# NER 调研

[命名实体识别](https://so.csdn.net/so/search?q=%E5%91%BD%E5%90%8D%E5%AE%9E%E4%BD%93%E8%AF%86%E5%88%AB&spm=1001.2101.3001.7020)（Named Entity Recognition， NER）是指在文本中识别出特殊对象，如人、地点、组织机构等。

## NER实现方法

基于规则的方法：利用专家手工制订的规则进行命名实体识别。举例：“赵某出生于山东省菏泽市曹县，于 11 月 22 日将刘某诉至菏泽市曹县人民法院”，构建规则，满足“地名+人民法院”的词认定为组织机构。

基于传统机器学习的方法：基于传统机器学习的方法又可分为有监督和无监督的方式。有监督的方法将 NER 转换为多分类或序列标记任务。根据标注好的数据，人工构建特征工程，然后应用机器学习算法训练模型使其对数据的模式进行学习。例如隐马尔可夫模型（HMM）、支持向量机（SVM）和条件随机场（CRF）等。

基于深度学习的方法：以端到端的方式自动检测对应输入语料中的实体类别，通过深度学习的方式自动发现隐藏的特征，抽取与实体相对应的语义信息，是现在主流的做法。

本次调研主要是在深度学习的基础上，调研开源的软件及工具。原因是基于规则的方法需要大量的工程实践来做规则处理，是针对具体问题具体分析的方法；而传统的机器学习方法在NLP领域已经被明证不如深度学习的方法。

## 调研方向

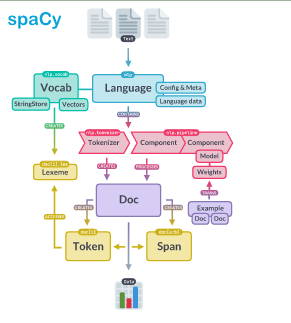
为试验深度学习的效果，调研了多个工具，包含国内外工业界和学术界的开源工具，其中一些采用了谷歌的BERT模型，另外一些采用了transformers模型。这些工具都提供了一个或多个训练好的模型，直接通过调用相关模型就可以进行命名实体数据的识别。

### Spacy实体识别

spaCy是一个用于Python和Cython中高级自然语言处理的库。它建立在最新研究的基础上，设计用于实际产品的工业界开源库。

spaCy带有预训练的管道，目前支持70多种语言的标记化和训练。它具有最先进的速度和神经网络模型，用于标记、解析、命名实体识别、文本分类等，使用BERT等预训练转换器进行多任务学习，以及生产就绪的培训系统和简单的模型打包、部署和工作流管理。spaCy是商业开源软件，在麻省理工学院的许可下发布。

Spacy的架构如下图，详细的内容可参考官方文档。



1. 相关文档

<https://spacy.io/api/entityrecognizer> spacy实体识别文档

<https://spacy.io/usage/spacy-101> spacy 快速开始

<https://spacy.io/usage> spacy 安装文档

1. 快速开始

|  |
| --- |
| 环境准备:  Python安装spacy  pip install -U pip setuptools wheel  pip install -U spacy  python -m spacy download zh\_core\_web\_sm  python -m spacy download zh\_core\_web\_trf  python -m spacy download zh\_core\_web\_lg |
| import spacy  def read\_file(file\_path):  with open(file\_path, 'r', encoding='utf-8') as file:  return file.read()  if \_\_name\_\_ == '\_\_main\_\_':  l = ['EVENT', 'ORG', 'PERSON', 'LANGUAGE', 'LAW', 'LOC', 'PRODUCT', 'WORK\_OF\_ART']  nlp = spacy.load("zh\_core\_web\_trf")*# 加载对应的模型*  text = read\_file("D:/dataner/sample4.txt")  doc = nlp(text)  seen = set()  for ent in doc.ents:  if ent.label\_ in l:  a = (ent.text, ent.label\_)  if a not in seen:  seen.add(a)  print(ent.text, ent.start\_char, ent.end\_char, ent.label\_)  print(seen) |

1. 模型及标注

采用模型，比较两者处理效果：

* zh\_core\_web\_trf 深度神经网络模型 Bert
* zh\_core\_web\_lg

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **标签** | **含义** | **标签** | **含义** | **标签** | **含义** |
| ORG | 机构名 | PERSON | 人 | EVENT | 事件 |
| WORK\_OF\_ART | 艺术工作 | LOC | 地区 | LANGUAGE | 语言 |

1. 环境要求

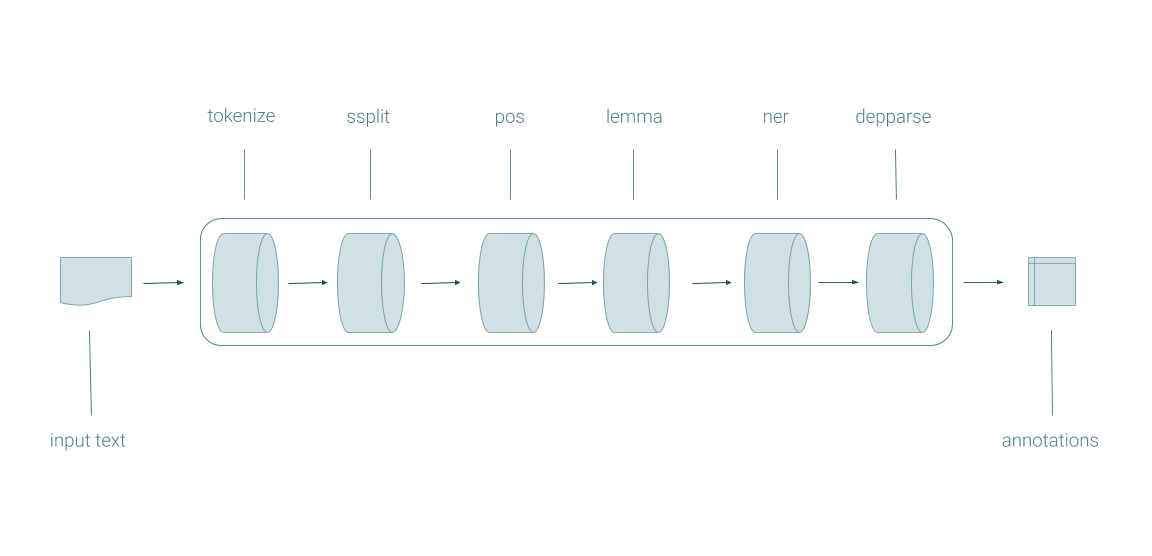
基于BERT的transformers实现，实测在8G内存电脑可以运行。

### Stanfordnlp

斯坦福CoreNLP提供了一套用Java编写的自然语言分析工具。它可以接受原始的人类语言文本输入，给出单词的基本形式、词性、是否是公司名称、人名等，规范和解释日期、时间和数字量，标记句子结构根据句法短语或依赖关系，并指示哪些名词短语指代相同的实体。它为阿拉伯语、中文、法语、德语、匈牙利语、意大利语和西班牙语提供不同级别的支持。

斯坦福CoreNLP是一个集成框架，可以非常轻松地将一堆语言分析工具应用于一段文本。从纯文本开始，其分析为更高级别和特定领域的文本理解应用程序提供了基础构建块。CoreNLP 是一套稳定且经过充分测试的自然语言处理工具，被学术界、工业界和政府的各个团体广泛使用。这些工具分别使用基于规则的概率机器学习和深度学习组件。

CoreNLP 的核心是管道，管道接收文本信息，在文本上运行一系列NLP执行器，并生成最终的结果。



1. 线上参考文档

https://stanfordnlp.github.io/CoreNLP/ CoreNLP官网

1. 构建运行项目，打包

|  |
| --- |
| 环境准备:  JAVA 1.8+环境  Maven环境安装 |
| <dependency>  <groupId>edu.stanford.nlp</groupId>  <artifactId>stanford-corenlp</artifactId>  <version>4.5.5</version> </dependency> <dependency>  <groupId>edu.stanford.nlp</groupId>  <artifactId>stanford-corenlp</artifactId>  <version>4.5.5</version>  <classifier>models</classifier> </dependency> <dependency>  <groupId>edu.stanford.nlp</groupId>  <artifactId>stanford-corenlp</artifactId>  <version>4.5.5</version>  <classifier>models-chinese</classifier> </dependency> |
| public static void pipeline(String[] args) {  *// set up pipeline properties* Properties props = new Properties();  props.setProperty("annotators", "tokenize,pos,lemma,ner"); *// set up pipeline* StanfordCoreNLP pipeline = new StanfordCoreNLP(props);  *// make an example document* CoreDocument doc = new CoreDocument("Joe Smith is from Seattle.");  *// annotate the document* pipeline.annotate(doc);  *// view results* System.*out*.println("---");  System.*out*.println("entities found");  for (CoreEntityMention em : doc.entityMentions())  System.*out*.println("\tdetected entity: \t"+em.text()+"\t"+em.entityType());  System.*out*.println("---");  System.*out*.println("tokens and ner tags");  String tokensAndNERTags = doc.tokens().stream().map(token -> "("+token.word()+","+token.ner()+")").collect(  Collectors.*joining*(" "));  System.*out*.println(tokensAndNERTags); } |

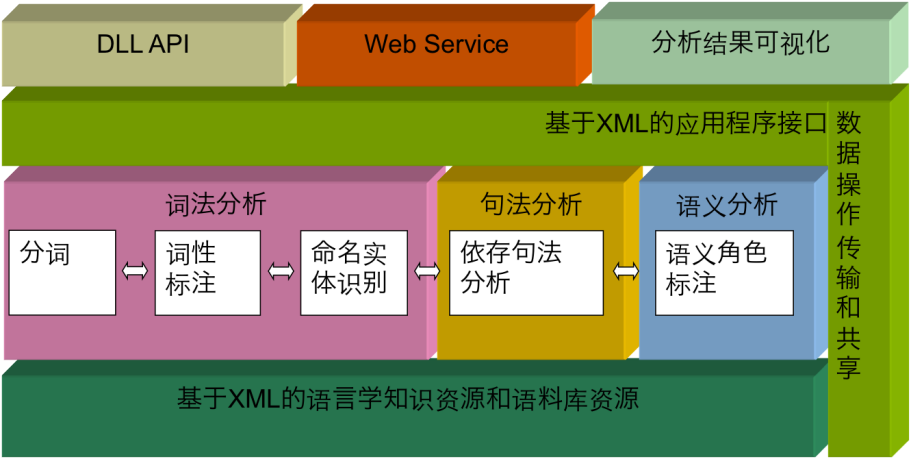
1. 标注类型

|  |  |  |
| --- | --- | --- |
| **标记1** | **标记2** | **含义** |
| P | PERSON | 人名 |
| L | LOCATION | 地区 |
| O | ORGANIZATION | 机构名 |
| C | COUNTRY | 国家 |
| CI | CITY | 城市 |
| SP | STATE\_OR\_PROVINCE | 州、省 |

1. 环境要求

斯坦福大学的corenlp识别，采用java语言开发，对内存的要求比较高，实地采用8G内存的电脑，会OOM，最后采用服务器运行程序。

### 哈工大LTP



1. 线上可用文档

<http://ltp.ai/index.html> 哈工大LTP官方网址，提供在线演示、模型下载等功能。

https://github.com/HIT-SCIR 哈工大代码在github上的地址。

<http://www.ltp-cloud.com/intro> LTP云服务地址

1. 快速使用

|  |
| --- |
| 环境准备:  Python安装LTP  下载对应的模型文件 |
| from ltp import LTP ltp = LTP("LTP/small") *# 默认加载 Small 模型* output = ltp.pipeline(["他叫汤姆去拿外衣。"], tasks=["cws", "pos", "ner", "srl", "dep", "sdp"]) *# 使用字典格式作为返回结果* print(output.cws) *# print(output[0]) / print(output['cws']) # 也可以使用下标访问* print(output.pos) print(output.sdp) |
| *# 使用感知机算法实现的分词、词性和命名实体识别，速度比较快，但是精度略低* ltp = LTP("LTP/legacy") *# cws, pos, ner = ltp.pipeline(["他叫汤姆去拿外衣。"], tasks=["cws", "ner"]).to\_tuple() # error: NER 需要 词性标注任务的结果* cws, pos, ner = ltp.pipeline(["他叫汤姆去拿外衣。"], tasks=["cws", "pos", "ner"]).to\_tuple() *# to tuple 可以自动转换为元组格式 # 使用元组格式作为返回结果* print(cws, pos, ner) |

1. 模型及任务

|  |  |
| --- | --- |
| **模型** | **任务** |
| cws | 分词 |
| pos | 词性标注 |
| ner | 命名实体识别 |
| srl | 语义角色标注 |
| dep | 依存语法分析 |
| sdp | 依存语法分析树 |

命名实体识别（Named Entity Recognition，简称NER），又称作“专名识别”，是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。命名实体识别是信息提取、问答系统、句法分析、机器翻译、面向Semantic Web的元数据标注等应用领域的重要基础工具，在自然语言处理技术走向实用化的过程中占有重要地位。

在哈工大Pyltp中，NE识别模块的标注结果采用O-S-B-I-E标注形式，其含义如下（参考）：

|  |  |
| --- | --- |
| **标记** | **含义** |
| O | 这个词不是NE |
| S | 单独构成一个NE |
| B | 构成一个NE的开始 |
| I | 构成一个NE的中间 |
| E | 构成一个NE的结尾 |
| **标记** | **含义** |
| nh | 人名 |
| ni | 机构名 |
| ns | 地名 |

1. 其他参考官方文档

<https://pyltp.readthedocs.io/zh-cn/latest/api.html>

1. 环境要求

模型参数较少，基于 pytorch与transformers实现，实测8G内存电脑可以运行。

### Jieba分词

“结巴”中文分词：做最好的 Python 中文分词组件。Jieba实体识别有默认模式和paddle模式，paddle模式采用百度的AI库训练，这里测试实体识别采用paddle模式进行处理。

1. Jieba分词地址

<https://github.com/fxsjy/jieba> jieba分词地址

<https://www.paddlepaddle.org.cn/documentation/docs/zh/install/index_cn.htm> PaddlePaddle安装指南

1. 快速使用

|  |
| --- |
| 环境准备:  Python安装Jieba  安装PaddlePaddle工具 |
| 1. PaddlePaddle仅支持64位的操作系统，查询命令python -c "import platform;print(platform.architecture()[0]);print(platform.machine())" 2. PaddlePaddle 有对CPU和GPU的支持，查询命令python -m pip install paddlepaddle==2.5.0 -i https://mirror.baidu.com/pypi/simple |
| import jieba import jieba.posseg as pseg if \_\_name\_\_ == '\_\_main\_\_':  *# words = pseg.cut("我爱北京天安门") #jieba默认模式* jieba.enable\_paddle() *#启动paddle模式。 0.40版之后开始支持，早期版本不支持* words = pseg.cut("他叫汤姆去拿外衣。",use\_paddle=True) *#paddle模式* for word, flag in words:  print('%s %s' % (word, flag)) |

1. 词性标注

|  |  |
| --- | --- |
| **标记** | **含义** |
| nr | 人名 |
| nt | 机构名 |
| ns | 地名 |
| nw | 作品名 |
| PER | 人名 |
| LOC | 机构名 |
| ORG | 地名 |

1. 环境要求

基于百度的PaddlePaddle，对性能要求不高，大部分主流电脑都可运行，实测8G内存电脑可以运行。

### HanLP分词

HanLP是一款自然语言处理工具包，由中国人民大学自然语言处理与人文计算研究所开发。它提供了丰富的文本处理功能，包括分词、词性标注、命名实体识别等。HanLP使用Java语言实现，并提供了Python、C++等多种编程语言的接口。

HanLP官网提供了多个分词模型，能够在特定领域有更好的效果。

1. 地址

<https://hanlp.hankcs.com/demos/ner.html> hanlp在线测试地址，商业接口

<https://www.hanlp.com/index.html> 官网

<https://hanlp.hankcs.com/docs/tutorial.html> 手册

1. 快速开始

HanLP有商业版和开源模型版，这里分开介绍。商业版可以通过官网进行测试，也可以注册账号申请免费的token调用，但是在数量和频率上有限制，HanLP在接口调用上提供了多语言版本，为方便期间本文测试采用的Python语言版。本地部署版是HanLP整理开源的各种模型，需要在本地通过管道的方式，单任务执行。MSRA是最大的中文命名实体识别语料库，考虑可能在MSRA训练出来的模型在准确度有更好的效果，本文档的测试也基于MSRA预料库的模型，包含tok和ner两种。需要准备环境如下：

|  |
| --- |
| 环境准备:  Python安装HanLp(本地部署)  Python安装hanlp\_restful(商业版调用)  pip install hanlp[full] -U  配置环境变量 |
| 模型参数文件:   1. msra\_crf\_electra\_base\_20220507\_113936 359Mb hanlp.pretrained.tok.MSR\_TOK\_ELECTRA\_BASE\_CRF 2. msra\_ner\_albert\_base\_20211228\_173323 37.5Mb   hanlp.pretrained.ner.MSRA\_NER\_ALBERT\_BASE\_ZH 3 entity   1. msra\_ner\_electra\_small\_20220215\_205503 44.9Mb   hanlp.pretrained.ner.MSRA\_NER\_ELECTRA\_SMALL\_ZH 26 entity   1. ner\_bert\_base\_msra\_20211227\_114712 362Mb   hanlp.pretrained.ner.MSRA\_NER\_BERT\_BASE\_ZH 3 entity |
| from hanlp\_restful import HanLPClient if \_\_name\_\_ == '\_\_main\_\_':  HanLP = HanLPClient('https://www.hanlp.com/api', auth=None, language='zh') *# auth不填则匿名，zh中文，mul多语种* print(HanLP('2021年HanLPv2.1为生产环境带来次世代最先进的多语种NLP技术。阿婆主来到北京立方庭参观自然语义科技公司。', tasks='ner\*')) |
|  |

1. 标注类型

|  |  |
| --- | --- |
| **标记** | **含义** |
| PER | 人名 |
| LOC | 机构名 |
| ORG | 地名 |

1. 环境要求

本地部署运行要求，本地部署因为内存机器原因失败。HanLP依赖于tensorflow框架。

### 大模型GPT分词

大模型分词通常使用的是预训练生成式模型，以ChatGPT效果最强，我们此处的测试并未采用OpenAi的ChatGPT，而是以国内的一款类似于ChatGPT的商业工具ChatGo（工具是收费的，目的是为了对比识别效果），ChatGo以OpenAi开源的GPT3.5为基础训练的模型。

1. 地址

<https://chatgo.pro/web> ChatGo Web地址

1. Prompt提示词

|  |
| --- |
| **Prompt** |
| 1. 请找到上述文字中设计到的企业、人员和组织，并按照类别返回。 |

# NER 结果实测

测试结果详情查看附属文档- <NER样本实测.pdf>。