# Lecture 7 超基础复习笔记 – 命名实体识别（NER）

## 零、先理解问题（什么是命名实体识别？）

### 什么是命名实体（Named Entity）？

**定义：**文本中具有特定意义的名称

**常见类型：**

```
人名（PERSON）：
  - "Rowan Atkinson"
  - "Mike Pence"

地名（LOCATION）：
  - "Dutch"（荷兰语的，指荷兰）
  - "British"（英国的）

组织名（ORGANIZATION）：
  - "BBC"
  - "Google"

日期/时间（DATE/TIME）：
  - "January 20, 2017"
  - "2013"

其他（MISC）：
  - 语言名
  - 货币
  - 百分比
```

### 什么是命名实体识别（NER）？

**任务：**在文本中自动识别并标注命名实体

**例子：**

```
输入文本：
"British actor Rowan Atkinson, best known as 'Mr. Bean,' has died."

NER输出：
British ← NORP（国籍/宗教/政治团体）
Rowan Atkinson ← PERSON（人名）
Mr. Bean ← PERSON（人名）
```

---

### 为什么重要?

**应用场景：**

1. **信息提取** 从新闻中提取：谁在哪里做了什么 → 构建知识图谱
2. **问答系统** 问题："谁是英国首相？" → 需要识别"英国"（地名）和"首相"（职位）
3. **事实核查** 识别声明中提到的人物、地点 → 验证真实性
4. **内容推荐** 识别用户关注的人物、地点 → 推荐相关内容

---

## 一、两种NER方法

### 方法1：基于规则（Gazetteer）

**核心思想：**使用预定义的实体列表进行匹配

```
有一个国家列表：
countries = ["USA", "UK", "China", "Brazil", ...]
```

```
在文本中查找这些词
→ 如果找到，标注为"COUNTRY"
```

**使用工具：**GATE（General Architecture for Text Engineering）

---

### 方法2：基于机器学习（Neural Network）

**核心思想：**训练神经网络模型自动学习实体特征

```
训练数据：
"Barack Obama was born in Hawaii" → Obama=PERSON, Hawaii=LOCATION

模型学习：
- 人名特征：通常是大写开头的连续词
- 地名特征：通常跟在"in", "at"等介词后
```

**使用工具：**spaCy

---

## 二、Gazetteer方法（GATE）

### 1. 什么是Gazetteer?

**定义：**地名词典，存储预定义实体的列表

**通俗理解：**一本"花名册"

```
就像学校的学生名单：
- 一班：张三、李四、王五
- 二班：赵六、孙七、周八

Gazetteer就是各种实体的名单：
- 国家：USA, UK, China, ...
- 语言：English, Chinese, Spanish, ...
- 公司：Google, Microsoft, Apple, ...
```

---

## 2. Gazetteer的结构

**基础结构：**

```
实体名称 + 特征（可选）
```

**例子：**

```
# countries.csv
Country,Code,WikiURL
Brazil,BR,https://en.wikipedia.org/wiki/Brazil
China,CN,https://en.wikipedia.org/wiki/China
USA,US,https://en.wikipedia.org/wiki/United_States

# languages.csv
Language,WikiURL
English,https://en.wikipedia.org/wiki/English_language
Chinese,https://en.wikipedia.org/wiki/Chinese_language
```

---

## 3. 创建Gazetteer

### 步骤1：准备数据

```
import pandas as pd

# 读取国家列表
countries = pd.read_csv("countries.csv")

# 数据示例
#   Country  Code
# 0 Brazil   BR
# 1 China    CN
# 2 USA      US
```

---

### 步骤2：构建Wikipedia URL

**规则：**

```
基础URL：https://en.wikipedia.org/wiki/
完整URL：基础URL + 实体名称

例子：
Brazil → https://en.wikipedia.org/wiki/Brazil

特殊情况（多个单词）：
"United States" → https://en.wikipedia.org/wiki/United_States
（空格替换为下划线）
```

**代码实现：**

```
def create_wiki_url(entity_name):
    """
    创建Wikipedia URL

    参数：
    entity_name：实体名称

    返回：
    Wikipedia URL
    """
    # 替换空格为下划线
    name = entity_name.replace(" ", "_")

    # 构建URL
    url = f"https://en.wikipedia.org/wiki/{name}"

    return url
```

**例子：**

```
create_wiki_url("Brazil")
# 'https://en.wikipedia.org/wiki/Brazil'

create_wiki_url("United States")
# 'https://en.wikipedia.org/wiki/United_States'
```

```
create_wiki_url("Bosnia and Herzegovina")
# 'https://en.wikipedia.org/wiki/Bosnia_and_Herzegovina'
```

---

### 步骤3：使用GATE创建Gazetteer

#### GATE的Gazetteer格式：

```
from gatenlp import Document
from gatenlp.processing.gazetteer import StringGazetteer

# 创建Gazetteer
gaz = StringGazetteer()

# 添加实体
gaz.add("Brazil", {"code": "BR", "type": "COUNTRY"})
gaz.add("China", {"code": "CN", "type": "COUNTRY"})
gaz.add("USA", {"code": "US", "type": "COUNTRY"})

# 或者从列表批量添加
for _, row in countries.iterrows():
    gaz.add(row['Country'], {
        "code": row['Code'],
        "type": "COUNTRY",
        "wiki_url": create_wiki_url(row['Country'])
    })
```

---

### 步骤4：应用Gazetteer标注文本

```
# 创建文档
text = "British actor Rowan Atkinson has died."
doc = Document(text)

# 应用Gazetteer
gaz.run(doc)

# 查看标注结果
for ann in doc.annset():
    print(f"实体: {doc[ann.start:ann.end]}")
```

```
    print(f"类型: {ann.type}")
    print(f"特征: {ann.features}")
    print()
```

**输出示例：**

```
实体: British
类型: COUNTRY
特征: {'code': 'GB', 'wiki_url': 'https://en.wikipedia.org/wiki/United_
```

---

## 4. Gazetteer的优缺点

**优点：**

```
1. 简单直接
   → 只需要准备列表

2. 准确率高（对于列表中的实体）
   → 完全匹配，不会出错

3. 可控性强
   → 可以随时添加/删除实体

4. 无需训练
   → 不需要标注数据
```

**缺点：**

```
1. 覆盖不全
   → 只能识别列表中的实体
   → 新实体无法识别

2. 歧义问题
   例子:
   "Washington"
   → 可能是人名（乔治·华盛顿）
   → 可能是地名（华盛顿州、华盛顿特区）
   → Gazetteer无法区分
```

```
3. 上下文问题
   例子:
   "I like turkey"
   → "turkey"可能是国家土耳其
   → 也可能是火鸡肉
   → Gazetteer只看词，不看上下文

4. 维护成本
   → 需要不断更新列表
   → 新的实体（新公司、新产品）需要手动添加
```

---

# 三、机器学习方法（spaCy）

## 1. spaCy是什么?

**定义：**一个现代的NLP库，使用神经网络进行文本处理

**核心组件：**

```
Processing Pipeline（处理流水线）:

文本 → 分词器 → 词性标注器 → 依存句法分析器 → NER → 输出
       (Tokenizer)  (POS Tagger)   (Dependency Parser)  (Named Entity
```

**NER模型：**

```
使用神经网络（深度学习）
→ 自动学习实体的特征
→ 不需要人工规则
```

---

## 2. spaCy的基本使用

### 步骤1：安装和加载模型

```
import spacy

# 加载英语模型
nlp = spacy.load("en_core_web_sm")
# en_core_web_sm: 英语小型模型
# 其他选项:
# - en_core_web_md: 中型模型（更准确）
# - en_core_web_lg: 大型模型（最准确）
```

---

### 步骤2：处理文本

```
text = "British actor Rowan Atkinson, best known as 'Mr. Bean,' has di

# 处理文本
doc = nlp(text)

# doc是一个Document对象，包含了所有的标注信息
```

---

### 步骤3：提取实体

```
# 遍历所有实体
for ent in doc.ents:
    print(f"实体: {ent.text}")
    print(f"类型: {ent.label_}")
    print(f"位置: {ent.start_char} - {ent.end_char}")
    print()
```

**输出示例：**

```
实体: British
类型: NORP
位置: 0 - 7

实体: Rowan Atkinson
类型: PERSON
位置: 14 - 28
```

```
实体： Mr. Bean
类型： PERSON
位置： 48 - 56
```

---

## 3. spaCy的实体类型

**常见类型：**

| 标签 | 英文全称 | 中文 | 例子 |
| --- | --- | --- | --- |
| **PERSON** | Person | 人名 | Rowan Atkinson, Mike Pence |
| **ORG** | Organization | 组织/公司 | BBC, Google, United Nations |
| **GPE** | Geopolitical Entity | 地理政治实体 | London, USA, China |
| **LOC** | Location | 地点 | Pacific Ocean, Mount Everest |
| **DATE** | Date | 日期 | January 20, 2017, yesterday |
| **TIME** | Time | 时间 | 3 PM, morning |
| **MONEY** | Money | 货币 | $100, £50 |
| **PERCENT** | Percentage | 百分比 | 50%, 3.5% |

| 标签 | 英文全称 | 中文 | 例子 |
| --- | --- | --- | --- |
| NORP | Nationalities/Religious/Political groups | 国籍/宗教/政治团体 | British, American, Muslim |
| FAC | Facility | 设施 | Brooklyn Bridge, Heathrow Airport |
| PRODUCT | Product | 产品 | iPhone, Windows 10 |
| EVENT | Event | 事件 | World War II, Olympics |
| WORK_OF_ART | Work of Art | 艺术作品 | Hamlet, Mona Lisa |
| LAW | Law | 法律 | Constitution, Bill of Rights |
| LANGUAGE | Language | 语言 | English, Chinese |

---

## 4. 完整示例

```
import spacy

# 加载模型
nlp = spacy.load("en_core_web_sm")
```

```
# 处理多条文本
texts = [
    "A video shows Dutch politician Tunahan Kuzu putting a grilled chee
    "British actor Rowan Atkinson, best known as 'Mr. Bean,' has died.
    "Mike Pence once said that smoking doesn't kill people."
]

# 分析每条文本
for text in texts:
    doc = nlp(text)

    print(f"文本: {text}")
    print(f"实体:")

    for ent in doc.ents:
        print(f"  - {ent.text} ({ent.label_})")

    print()
```

**输出：**

```
文本: A video shows Dutch politician Tunahan Kuzu putting a grilled che
实体:
  - Dutch (NORP)
  - Tunahan Kuzu (PERSON)

文本: British actor Rowan Atkinson, best known as 'Mr. Bean,' has died.
实体:
  - British (NORP)
  - Rowan Atkinson (PERSON)
  - Mr. Bean (PERSON)

文本: Mike Pence once said that smoking doesn't kill people.
实体:
  - Mike Pence (PERSON)
```

---

## 5. 自定义Pipeline（优化性能）

**问题：**默认pipeline包含很多组件，但我们只需要NER

```
# 默认pipeline
nlp = spacy.load("en_core_web_sm")
# 包含：tok2vec, tagger, parser, ner, attribute_ruler, lemmatizer

# 这些组件会消耗时间和内存
```

**解决方案：**只加载需要的组件

```
# 只加载tokenizer和ner
nlp = spacy.load("en_core_web_sm",
                 disable=["tagger", "parser", "attribute_ruler", "lemm

# 或者明确指定要加载的组件
nlp = spacy.load("en_core_web_sm",
                 disable=[pipe for pipe in nlp.pipe_names if pipe not
```

**效果：**

```
加载速度：快 3-5 倍
处理速度：快 2-3 倍
内存占用：减少 50%
```

---

# 四、实体统计与可视化

## 1. 统计每条文本的实体数量

```
import pandas as pd
import spacy

# 加载数据
df = pd.read_csv("snopes.csv")

# 加载spaCy模型
nlp = spacy.load("en_core_web_sm", disable=["tagger", "parser"])

# 统计实体数量
entity_counts = []
```

```
for text in df['claim']:
    doc = nlp(text)
    entity_counts.append(len(doc.ents))

# 添加到DataFrame
df['entity_count'] = entity_counts

# 查看统计
print(df['entity_count'].describe())
```

**输出示例：**

```
count    500.0
mean       2.3
std        1.5
min        0.0
25%        1.0
50%        2.0
75%        3.0
max        8.0
```

**解释：** – 平均每条claim有2.3个实体 – 最少0个，最多8个 – 50%的claim有2个或更少实体

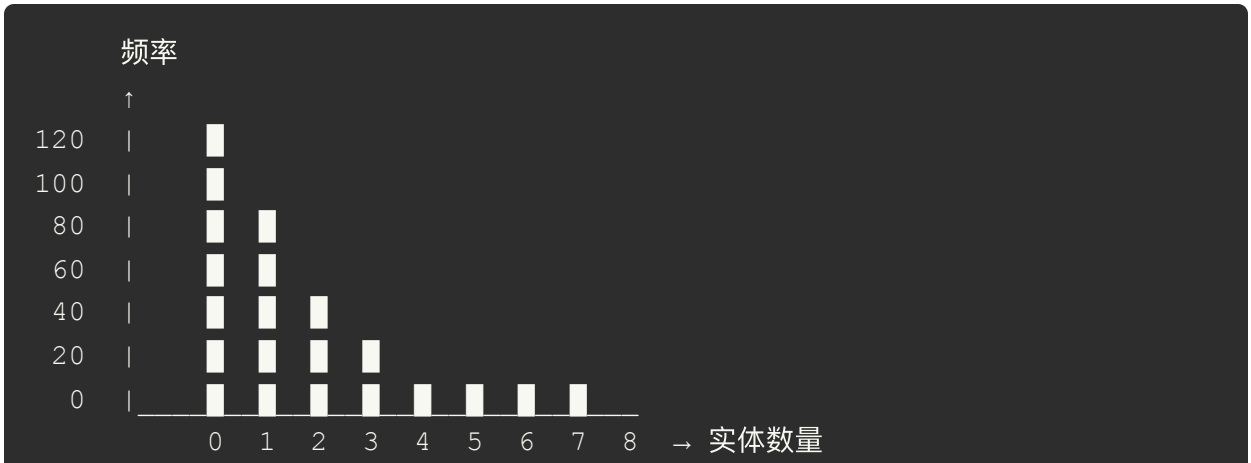
---

## 2. 绘制分布图

```
import matplotlib.pyplot as plt

# 绘制直方图
plt.figure(figsize=(10, 6))
plt.hist(df['entity_count'], bins=range(0, df['entity_count'].max()+2)
         edgecolor='black', alpha=0.7)
plt.xlabel('Number of Entities per Claim')
plt.ylabel('Frequency')
plt.title('Distribution of Entity Counts')
plt.grid(axis='y', alpha=0.3)
plt.show()
```

**图形示例：**




大部分claim有1-3个实体

---

## 3. 统计最常见的实体

```
from collections import Counter

# 收集所有实体
all_entities = []

for text in df['claim']:
    doc = nlp(text)
    for ent in doc.ents:
        all_entities.append(ent.text)

# 统计频率
entity_counter = Counter(all_entities)

# 找出最常见的20个
top_20 = entity_counter.most_common(20)

# 打印
for entity, count in top_20:
    print(f"{entity}: {count}")
```

**输出示例：**

```
United States: 45
Trump: 32
Obama: 28
Clinton: 25
2016: 20
New York: 18
Facebook: 15
Russia: 14
...
```

---

### 4. 按类型统计实体

```
# 收集实体及其类型
entity_type_counter = Counter()

for text in df['claim']:
    doc = nlp(text)
    for ent in doc.ents:
        entity_type_counter[ent.label_] += 1

# 打印统计
for ent_type, count in entity_type_counter.most_common():
    print(f"{ent_type}: {count}")
```

**输出示例：**

```
PERSON: 250
GPE: 180
DATE: 120
ORG: 95
NORP: 60
MONEY: 30
...
```

---

## 五、实体共现分析

## 1. 什么是实体共现?

**定义：**两个实体在同一条文本中出现

**例子：**

```
文本："Donald Trump met Vladimir Putin in Moscow"

实体：
- Donald Trump (PERSON)
- Vladimir Putin (PERSON)
- Moscow (GPE)

共现对：
- (Donald Trump, Vladimir Putin)
- (Donald Trump, Moscow)
- (Vladimir Putin, Moscow)
```

**意义：**

```
共现频率高 → 两个实体关系密切

例如：
"Obama" 和 "White House" 经常共现
→ Obama是前总统，在白宫工作过
```

---

## 2. 提取共现关系

```
from itertools import combinations
from collections import Counter

# 收集所有共现对
cooccurrences = Counter()

for text in df['claim']:
    doc = nlp(text)

    # 提取本文中的所有实体
    entities = [ent.text for ent in doc.ents]
```

```
    # 如果少于2个实体，跳过
    if len(entities) < 2:
        continue

    # 生成所有可能的实体对
    for ent1, ent2 in combinations(entities, 2):
        # 按字母顺序排序（确保(A,B)和(B,A)被认为是同一对）
        pair = tuple(sorted([ent1, ent2]))
        cooccurrences[pair] += 1

# 打印最常见的共现对
for (ent1, ent2), count in cooccurrences.most_common(20):
    print(f"{ent1} <-> {ent2}: {count}")
```

**详细解释：**

`combinations` **函数：**

```
from itertools import combinations

entities = ['A', 'B', 'C']

# 生成所有2个元素的组合
for pair in combinations(entities, 2):
    print(pair)

# 输出：
# ('A', 'B')
# ('A', 'C')
# ('B', 'C')

# 注意：不包括 ('A', 'A'), ('B', 'B'), ('C', 'C')
# 也不包括重复的，如 ('B', 'A')（因为已经有('A', 'B')了）
```

**为什么要排序?**

```
# 假设有两条文本：
# 文本1："Obama met Trump"  → (Obama, Trump)
# 文本2："Trump criticized Obama"  → (Trump, Obama)

# 不排序：
```

```
cooccurrences[('Obama', 'Trump')] = 1
cooccurrences[('Trump', 'Obama')] = 1
# 两个不同的键！

# 排序后：
pair1 = tuple(sorted(['Obama', 'Trump']))  # ('Obama', 'Trump')
pair2 = tuple(sorted(['Trump', 'Obama']))  # ('Obama', 'Trump')
cooccurrences[('Obama', 'Trump')] = 2
# 同一个键！
```

---

**输出示例：**

```
United States <-> Donald Trump: 25
Barack Obama <-> White House: 18
Hillary Clinton <-> Donald Trump: 15
Russia <-> Vladimir Putin: 12
Facebook <-> United States: 10
...
```

---

## 3. 可视化共现网络

**目标：**创建一个网络图，展示实体之间的关系

**使用工具：**NetworkX + Matplotlib

```
import networkx as nx
import matplotlib.pyplot as plt

# 创建图
G = nx.Graph()

# 只添加频率 >= 5 的共现对
for (ent1, ent2), count in cooccurrences.items():
    if count >= 5:
        G.add_edge(ent1, ent2, weight=count)

# 绘制网络图
plt.figure(figsize=(15, 15))
```

```
# 使用spring layout
pos = nx.spring_layout(G, k=0.5, iterations=50)

# 绘制节点
nx.draw_networkx_nodes(G, pos, node_size=500, node_color='lightblue')

# 绘制边（粗细根据权重）
edges = G.edges()
weights = [G[u][v]['weight'] for u, v in edges]
nx.draw_networkx_edges(G, pos, width=[w*0.5 for w in weights], alpha=0

# 绘制标签
nx.draw_networkx_labels(G, pos, font_size=10)

plt.title("Entity Co-occurrence Network")
plt.axis('off')
plt.tight_layout()
plt.show()
```

**图形解释：**

```
- 节点：实体
- 边：共现关系
- 边的粗细：共现频率（越粗越频繁）

例如：
    Obama ———————— White House   （粗线，频繁共现）
      |                  |
      |                  |
    Trump ————————— United States   （粗线）
      |
      |
    Putin —— Russia   （细线，较少共现）
```

---

# 六、大小写对NER的影响

## 问题

**大写：**

```
text = "BARACK OBAMA WAS BORN IN HAWAII"
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)

# 可能输出：
# BARACK OBAMA PERSON ✅ 正确
# HAWAII GPE ✅ 正确
```

**小写：**

```
text = "barack obama was born in hawaii"
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)

# 可能输出：
# （没有实体！）❌ 错误
```

---

## 为什么？

**NER模型依赖大小写特征：**

```
英语中，专有名词（人名、地名）通常首字母大写
→ 模型学习到了这个特征
→ 小写文本失去了这个重要线索
```

**例子：**

```
"I saw a turkey in Turkey"

正常大小写：
- turkey（小写）→ 可能是动物
- Turkey（大写）→ 可能是国家
```

```
全小写：
- turkey → 无法区分
```

---

### 解决方案

#### 方案1：保持原始大小写

```
# 不要预处理时就转小写
# 错误：
text = text.lower()  ❌

# 正确：
text = text  # 保持原样
```

---

#### 方案2：使用大小写不敏感的模型

```
# spaCy的某些模型对大小写不那么敏感
# 但准确率可能下降
```

---

#### 方案3：恢复大小写（如果必须要小写）

```
# 如果文本已经是小写，尝试恢复
# 这通常很困难且不准确
```

---

## 七、模型的缺陷（"Fool the Model"）

### 1. 新实体无法识别

**问题：**

```
text = "Elon Musk founded SpaceX in 2002"
doc = nlp(text)

# 如果模型训练数据是2010年之前的
# 可能无法识别"Elon Musk"和"SpaceX"（因为那时它们不出名）
```

---

## 2. 上下文歧义

### 例子1："Washington"

```
text1 = "George Washington was the first president"
# Washington → PERSON ✅

text2 = "I visited Washington last year"
# Washington → GPE ✅

text3 = "Washington approved the budget"
# Washington → ??? （可能是人名，可能是地名）
# 模型可能搞混
```

---

### 例子2："Turkey"

```
text1 = "I visited Turkey last summer"
# Turkey → GPE ✅

text2 = "I ate turkey for Thanksgiving"
# turkey → 不是实体 ✅

text3 = "Turkey is delicious"
# Turkey → GPE? 还是食物?
# 模型可能错误
```

---

## 3. 缩写和简称

```
text = "The UK voted to leave the EU"
# UK → GPE ✅
# EU → ORG ✅

# 但是：
text = "My friend lives in the uk"
# uk（小写）→ 可能无法识别 ❌
```

---

## 4. 嵌套实体

```
text = "University of California, Berkeley"
# 这是一个实体
# 但可能被识别为：
# - "University of California" (ORG)
# - "Berkeley" (GPE)
# 两个独立实体，而不是一个

# 或者完全识别为：
# - "University of California, Berkeley" (ORG) ✅ 正确
```

---

## 5. 创造性/非正式表达

```
text = "The Donald tweeted again"
# "The Donald" → 指Donald Trump
# 但模型可能无法识别

text = "Beyoncé dropped a new album"
# "Beyoncé" → 可能因为特殊字符无法识别

text = "I met J.Lo at the airport"
# "J.Lo" → Jennifer Lopez的昵称
# 模型可能无法识别
```

---

# 八、评估NER模型

## 1. 评估指标

### 精确率（Precision）：

```
精确率 = 正确识别的实体数 / 所有识别出的实体数

例如：
模型识别了100个实体
其中80个是正确的
20个是错误的（误报）

精确率 = 80 / 100 = 80%
```

---

### 召回率（Recall）：

```
召回率 = 正确识别的实体数 / 应该识别的实体数

例如：
文本中实际有120个实体
模型只识别出了80个

召回率 = 80 / 120 = 66.7%
```

---

### F1分数（F1 Score）：

```
F1 = 2 × (Precision × Recall) / (Precision + Recall)

调和平均数，综合考虑精确率和召回率

例如：
Precision = 80%
Recall = 66.7%

F1 = 2 × (0.8 × 0.667) / (0.8 + 0.667)
```

```
   = 2 × 0.534 / 1.467
   = 0.728
   = 72.8%
```

---

### 2. 混淆矩阵示例

**对于多类别NER：**

```
          预测
实际     PERSON  ORG   GPE   其他
PERSON [  80     5     2     13  ]
ORG    [  3      60    1     6   ]
GPE    [  1      2     70    7   ]
其他   [  10     8     5     177 ]

对角线：正确识别
非对角线：错误识别

例如：
- 80个PERSON被正确识别为PERSON
- 5个PERSON被错误识别为ORG
- 3个ORG被错误识别为PERSON
```

---

## 九、实际应用建议

### 1. 选择合适的方法

**使用Gazetteer，如果：**

```
1. 实体列表固定且可枚举
   例如：国家、州、城市列表

2. 准确率要求高
   例如：法律文件、医疗记录
```

```
3. 特定领域
   例如：公司名称、产品型号
```

---

**使用机器学习（spaCy），如果：**

```
1. 实体种类多样
   例如：新闻文章、社交媒体

2. 需要处理新实体
   例如：识别新公司、新人物

3. 需要考虑上下文
   例如：区分"Apple"（公司）和"apple"（水果）
```

---

## 2. 组合两种方法

**最佳实践：**

```
步骤1：用spaCy识别大部分实体
步骤2：用Gazetteer补充特定领域的实体
步骤3：合并结果

例如：
spaCy识别：人名、日期、地名等通用实体
Gazetteer补充：公司内部的产品名称、部门名称等
```

---

## 3. 预处理建议

**保持大小写：**

```
# 不要：
text = text.lower()  ❌
```

```
# 要：
text = text.strip()  ✅ 只去除首尾空格
```

---

**去除重复：**

```
import pandas as pd

df = pd.read_csv("data.csv")
df = df.drop_duplicates()  # 去除重复行
```

---

**处理缺失值：**

```
# 去除空白文本
df = df[df['text'].notna()]
df = df[df['text'].str.strip() != '']
```

---

# 十、纸笔考试重点

## 必须掌握

### 1. ✅ NER的定义和目的

```
定义：识别文本中的命名实体（人名、地名、组织名等）
目的：信息提取、知识图谱构建、问答系统等
```

---

### 2. ✅ 两种NER方法的对比

```
Gazetteer（基于规则）：
  优点：准确率高、可控
  缺点：覆盖不全、维护成本高
```

```
机器学习（神经网络）：
   优点：泛化能力强、可处理新实体
   缺点：可能出错、需要训练数据
```

---

### 3. ✅ 常见实体类型

```
PERSON：人名
GPE：地理政治实体（国家、城市等）
ORG：组织
DATE：日期
NORP：国籍/宗教/政治团体
```

---

### 4. ✅ 实体共现的概念

```
定义：两个实体在同一文本中出现
意义：发现实体之间的关系
```

---

### 5. ✅ 大小写对NER的影响

```
大写：提供重要的特征信息
小写：可能导致实体无法识别
```

---

## 可能的考题

### 题型1：识别实体

**题目：**标注以下文本中的命名实体

```
"Mike Pence once said that smoking doesn't kill people."
```

▶ 答案

---

### 题型2：实体类型判断

**题目：**判断以下实体的类型

```
1. "United Nations"
2. "January 20, 2017"
3. "British"
4. "$100"
```

▶ 答案

---

### 题型3：方法选择

**题目：**以下场景应该使用Gazetteer还是机器学习方法?

```
场景1：识别所有美国50个州的名称
场景2：识别新闻文章中的所有人名
场景3：识别公司内部的产品代号
```

▶ 答案

---

### 题型4：共现分析

**题目：**给定以下3条文本，找出所有实体共现对

```
文本1："Obama met Putin in Moscow"
文本2："Trump criticized Obama"
```

```
文本3："Putin visited Moscow"
```

▶ 答案

---

### 题型5：模型缺陷分析

**题目：**为什么以下文本可能会"欺骗"NER模型？

```
"I like turkey"
```

▶ 答案

---

## 十一、Python实现要点

### 1. spaCy基础

```
import spacy

# 加载模型
nlp = spacy.load("en_core_web_sm")

# 处理文本
doc = nlp("Your text here")

# 获取实体
entities = [(ent.text, ent.label_) for ent in doc.ents]

# 获取tokens
tokens = [token.text for token in doc]
```

---

### 2. 批量处理（提高效率）

```
# 不高效：
for text in texts:
    doc = nlp(text)  # 每次单独处理

# 高效：
docs = nlp.pipe(texts, batch_size=50)  # 批量处理
for doc in docs:
    # 处理doc
```

---

## 3. Pandas整合

```
import pandas as pd
import spacy

nlp = spacy.load("en_core_web_sm")

# 读取数据
df = pd.read_csv("data.csv")

# 应用NER
df['entities'] = df['text'].apply(lambda x: [(ent.text, ent.label_) fo

# 统计实体数量
df['entity_count'] = df['entities'].apply(len)
```

---

## 4. 可视化

```
from spacy import displacy

# 可视化NER结果（Jupyter Notebook中）
doc = nlp("Barack Obama was born in Hawaii")
displacy.render(doc, style='ent', jupyter=True)

# 输出HTML
html = displacy.render(doc, style='ent', page=True)
```

```
with open('ner_result.html', 'w') as f:
    f.write(html)
```

---

# 十二、记忆口诀

### NER基础

```
人名地名组织名，
日期货币不能忘，
实体识别找关键，
信息提取第一步。
```

### 两种方法

```
Gazetteer列表查，
准确但要常更新，
spaCy神经网络强，
新词也能猜一猜。
```

### 共现分析

```
同文共现有联系，
频率高则关系密，
网络图来可视化，
一眼看清谁和谁。
```

---

# 十三、常见错误

### ❌ 错误1：预处理时转小写

```
# 错误
text = "Barack Obama"
```

```
text = text.lower()  # "barack obama"
doc = nlp(text)
# 可能无法识别实体 ❌

# 正确
text = "Barack Obama"
doc = nlp(text)  # 保持大小写
# 正确识别 ✅
```

---

## ❌ 错误2：忘记去重

```
# 错误
df = pd.read_csv("data.csv")
# 不去重，浪费时间处理重复数据 ❌

# 正确
df = pd.read_csv("data.csv")
df = df.drop_duplicates(subset=['text'])  ✅
```

---

## ❌ 错误3：共现对不排序

```
# 错误
cooc[('A', 'B')] += 1
cooc[('B', 'A')] += 1
# 两个不同的键 ❌

# 正确
pair = tuple(sorted(['A', 'B']))
cooc[pair] += 1  # 总是 ('A', 'B')
pair = tuple(sorted(['B', 'A']))
cooc[pair] += 1  # 也是 ('A', 'B') ✅
```

---

## ❌ 错误4：单独处理而不批量

```
# 错误（慢）
for text in texts:
    doc = nlp(text)  ❌

# 正确（快）
for doc in nlp.pipe(texts):
    # 处理  ✅
```

---

**祝你考试顺利！** 🎉

**Lecture 7核心要点：** – NER：识别文本中的命名实体 – Gazetteer：基于列表匹配，准确但覆盖有限 – spaCy：基于神经网络，泛化能力强 – 实体共现：分析实体之间的关系 – 大小写很重要：影响识别准确率 – 常见实体类型：PERSON, GPE, ORG, DATE, NORP等

**记住：**理解两种方法的优缺点，知道何时使用哪种！会计算实体共现！