第3章

实体与关系挖掘

★本章导读★

本章是关于知识图谱中实体和关系的挖掘与抽取的内容，分别介绍领域短语挖掘、同义词挖掘、缩略词挖掘、实体识别。

3.1节介绍领域短语挖掘的定义、与LDA的区别、与关键词抽取的区别，以及领域短语挖掘的无监督方法和挖掘结果的统计指标特征：词频-逆文档频率（Term Frequency-Inverse Document Frequency，TF-IDF）、点互信息（Pointwise Mutual Information，PMI）、子短语频次减去负短语频次（C-value）、左邻字熵与右邻字熵（NC-value）等。3.2节介绍同义词的定义以及同义词挖掘的方法，包括基于同义词资源的方法、基于模式匹配的方法和自举法。3.3节介绍缩略词挖掘的内容，包括缩略词的概念和形式、缩略词的检测与提取方法以及缩略词的预测等内容。通过对缩略词的挖掘可以更好地理解文本中的含义，并且可以提高实体识别和关系抽取的精度。3.4节主要介绍实体识别的内容，包括任务概述、传统的实体识别方法和基于深度学习的实体识别方法。实体识别是知识图谱构建的关键技术之一，它可以识别文本中的实体并将其映射到知识图谱的实体节点上。传统的实体识别方法主要是基于规则和模板的方法，而深度学习的实体识别方法则是利用神经网络模型进行实体识别，其准确率和效率均有很大的提高。

本章内容是知识图谱实现中不可或缺的一部分，对于构建高质量的知识图谱具有重要的作用。掌握本章的内容，可以帮助读者更好地理解知识图谱的实现过程，并具备实际操作的能力。

★知识要点★

（1）知识图谱中的信息提取技术包括领域短语挖掘、同义词挖掘、缩略词挖掘、实体识别以及基于规则模型的实体抽取与关系挖掘构建知识图谱的方法。

（2）领域短语挖掘的统计指标特征包括TF-IDF、PMI、C-value、NC-value等。

（3）同义词挖掘包括基于同义词资源、基于模式匹配和自举法的方法。

（4）缩略词挖掘包括检测与提取方法以及预测方法。

（5）实体识别方法包括传统和深度学习方法。

3.1 领域短语挖掘

3.1.1 问题描述

人类学习一个领域的知识一般是从该领域的词汇和术语开始。例如，对于知识图谱领域的学习，就要从“关系抽取”“词汇挖掘”“实体识别”等领域词汇的理解开始。短语挖掘一般应用于构建领域知识图谱，用于发现领域相关的短语，进而找到其中领域相关的实体。

领域短语挖掘是指从给定的领域语料（将大量的文档融合在一起组成一个语料）中自动挖掘该领域的高质量短语的过程。领域短语挖掘的输入是领域语料，输出是该领域中的高质量短语（high quality phrase）。

在给定的文档中，一个高质量短语是指连续出现的单词序列：$w_1\ w_2\ w_3 \cdots\ w_n$，其本质上是一个 n-gram，其中 n 是指短语的长度。对于中文短语挖掘，可以是词（比如，可以认为是由“四川”与“大学”构成的词序列），也可以是字符（比如，“四川大学”可以认为是由 4 个字符构成的字符序列）。一个高质量的短语通常独立描述了一个完整、不可分割的语义单元。

对于短语的质量，一般从以下几个角度来评估。

（1）频率。一般来说，一个 n-gram 在给定的文档集合中要出现得足够频繁才能被视作高质量短语。

（2）一致性。一致性是指 n-gram 的搭配频率明显高于其各部分偶然组合在一起的可能性，即反映的是 n-gram 中不同单词的搭配是否合理或者是否常见。

（3）信息量。一般来说，一个高质量短语应该传达一定的信息，即表达一定的主题或者概念。

（4）完整性。一个高质量短语在特定的上下文中还必须是一个完整的语义单元。

领域短语挖掘和隐含狄利克雷分布（Latent Dirichlet Allocation，LDA）主题模型的区别在于，LDA 主题模型的输入是若干篇文档，输出是每篇文档的主题分布和每个主题的词分布，根据这两个分布可以得到每篇文档中不同词的分数。领域短语挖掘的输入不区分多篇文档，而是直接将它们合并为一个大文档，输出是该领域的高质量短语。LDA 关注的是主题下的字（词）分布，并不关心如何得到短语。

领域短语挖掘和关键词抽取的区别在于：关键词抽取是从语料中提取最重要、最有代表性的短语，抽取的短语数量一般比较少。例如，写论文的时候在摘要（abstract）下面一般会附上 5 或 6 个关键词（keyword）。

领域短语挖掘和新词发现的区别在于：新词发现的主要目标是发现词汇库中不存在的新词汇，而领域短语挖掘不区别新短语和词汇库中已有的短语。新词发现可以通过在领域短语挖掘的基础上进一步过滤已有词汇来实现。

早期的短语挖掘主要基于规则来挖掘名词性短语。最直接的方法是通过预定义的词性标签（POS tag）规则来识别文档中的高质量名词短语。为了避免人工定义规则的高昂代价，可利用标注好词性的语料来自动学习规则，如使用马尔可夫模型来完成这一任务。但是，词性标注不能做到百分百准确，这会在一定程度上影响后续规则学习的准确率。

近年来，利用短语的统计指标特征来挖掘词汇成为主流方法之一。基于统计指标的领

域短语挖掘方法可以分为无监督学习和监督学习两大类方法。无监督学习适用于缺乏标注数据的场景，监督学习适用于有标注数据的场景。

3.1.2 领域短语挖掘方法

无监督方法主要通过计算候选短语的统计指标特征来挖掘领域短语，主要流程如图 3-1 所示。

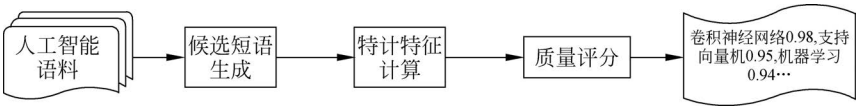


图 3-1 无监督方法挖掘领域短语主要流程

无监督方法主要包括以下几步。

(1) 候选短语生成。这里的候选短语就是高频的 n-gram(连续的 n 个字/词序列)。首先设定 n-gram 出现的最低阈值(阈值和语料的大小成正比，语料越大，阈值越大。对于较大的语料，阈值一般取 30)，通过频繁模式挖掘得到出现次数大于或等于阈值的 n-gram 作为候选短语。

(2) 统计特征计算。根据语料计算候选短语的统计指标特征，如 TF-IDF、PMI、左邻字熵与右邻字熵等。

(3) 质量评分。将这些特征的值融合(如加权求和等)得到候选短语的最终分数，用该分数来评估短语的质量。

(4) 排序输出。对所有候选短语按照分数由高到低排序，通常取前 K 个短语或者取根据阈值筛选出的短语作为输出。

基于监督学习的领域短语挖掘在无监督方法的基础上增加了两个步骤：样本标注和分类器学习。前者负责构造训练样本，后者根据样本训练一个二元分类器以预测候选短语是否是高质量短语。基于监督学习的领域短语挖掘主要流程如图 3-2 所示。

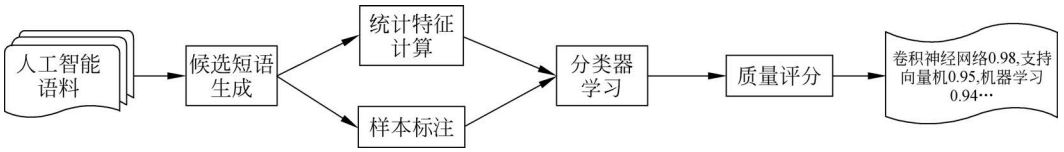


图 3-2 基于监督学习的领域短语挖掘主要流程

样本标注的具体实现可以是人工标注或者远程监督标注两种常见形式。人工标注指由人手工标注候选短语是否是高质量的。远程监督标注一般用在线知识库(如百度百科、维基百科等)作为高质量短语的来源，如果候选短语是在线知识库的一个词条，则其被视作高质量短语，否则被视作负例样本。

分类器学习是根据正负例样本，学习一个二元分类器。分类器模型可以是决策树、随机森林或者支持向量机。对于每个样本，使用 TF-IDF、C-value、NC-value 以及 PMI 等构造相应的特征向量。

上述方法根据原始词频的相关统计特征(如词频、PMI 和左/右邻字熵等)来判定候选短语的质量，因此词频统计的准确性会对最终的打分产生显著影响。直接的统计方法会从

文本中枚举所有的 n-gram 并统计其相应的出现次数作为词频，这就导致了子短语的词频一定大于父短语，因此基于原始词频的质量估计有偏差，不足以采信。导致这一估计偏差的根本原因在于：一旦认定某个父短语（如“支持向量机”）是高质量短语，那么它的出现就不应该重复累积到其任何子短语上。

因此，基于 n-gram 的原始频次统计方法需要修正与优化。考虑到在构建了高质量候选短语的判定模型之后，可以尝试利用模型来识别高质量短语，再根据已经发现的高质量短语对语料进行切割，在切割的基础上重新统计词频，改进词频统计的精度。

基于监督学习的领域短语挖掘方法经过优化后，采取迭代式计算框架，在迭代的每一轮先后进行语料切割和统计指标更新。由于切割可以提升频次统计的精度，基于相应统计特征构建的高质量短语识别模型也就更加精准，从而能更好地识别高质量短语。而高质量短语的精准识别又可以进一步更好地指导语料切割。语料切割与高质量短语识别两者之间相互增强。经过多次迭代，直至候选短语得分收敛。最终，依据每个候选短语的最后得分识别语料中的高质量短语。

迭代式短语挖掘过程，如图 3-3 所示。

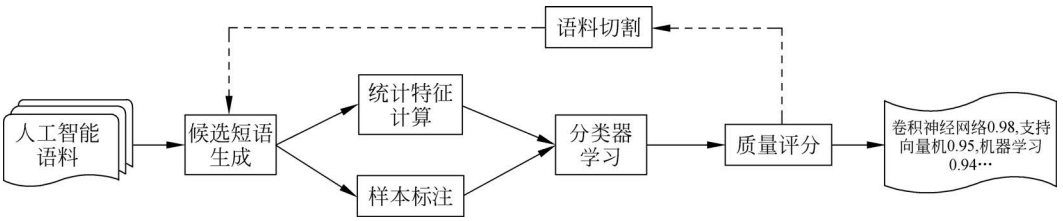


图 3-3　迭代式短语挖掘过程

3.1.3　统计指标特征

1. TF-IDF

TF-IDF 一般用来评价一个短语在语料中的重要性。考虑到通用词汇通常在外部文档中也以很高的频率出现，但是领域词汇在外部文档中出现的频率则要低很多，因此通常引入逆文档频率来识别领域特有的高质量短语。如果某个短语在领域语料中频繁出现但在外部文档中很少出现，则该短语很可能是该领域的高质量短语。

TF-IDF 形式化地表示为 TF 乘以 IDF。对于某个词汇 u，其 TF 值定义为语料中该词汇出现的频次 $f(u)$ 除以该语料中所有词汇的累计词频：

$$\mathrm{TF}(u)=\frac{f(u)}{\sum_{u'} f(u')}$$

IDF 定义为外部文档总数除以包含该词汇的外部文档数（通常使用比值的对数形式）：

$$\mathrm{IDF}(u)=\log\frac{|D|+\delta}{|\{j:u\in d_j\}|+\delta}$$

其中，d_j 是第 j 篇外部文档，$|D|$ 是外部文档的总数。

一个词的重要程度与其在该语料中出现的频次（TF）呈正向关系，与其在外部文档中出现的频次（DF）呈反向关系（也就是与 IDF 呈正向关系）。

2. C-value

C-value 在词频基础上还考虑了短语的长度，以及父子短语对于词频统计的影响。C-value 首先考虑候选短语长度对其质量的影响。一般而言，在很多专业领域（比如医学领域），越长的短语越有可能是专有名词，从而极可能是高质量短语。其次，C-value 考虑了统计候选短语频率时父短语的重复统计对于短语频次估计所带来的偏差。在统计一个短语的频次时，需要减去该短语所有父短语的频次，其具体定义为

$$\text{C-value}(u)=\begin{cases}\log_2 |u| \cdot f(u), & u \text{ 没有父短语}\\ \log_2 |u| \left(f(u)-\dfrac{1}{|T_u|}\sum_{b\in T_u} f(b)\right), & u \text{ 有父短语}\end{cases} \tag{3-1}$$

其中，T_u 是 u 的所有父短语；$|T_u|$是父短语的数量。同时式(3-1)的第二个式子中减去了短语 u 的父短语出现的平均频次，可以消除因父短语重复计数所带来的偏差。

3. NC-value

C-value 只使用了短语及其父短语出现的频次信息，没有充分利用短语丰富的上下文信息。充分考虑上下文信息可以改进对高质量短语的识别，NC-value 正是基于这一思想对 C-value 进行了改进：

$$\text{NC-value}(u)=0.8\text{C-value}(u)+0.2\sum_{b\in C_u} f_u(b)\text{weight}(b)$$

$$\text{weight}(b)=\frac{t(b)}{n}$$

NC-value 在 C-value 的基础上，考虑候选短语 u 的上下文单词 $b\in C_u$ 的影响，其中，$f_u(b)$是指 b 作为 u 的上下文出现的次数，weight(b)是衡量 b 重要性的权重。

4. PMI

PMI 也是在抽取领域短语时常用的指标。PMI 值刻画了短语组成部分之间的一致性(concordance)程度。假设某个短语 u 由 u_l 和 u_r 两部分组成，u_l 和 u_r 的 PMI 值越大，u 越可能是 u_l 和 u_r 的一个有意义的组合。PMI 具体定义如下：

$$\text{PMI}(u_l,u_r)=\log\frac{p(u)}{p(u_l)p(u_r)}$$

5. 左邻字熵与右邻字熵

左邻字熵与右邻字熵用来刻画短语的自由运用程度，即用来衡量一个词左邻字集合与右邻字集合的丰富程度。熵表达了事件的不确定性(随机性)，熵越大则不确定程度越高。给定某候选短语 u，其左(右)邻字熵计算为

$$H(u)=-\sum_{x\in\chi} p(x)\log p(x)$$

其中，$p(x)$为某个左邻(右邻)字 x 出现大概率，χ 是 u 的所有左邻(右邻)字的集合。

统计指标特征及其作用如表 3-1 所示。

表 3-1 统计指标特征及其作用

统 计 指 标	特征及其作用
TF-IDF	挖掘能够有效代表某篇文档特征的短语
C-value	考虑了短语与其父短语的关系来挖掘高质量短语

续表

统 计 指 标	特征及其作用
NC-value	在 C-value 的基础上进一步考虑了上下文来挖掘高质量短语
PMI	挖掘组成部分一致性较高(经常一起搭配)的短语
左(右)邻字熵	挖掘左(右)邻丰富的短语

3.2 同义词挖掘

3.2.1 概述

同义词是指意义相同或相近的词。同义词的主要特征是它们在语义上相同或相似。语言中的同义关系十分复杂,同义关系至少包含以下几类。

(1) 不同国家的语言互译。

(2) 具有相同含义的词。

(3) 中国人的字、名、号、雅称、尊称、官职、谥号等。

(4) 动植物、药品、疾病等的别称或俗称。

(5) 简称。

需要注意的是,同义词表达的是词汇之间的语义相似性,而不是相关性。此外,同义词的一种特例是缩略词,但同义词远不止缩略词一种形式。缩略词是与全称强相关的,一般要求保留来自全称的部分字符,而同义词之间在形式上可能完全不相关。

3.2.2 典型方法

1. 基于同义词资源的方法

已有的同义词资源主要来自字典、网络字典以及百科词条。其中,字典中的同义词资源是经由专家手工整理而成的,通常质量较高,但难以完整收录(因为人力成本高昂)。近年来,随着网络百科的发展,互联网上的知识资源日益丰富,维基百科、百度百科成了新的同义词来源。利用字典和百科词条挖掘同义词准确度极高,所以是一种十分流行、广为采用的方法。这些同义词资源也为构建基于学习模型的同义词挖掘方法提供了丰富的样本。从同义词资源挖掘得到的同义词往往只包含书面用语,收录不完整,这是该方法的缺点所在。

首先介绍字典和网络字典,典型的字典资源包括 WordNet、汉语大词典等。通过查询一个词在这些字典中收录的同义词,便可以挖掘其同义词,这种方法简单、有效,但挖掘出的词条偏向书面用语。

类似地,通过查询百科词条,也可以获得一个词语的同义词。常见的百科词条资源有维基百科、百度百科等。通过爬取一个词语的百科词条页面,并解析其 Infobox 中的信息,便可获取同义词。通过这种方法挖掘出来的同义词通常质量较高,并且百科词条往往涵盖了各领域的词汇,覆盖面很广。

以百度百科词条“彼岸花”为例,在其 Infobox 中可以提取“别称”“拉丁名”等属性,从而可以提取“龙爪花、蟑螂花”等同义词,如图 3-4 所示。

2. 基于模式匹配的方法

基于模式匹配的方法,利用同义词在句子中被提及的文本模式,从句子中挖掘同义词。

中文学名	红花石蒜	纲	单子叶植物纲
拉丁学名	*Lycoris radiata* (L'Her.) Herb.	目	百合目
别　称	龙爪花、蟑螂花	科	石蒜科
界	植物界	属	石蒜属
门	被子植物亚门	种	石蒜
		命名者及年代	Herb., 1820

图 3-4　“彼岸花”的同义词

首先需要定义同义词抽取的模式(pattern),常见的中文模式有“又称”“亦称”和“括号”等。在定义好模式后,输入语料,将语料中的文本与模式进行匹配,若匹配成功,即可识别出同义词。表 3-2 列出常见的中文同义词模式。

表 3-2　中文同义词模式

模式(*X*、*Y* 表示一组同义词对)	举　例
X 又称 *Y*	番茄又称西红柿
X(*Y*)	明太祖(朱元璋)
X 简称 *Y*	巴塞罗那简称巴萨
X,亦称 *Y*	计量,亦称测量
X,别名 *Y*	曼珠沙华,别名彼岸花
X 的全称是 *Y*	皇马的全称是皇家马德里
X,俗称 *Y*	脊髓灰质炎,俗称小儿麻痹症

基于模式匹配的同义词挖掘通常具有较高的准确率,但在召回率方面存在局限性。每种文本模式的表达能力有限,只能召回基于该模式表达的同义词对,对于超出预定义模式的同义词对就无能为力了。此外,不同语言、不同语料中的同义关系的表达模式也不尽相同,很难穷举各种同义关系的表达模式,而且每个新语料都需要花费大量的人力手工定义匹配模式,代价十分高昂。

3. 自举法

自举法(bootstrapping)是对基于模式匹配的方法的改进,从一些种子样本或者预定义模式出发,不断地从语料中学习同义词在文本中的新表达模式,从而提高召回率。自举法是一个循环迭代的过程,每轮循环发现新模式、召回新同义词对,循环往复直至达到终止条件。自举法挖掘同义词的基本流程,如图 3-5 所示。

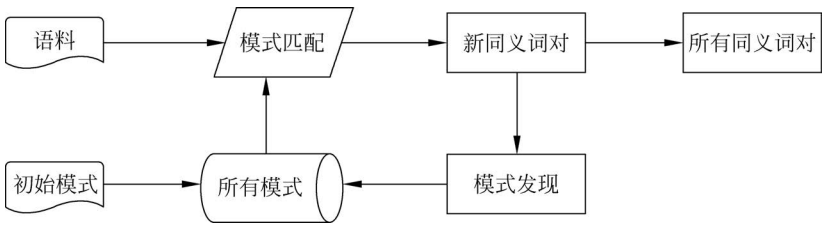


图 3-5　自举法挖掘同义词的基本流程

自举法可以自动挖掘新模式。新模式召回了更多的同义词对,因此提高了召回率。但是自动学习获得的新模式的质量难以得到保证,导致挖掘出的同义词对的准确率有所下降。一种直接的解决方法是对模式质量进行评估。一个好的模式应该尽量召回正确的同义词

对，而尽量少召回非同义词对，因此，可以事先构造一些正负例样本用于模式评估。

4. 其他方法

除了上述方法外，还有一些思路。

(1) 借助序列标注模型自动挖掘同义词的文本描述模式。例如，定义 ENT 表示实体，S_B 表示模式开始的位置，S_I 表示模式继续的位置，O 表示其他成分。基于标注好的文本数据，通过序列标注模型学习出新的模式。

(2) 基于图模型挖掘同义词。基于词与词之间的各种相似性可以构建一张词汇关联图。同义词在图上往往呈现出“抱团”的结构特性。也就是同义词之间关联紧密，不同义的词之间关联稀疏。这个特性就是复杂网络中的社团结构。因此，可以将同义词发现问题建模为图上的社团发现问题。它将词语组成的语义关联图作为输入并返回图上的社团，每个社团对应一组同义词。

3.3 缩略词挖掘


视频讲解

3.3.1 缩略词的概念和形式

缩略词是指一个词或者短语的缩略形式。缩略词的英文 Abbreviation 出自拉丁语，原意同 short。缩略词通常由原词中的一些组成部分构成，同时保持原词的含义，其检测与抽取在方法上与同义词类似，但是与同义词相比，缩略词在文本中出现的规则往往更简单。

在不同的语言中，缩略词的形式有所不同。主要介绍缩略词在表音(字母)文字(如拉丁语系)以及表意文字(如中文)中的形式。

表音文字以拉丁语系为例，缩略词的形式包括 contractions(简称)、crasis(元音融合)、acronyms(首字母缩略词)和 initialisms(首字母缩写)。

拉丁语系中不同类别的缩略词形式，如表 3-3 所示。

表 3-3 拉丁语系中不同类别的缩略词形式

缩略词形式	原 词	新 词
contractions	Doctor，I am(英语)	Dr，Fm
crasis	De le，de les(法语)	Du，des
acronyms	Severe Acute Respiratory Syndrome(英语)	SARS
initialisms	British Broadcasting Corporation(英语)	BBC

表意文字以中文为例，缩略词形式更加复杂。一个实体或者短语常由多个词组成，每个词包含的字数不同。缩略词往往是从每个词中选取一个或者多个字组成的，剩下的那些字则直接省略。中文缩略词的主要形式，如表 3-4 所示。

表 3-4 中文缩略词的主要形式

全 称	分 词 结 果	缩略词
中国中央电视台	中国 中央 电视台	央视
安全理事会	安全 理事 会	安理会
中国电子系统工程第四建设有限公司	中国 电子 系统工程 第四建设 有限公司	中电四公司

3.3.2 缩略词的检测与提取

1. 基于文本模式的抽取

基于文本模式构建抽取规则是缩略词抽取最常用的方法。由于缩略词本质上是同义词的一种形式，因此缩略词抽取中使用的规则与同义词抽取中的很相似。在缩略词抽取中，常见的基于文本模式的抽取规则，如表 3-5 所示。

表 3-5 常见的基于文本模式的抽取规则

模式（*X* 表示原词，*Y* 表示缩略词）	示　例
X（*Y*）	Support vector machine（SVM） Support-vector-machine（SVM）
X. *（*Y*）	Support vector machine for gression（SVM）
Y is the abbreviation of *X*	SVM is the abbreviation of Support vector machine
X and *Y* are synonyms	Support vector machine and SVM are synonyms
X，also known as *Y*	Support vector machine also known as SVM

2. 抽取结果的清洗和筛选

对缩略词搜索结果的清洗和筛选方法主要分为两种。

（1）利用数据集有关缩写的统计指标进行识别。

（2）使用机器学习模型构建二元分类模型，以此判断抽取出的缩略词正确与否。

这类算法常常需要事先构建一定规模的标注数据集。同时，这类算法依赖人为设计的特征，这些特征既包括前面提到的一系列统计指标，也包括文本特征。

缩略词判定中常用的文本特征包括字符匹配程度和词性特征。

单纯利用统计信息的缩略词识别方法能够准确识别常见的缩略词模式，但对于长尾模式的识别往往效果较差；而机器学习模型通常具有一定的泛化能力，因此能够适应不同文本和不同领域，对低频缩略词模式的识别能力更强，但是对训练数据和训练模型的依赖也更强。

3. 枚举并剪枝

枚举并剪枝是针对中文缩略词提出的一种有效方法。对于中文缩略词而言，缩略词中常常仅包含原词中的字符，并且字符间保持原有顺序。枚举并剪枝方法的输入是语料以及某个给定实体。这一方法首先穷举目标实体名称所有子序列，即所有可能的缩略形式，进一步排除没有在文本中出现过的或者出现次数太少的候选缩略词。

缩略词抽取方法虽然能够获取大量的缩略词对，但受限于语料大小，其对于新登录词往往效果较差。目前一些相关研究着眼于分析缩略词的规则，自动习得缩略词形式并进行预测。这种方法不依赖于语料，仅依靠输入的全称的相关文本，通过自然语言模型预测该全称可能的缩略词形式。以下以中文缩略词预测为例，介绍几种典型的预测方法。

3.3.3 缩略词的预测

1. 基于规则的方法

（1）针对特定字符和词语形式的局部规则，大致包括基于词性、基于位置或基于词之间的相互关联等规则。

（2）依赖语言环境的全局规则。

对于基于规则的方法来说，其规则是可控的、可解释的。但基于规则的方法也存在如下缺点：

① 很多规则往往是很复杂且难以被明确定义的；

② 专家成本高、泛化能力弱；

③可能存在同一个全称适用于多个匹配规则的情况。

2. 条件随机场

序列标注模型是预测缩略词的常用模型。在序列标注问题的场景下，全称中的每个字符分别被打上 1 或者 0 的标签，分别表示当前字符在结果缩略词中是否出现，如图 3-6 所示。

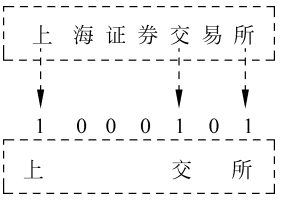


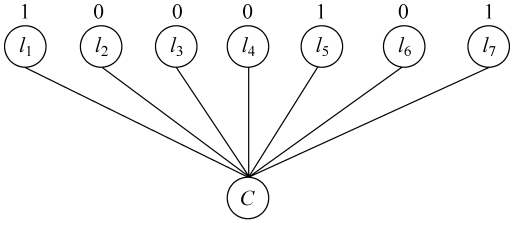
图 3-6　缩略词标签

条件随机场（CRF）是 Lafferty 等于 2001 年提出的一种建立在马尔可夫链基础上的无向图模型。CRF 每次标注时都会充分考虑已有的标注结果的影响。给定输入字符序列 $C=\{c_1,c_2,\cdots,c_T\}$，输出标签序列 $L=\{l_1,l_2,\cdots,l_T\}$，L 的计算过程为

$$P(L\mid C)=\frac{1}{Z(C)}\exp\left(\sum_{t=1}^{T}\sum_{k}\lambda_k f_k(l_t,l_{t-1},C,t)\right)$$

其中，f_k 表示定义在观测序列的两个相邻标签位置上的状态转移函数，并用于刻画相邻标签变量间的相关关系以及输入序列 C 对它们的影响；λ_k 为第 k 个特征的权重参数，$Z(C)$ 是规范化因子。

CRF 最常见的链式结构如图 3-7 所示。基于 CRF 构建缩略词预测模型常用到以下特征：字符级特征、词级别特征、位置特征、词的关联特征。



$C=\{c_1,c_2,c_3,c_4,c_5,c_6,c_7\}$
上海证券交易所
图 3-7　CRF 链式结构

3. 深度学习

在神经网络方法中，词或字符被表示为一个低维稠密空间中的向量。基于这些向量表示，可使用典型的网络结构抽取字词之间的组合特征，典型网络结构包括卷积神经网络（Convolutional Neural Network，CNN）、循环神经网络（Recurrent Neural Network，RNN）等。图 3-8 展示了一种典型的用于缩略词预测的深度神经网络模型，其主体结构为长短期记忆（Long and Short Term Memory，LSTM）网络。

深度神经网络模型常常需要使用预训练好的词向量来提升模型的性能。对于中文缩略词问题而言，字符本身的语义和字符在整个词语中的语义常常存在很大的差别。在中文相关的处理中，通常要将字符级向量表示及词汇级向量表示等不同粒度的语言信息输入到深度神经网络模型中，才能取得较好的效果。基于深度学习的缩略词预测的主要缺陷在于其

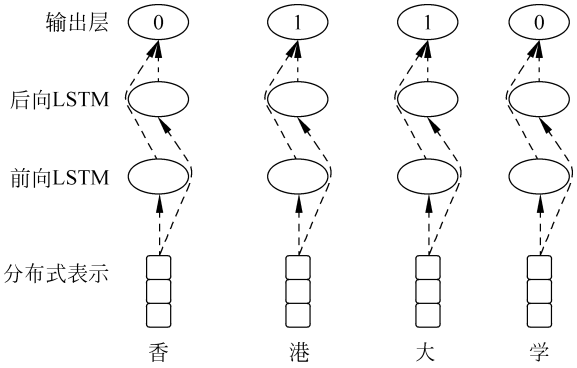


图 3-8　用于缩略词预测的深度神经网络模型

不可解释性，用户往往很难理解究竟是什么样的特征产生了最终的结果。这一点在很大程度上限制了对其性能的进一步提升。

视频讲解

3.4　实体识别

3.4.1　概述

命名实体是一个词或短语，它可以在具有相似属性的一组事物中清楚地标识出某一个事物。命名实体识别(Named Entity Recognition，NER)则是指在文本中定位命名实体的边界并将之分类到预定义类型集合的过程[38]。命名实体可以理解为有文本标识的实体。实体在文本中的表示形式通常被称作实体指代(Mention，或者直接被称为指代)。

NER 的输入是一个句子对应的单词序列 $s=\langle w_1, w_2, \cdots, w_N \rangle$，输出是一个三元组集合，其中每个元组形式为$\langle I_s, I_e, t \rangle$，表示 s 中的一个命名实体，其中 $I_s \in [1, N]$ 和 $I_e \in [1, N]$分别表示命名实体在 s 中的开始和结束位置，而 t 是实体类型。

命名实体识别任务示例如图 3-9 所示。

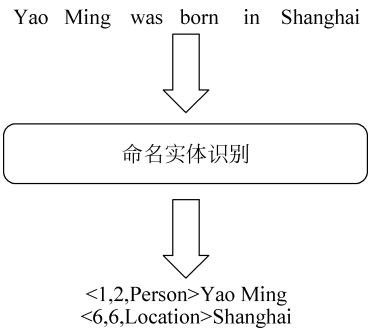


图 3-9　命名实体识别任务示例

3.4.2　传统的 NER 方法

1. 基于规则、词典和在线知识库的方法

这类方法基于规则、词典和在线知识库，依赖语言学专家手工构造规则。通常每条规则都被赋予权值，当遇到规则冲突的时候，选择权值最高的规则来判别命名实体的类型。比较著名的基于规则的 NER 系统包括 LaSIE-Ⅱ、NetOwl、Facile、SRA、FASTUS 和 LTG 等系统。这些系统主要基于人工制定的语义和句法规则来识别实体，LTG 系统使用的部分规则如表 3-6 所示。

表 3-6　LTG 系统使用的部分规则

规　　则	标　　注	举　　例
Xxxx+,DD+	人物	Wliite,33
Xxxx+ is? a? JJ * PROF	人物	Yuri Gromov,a former director

续表

规 则	标 注	举 例
Xxxx+ is? a? JJ * REL	人物	John White is beloved brother
Xxxx+ himself	人物	White himself
Xxxx+ area	地点	Beribidjan area
PROF of/at/with Xxxx+	组织机构	director of Trinity Motors
shares in Xxxx+	组织机构	shares in Trinity Motors

其中，Xxxx+代表大写单词序列，DD 代表数字，PROF 代表职业，REL 代表人物关系，JJ * 代表形容词序列。

基于规则的实体识别系统往往还需要借助实体词典，对候选实体进行进一步的确认。当词典详尽无遗时，基于规则的系统效果很好。但是基于特定领域的规则和并不完整的词典，往往会导致 NER 系统有着较低的召回率，而且这些规则难以应用到其他领域。

E. Alfonseca 和 Manandhar 提出了一种基于 WordNet 的实体分类方法。该方法的基本思想是计算某个词或实体与 WordNet 中的概念或者实例的语义相似性，将目标词挂载到相应的概念或者实例的上位词下，从而完成实体分类。WordNet 具有丰富的类别体系，因此这一方法可以极大地拓展普通 NER 模型类别的数量。这一方法无须人为定义模式，也无须标注样本，有时也被归类到无监督学习方法中。

2. 监督学习方法

应用监督学习方法时，NER 被建模为序列标注问题。NER 任务使用 BIO 标注法，其中 B 表示实体的起始位置，I 表示实体的中间或结束位置，O 表示相应字符不是实体。BIO 标注法示例如图 3-10 所示。

亚	里	士	多	德	出	生	于	斯	塔	基	拉
B-PER	I-PER	I-PER	I-PER	I-PER	O	O	O	B-LOC	I-LOC	I-LOC	I-LOC

图 3-10 BIO 标注法示例

基于序列标注的建模接收文本作为输入，产生相应的 BIO 标注为输出。常见的序列标注问题的建模模型包括 HMM 和 CRF。HMM 是一种生成式模型，直接对输入文本 X 和输出标签序列 Y 的联合概率 $P(Y,X)$ 建模。HMM 将序列 Y 视作隐变量，将文本 X 视作由这些隐变量经由马尔可夫随机过程生成的结果。因此，对 X 求解最优标签序列的过程可以建模为 $Y=\underset{Y}{\arg\max}P(Y,X)$。

基于 HMM 的建模假定了标签序列之间具有较强的马尔可夫性，这一假设太强，限制了其实际应用的效果。CRF 是一种判别式模型，直接建模并求解使 $P(Y \mid X)$ 最大的 Y。

NER 系统常会用到如表 3-7 所示的几类典型特征。

表 3-7 NER 系统常用到的典型特征

特 征	含义/说明	示 例
核心词特征	核心词(Head Noun)是名词短语中的一个主要名词，可以通过查表来识别候选实体是否存在核心词	例如，“第二人民医院”中的“医院”是表述核心功能和性质的核心词，因此相应短语有较大可能是一个实体

续表

特　　征	含义/说明	示　　例
词典特征	指基于词典的特征，比如是否存在于词典中，或者与词典词项的匹配程度	例如，若 Shanghai 出现在词典中，则对应的词典特征可以表示为 TRUE
构词特征	构词特征是较为明显的特征，主要描述词语的构成情况，而且不依赖于特定领域	以大写字母组成的实体名称很有可能属于机构类别，例如，Sun、Microsoft 等
词形特征	相同类别的词可能具有相同的词形	例如，Intel 的 CPU 名称通常以“Intel＋数字”的形式出现，包括 Intel 8008、Intel 8080、Intel 8086、Intel 8088 等
词缀特征	单词的词缀可以作为划分实体类型的依据，在专业领域效果明显，被广泛使用	例如，在化学元素的命名体系中，常常使用“-ium”作为后缀，包括 Rubidium、Strontium、Yttrium 等
词性特征	又称为 POS(Part-Of-Speech)特征，对于识别命名实体的边界有较大的作用	例如，“Yao Ming was born in Shanghai”对应的词性特征为：('Yao Ming','NOUN'),('was','VERB'),('brn','VERB'),('in','ADP'),('Shanghai','NOUN')

3. 半监督学习方法

在 NER 任务中，监督学习方法面临着缺少标注语料和数据稀疏的问题，因此半监督学习(Semi-Supervised Learning，SSL)作为监督学习与无监督学习相结合的一种学习方法被应用于 NER 任务中。

一类典型的半监督学习方法是自举法，通常从少量标注数据、大量未标注数据和一小组初始假设或分类器开始，迭代生成更多的标注数据，直至到达某个阈值。半监督 NER 在某些类型的数据上已经取得了与监督学习方法可比拟的效果。

M. Collins 和 Singer 提出了一种基于协同训练(co-training)的方法来解决命名实体识别问题。该方法旨在学习两套不同的实体识别规则。在学习过程中，每一类规则为另一类规则的学习提供弱监督。该算法中的分类规则包括两种：一种是拼写规则，另一种是上下文规则。

基于协同训练的 NER 方法过程如图 3-11 所示。

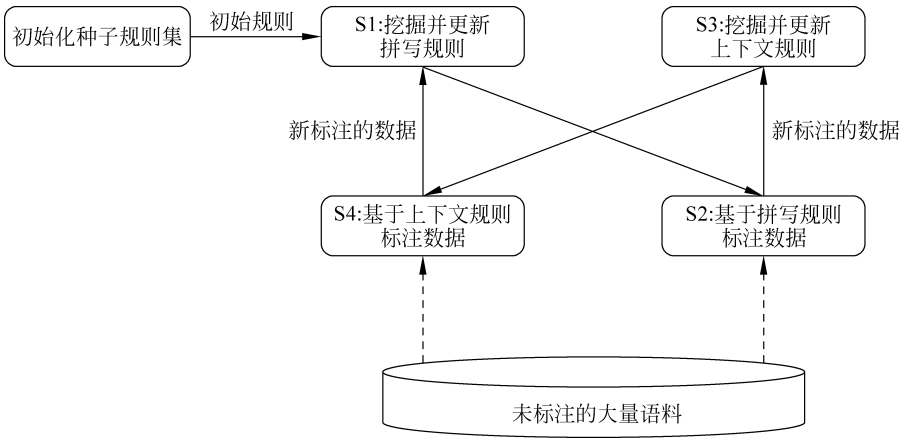


图 3-11　基于协同训练的 NER 方法过程

3.4.3　基于深度学习的 NER 方法

基于深度学习的方法通常将 NER 问题建模为序列标注问题。在 NER 任务中，常用的深度神经网络有 RNN 和 CNN，其中 CNN 主要用于向量特征学习，RNN 则可以同时用于向量特征学习和序列标注。

基于深度学习的 NER 框架如图 3-12 所示，其主要包含 3 个模块：输入的分布式表示(distributed representation)、上下文编码器(context encoder)和标签解码器(tag decoder)，是一个典型的编码器-解码器(encoder-decoder)框架。

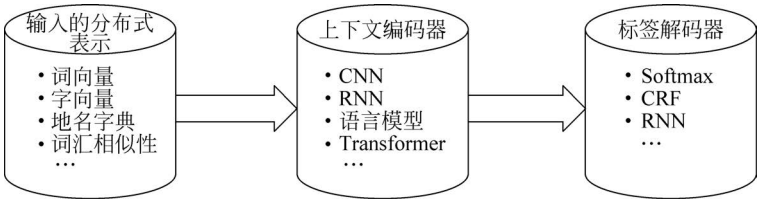


图 3-12　基于深度学习的 NER 框架

1. 输入的分布式表示

深度学习模型无法直接接收符号化文本作为输入，而只能接收数值向量。因此，基于深度学习的 NER 方法，首先需要将输入的句子表示成一组向量。

(1) 词向量。词是句子的基本组成单位，为了将句子表示成一组向量，一个简单的思路是将句子中的每个词表示成一组向量，再通过特征融合得到整个句子的向量表示。词向量往往通过无监督算法，如词袋模型(CBOW)和 Skip-Gram 模型等，并经过大量文本的预训练得到。

(2) 字向量。除了词向量外，另一种思路是将词中的每个字用向量表示，这样可以得到词向量难以表示的一些信息，如词中的前缀和后缀等字符信息。而且，字符级的向量能够很自然地处理词典外的词汇。通常使用 CNN 和 RNN 等模型提取字向量。

(3) 混合表示。除了词级和字符级表示外，一些研究还将其他信息(例如，是否出现在词典的列表中)纳入词的最终表示。换言之，基于深度神经网络模型习得的向量表示可以与传统特征工程得到的向量表示组合，以融合更多的额外信息，从而提高模型的准确率。但这一做法也有可能降低泛化能力。

2. 上下文编码器

基于深度学习的 NER 方法的第二个阶段是从输入表示中学习上下文编码器。上下文编码器有两种常用的模型结构：CNN 和 RNN。

(1) 基于 CNN 的编码器一般以整个句子作为输入，使用一维卷积对句子进行特征提取。在输入表示学习阶段，已经将句子中的每个单词表达为向量，通常还会考虑单词在句子中的相对位置特征，以增强单词的表示效果。

(2) 循环神经网络的特点是考虑了句子中前后字符之间的相互影响。循环神经网络及其变体在序列数据建模方面都取得了显著成效。特别是双向循环神经网络(Bi-RNN)能从两个方向(正向和逆向)来处理一个句子，能够捕捉被单向 RNN 所忽略的模式。

3. 标签解码器

标签解码器将经过编码的上下文表示作为输入并产生对应于输入句子的标签。下面将分别介绍标签解码器的 3 种架构：全连接层＋Softmax、CRF 和 RNN。

(1) 全连接层＋Softmax。全连接层接收每个单词中间层向量表示，产生标签分值向量

$\boldsymbol{Y}=(y_1, y_2,\cdots,y_i)$作为输出。向量$\boldsymbol{Y}$被输送到 Softmax 层，产生最终的标签概率分布。

(2) CRF 是一类能够充分考虑输出标签之间关系的序列标注模型，可以有效建模，最终预测标签之间的约束关系，从而提高预测准确率。

(3) RNN 可用于解码，当将编码层的向量映射为标签序列，且实体类型的数量很大时，相比于其他的解码器，RNN 解码器的训练速度更快。

3.5 实验：实体识别

3.5.1 实验内容

实体是知识图谱的基本单元，也是文本中承载信息的重要语言单位。实体识别又称命名实体识别，其目的就是识别出文本中实体的命名性指称项，并标明其类别。一般来说，命名实体识别需要识别出实体类、时间类和数字类三大类，如人名、机构名、地名、时间、日期、货币和百分比。

本实验主要使用 Python 语言，训练 BiLSTMCRF 模型，然后输入测试句子，识别出其中的命名实体并输出。

3.5.2 实验目标

(1) 理解实体识别的概念。

(2) 掌握 BiLSTMCRF 模型。

(3) 掌握使用 Python 进行实体识别的方法。

3.5.3 实验步骤

1. 创建项目

在 PyCharm 桌面图标上双击打开 PyCharm，单击 New Project。如图 3-13 所示，在弹出 New Project 窗口的 Location 栏中输入项目所在位置，取消选中 Create a main. py welcome script 复选框，并选中 Existing Interpreter 单选按钮，单击“...”按钮进入 Add Python Interpreter 窗口。

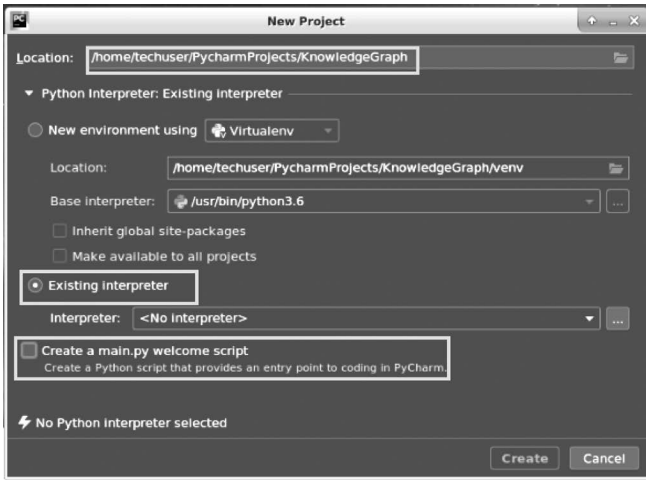


图 3-13 创建项目

选择 System Interpreter，在 Interpreter 栏中选择/usr/bin/python3，单击 OK 按钮，完成 Python 环境的设置，如图 3-14 所示。

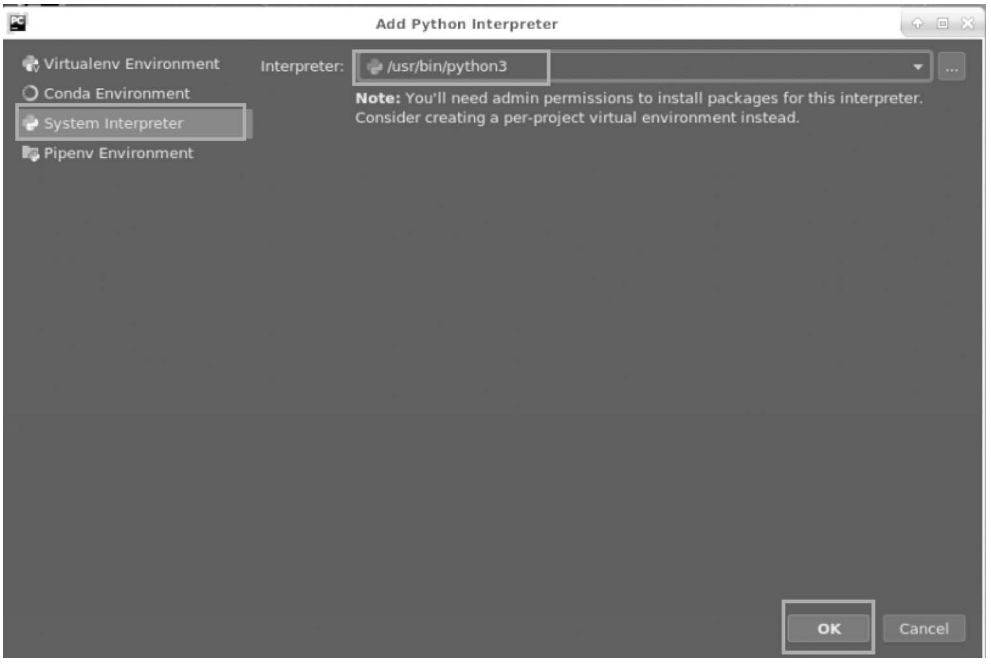


图 3-14　设置 Python 环境

回到 New Project 窗口，单击 Create 按钮，完成效果如图 3-15 所示。

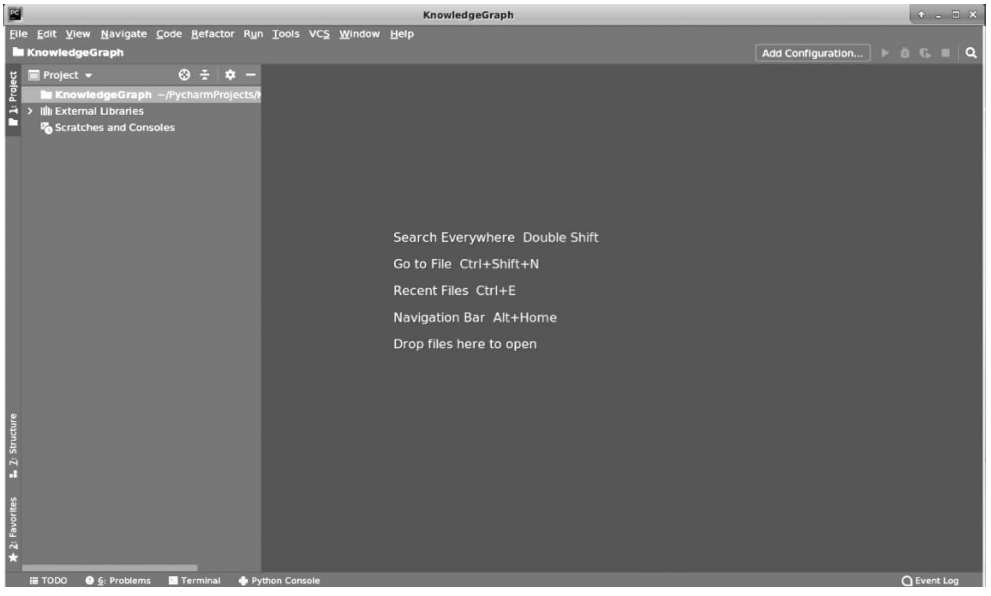


图 3-15　新项目创建完成

2. 创建文件

选择 Project 下方的 Algorithm 项目标志，可以发现下面并没有任何文件，需要先创建 Python 文件。右击 Algorithm 项目标志，选择 New→Python File 命令（见图 3-16）。在弹

出的 New Python file 窗口(见图 3-17)的 Name 输入框中输入文件名 EntityRecognition,按 Enter 键即可完成文件创建。完成效果如图 3-18 所示。

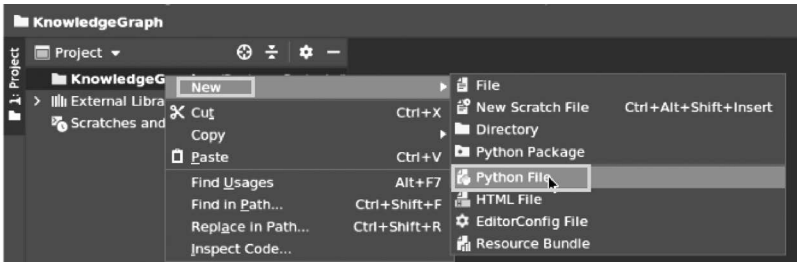


图 3-16 选择 New→Python File 命令

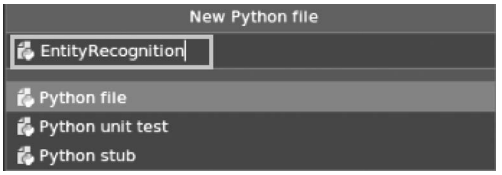


图 3-17 New Python file 窗口

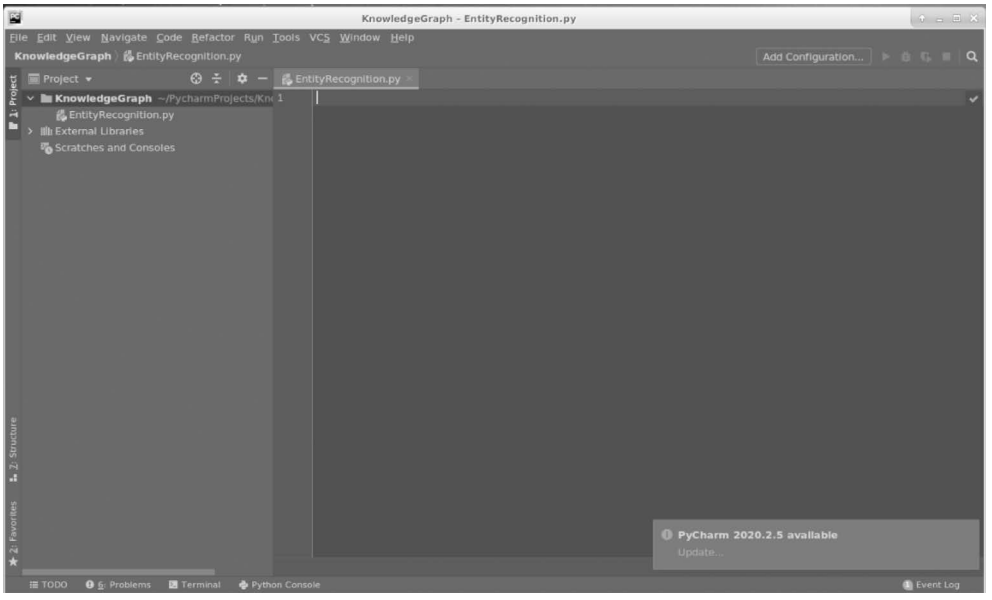


图 3-18 创建新文件

3. 数据导入

如图 3-19 所示,在网络浏览器中输入网址下载文件。文件下载完成后,进入/home/techuser/Downloads 目录,找到下载的文件 models.zip,右击,在弹出的快捷菜单中选择 Open Terminal Here 命令,如图 3-20 所示。

在 Terminal 界面,创建 models 文件夹,将 models.zip 中的文件解压到 models 文件夹下,同时将 models 文件夹移动到 KnowledgeGraph 项目下:

```
mkdir models
unzip -d ./models models.zip
```

图 3-19　在网络中下载文件

图 3-20　打开 Terminal 界面

```
mv  models ../PycharmProjects/KnowledgeGraph/
```

如图 3-21 所示，打开 KnowledgeGraph 项目，可以看到 models 文件夹出现在目录中。

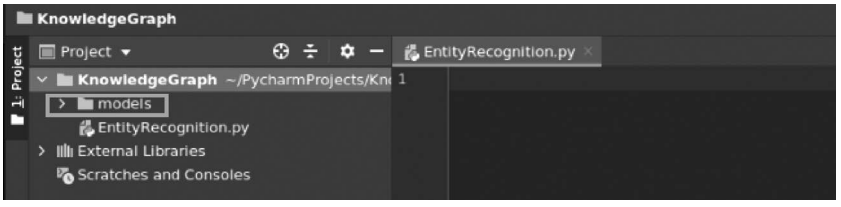


图 3-21　成功创建 models 文件夹

4. 算法代码

打开 EntityRecognition.py 文件，会看到里面一片空白，依次输入以下代码段（以下代码段均须保存并运行才能通过检测）。

（1）导入工具包。

```
#coding = utf - 8
import pickle                    #加载数据
import torch                     #深度学习框架
from torch import nn             #用于构建模型
```

（2）获取实体函数，通过预测结果、输入文本、标签等获取实体信息。

```
def get_entity(result, input_str, tag):
'''
:param result:
```

```
    :param input_str:输入文本
    :param tag:实体标签
    :return:获取到的实体
    '''
    entities = []
    for (start,end) in result:
        entities.append({
            "start":start,
            "stop":end + 1,
            "word":input_str[start:end+1],
            "type":tag
        })
    return entities
```

(3) 获取标签函数,根据输入的预测结果去获取对应的标签。

```
def get_tags(results, tag, tag_map):
    '''
    获取标签类
    :param results:路径首位
    :param tag:当前标签
    :param tag_map:所有标签
    :return:
    '''
    start_tag = tag_map["B-" + tag]
    mid_tag = tag_map["I-" + tag]
    end_tag = tag_map["E-" + tag]
    single_tag = tag_map["STOP"]
    o_tag = tag_map["O"]
    begin = -1
    end = 0
    tags_arr = []
    last_tag = 0
    for index, tag in enumerate(results):
        if tag == start_tag and index == 0:
            begin = 0
        elif tag == start_tag:
            begin = index
        elif tag == end_tag and last_tag in [mid_tag, start_tag] and begin > -1:
            end = index
            tags_arr.append([begin, end])
        elif tag == o_tag or tag == single_tag:
            begin = -1
        last_tag = tag
    return tags_arr
```

(4) 实体识别类,该类主要实现命名实体识别功能,通过获取预训练的模型,预测输入文本的实际标签。

```
class NER(object):
    def __init__(self):
    self.embedding_size = 100
```

```
    self.hidden_size = 128
    self.batch_size = 160
    self.tags = ["ORG", "PER"]
    self.dropout = 0.5
    self.init_model()
def init_model(self):
     '''
    初始化模型
    :return:
    '''
    model_params = self.load_model_params()
    self.tag_map = model_params["tag_map"]
    input_size = model_params["input_size"]
    self.vocab = model_params["vocab"]
    #声明模型
    self.model = BiLSTMCRF(vocab_size = input_size, embedding_dim = self.embedding_size,
hidden_dim = self.hidden_size, tag_map = self.tag_map
    )
    self.restore_model()
def restore_model(self):
    '''
    加载模型
    :return:
    '''
    self.model.load_state_dict(torch.load("models/model.pkl"))
def load_model_params(self):
    '''
    加载模型参数
    :return:模型参数
    '''
    with open("models/params.pkl", "rb") as f:
        model_params = pickle.load(f)
    return model_params
def predict(self, input_str = ""):
    '''
    预测函数
    :return:
    '''
    if input_str == "":
        input_str = input("请输入测试文本: ")
    input_emb = [self.vocab.get(i, 0) for i in input_str]
    sentences = torch.tensor(input_emb).view(1, -1)
    scores, results = self.model(sentences)
    entities = []
    for tag in self.tags:
        tags_arr = get_tags(results[0], tag, self.tag_map)
        entities += get_entity(tags_arr, input_str, tag)
    return entities
```

(5) 模型类,该类通过双向长短期记忆模型和条件随机场来预测测试文本中的实体。

```
class BiLSTMCRF(nn.Module):
```

```
def __init__(self,batch_size = 32,vocab_size = 20,hidden_dim = 128,dropout = 0.0,embedding_
dim = 300,tag_map = {}):
    super(BiLSTMCRF, self).__init__()
    self.batch_size = batch_size
    self.hidden_dim = hidden_dim
    self.embedding_dim = embedding_dim
    self.vocab_size = vocab_size
    self.dropout = dropout

    self.tags_len = len(tag_map)
    self.tag_map = tag_map
    self.transitions = nn.Parameter(torch.randn(self.tags_len, self.tags_len))
    self.transitions.data[:, self.tag_map["START"]] = -1.
    self.transitions.data[self.tag_map["STOP"], :] = -1.
    self.word_embeddings = nn.Embedding(vocab_size, self.embedding_dim)
    self.lstm = nn.LSTM(self.embedding_dim, self.hidden_dim // 2,
            num_layers = 1, bidirectional = True, batch_first = True, dropout = self.dropout)
    self.hidden2tag = nn.Linear(self.hidden_dim, self.tags_len)
    self.hidden = self.get_hidden()
def get_hidden(self):
    '''
    获取隐藏层
    :return:
    '''
    return (torch.randn(2, self.batch_size, self.hidden_dim // 2),
          torch.randn(2, self.batch_size, self.hidden_dim // 2))
def get_lstm_features(self, sentence):
    '''
    获取 lstm 的特征
    :param sentence: 输入句子
    :return:
    '''
    self.hidden = self.get_hidden()
    length = sentence.shape[1]
    embeddings = self.word_embeddings(sentence).view(self.batch_size, length, self.
embedding_dim)
    lstm_out, self.hidden = self.lstm(embeddings, self.hidden)
    lstm_out = lstm_out.view(self.batch_size, -1, self.hidden_dim)
    features = self.hidden2tag(lstm_out)
    return features
def forward(self, sentences):
    '''
    模型运行函数
    :param sentences:输入句子
    :return:分数,分类结果
    '''
    sentences = torch.tensor(sentences, dtype = torch.long)
    lengths = [i.size(-1) for i in sentences]
    self.batch_size = sentences.size(0)
    features = self.get_lstm_features(sentences)
    scores = []
```

```
        results = []
        for feature, len in zip(features, lengths):
            feature = feature[:len]
            score, path = self.viterbi_decode(feature)
            scores.append(score)
            results.append(path)
        return scores, results
    def viterbi_decode(self, features):
        '''
        维特比解码
        :param features:lstm 提取到的特征
        :return:维特比分数,维特比解码结果
        '''
        trellis = torch.zeros(features.size())
        return_pointers = torch.zeros(features.size(), dtype=torch.long)
        trellis[0] = features[0]
        for t in range(1, len(features)):
            v = trellis[t - 1].unsqueeze(1).expand_as(self.transitions) + self.transitions
            trellis[t] = features[t] + torch.max(v, 0)[0]
            return_pointers[t] = torch.max(v, 0)[1]
        viterbi = [torch.max(trellis[-1], -1)[1].cpu().tolist()]
        return_pointers = return_pointers.numpy()
        for rp in reversed(return_pointers[1:]):
            viterbi.append(rp[viterbi[-1]])
        viterbi.reverse()
        viterbi_score = torch.max(trellis[-1], 0)[0].cpu().tolist()
        return viterbi_score, viterbi
```

(6) 主函数,主要负责声明 NER 类进行预测,并存储预测结果:

```
if __name__ == "__main__":
 ner = NER()
 result = ner.predict("对于这种问题,我们将交由天津市合佳威立雅环保服务公司进行处理")
 print(result)
 with open("results.txt",'w',encoding="utf-8") as f:
    f.write(str(result))
```

5. 运行代码

如图 3-22 所示,打开 Terminal 界面,在出现的命令行窗口中输入命令即可运行程序。

图 3-22　运行程序

等待程序运行完成后,打开 models 文件夹,可以看到其中多了 results.txt 文件,如图 3-23 所示。打开 results.txt 文件,可以看到测试文本的实体识别结果,如图 3-24 所示。

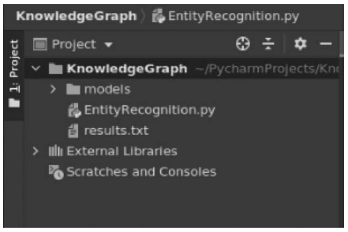


图 3-23　文件夹中出现 results.txt 文件

图 3-24　运行结果

3.5.4　实验总结

本实验使用 Python 语言实现了中文命名实体识别的功能。其中借助 BiLSTM 双向 LSTM 模型＋CRF 模型的方式预测输入句子中的命名实体。虽然因为时间问题，并未进行该模型的训练过程的展示，仅仅只是进行了输入案例的预测，但相信能够帮助实验者理解命名实体的实现过程。

课后习题

1. 选择题

1-1 下列哪些是表示高质量短语的统计指标？（　　）

A. TF-IDF　　B. PMI　　C. C-value　　D. 以上全都是

1-2 下列哪个统计指标可以判断出“马克思”比“马克”更好？（　　）

A. TF-IDF　　B. PMI　　C. 左/右邻字熵　　D. C-value

1-3 下列哪个统计指标可以判断出短语“数据结构”比“数据”和“结构”更好？（　　）

A. TF-IDF　　B. PMI　　C. 左/右邻字熵　　D. C-value

1-4 下列不是同义词形式转述的是（　　）。

A. X 属于 Y　　B. X 等价 Y　　C. $X(Y)$　　D. X 简称 Y

1-5 同义词短语挖掘的一般方法不包括（　　）。

A. 基于同义词词典匹配　　B. 左/右邻字熵

C. 语料对齐挖掘　　D. 元数据搜索

1-6 缩略词挖掘内容主要包括（　　）。

A. 缩略词的检测　　B. 缩略词的抽取

C. 缩略词的预测　　D. 以上都是

1-7 将“北京市第十四中学”缩略成“北京十四中”是运用了（　　）规则。

A. 基于词性　　B. 基于位置

C. 基于词之间的相互关联　　D. 全局规则

1-8（多选题）英文缩略词常常有哪些缩写形式？（　　）

A. 采用短语中每个词或关键词的首字母组成缩写

B. 采用词的前两个或三个字母组成缩略词

C. 省略单词的元音字母，保留辅音字缩略词形式

D. 省略单词的元音字母，保留关键辅音字缩略词形式

2. 判断题

2-1 缩略词是同义词的一种重要形式。缩略词的检测与抽取在方法上与同义词相似，但是比起同义词，缩略词在文本中出现的规则往往更复杂。（　　）

2-2 实体识别的方法主要包括基于规则、词典和在线知识库的方法，监督学习方法，半监督学习方法以及基于深度学习的 NER 方法。（　　）

2-3 实体识别是对于领域语料库而言，我们想要抽取的一些人名、地名、机构名信息。（　　）

2-4 实体类型主要包括组织名、人名、地名、时间表达式以及数值表达式等。（　　）

3. 简答题

3-1 领域短语挖掘主要的目的是什么？有什么方法？

3-2 同义词短语挖掘指的是什么？一般方法有哪些？

3-3 缩略词挖掘关键是什么？

3-4 实体识别时识别的是什么？实体识别常用的方法有哪些？