# 第5章 NLP应用

在这一章中，我们要来具体讨论一下NLP应用。也就是说，我们接下来会将用到在之前章节中所学到的所有概念，看看用这些概念究竟能开发出何种应用程序。因此，这将会是一个完全需要动手实践的章节。在前面的章节中，我们已经学习了所有NLP应用都需要执行的大部分预处理步骤。我们了解了如何使用标识器、POS标签、NER以及如何进行文本解析。本章要提供的是一种思路，让您了解应该如何运用之前所学到的知识开发出一些复杂的NLP应用。

如今，我们的现实世界中已经存在着非常多的NLP应用程序，譬如Google Search、Siri、机器翻译，Google News、Jeopardy[[1]](#footnote-1)和拼写检查等都是一些大家最为耳熟能详的例子。这其中的一些技术层次是研究人员多年努力的成果，他们将这些技术应用到了当前的水平。 NLP太复杂了，正如我们在之前章节中所看到的那样，像POS和NER这样的预处理步骤大部分也还都是研究性的问题。但通过使用NLTK库，我们已经在恰当的精确度范围内解决了其中的许多问题。我们在这本书中不会涉及到机器翻译和语音识别这样较为复杂的应用。但您现在应该已经具备了足够多的背景知识，也是时候去了解该领域的一些基本应用了。即使是作为一个NLP爱好者，我们也应该对这些NLP应用有一个基本的了解。我们也建议读者可以去互联网上找一些NLP应用来看看，并试着去了解它们。

总而言之，在本章：

* 我们将为读者介绍几个常见的NLP应用。
* 我们将会利用到目前为止所学习的知识开发一个NLP应用（新闻摘要器）。
* 我们还会介绍不同NLP应用的侧重点，以及它们各自的基本细节。

## 构建第一个NLP应用

让我们先来看一种非常复杂的NLP应用：**信息摘要（summarization）**。该应用的概念非常简单：对于我们所提供的文章/短文/故事，您通常会需要针对其内容自动生成一些摘要。事实上，信息摘要这个应用需要我们具备一些深层次的NLP知识，因为这里需要了解的不单是句子的结构，而是整个文本的结构，除此之外，我们还得要了解该文本的体裁和主题内容。

鉴于这一切看上去都太过于复杂，所以还是先来尝试一种很直观的方法吧。我们姑且假设这里所要做的信息摘要只不过就是要针对句子的重要性和意义进行一次排名。为此，我们要在理解句子的基础上创建一系列规则，然后用我们到目前为止所学到的处理工具来对新闻文章进行一些可接受的信息摘要处理。

在下面的例子中，我们会将从*纽约时报*上搜刮来的一篇文章保存在nyt.txt这个文本文件中。在这里，我们要对这篇新闻稿进行信息摘要。下面就让我们来创建一个个人版的Google News吧。

一开始，我们首先需要记住一件事：在通常情况下，拥有较多实体和名词的句子的重要性往往会相对比较高。现在，我们的任务是要用某种可被标准化的统一逻辑来计算**重要性评分（importance score）**。即如果我们想获取前n个句子的信息情况，就要去选择一个与其重要性评分的阈值。

现在我们来看看新闻稿的内容。在这里，您也可以选择将我这篇新闻稿以纯新闻内容的形式转储到一个文本文件中。这段内容具体如下：

>>> import sys

>>> f=open('nyt.txt','r')

>>> news\_content=f.read()

""" President Obama on Monday will ban the federal provision of some types of military-style equipment to local police departments and sharply restrict the availability of others, administration officials said.

The ban is part of Mr. Obama's push to ease tensions between law enforcement and minority communities in reaction to the crises in Baltimore; Ferguson, Mo.; and other cities. - - blic." It contains dozens of recommendations for agencies throughout the country."""

一旦我们要对这段新闻内容的解析，我们就会整个新闻稿分解成一个句子列表。这样，我们就回到了之前讨论过的句子标识器上了，后者会将整个新闻片段分解成若干个句子。在这里，我们会提供一些句型编号，便于我们识别这些句子并对其进行排名。一旦我们得到了这些句子，我们会让其在单词标识器中过一遍，最后再来过NER标注器和POS标注器。

>>> import nltk

>>> results=[]

>>> for sent\_no,sentence in enumerate(nltk.sent\_tokenize(news\_content)):

>>> no\_of\_tokens=len(nltk.word\_tokenize(sentence))

>>> #print no\_of\_toekns

>>> # Let's do POS tagging

>>> tagged=nltk.pos\_tag(nltk.word\_tokenize(sentence))

>>> # Count the no of Nouns in the sentence

>>> no\_of\_nouns=len([word for word,pos in tagged if pos in ["NN","NNP"] ])

>>> #Use NER to tag the named entities.

>>> ners=nltk.ne\_chunk(nltk.pos\_tag(nltk.word\_tokenize(sentence)), binary=False)

>>> no\_of\_ners= len([chunk for chunk in ners if hasattr(chunk, 'node')])

>>> score=(no\_of\_ners+no\_of\_nouns)/float(no\_of\_toekns)

>>>

>>> results.append((sent\_no,no\_of\_tokens,no\_of\_ners,\ no\_of\_nouns,score,sentence))

在上面的代码中，我们对一个句子列表进行了迭代，并根据公式计算出了这些句子的评分。当然，该公式只是个以被标识实体为分子，以普通标识词为分母的分子式。我们会将所有的这些结果创建成一个元组。

现在，结果就是一个包含了所有评分的元组，例如其中的名词数量、实体数量等。下面我们要对评分来一个降序排序，代码如下：

>>> for sent in sorted(results,key=lambda x: x[4],reverse=True):

>>> print sent[5]

这样一来，我们就等于完成了对这些句子的排名。您会为这篇新闻稿能得到这样的结果而感到惊讶。

一旦我们手里有了no\_of\_nouns和no\_of\_ners的评分列表，就可以围绕着它们去建立一些更复杂的规则。 例如，一篇典型的新闻稿通常都会在文章的开头来一个主题说明，并在文章的最后一句也会对整个故事做一个总结。

我们可以修改之前那段代码，将上述逻辑整合进去吗？

当然，这种信息摘要应用中还包含着另一种理论逻辑：重要的句子中通常包含着重要的词汇，而跨语料库的差异词（discriminatory word）绝大多数都是重要的词汇。因此，只要句子中包含具有很大差异性的词汇，它就是重要的。这样，我们得到了 一个非常简单的测量方法，就是计算每个词各自的**TF-IDF（term frequency–inverse document frequency）**[[2]](#footnote-2)分值，然后根据词汇的重要性找出一种标准化的平均评分。这个评分就可以用来充当我们在信息摘要中选取句子的标准。

为了解释清楚概念，我们这里不会拿整篇文章来举例，这里将只采用文章的前三个句子。下面，我们就来看看如何用寥寥几行代码实现这个复杂的东西：

|  |
| --- |
| 这段代码需要您安装一下scikit这个库。如果您已经安装了anaconda或canopy，那么其实就已经安装了这个库，否则请按照下面链接中的指示安装scikit。  <http://scikit-learn.org/stable/install.html> |

>>> import nltk

>>> from sklearn.feature\_extraction.text import TfidfVectorizer

>>> results=[]

>>> news\_content="Mr. Obama planned to promote the effort on Monday during a visit to Camden, N.J. The ban is part of Mr. Obama's push to ease tensions between law enforcement and minority \communities in reaction to the crises in Baltimore; Ferguson, Mo. We are, without a doubt, sitting at a defining moment in American policing, Ronald L. Davis, the director of the Office of Community Oriented Policing Services at the Department of Justice, told reporters in a conference call organized by the White House"

>>> sentences=nltk.sent\_tokenize(news\_content)

>>> vectorizer = TfidfVectorizer(norm='l2', min\_df=0, use\_idf=True, smooth\_ idf=False, sublinear\_tf=True)

>>> sklearn\_binary=vectorizer.fit\_transform(sentences)

>>> print countvectorizer.get\_feature\_names()

>>> print sklearn\_binary.toarray()

>>> for i in sklearn\_binary.toarray():

>>> results.append(i.sum()/float(len(i.nonzero()[0]))

在上述代码中，我用到了一些未知的方法，譬如说TfidfVectorizer()，这是一个评分方法，它会为给定句子列表中每个句子计算出一个TF-IDF评分的向量。现在先别操心这个，我们以后会更详细地来讨论它。本章先暂且将其视为一个黑盒子函数，即对于一个给定的句子/文档列表，它将会给出每个句子所对应的评分，并且还会提供能构建出一个term-doc的矩阵，以作为我们的输出。

现在，我们从所有的句子中得到了一个容纳所有单词的字典以及一个评分列表的列表，后者的每一个元素都代表了一个单词所被赋予的TF-IDF评分。如果我们得到的是正确的结果，应该就可以看到一些停用词的评分值是接近0的，而一些差异词（譬如ban和obama）则通常会得到一个很高的评分。一旦我们在代码中获得了这些数据，就可以通过那些TF-IDF值非零的单词来得出TF-IDF的平均分值。最后我们就会得到一个与第一个方法类似的评分结果。

您一定会为这个简单的算法能得到这样的结果而感到惊讶。现在，我想我们应该已经做好了所有的准备，可以去编写属于自己的新闻摘要器了。该摘要器会利用上述两种算法对任意两篇给定的新闻搞进行信息摘要处理，并获取不错的摘要结果。虽然这种方法可以实现一个相对还不错的信息摘要程序，但它与信息摘要方面的当前研究水平相比，实际上还差得很远。我们建议读者们多去找一些信息摘要方面的文献来阅读。同时，我们也希望读者能试着混合使用这两种信息摘要的方法。

## 其它NLP应用

我们另外还有一些别的NLP应用，其中包括了文本分类、机器翻译、语音识别、信息检索、信息提取、主题划分和话语分析。这其中有一些问题其实到目前位置都还是一个非常难以实现的NLP任务，相关的领域仍在进行着大量的研究。我们将会在下一章中深入地讨论其中的一些话题，但既然是在学习NLP，我们就应该先要对这些应用的基本情况有一个了解。

### 机器翻译

对机器翻译（machine translation）最简单直接的理解方法就看我们自己是如何将某种语言翻译成另一种语言的。我们会在头脑中对相关句子的结构进行解析，以便试着理解该句子。 一旦理解了句子的含义，我们就会试着将原始语言中的单词替换成目标语言的对应词汇。并且在替换过程中，遵守目标语言的语法规则，以便最终实现正确的翻译。

（图：图中翻译

Interlingua：中间语言

Semantic composition：语义组合

Semantic decomposition：语义分解

Semantic structure：语义结构

Semantic transfer：语义转换

Semantic analysis：语义分析

Semantic generation：语义生成

Syntactic structure：句法结构

Syntactic analysis：句法分析

Syntactic transfer：句法转换

Syntactic generation：句法生成

Word structure：单词结构

Direct：直接替换

Morphological analysis：词法分析

Morphological generation：词法生成

Source text：原始文本

Target text：目标文本

）

大致上而言，机器翻译就是上图中那样一个金字塔状的过程。如果以原始语言的文本为出发点，我们就必须先要目标句子进行标识化处理，后者会被解析成树状结构（简而言之就是它的语法结构），以确保这些句子的正确表达。紧接着就是语义结构，这代表的这些句子所表达的含义，然后，我们来到了中间语言这个层面，该中间语言是一种独立于所有语言之外的抽象状态。到目前为止，人们已经开发出了许多种翻译方法，这些方法的走向越接近上述金字塔的塔顶，就越需要用到NLP技术。因此，根据各种不同的翻译层次，我们可以使用各种不同的方法来应对。下面我们就来看看其中的两种方法：

* **直接翻译：**这更像是一种基于字典的机器翻译，当我们拥有大型语料库以及海量的目标语言词汇时，依赖于相关语言的大型语料库来实现某种类型的翻译应用是有可能的。而且因它简单而流行。
* **语法翻译：**在这种翻译方法中，我们会试着去构建一个针对原始语言的解析器。到目前为止，人们已经累积了各种各样的关于解析问题的方法。其中有些深层解析器实际上已经拥有了处理一部分语义的能力。一旦我们搞定了解析器，目标词汇的替换问题就迎刃而解了，目标解析器会自行产生出最终的目标语言的句子。

### 统计型机器翻译

统计型机器翻译（**Statistical machine translation** ，简称SMT）是一种最新型的机器翻译方法。在这种方法中，人们提出了各种运用统计学方法，几乎涵盖了机器翻译方面的所有面向。 这一类算法背后的思路是依靠我们所拥有的海量语料库、并行化文本以及可用目标语言产生语言翻译的语言模型。Google Translate就是一个很好的SMT应用实例，它会从不同语言的语料库中学习到相关信息，并围绕这些信息来创建SMT应用。

### 信息检索

**信息检索（Information retrieval ，简称 IR）**也是最受欢迎且被广泛使用的NLP应用之一。这类应用最好的实例就是Google Search，它会根据既定的用户输入，利用信息检索算法检索出与用户查询相关的信息。

简而言之，IR就是一个根据用户需求来获取最具相关性的信息的过程。在这里系统可以用多种不同的方式来查找信息，当它最终必须要能检索出最具相关性的信息。

典型的IR系统通常的做法是要产生一种索引机制，我们称之为**反向索引（inverted index）**。这类机制与书籍中所使用的索引方案非常相似，我们通常会在一本书的最后一页上找到整本书中出现过的单词的一份索引。IR系统也会创建一份类似的反向索引的poslist下面我们就来看一份典型的poslist：

< Term , DocFreq, [DocId1,DocId2] >

{"the",2 --->[1,2] }

{"US",1 --->[2] }

{"president",2 --->[1,2] }

如果有任何单词同时出现在文档1和文档2中，出处列表中就会出现有一份指向该单词的文档列表。一旦我们搞定了数据结构的类型，就可以将其应用到不同的检索模型中。不同的检索模型操作的是不同数据类型。下面我们就来介绍其中的几个。

#### 布尔检索

在布尔模型中，我们只需要在poslist上执行某种布尔操作即可。 例如，如果我们正在执行类似于“US president”这样的搜索查询，系统就应该去查询“US”和“president”这两个单词poslist的交集。

{US}{president}=> [2]

在这里，第二个文档被证明是具有相关的文档。

#### 向量空间模型

**向量空间模型（vector space model** ，简称**VSM）**的概念来自几何学。它会以可视化的方式将文档以一个向量的形式出现在一个高维度的词汇空间中。 因此，每个文档在该空间中都可以各自用一个向量来表示。虽然向量的表示方式可以各不相同，但TF-IDF无疑是其中最为实用而有效的方式之一。

对于给定的单词和语料库，它们的**单词频率（TF）**和**逆文档频率（IDF）**的计算公式如下：

（公式）

在这里，TF指的只是单词在文档中的出现频率。而IDF则指的是文档频率的反比值，也就是该语料库中出现该单词的文档数的累计值：

（公式）

虽然这些公式的标准变化可以各种各样，但我们可以将它们合二为一，创建一个更可靠的评分机制来为文档中的各个单词评分。然后想要获得TF-IDF评分，我们还需要将这两个评分相乘：

（公式）

在TF-IDF中，我们所评分的是一个单词在当前文档中出现的次数和它在语料库间的散播程度。这让我们认识到一个事实，即跨语料库且出现频率高的单词并不常见。因此我们可以区别检索这些文档。我们在上一节中也使用了TF-IDF，用来描述我们的信息摘要器。相同评分的文档可用一个向量来表示。 一旦我们将所有文档都表示成了某种向量形式，接下来就可以制定向量空间模型了。

在VSM中，用户的搜索查询也被当作是一种文档并表示成一个向量。直观地看，通过计算这两种向量间的点积可以获得目标文档与用户查询之间的余弦相似度。

（图：图中文字无须翻译）

在上图中，我们看到这些相同的文档可以用各单词轴线来表示，查询Obama与*D1*的相关性要高于*D2*。因此该查询的文档相关性评分可表示如下：

（公式）

#### 概率模型

概率模型会试着去评估相关文档被用户所需要的概率。该模型会假设这个相关概率取决于用户查询和文档表示方式。这里的主要思路是某文档出现在相关性集合中，但不存在于非相关性集合中。下面，我们用dj来表示文档，q表示用户的查询，R表示文档的相关性集合，P表示非相关性集合。那么这里的评分计算应该如下：

（公式）

|  |
| --- |
| 如果想了解更多关于IR的话题，我推荐您可以阅读一下下面链接中的内容：  <http://nlp.stanford.edu/IR-book/html/htmledition/irbook.html> |

### 语音识别

语音识别是一个非常古老的NLP问题。人们自第一次世界大战时代以来就一直在尝试着解决这个问题，目前它还仍然是计算领域最热门的话题之一。这个问题的思路其实很直观。即我们需要将一段给定的某个人的语音转换成文本、与语音相关的问题都需要生成一个声音序列，我们称之为**音素（phonemes）**，这是非常难以处理的，因此语音分割本身就是一个大问题。只要语音是可处理的，那么下一步就可以通过一些可用的训练数据来形成某种约束（模型）了，这里就涉及重度的机器学习应用了。如果我们用图示法将这个应用约束表示成某种方框，就会发现这个方框应该会是整个系统中最复杂的组件之一。其中，它的声学建模涉及到的是基于音素的建模，而词汇模型所要解决的模型问题是基于小型的句子分段来进行的，它要将各个分段的含义关联起来。单独的语言建模都是基于单元词法和二元词法来进行的。

一旦我们构建好了这些模型，我们就会用它来处理相关的语句表达。当这些初始化处理过程完成之后，我们就会对这些语句进行那些声学的、词汇的以及语言建模上的处理，产生相关的标记并输出。

（图：图中翻译

Training data：训练数据

Applying constraints：应用约束

Acoustic models：声学模型

Lexical models：词汇模型

Language models：语言模型

Speech signal：语音符号

Representation：表示方式

Search：搜索

Recognized words：被识别单词

）

### 文本分类

文本分类是NLP问题中非常有趣且常见的应用。在日常工作中，我们会需要与许多文本分类器进行交互。例如我们所使用的垃圾邮件过滤器，优先收件箱，新闻聚合器等，所有的这些都是用文本分类技术所构建的应用。

文本分类是一个定义明确且已经部分得到解决的问题，已被用于多个领域。一般情况下，所谓文本分类本质上就是一个利用单词或者词组对文本文档进行分类的过程。虽然这是一个典型的机器学习问题，但文本分类中所用到的许多预处理步骤都来自NLP问题。

下面，我们就来看看文本分类应用的抽象图：

（图：图中翻译

Tokenization：标识化处理

Stop-word：停用词

Stemming：词干提取

Pos\_tag：词性标注

Preprocessing：预处理过程

Text classifier：文本分类器

Pos/neg class：pos/neg类

）

如图所示，我们手里现在有一堆文档组成了一个类集合。在这里我们为了让事情简单一点，只讨论了用二进制来进行1/0分类的情况。即我们假设这是在解决一个垃圾邮件检测的问题，用1来表示垃圾邮件，而0则表示不被视为垃圾邮件的正常文本。

在这个处理过程中，我们会涉及到一些之前章节中所介绍的预处理步骤。其中有些步骤的必要性取决于我们要解决的是何种类型的文本分类问题。因此在某些少见的情况下，我们会需要根据某特征工程的情况来放弃部分的预处理步骤。该特性工程的最终目标是要生成一个**词汇文档矩阵（Term doc matrix，简称TDM）**，其中保存的是整个语料库的词汇表：它的列和行都是文档，该矩阵所表现的是一种评分机制，用的是**词袋（Bag of word，简称BOW）**表示法。我们可以通过加权方案将其转变为TF、TF-IDF。Bernoulli等其它词频表示法。

除此之外，我们还可以通过譬如给定特征的POS、上下文语境的POS等其它方式来进行特征导入。以获取比NLP本身更大的特征空间。一旦我们生成了相关的TDM，文本分类问题就变成了典型的监督型/非监督性分类问题：即对于给定的一组样本，我们要预测它们各自的分类。我们会下一章中专门讨论这个话题。这绝对是NLP / ML领域中的一个辉煌应用，并且往往都是商业级的应用。

其中有一些还是我们日常生活在最常见的应用，例如情绪分析、垃圾邮件分类、电子邮件分类、新闻分类、专利分类等。我们将会在下一章中更具体地讨论文本分类的问题。

### 信息提取

**信息提取（IE）**是一个从非结构化文本中提取有意义的信息的过程。同时，IE也是一种被广泛使用且非常重要的应用。通常情况下，信息提取的引擎会利用海量的非结构化文档来生成某种结构化/半结构化的**知识库（knowledge base，简称KB）**，然后在围绕着该知识库来部署、构建相关的应用。举个简单的例子，我们可以用一组海量的非结构化的文本文档来生成一个非常不错的知识本体（ontology）。Dbpedia就是这样的一个项目，Wikipedia中所有的文章都被用作了产生知识本体的构件，这些构件之间都有着一些相互联系或者其它形式的关系。

执行信息提取主要有两种方式：

* **基于规则的提取**：这种方法中会用一种模板填充机制。它的思路是我们要为自己所预期的结果去寻找一些被预定义好的用例，并试着从该特定的非结构化文本中挖掘出特定的模板。例如如果要构建足球知识库，我们要获取所有球员的信息，其中会涉及他们的简介、统计数据以及部分个人信息等。所有的这些信息都可以被先定义好，然后用基于模式的规则或POS标签，NER和关系提取法来提取。
* **基于机器学习的提取**：这也是一种方法，一种更深度的基于NLP的方法，譬如我们可以专门针对自己的知识库需求来构建一个解析器​​。某些知识库将会需要挖掘一些不能用预训练的NER来提取的实体，因此我们就有必要构建一个自定义的NER。在这种情况下，我们可能会想要去开发一个专门针对待构建知识库的关系提取算法。这会是一种更密集使用NLP技术的方法，因为这里要开发的是一个基于NLP的解析器或标记器，属于重度的机器学习应用。

### 问答系统

**问答（Question answering，简称QA）**系统是一种基于自身的知识库来解决相关问题的智能系统。这方面最主要的一个例子就是IBM Watson，因为它参加了电视节目Jeopardy，并且在比赛中赢了人类对手。QA系统可被分解成若干个构成组件，其中包括用于查询知识库的语音识别，以及用于生成知识库的信息检索和信息提取。

一旦我们对系统提出了一个问题，对其进行各种不同的分别别类就是它面临的一大问题。另一方面是它要对知识库进行有效的搜索并能检索出最确切的文档。甚至在这之后，我们还必须通过其它的一些应用（例如信息摘要和解析）以自然的方式产生出答案。

### 对话系统

对话系统一直被当作是一种梦幻般的应用，当该系统收到某种既定源语言的语音时，它会自动执行语音识别，将其转换成文本。然后，该文本会被传递给某种机器翻译系统，该系统可以将该语音的文本翻译成目标语言。接下来，我们会用某种文本转语音系统将其结果再转换成目标语言的语音。这就是我们最为理想化的NLP应用之一，因为只要有了它，我们可以用任何语言与计算机进行通信，同时计算机也将用相同的语言来进行回复。这样一来，这种应用就等于破除了世界上所有现存的语言障碍。

其中，Apple Siri和Google Voice是对话系统中商业应用的典型例子，它们的智能程度都已经足以了解我们的信息需求，并且都能试着用一组动作和信息来解决这些需求，作出与人类相似的反应。

### 词义消歧

**词义消歧（Word sense disambiguation，简称WSD）**也是一个人们研究多年但仍未得到解决的困难挑战，同时它也是问答系统、信息摘要和搜索等应用所面临的主要难点之一。理解这个概念有一个简单的方法：即就是当我们遇到不同的上下文语境时，有许多单词的具体含义是不一样的。 例如下列例句中都有“cold”这个单词：

* The ice-cream is really cold
* That was cold blooded!

但这两个“cold”的含义并不相同，这种区别的概念对于计算机来说确实是很难理解的。词性标注和NER这些NLP预处理选项也只能解决其中的部分问题。

### 主题建模

在大规模非结构化文本内容的处理条件下，主题建模算是一个非常了不起的应用了。它的主要任务是识别出语料库中新出现的主题，然后根据这些主题将其文档分类存放到语料库中。我们会在下一章中对这个问题进行一些简单的讨论。

主题建模应用使用了相同的NLP预处理，例如句子分割、标识化处理和词干提取等。 该算法的优点是我们有了一种无人监管的文档分类法;。另外，主题生成的过程并没有明显涉及到其它东西。 我们鼓励读者更进一步了解主题建模，推荐您去阅读一下与潜在狄氏分配（**latent dirichlet allocation**，简称LDA）和**潜在语义索引（latent semantics indexing，简称LSI）**相关的详细信息。

### 语言检测

对于一段给定的文本来说，对其进行语言检测其实也是一个问题。并且，语言检测的应用对于其它一些NLP应用（譬如搜索、机器翻译、语音等）也非常重要。其主要的想法是要从文本中学习到相关语言的特性。 在这个特性工程中，我们会用到各种与机器学习以及NLP相关的技术。

### 光符识别

**光符识别（Optical character recognition，简称OCR）**是一种NLP与计算机视觉技术相结合的应用，它会对对于给定的手写文档/非数字文档进行文本识别，并将其提取为数字格式。 这项应用在机器学习领域中也已经被广泛研究了许多年。Google Book算是其中一个比较大型OCR项目，他们使用OCR技术将非数字图书转换到了一个集中式的图书馆中。

## 本章小结

总而言之，我们周围存在许多NLP应用，它们充斥着我们的日常交互。NLP是有一定难度和复杂度的，其中有些问题至今为止也没得到解决，或者没有完美的解决方案。所以每一个探寻NLP问题的人都在试着搜索这方面的文献。现在正是成为一名NLP研究者的大好时机。因为在大数据时代，NLP应用将非常受欢迎。许多研究实验室和组织目前都正在致力于开发像语音识别、搜索和文本分类这一类的NLP应用。

相信我们到本章位置已经学到了许多基础知识。在接下来的几章中，我们将会深入探讨本章所介绍的一些应用。也就是说，我们目前已经来到了一个学习节点上，我们已经充分掌握了那些与NLP相关的预处理工具，并且对一些最流行的NLP应用也已经有了一个基本的了解。下面，我们会希望能利用学到的知识来构建某种版本的NLP应用。

在下一章中，我们将会开始介绍一些重要的NLP应用，譬如文本分类、文本聚类和主题建模。而且，我们还会稍稍离开一下纯NLTK应用程序，去了解一下NLTK与其他库的配合使用方法。

1. 译者注：这是一款文字类问答游戏，这些问答题非常考验玩家的英文水平、以及各个领域的知识。玩家要有能力解析题目中的隐晦含义，反讽或者谜题。这也是目前计算机最欠缺的能力。 [↑](#footnote-ref-1)
2. 译者注：TF-IDF是一种统计方法，主要用来评估某一单词在一个文件集或某个语料库中某一份文件的重要程度。单词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。 [↑](#footnote-ref-2)