词串集合。“概率”意味着文法给每个字符串分配一个概率，下面是一个PCFG规则：

其中，VP（verb phrase，动词短语）和NP（noun phrase，名词短语）是非终结符（non-terminal symbol）。文法也用到真正的单词，即终结符（terminal symbols）。上面的规则表示，动词短语单独由动词组成的概率为0.70，由一个VP后面跟上一个NP组成的概率为0.30．附录B讲述了非概率的上下文无关文法。

现在我们针对一个小的英语片段定义文法，这个英语片段适用于Agent在探索 wumpus世界时进行相互交流。我们称之为语言。在后续章节里，我们将改进使其更接近于真实英语。要为英语设计一个完整的语法是非常困难的，因为就算是两个人也不会在有效英语的构成上达成一致。

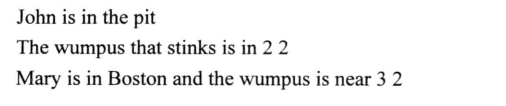
### 2.2 的词典

首先我们定义一个词典（lexicon），或者说合法的词语列表。这些词语按照一种字典使用者熟悉的词法范畴（lexical categories）进行分类：指示事物的名词、代词、名字，指示事件的动词，修饰名词的形容词，修饰动词的副词，以及功能词：冠词（比如the）、介词（in）和连词（and）。下图描述了一个小型的语言的词典。



每个以“···”结尾的范畴说明在这个范畴中还有其他词语。对于名词、名字、动词、形容词和副词而言，即便从原则上说，将它们全部列出也是不可行的。不仅是因为每类中有成千上万的词语，而且经常还有新词加入—如iPod或者biodiesel。这5个范畴被称为开放类（open class）。对于代词、关系代词、冠词、介词以及连词这些范畴，我们努点力就能够完全列举出这些词语。这些范畴被称为封闭类（elosed class），它们只包含少量词语（十来个或更多）。封闭类经过几个世纪才发生变化，而不是几个月。比如，词语“thee”和“thou”在17世纪是被普遍使用的代词，到了19世纪这种趋势才减弱，而今天仅仅在诗歌和某些地区方言中才能见到。

下一步则是将单词组合成短语。图23.2描述了的语法，6个句法范畴的规则，并且为每条重写规则都提供了一个例子，图23.3描述了句子“Every wumpus smells．”的分析树（parse tree）.分析树给出了一各构造性的证明：该单词串的确是符合规则的句子。而语法能够生成很多英语语句，如：



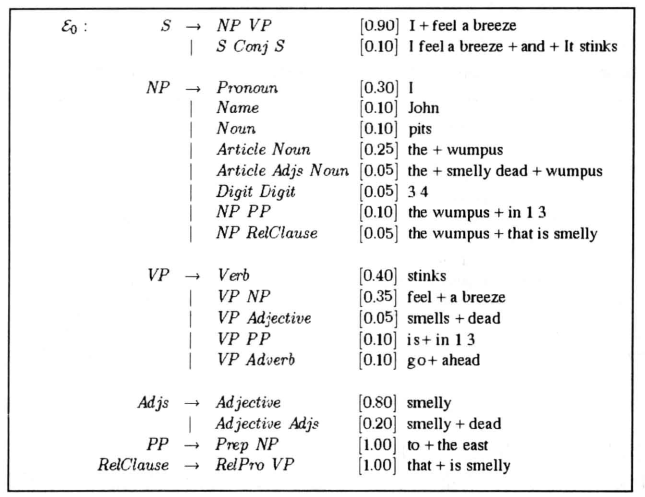


图23.2 的语法，每条规则都给出了一个短语作为例子。句法范畴为句子（S）、名词短语（NP）、

动词短语（VP）、形容词列表（Adjs）、介词短语（PP）和关系从句（RelClause）

不幸的是，该语法会过生成（overgenerate）；也就是说，它可以生成不符合语法的语句。比如“Me go Boston”和“I smell pits wumpus John.”同时它也可能欠生成（undergenerates）： 它会拒绝很多英语中的语句，比如“I think the wumpus is smelly.”下面我们将看到如何学习获得更好的语法，现在我们暂时把注意力集中在运用现有语法我们能做些什么。

### 2.3 句法分析

句法分析（Parsing）是按照语法规则分析单词串从而得到其短语结构的过程。图23.4说明了我们可以从S符号开始，自顶向下搜索并构造以单词作为叶子结点的树，或者我们也可以从单词出发，自底向上搜索并构造树直到顶端S。然而，自顶向下和自底向上句法分析方法都可能比较低效，二者都会对搜索空间中的某些区域做重复工作而浪费时间，而这些区域可能并不能引向成功的分析。考虑以下两个语句：

Have the students in section 2 of Computer Science 101 take the exam.

Have the students in section 2 of Computer Science 101 taken the exam?

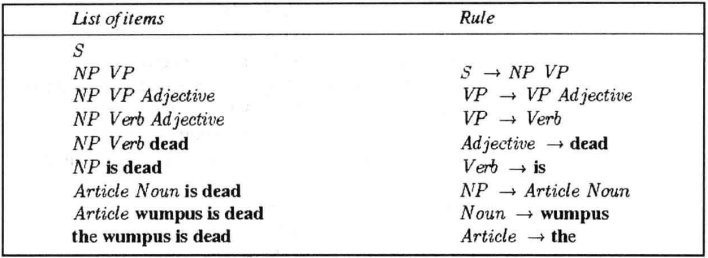


图23.4跟踪“The wumpus is dead”串的分析过程，根据语法寻找它的分析结果。如采取自顶

向下短语分析，项目列表首先为S，之后每一步都是使用形式为的规则匹配项目X，并在项目列表中将X替换为。如采取自底向上分析，项目列表首先为由单词组成的句子，之后每一步都使用形式为的规则匹配项目列表中的符号串，并将列表中的替换为X

尽管它们的前10个单词都是相同的，但是这两个句子却有截然不同的句法分析结果，因为第一句是一个命令句，而第二句是一个疑问句。从左到右的句法分析算法将不得不猜测第一个单词到底是命令句还是疑问句的组成部分，直到处理到第十一个单词take或 taken时，算法才能够确定到底哪个猜测是正确的。当算法猜测错误时，它必须一直回湖到第一个单词，并在另一种解释下重新分析整个句子。

为了这种导致低效率的问题，我们可以使用动态规划方法：每次分析子串时，就将结果存储起来，将来再碰到就无需重新分析了。例如，一旦我们发现“the students in section 2 of Computer Science 101”是一个名词短语，我们可以将结果记录在一种被称为图（chart）的数据结构中。做这种工作的算法被称为图分析器（chart parser）。因为我们使用的是上下文无关文法，因此在搜索空间中，我们在某一分支的上下文中对任何短语的分析，也可以在其他分支中适用。图分析器有很多种类，我们介绍的是一种自底而上的方法，被称为CYK算法（CYK algorithm），该算法以其发明者金名：John Cocke.Daniel Younger Tadeo Kasami.

算法要求文法的所有规则都符合下面两种形式之一：词法规则的形式是。而句法规则的形式是.这种文法形式被称为Chomsky范式（Chomsky Normal Form），虽然看上去是受限的，其实不然：任何上下文无关语法的都可以被自动转换为Chomsky范式，习题23.8可以引导你思考这个转换过程。

CYK算法中表P占用的空间为，其中n是句子中单词的数日，m是文法中非终结符的数日，其时间花费为。（由于m对于某一具体语法而言是常量，通常也被描述为虽然对于限制更加严格的文法可以找到更快的算法，但对于一般的上下文无关语法面言，已经没有比CYK更好的算法。事实上，让一个算法在的时间内完成分析，也是件很技巧的工作，因为一个句子可能会有指数级数目的分析树。

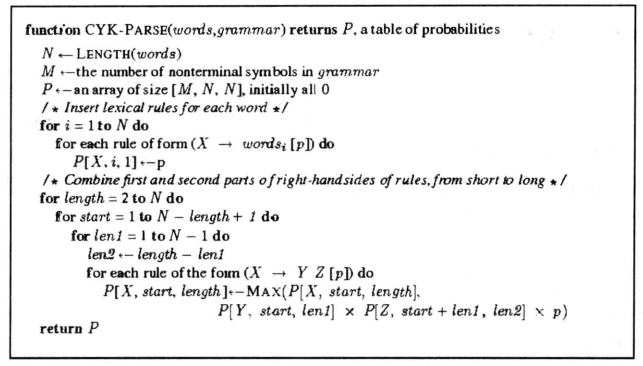
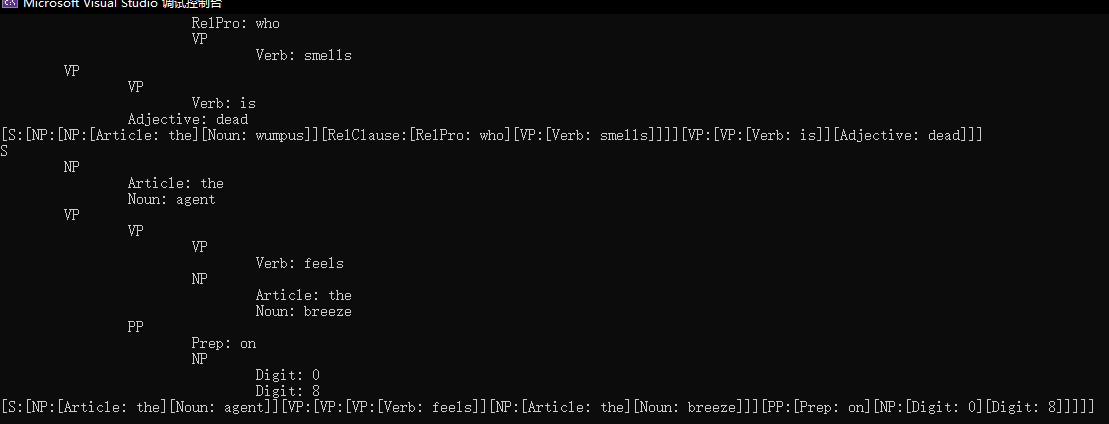


图23.5 CYK句法分析算法。给定一个单词串，算法将找出整个串及其所有子串的最可能的推

导。算法的返回值是整个表P，其中数组表示的是从位置start开始、长度为len的子串构成范畴X的最可能概率。如果在该子串不能构成X，那么概率为0

## 3实验结果



## 4实验分析

[S:[NP:[Article: the][Noun: wumpus]][VP:[VP:[Verb: is]][Adjective: dead]]]

[S:[NP:[NP:[Article: the][Noun: wumpus]][RelClause:[RelPro: who][VP:[Verb: smells]]]][VP:[VP:[Verb: is]][Adjective: dead]]]

[S:[NP:[Article: the][Noun: agent]][VP:[VP:[VP:[Verb: feels]][NP:[Article: the][Noun: breeze]]][PP:[Prep: on][NP:[Digit: 0][Digit: 8]]]]]

## 5 结论

优点：简单易行，执行效率高。  
  弱点：必须对文法进行范式化处理；无法区分歧义。

## 6 代码

