# 从文本提取信息

对于任何给定的问题，很可能已经有人把答案写在某个地方了。以电子形式提供的自然语言文本的数量真的惊人，并且与日俱增。然而，自然语言的复杂性使访问这些文本中的信息非常困难。NLP 目前的技术水平仍然有很长的路要走，才能够从不受限制的文本建立通用的意义重现。如果我们不是集中我们的精力在问题或“实体关系”的有限集合，例如：“不同的设施位于何处”或“谁被什么公司雇用”上，我们就能取得重大进展。本章的目的是要回答下列问题：

1. 我们如何能构建一个系统，从非结构化文本中提取结构化数据？

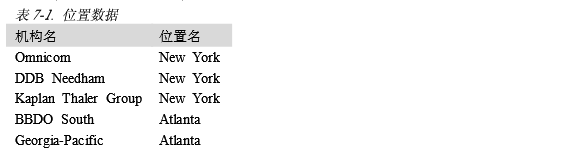
2. 有哪些稳健的方法识别一个文本中描述的实体和关系？

3. 哪些语料库适合这项工作，我们如何使用它们来训练和评估我们的模型？

一路上，我们将应用最后两章中的技术来解决分块和命名实体识别。

## 信息提取

信息有很多种“形状”和“大小”。一个重要的形式是结构化数据：实体和关系的可预测的规范的结构。例如：我们可能对公司和地点之间的关系感兴趣。给定一个公司，我们希望能够确定它做业务的位置；反过来，对于给定位置，我们会想发现哪些公司在该位置做业务。如果我们的数据是表格形式，如表7.1中的例子，那么回答这些问题就很简单了。

如果这个位置数据被作为一个元组(entity, relation, entity)的链表存储在Python 中，那么这个问题：“哪些组织在亚特兰大经营？”可翻译如下：

>>> print [org for (e1, rel, e2) if rel=='IN' and e2=='Atlanta']

['BBDO South', 'Georgia-Pacific']

如果我们尝试从文本中获得相似的信息，事情就比较麻烦了。例如：思考下面的片段(来自nltk.corpus.ieer，fileid为NYT19980315.0085)

The fourth Wells account moving to another agency is the packaged paperproducts division of Georgia-Pacific Corp., which arrived at Wells only last fall. Like Hertz and the History Channel, it is also leaving for an Omnicom-owned agency, the BBDO South unit of BBDO Worldwide. BBDO South in Atlanta, which handles corporate advertising for Georgia-Pacific, will assume additional duties for brands like Angel Soft toilet tissue and Sparkle paper towels, said Ken Haldin, a spokesman for Georgia-Pacific in Atlanta.

如果你通读了上述片段，你将收集到回答例子问题所需的信息。但我们如何能让一台机器理解上述片段返回链表['BBDO South', 'Georgia-Pacific']作为答案呢？这显然是一个困难得多的任务。与表7.1不同，上述片段不包含连结组织名和位置名的结构。 这个问题的解决方法之一是建立一个非常通用的意义重现（第10 章）。在这一章中，我们采取不同的方法，提前定为我们将只查找文本中非常具体的各种信息，如：组织和地点之间的关系。不是试图用文字像上述片段那样直接回答这个问题，我们首先将自然语言句子这样的非结构化数据转换成表 7-1的结构化数据。然后，利用强大的查询工具，如SQL语句。这种从文本获取意义的方法被称为信息提取。

信息提取有许多应用，包括商业智能、简历收获、媒体分析、情感检测、专利检索、电子邮件扫描。当前研究的一个特别重要的领域是提取出电子科学文献的结构化数据，特别是在生物学和医学领域。

**信息提取结构**

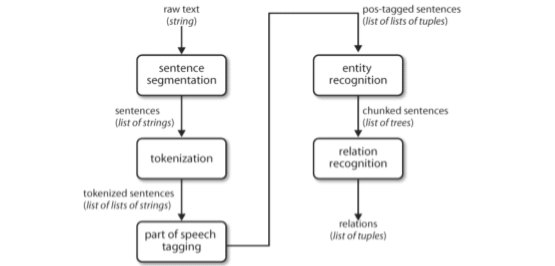
图7-1 显示了一个简单的信息提取系统的结构。它开始于使用第3 章和第5 章讨论过的几个程序处理文档：首先，使用句子分割器将该文档的原始文本分割成句，使用分词器将每个句子进一步细分为词。接下来，对每个句子进行词性标注，在下一步命名实体识别中将证明这是非常有益的。在这一步，我们寻找每个句子中提到的潜在的有趣的实体。最后，我们使用关系识别搜索文本中不同实体间的可能关系。

图7.1 信息提取的简单流水线结构

该系统以一个文档的原始文本作为其输入，生成 (entity, relation, entity)元组的一个链表作为输出 。 例如 ：假设一个文档表明 Georgia -Pacific 公司位于 Atlanta，它可能产生元组 ([ORG: 'Georgia-Pacific'] 'in' [LOC: 'Atla nta']) 。

要执行前面三项任务，我们可以定义一个函数，简单地连接NLTK 中默认的句子分割器，分词器和词性标注器：

>>> def ie\_preprocess(document):

... sentences = nltk.sent\_tokenize(document)

... sentences = [nltk.word\_tokenize(sent) for sent in sentences]

... sentences = [nltk.pos\_tag(sent) for sent in sentences]

接下来，命名实体识别中，我们分割和标注可能组成一个有趣关系的实体。通常情况下，这些将被定义为名词短语，例如 the knights who say "ni"或者适当的名称如 Monty Python。 在一些任务中，同时考虑不明确的名词或名词块也是有用的，如every student或 cats，这些不必要一定与定义NP 和适当名称一样的方式指示实体。

最后，在提取关系时，我们搜索对文本中出现在附近的实体对之间的特殊模式，并使用这些模式建立元组记录实体之间的关系。

## 分块

我们将用于实体识别的基本技术是分块（chunking），分割和标注图 7-2所示的多标识符序列。小框显示词级标识符和词性标注，大框显示较高级别的分块。每个这种较大的框叫做一大块（chunk）。就像分词忽略空白符，分块通常选择标识符的一个子集。同样像分词 一样，分块构成的源文本中的片段不能重叠。

在本节中，我们将在较深的层面探讨分块，以块的定义和表示开始。我们将看到正则表 达式和N-gram 的方法分块，使用 CoNLL-2000 分块语料库开发和评估分块器。我们将在7. 5节和 7.6节回到命名实体识别和关系抽取的任务。

**名词短语分块**

我们将首先思考名词短语分块，或NP-分块（NP-chunking），在那里我们寻找单独名 词短语对应的块。

例如：这里是一些《华尔街日报》文本，其中的NP-块用方括号标记：

1. [ The/DT market/NN ] for/IN [ system-management/NN software/NN ] for/ IN [ Digital/NNP ] [ ’s/POS hardware/NN ] is/VBZ fragmented/JJ enough/RB that/IN [ a/DT giant/NN ] such/JJ as/IN [ Computer/NNP Associates/NNPS ] should/MD do/VB well/RB there/RB ./.

正如我们可以看到，NP-块往往是比完整的名词短语的小片段。例如：the market for s ystem-management software for Digital's hardware 是一个单独的名词短语（含两个嵌套的名 词短语），它中间有一个简单的 NP-块 the market。这种差异的动机之一是NP-块被定义为不包含其他的NP-块。因此，修饰一个名词的任何介词短语或从句将不包括在相应的NP-块内， 因为它们几乎可以肯定包含更多的名词短语。

NP-分块信息最有用的来源之一是词性标记。这是在我们的信息提取系统中进行词性标 注的动机之一。我们在例 7-1中用一个已经标注词性的例句来演示这种方法。为了创建一个 NP-块，我们将首先定义一个块语法，规则句子应如何分块。在本例中，我们将用一个正则 表达式规则定义一个简单的语法。这条规则是说一个NP-块由一个可选的限定词（DT） 后面跟着任何数目的形容词（JJ）然后是一个名词（NN）组成。使用此语法，我们创建了一个块分析器，测试我们的例句④。结果是一棵树，我们可以输出⑤或图形显示⑥。

**标记模式**

组成一个块语法的规则使用标记模式来描述已标注的词的序列。一个标记模式是一个用尖括号分隔的词性标记序列，如<DT>?<JJ>\*<NN>。标记模式类似于正则表达式模式（3. 4节）。现在，思考下面的来自《华尔街日报》的名词短语：

another/DT sharp/JJ dive/NN

trade/NN figures/NNS

any/DT new/JJ policy/NN measures/NNS

earlier/JJR stages/NNS

Panamanian/JJ dictator/NN Manuel/NNP Noriega/NNP

我们可以使用轻微改进的上述第一个标记模式来匹配这些名词短语，如<DT>?<JJ.\* >\*<NN.\*>+。这将把任何以一个可选的限定词开头，后面跟零个或多个任何类型的形容词（包括相对形容词，如earlier/JJR），后面跟一个或多个任何类型的名词的标识符序列分块。

**用正则表达式分块**

要找到一个给定的句子的块结构，RegexpParser分块器以一个没有标识符被分块的平面结构开始。轮流应用分块规则，依次更新块结构。所有的规则都被调用后，返回块结构。

**加缝隙**

有时定义我们想从一个块排除什么比较容易。我们可以为不包括在一大块中的一个标识符序列定义一个缝隙。在下面的例子中，barked/VBD at/IN 是一个缝隙：

[ the/DT little/JJ yellow/JJ dog/NN ] barked/VBD at/IN [ the/DT cat/NN ]

加缝隙是从一大块中去除一个标识符序列的过程。如果匹配的标识符序列贯穿一整块，那么这一整块会被去除；如果标识符序列出现在块中间，这些标识符会被去除，在以前只有一个块的地方留下两个块。如果序列在块的周边，这些标记被去除，留下一个较小的块。表 7-2演示了这三种可能性。

## 开发和评估分块器

现在你对分块的作用有了一些了解，但我们并没有解释如何评估分块器。和往常一样， 这需要一个合适的已标注语料库。我们一开始寻找将IOB 格式转换成 NLTK树的机制，然 后是使用已分块的语料库如何在一个更大的规模上做这个。我们将看到如何为一个分块器相 对一个语料库的准确性打分，再看看一些数据驱动方式搜索 NP块。我们整个的重点在于扩 展一个分块器的覆盖范围。

**读取IOB格式与CoNLL2000分块语料库**

使用corpora 模块，我们可以加载已标注的《华尔街日报》文本，然后使用 IOB符号 分块。这个语料库提供的块类型有NP，VP 和PP。正如我们已经看到的，每个句子使用多 行表示，如下所示：

he PRP B-NP

accepted VBD B-VP

the DT B-NP

position NN I-NP

...

转换函数chunk.conllstr2tree()用这些多行字符串建立一个树表示。此外它允许我们选择使用三个块类型的任何子集，这里只是NP 块：

>>> text = '''

... he PRP B-NP

... accepted VBD B-VP

... the DT B-NP

... position NN I-NP

... of IN B-PP

... vice NN B-NP

... chairman NN I-NP

... of IN B-PP

... Carlyle NNP B-NP

... Group NNP I-NP

... , , O

... a DT B-NP

... merchant NN I-NP

... banking NN I-NP

... concern NN I-NP

... . . O

... '''

>>> nltk.chunk.conllstr2tree(text, chunk\_types=['NP']).draw()

我们可以使用NLTK 的 corpus 模块访问较大量的已分块文本。CoNLL2000 分块语料 库包含27 万词的《华尔街日报文本》，分为“训练”和“测试”两部分，标注有词性标记和 IOB 格式分块标记。我们可以使用nltk.corpus.conll2000 访问这些数据。

正如你看到的，CoNLL2000 分块语料库包含三种块类型：NP块，我们已经看到了；V P块如has already delivered；PP块如because of。因为现在我们唯一感兴趣的是 NP块，我们可以使用chunk\_types 参数选择它们：

>>> print conll2000.chunked\_sents('train.txt', chunk\_types=['NP'])[99]

**简单评估基准**

现在，我们可以访问一个已分块语料，可以评估分块器。我们开始为琐碎的不创建任何 块的块分析器cp建立一个基准（baseline）：

>>> from nltk.corpus import conll2000

>>> cp = nltk.RegexpParser("")

>>> test\_sents = conll2000.chunked\_sents('test.txt', chunk\_types=['NP'])

>>> print cp.evaluate(test\_sents)

ChunkParse score:

IOB Accuracy: 43.4%

Precision: 0.0%

Recall: 0.0%

F-Measure: 0.0%

IOB 标记准确性表明超过三分之一的词被标注为O，即没有在 NP块中。然而，由于我 们的标注器没有找到任何块，其精度、召回率和F-度量均为零。现在让我们尝试一个初级 的正则表达式分块器，查找以名词短语标记的特征字（如CD、DT 和 JJ）开头的标记。

>>> grammar = r"NP: {<[CDJNP].\*>+}"

>>> cp = nltk.RegexpParser(grammar)

>>> print cp.evaluate(test\_sents)

ChunkParse score:

IOB Accuracy: 87.7%

Precision: 70.6%

Recall: 67.8%

F-Measure: 69.2%

正如你看到的，这种方法达到相当好的结果。但是，我们可以采用更多数据驱动的方法 改善它，在这里我们使用训练语料找到对每个词性标记最有可能的块标记（I、O或 B）。换句话说，我们可以使用unigram 标注器（5.4节）建立一个分块器。但不是尝试确定每个词 的正确的词性标记，而是给定每个词的词性标记，尝试确定正确的块标记。

**训练基于分类器的分块器**

无论是基于正则表达式的分块器还是n-gram分块器，决定创建什么块完全基于词性标记。然而，有时词性标记不足以确定一个句子应如何分块。例如：考虑下面的两个语句：

a. Joey/NN sold/VBD the/DT farmer/NN rice/NN ./.

b. Nick/NN broke/VBD my/DT computer/NN monitor/NN ./.

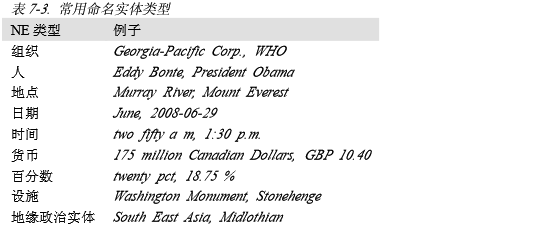
这两句话的词性标记相同，但分块方式不同。在第一句中，the farmer 和rice 都是单独 的块，而在第二个句子中相应的部分，the computer monitor，是一个单独的块。显然，如果我们想最大限度地提升分块的性能，我们需要使用词的内容信息作为词性标记的补充。

我们包含词的内容信息的方法之一是使用基于分类器的标注器对句子分块。如在上一节 使用的n-gram分块器，这个基于分类器的分块器分配IOB 标记给句子中的词，然后将这些标记转换为块。对于基于分类器的标注器本身，我们将使用与我们在 6.1节建立词性标注器相同的方法。

基于分类器的NP 分块器的基础代码包括两个类：第一个类几乎与例6-5 中ConsecutivePosTagger类相同。仅有的两个区别是它调用一个不同的特征提取器，使用MaxentClassifier而不是NaiveBayesClassifier；第二个类④基本上是标注 器类的一个包装器，将它变成一个分块器。训练期间，这第二个类映射训练语料中的块树到 标记序列；在parse()方法中，它将标注器提供的标记序列转换回一个块树。

## 命名体识别

在本章开头，我们简要介绍了命名实体（NEs）。命名实体是确切的名词短语，指示特 定类型的个体，如组织、人、日期等。表7-3 列出了一些较常用的NEs 类型。这些应该是 不言自明的，除了“FACILITY”：建筑和土木工程领域的人造产品；以及“GPE”：地缘政 治实体，如城市、州/省、国家。

命名实体识别（NER）系统的目标是识别所有文字提及的命名实体。可以分解成两个子 任务：确定NE 的边界和确定其类型。命名实体识别经常是信息提取中关系识别的前奏，它也有助于其他任务。例如：在问答系统（QA）中，我们试图提高信息检索的精确度，不是 返回整个页面而只是包含用户问题的答案的那些部分。大多数 QA 系统利用标准信息检索返 回的文件，然后尝试分离文档中包含答案的最小的文本片段。

我们如何识别命名实体呢？一个办法是查找一个适当的名称列表。例如：识别地点时， 我们可以使用地名辞典，如亚历山大地名辞典或盖蒂地名辞典。然而，盲目这样做会出问题， 如图7-5 所示。

图7-5 命名体识别示意图

通过在新闻故事中简单的查找：查找地名辞典中的每个词是容易出错的；案例区分可能有所帮助，但它们不是总会有的。

请看地名辞典很好的覆盖了很多国家的地点，却错误地认为 Sanchez 在多米尼加共和国而On在越南。当然，我们可以从地名辞典中忽略这些地名，但这样一来当它们出现在一个文档中时，我们将无法识别它们。

人或组织的名称的情况更加困难。任何这些名称的列表都肯定覆盖不全。每天都有新的组织出现，如果我们正在努力处理当代文本或博客条目，使用名称辞典查找来识别众多实体是不可能的。

困难的另一个原因是许多命名实体措辞有歧义。May和North可能分别是日期和地点类型的命名实体的，但也可以都是人名；相反的，Chris-tian Dior 看上去像是一个人名，但更可能是组织类型。词Yankee 在某些上下文中是普通的修饰语，但在短语Yankee infielders 中会被标注为组织类型的一个实体。

更大的挑战来自如 Stanford University 这样的多词名称和包含其他名称的名称，如 Cec il H. Green Library 和 Escondido Village Conference Service Center。因此，在命名实体识别中，我们需要能够识别多标识符序列的开头和结尾。

命名实体识别是一个非常适合用基于分类器类型的方法来处理的任务，这些方法我们在名词短语分块时看到过。特别是，我们可以建立一个标注器，为使用 IOB 格式的每个块都加了适当类型标签的句子中的每个词加标签。这里是CONLL 2002（conll2002）荷兰语训练数据的一部分：

Eddy N B-PER

Bonte N I-PER

is V O

woordvoerder N O

van Prep O

diezelfde Pron O

Hogeschool N B-ORG

. Punc O

在上面的表示中，每个标识符一行，与它的词性标记及命名实体标记一起。基于这个训练语料，我们可以构造一个可以用来标注新句子的标注器，使用 nltk.chunk.conlltags2t ree()函数将标记序列转换成一个块树。 NLTK 提供了一个已经训练好的可以识别命名实体的分类器，使用函数nltk.ne\_chun k()访问。如果我们设置参数binary=True，那么命名实体只被标注为NE；否则，分类器会添加类型标签，如PERSON, ORGANIZATION, and GPE。

>>> sent = nltk.corpus.treebank.tagged\_sents()[22]

>>> print nltk.ne\_chunk(sent, binary=True)

## 关系抽取

一旦文本中的命名实体已被识别，我们就可以提取它们之间存在的关系。如前所述，我们通常会寻找指定类型的命名实体之间的关系。进行这一任务的方法之一是首先寻找所有 (X, α, Y)形式的三元组，其中X 和 Y是指定类型的命名实体，α表示 X和 Y之间关系的 字符串。然后我们可以使用正则表达式从α的实体中抽出我们正在查找的关系。下面的例子 搜索包含词in 的字符串。特殊的正则表达式(?!\b.+ing\b)是一个否定预测先行断言，允许我们忽略如success in supervising the transition of 中的字符串，其中in后面跟一个动名词。

>>> IN = re.compile(r'.\*\bin\b(?!\b.+ing)')

>>> for doc in nltk.corpus.ieer.parsed\_docs('NYT\_19980315'):

... for rel in nltk.sem.extract\_rels('ORG', 'LOC', doc,

... corpus='ieer', pattern = IN):

... print nltk.sem.show\_raw\_rtuple(rel)

[ORG: 'WHYY'] 'in' [LOC: 'Philadelphia']

[ORG: 'McGlashan &AMP; Sarrail'] 'firm in' [LOC: 'San Mateo']

[ORG: 'Freedom Forum'] 'in' [LOC: 'Arlington']

[ORG: 'Brookings Institution'] ', the research group in' [LOC: 'Washington'] [ORG: 'Idealab'] ', a self-described business incubator based in' [LOC: 'Los Angeles']

[ORG: 'Open Text'] ', based in' [LOC: 'Waterloo'] [ORG: 'WGBH'] 'in' [LOC: 'Boston']

[ORG: 'Omnicom'] 'in' [LOC: 'New York'] [ORG: 'DDB Needham'] 'in' [LOC: 'New York']

[ORG: 'Kaplan Thaler Group'] 'in' [LOC: 'New York']

[ORG: 'BBDO South'] 'in' [LOC: 'Atlanta']

[ORG: 'Georgia-Pacific'] 'in' [LOC: 'Atlanta']

搜索关键字in执行的相当不错，虽然它的检索结果也会误报，例如： [ORG: House T ransportation Committee] , secured the most money in the [LOC: New York]；一种简单的 基于字符串的方法排除这样的填充字符串似乎不太可能。

如前文所示，CoNLL 2002 命名实体语料库的荷兰语部分不只包含命名实体标注，也包 含词性标注。这允许我们设计对这些标记敏感的模式，如下面的例子所示。show\_clause() 方法以分条形式输出关系，其中二元关系符号作为参数relsym 的值被指定。

## 小结

1. 信息提取系统搜索大量非结构化文本，寻找特定类型的实体和关系，并用它们来填充有组织的数据库。这些数据库就可以用来寻找特定问题的答案。

2. 信息提取系统的典型结构以断句开始，然后是分词和词性标注。接下来在产生的数据中搜索特定类型的实体。最后，信息提取系统着眼于文本中提到的相互临近的实体，并试图确定这些实体之间是否有指定的关系。

3. 实体识别通常采用分块器，它分割多标识符序列，并用适当的实体类型给它们加标签。常见的实体类型包括组织、人员、地点、日期、时间、货币、GPE（地缘政治实体）。

4. 用基于规则的系统可以构建分块器，例如：NLTK 中提供的RegexpParser类；或使用机器学习技术，如本章介绍的ConsecutiveNPChunker。在这两种情况中，词性标记往往是搜索块时的一个非常重要的特征。

5. 虽然分块器专门用来建立相对平坦的数据结构，其中没有任何两个块允许重叠，但它们可以被串联在一起，建立嵌套结构。

6. 关系抽取可以使用基于规则的系统，它通常查找文本中的连结实体和相关的词的特定模式；或使用机器学习系统，通常尝试从训练语料自动学习这种模式