



華中師範大學

自然语言处理

主讲教师：曾江峰

华中师范大学 信息管理学院

jfzeng@ccnu.edu.cn

考评规则

❖ 平时成绩 (60%)

考勤10% (缺勤1次扣3分，缺勤4次10分全部扣除，缺勤5次及以上，书面作业成绩按照1次缺勤扣除3分来惩罚；事假1次扣除1分，病假不扣分)

课堂作业50% (5-6次作业，需提交作业报告，暂定！)

❖ 期末成绩 (40%) If (平时成绩<45) or (期末成绩<45)

课程论文40%

{

 总成绩视为不通过；
 重修；

}

推荐教材

- ❖ 《自然语言处理入门》，何晗著，中国工信出版社，2019年
- ❖ 《Python自然语言处理》，周元哲著，清华大学出版社，2021年

自然语言处理

- 1 自然语言处理概述
- 2 语料清洗
- 3 特征工程
- 4 评价指标
- 5 中文分词
- 6 文本分类
- 7 文本聚类
- 8 序列标注
- 9 Word2vec

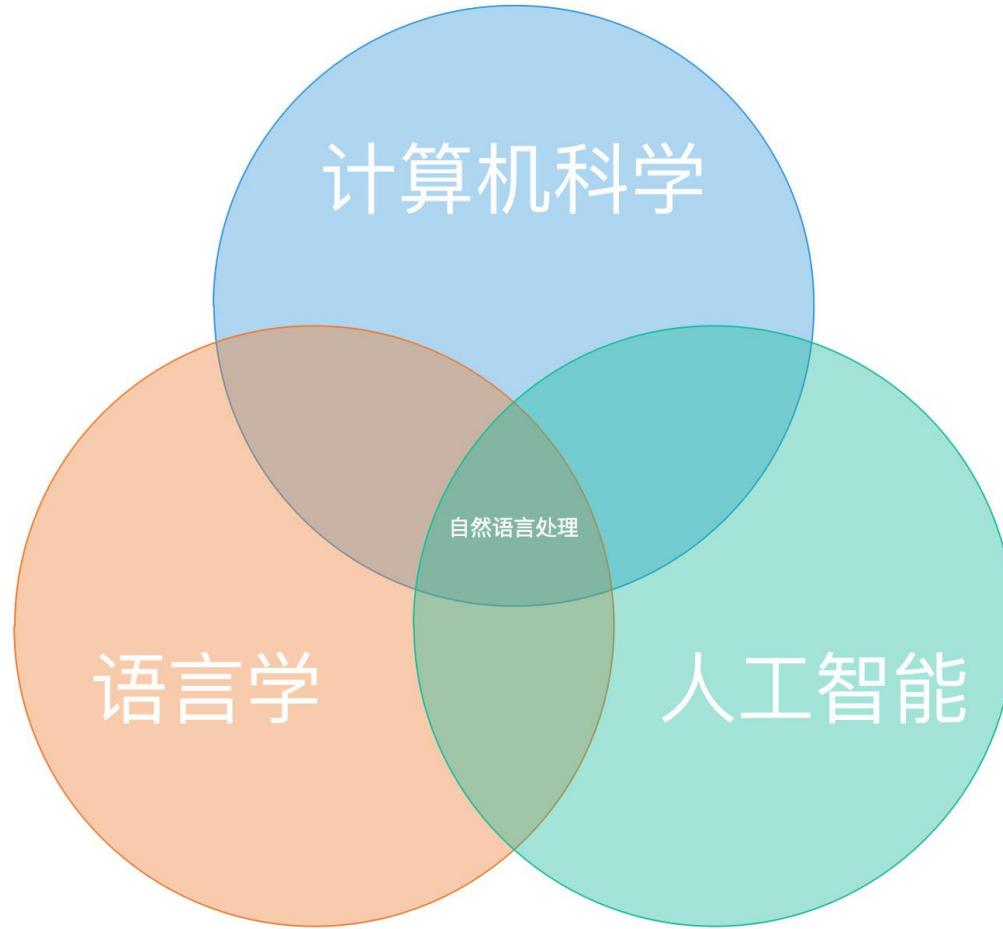


第一章 自然语言处理概述

- 1. 1 什么是自然语言处理
- 1. 2 自然语言与编程语言
- 1. 3 自然语言处理的层次
- 1. 4 自然语言处理的流派
- 1. 5 自然语言处理相关库
- 1. 6 NLP开发环境搭建

1. 1 什么是自然语言处理

- 计算机科学
- 人工智能
- 语言学



[《自然语言处理入门》](#)

自然语言处理NLP
= 计算语言学CL
= 自然语言理解NLU
+ 自然语言生成NLG



1. 1 什么是自然语言处理

猪立业之算法与科普 

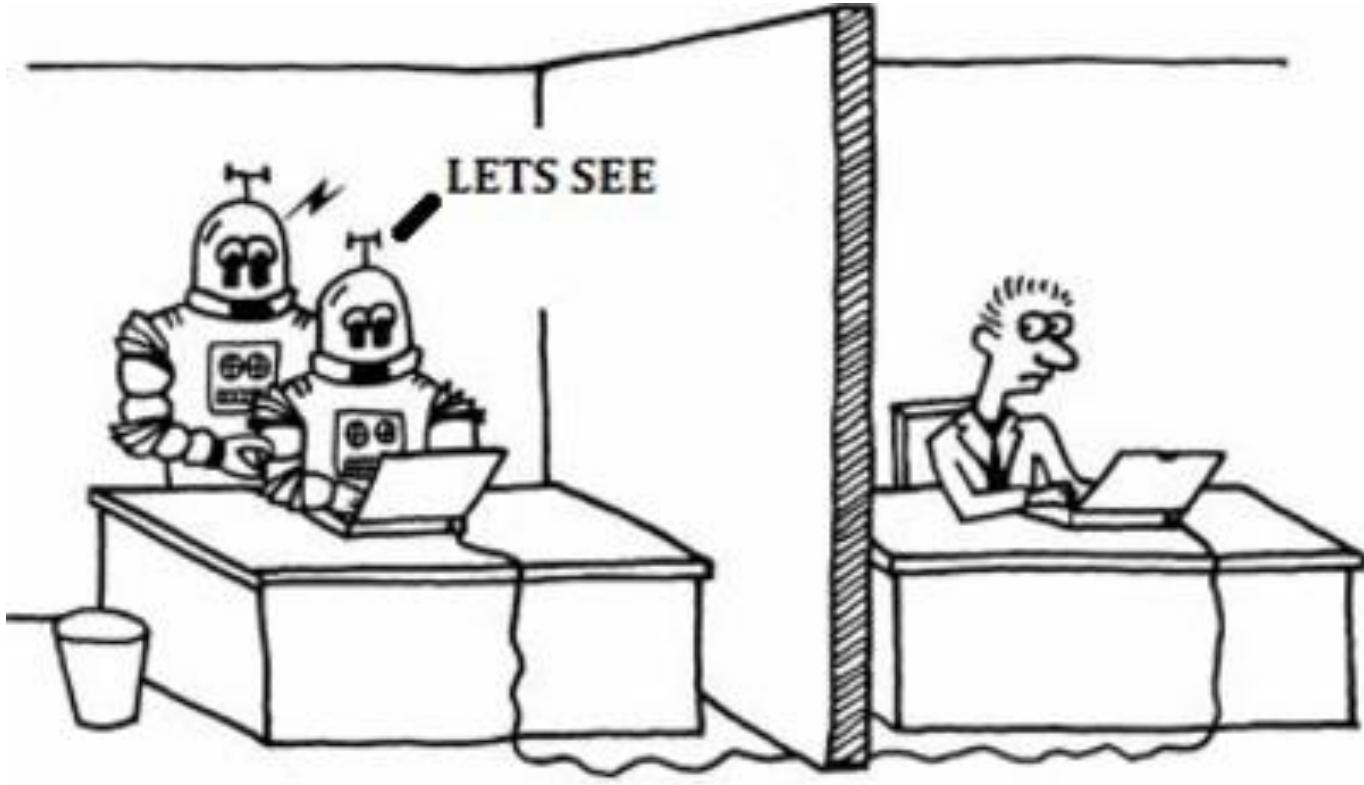
自然语言处理是人工智能皇冠上的明珠。

—— 比尔盖茨



自然语言处理与图灵测试

- 避免物理接触
- 自然语言文本交流
- 测试某机器是否能表现出与人等价或无法区分的智能



图灵测试与中文屋子

- 是否存在一个规则手册满足
 - 包罗万象
 - 输入通过规则映射到输出
 - 如果存在，则图灵测试不充分
 - 你的观点？

Chinese room argument

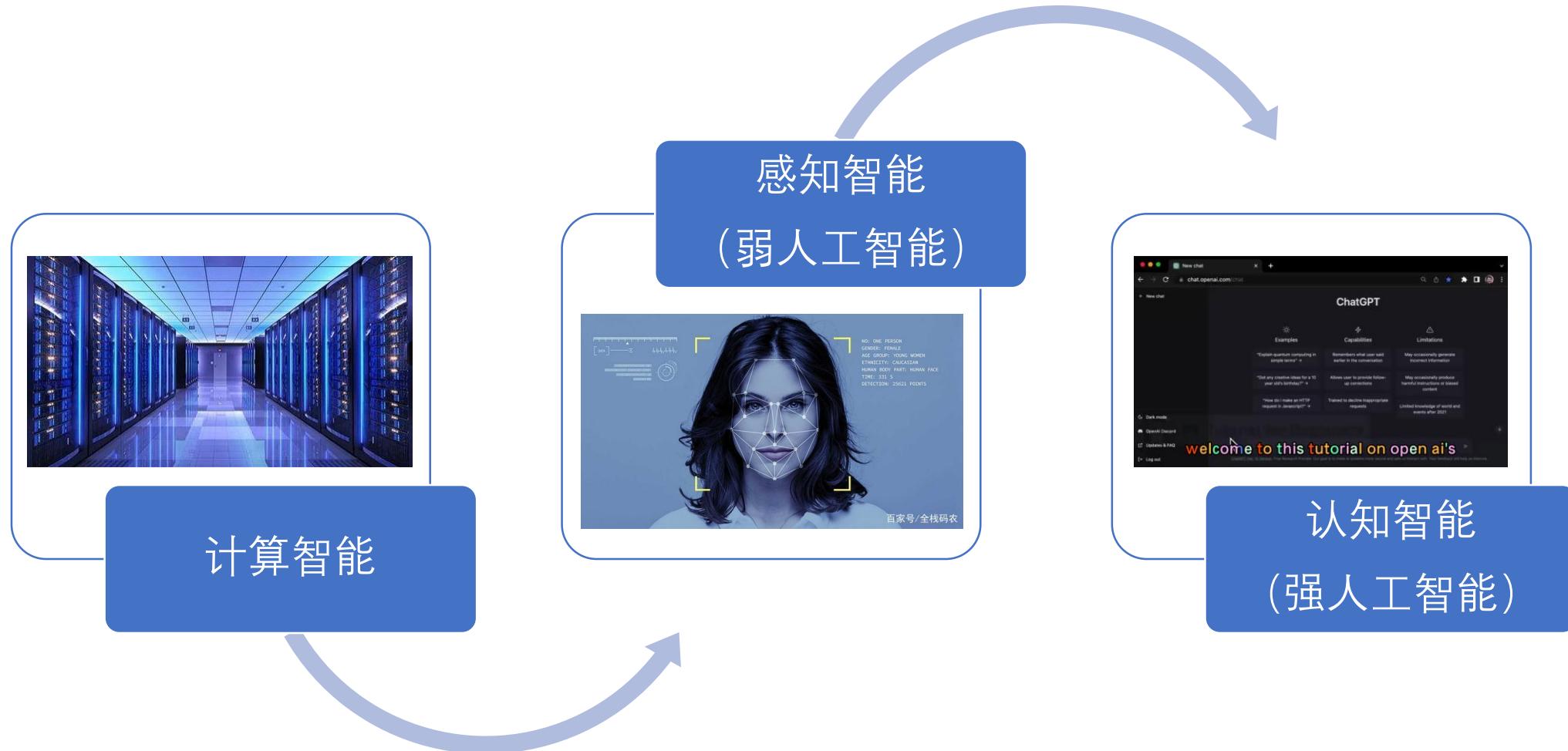


自然语言处理的难

- 这个星球上有许多生物拥有超过人类的视觉系统、听觉系统、嗅觉系统，但只有人类才拥有这么高级的语言。



自然语言处理的难



1.2 自然语言与编程语言

自然语言中的词汇比编程语言中的关键词丰富。

- 词汇量
 - C语言32个关键字
 - 汉语多少个?
 - 10万?
 - 100万?
 - 无限?
 - 宇宙中有多少天体
 - 无限, 至少数不清
 - 它们的名字独一无二
 - 所以?



1.2 自然语言与编程语言

- 结构化

自然语言是非结构化的，而编程语言是结构化的。

```
class Company(object):
    def __init__(self, founder, logo) -> None:
        self.founder = founder
        self.logo = logo
```

```
apple = Company(founder='乔布斯', logo='apple')
```

苹果 的 创始人 是 乔布斯 , 它 的 logo 是 



1.2 自然语言与编程语言

- 歧义性

自然语言含有大量歧义，这些歧义根据语境的不同而表现为特定的含义。
但在编程语言中，则不存在歧义性。

他说：“她这个人真有意思（**funny**）。”她说：“他这个人怪有意思的（**funny**）。”于是人们以为他们有了意思（**wish**），并让他向她意思意思（**express**）。他火了：“我根本没有那个意思（**thought**）！”她也生气了：“你们这么说是什么意思（**intention**）？”事后有人说：“真有意思（**funny**）。”也有人说：“真没意思（**nonsense**）”。（原文见《生活报》1994.11.13.第六版）〔吴尉天，1999〕

摘自宗成庆《统计自然语言处理》。



1.2 自然语言与编程语言

- 容错性

互联网上的文本更加随意，错别字或病句、不规范的标点符号等随处可见。

常见错误类型

用词错误

- 乾隆曾经除了一个上联，难道无数大臣
- 《王者荣耀》地方黄忠死守水晶到底怎么办？某玩家的答案真是亮眼了！

特点：形近，音近

文/句法错误

- 航拍黔西南深山农村，住在与世隔绝的两人家户，真像世外桃源！
- 去参加开宝马的初中同学取婚礼，我骑这个车不是有点掉价？！

特点：多/少字，乱序

知识错误

- 爸爸去哪儿5：讨论下邓论是否有点作，参加节目似演戏？
- 民警惊讶地发现，喝过酒的江某竟然开起了自己的试驾车！

特点：上下文知识，常识



1.2 自然语言与编程语言

- 易变性

自然语言不是由某个个人或组织发明或制定标准的。每个人都可以自由创造和传播新词汇和新用法，也在不停地赋予旧词汇以新含义，导致古汉语和现代汉语相差巨大。这也是自然语言明明是人类发明的，却还要称作“自然”的原因。

TOP DEFINITION



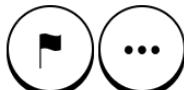
niubility

It's a [Chinglish](#). In Chinese **Niu** means cow, which also means that someone is very capable. **Bi**(pronunciation [Bee](#)), which is used to refer a person rudely, means pussy or formally genital.

Many people think they are full of niubility, and like to play [zhuangbility](#), which only reflect their [shability](#).

#niubility #bi #zhuangbility #shability #niu

by [Luo Mengyu](#) December 03, 2007



Get a **niubility** mug for your dog Jovana.



1.2 自然语言与编程语言

人类语言往往简洁、干练，经常省略大量背景知识或常识。

- 简略性
 - 暗语
 - 老地方见
 - 基本常识 (common sense)
 - 简称
 - 工行
 - 地税局

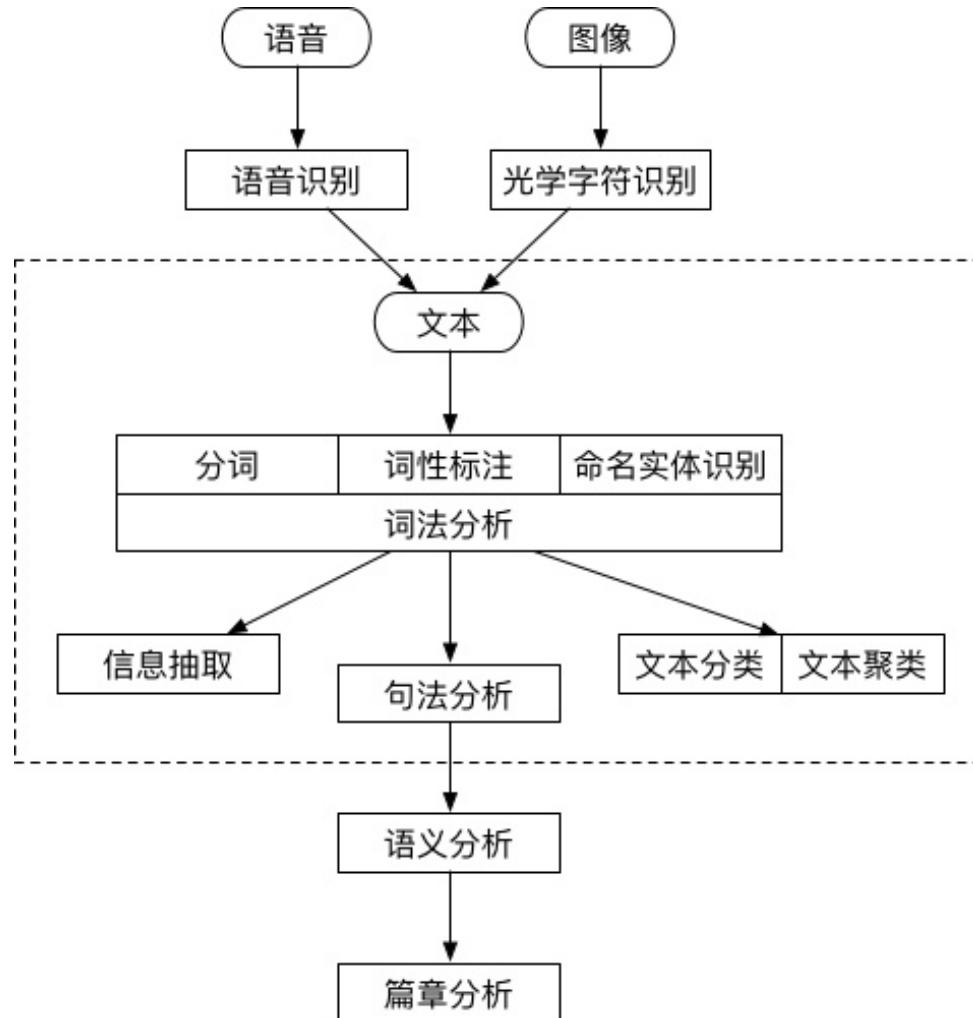


**老地方见，老地方见
Over Over**

1.2 自然语言与编程语言

比较	不同	例子
词汇量	自然语言中的词汇比编程语言中的关键词丰富，我们还可以随时创造各种类型的新词	蓝瘦、香菇
结构化	自然语言是非结构化的，而编程语言是结构化的	
歧义性	自然语言含有大量歧义，而编程语言是确定性的	这人真有意思：没意思
容错性	自然语言错误随处可见，而编程语言错误会导致编译不通过	的、地的用法错误
易变性	自然语言变化相对迅速嘈杂一些，而编程语言的变化要缓慢得多	新时代词汇
简略性	自然语言往往简洁、干练，而编程语言就要明确定义	“老地方”不必指出

1.3 自然语言处理的层次



1.3 自然语言处理的层次

- 输入层

- 语音
 - 语音识别
- 图像
 - 光学字符识别
- 文本

ASR
Automated Speech Recognition



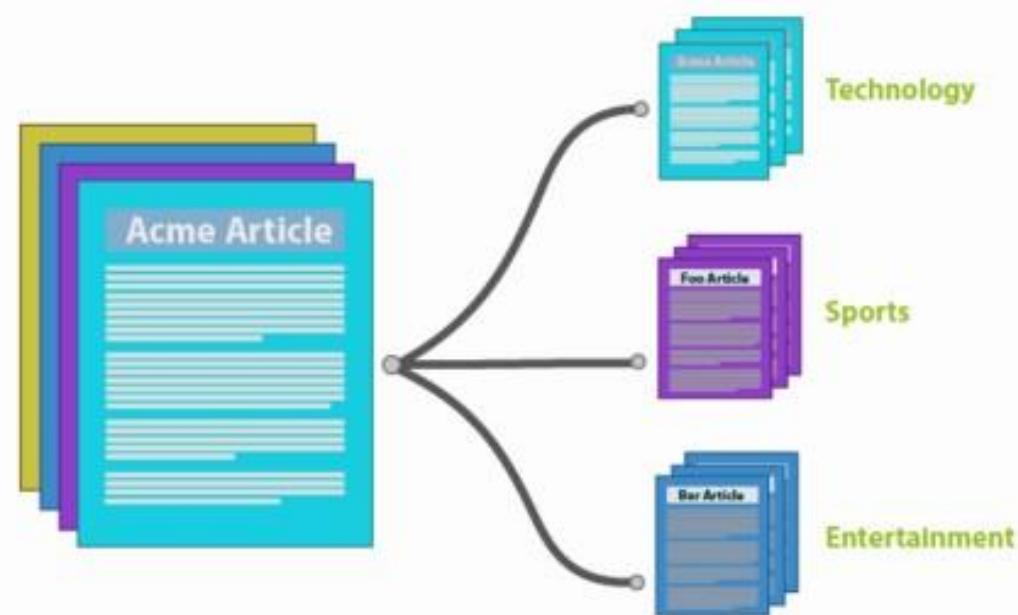
1.3 自然语言处理的层次

- 词法分析
 - 中文分词
 - 词性标注
 - 命名实体识别
- 信息抽取
 - 提取文本中部分有用的信息
 - 关键词
 - 实体抽取
 - 关系抽取
 - 发生在词法分析的基础上



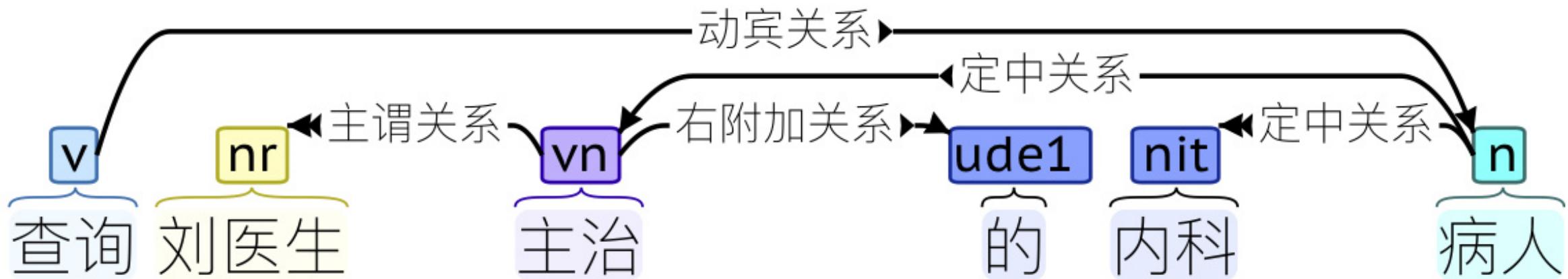
1.3 自然语言处理的层次

- 文档层
 - 文本分类
 - 正常邮件/垃圾邮件
 - 褒义/贬义
 - 文本聚类
 - 相似文档归档
 - 搜索结果中的类似结果



1.3 自然语言处理的层次

- 句法分析
 - 下图到底要查询什么？

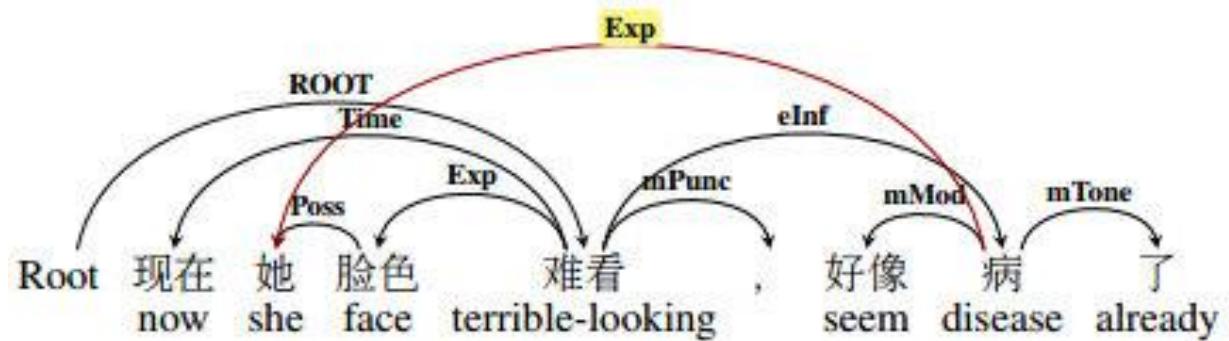


词法分析只能得到零散的词汇信息，通过句法分析可以发现词语之间的关系。
不仅是问答系统或搜索引擎，句法分析还经常用于基于短语的机器翻译，给译文的词语重新排序。



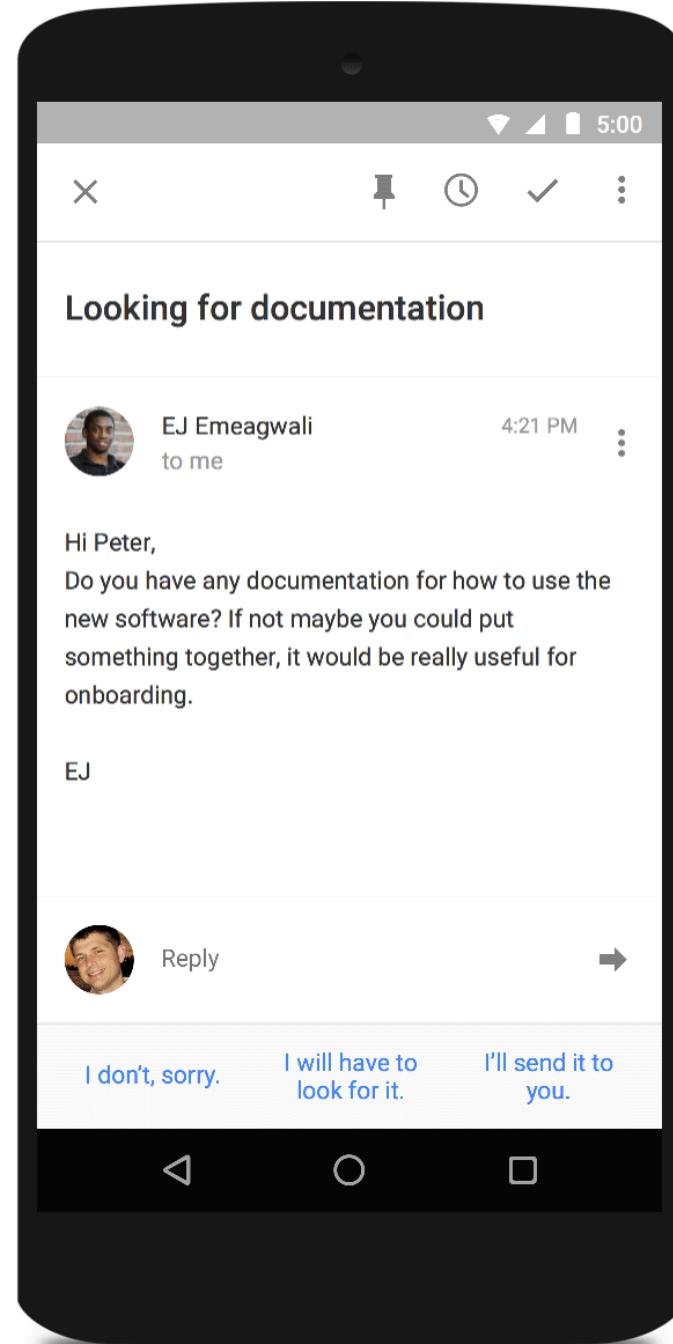
1.3 自然语言处理的层次

- 语义分析与篇章分析
 - 词义消歧
 - 语义角色标注
 - 语义依存分析
 - 指代消解等



1.3 自然语言处理的层次

- 其他高级任务
 - 自动问答
 - 自动摘要
 - 机器翻译
 - 自然场景下的文字识别
 - 对话系统
 -



机器翻译



对话系统



文本生成



《自然语言处理入门》

自然场景下文字识别



1.4 自然语言处理的流派



自然语言的发展历程

专家知识仍然有用武之地，可以根据语言学知识为统计模型设计特征模板（将语料表示为计算机理解的形式），这道工序被称为“特征工程”。



深度学习的魅力在于，它不再依赖专家制定的特征模板，而能够自动学习原始数据的抽象表示，所以它主要用于表示学习。



1.4.1 基于规则的专家系统

规则指由专家手工制定的确定性流程。
小到程序员日常使用的正则表达式，大到飞机的自动驾驶仪，都是固定的规则系统。

编号	如果后缀为	并且	则将后缀替换为	例子
1	eed	辅音+元音同时出现	ee	feed->feed agreed->agree
2	ed	含有辅音	空白	plastered->plaster bled->bled
3	ing	含有辅音	空白	eating->eat sing->sing

表1-1 波特词干算法规则集(部分)

专家系统要求设计者对所处理的问题具备深入的理解，并且尽量以人力全面考虑所有可能的情况。
它最大的弱点是难以拓展。

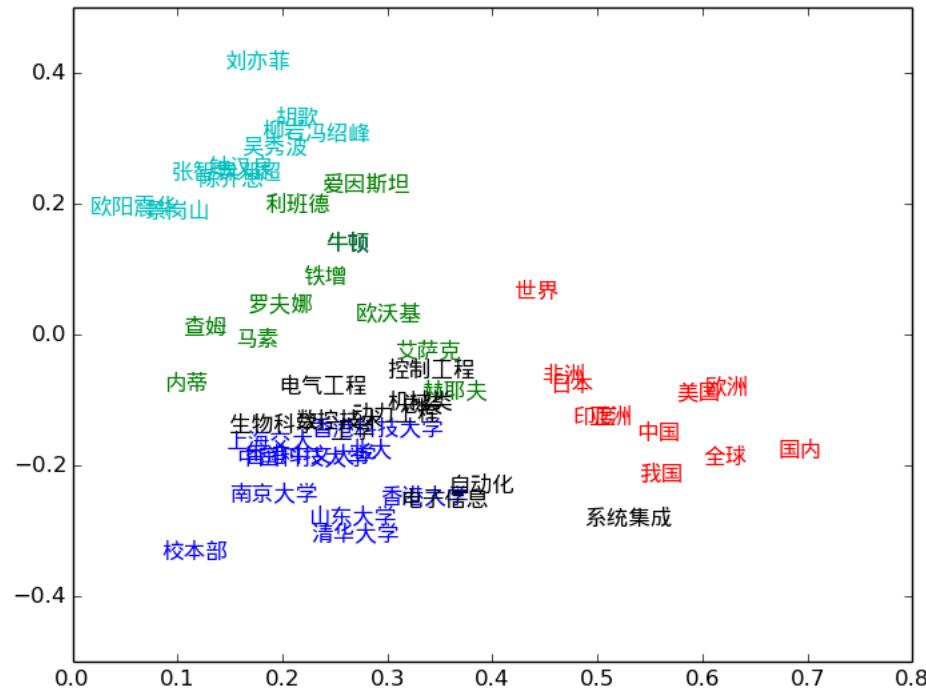
大多语言现象没有必然遵循的规则，也在时刻变化，使得规则系统显得僵硬、死板与不稳定。



1.4.2 基于统计的学习方法

- 以举例子的方式让机器自动学习语言规律

为了降低对专家的依赖，自适应灵活的语言问题，人们使用统计方法让计算机自动学习语言。所谓“统计”指的是在语料库上进行的统计。语料库指人工标注的结构化文本。



规则与统计

随着机器学习的日渐成熟，领域专家的作用越来越小。实际工程中，语言学知识的作用有两方面：一是帮助我们设计更简洁、高效的特征模板，二是在语料库建设中发挥作用。

“以统计为主、规则为辅”的方式成为NLP的主流。

**Every time I fire a linguist,
the performance of our
speech recognition system
goes up.**

Fred Jelinek

WWW.STOREMYPIC.COM



机器学习

机器学习是从人工智能中产生的一个重要学科分支，是实现智能化的关键。

经典定义：利用经验改善系统自身的性能



经验 → 数据



随着该领域的发展，目前主要研究**智能数据分析**的理论和算法，并已成为智能数据分析技术的源泉之一

图灵奖连续授予在该方面取得突出成就的学者



Leslie Valiant
(1949 -)
(Harvard Univ.)

“计算学习理论”奠基人

2011
年度



Judea Pearl
(1936 -)
(UCLA)

“图模型学习方法”先驱

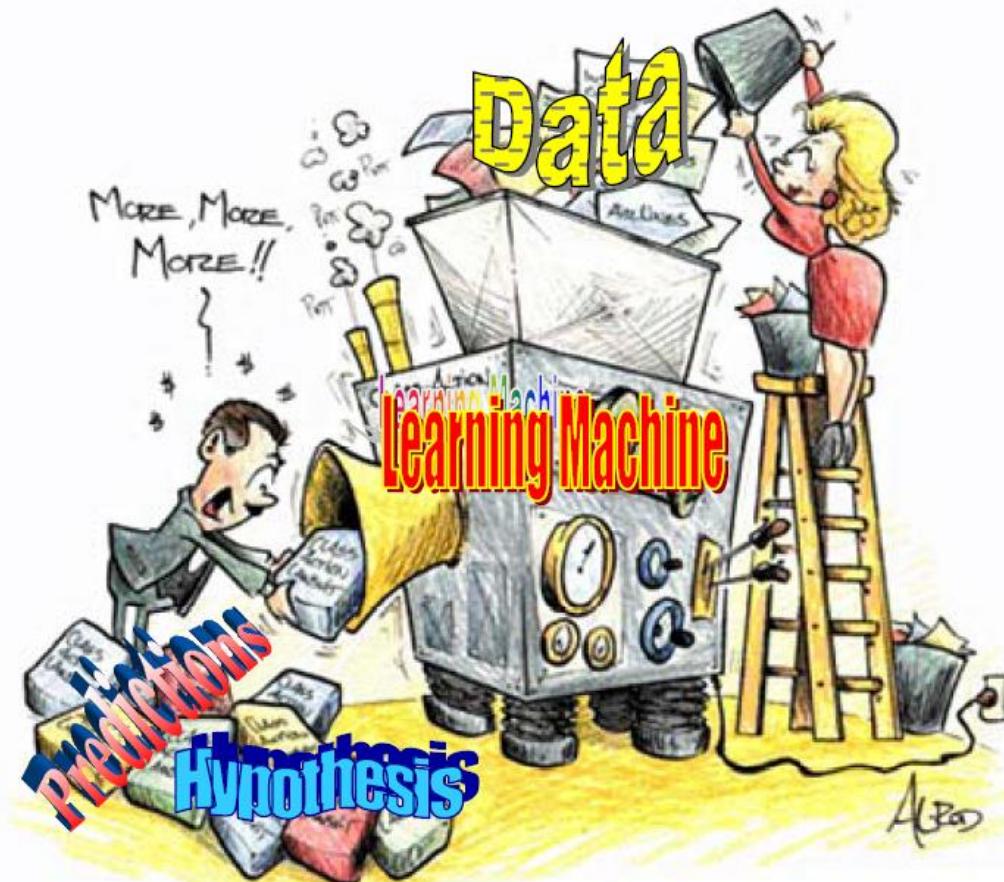
2012
年度

2018年图灵奖获得者

Yann LeCun、Geoffrey Hinton、Yoshua Bengio

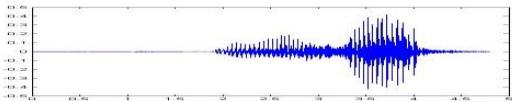


机器学习的一个形象描述



Machine Learning ≈ Looking for Function

- Speech Recognition

$$f($$

$$) = \text{“How are you”}$$

- Image Recognition

$$f($$

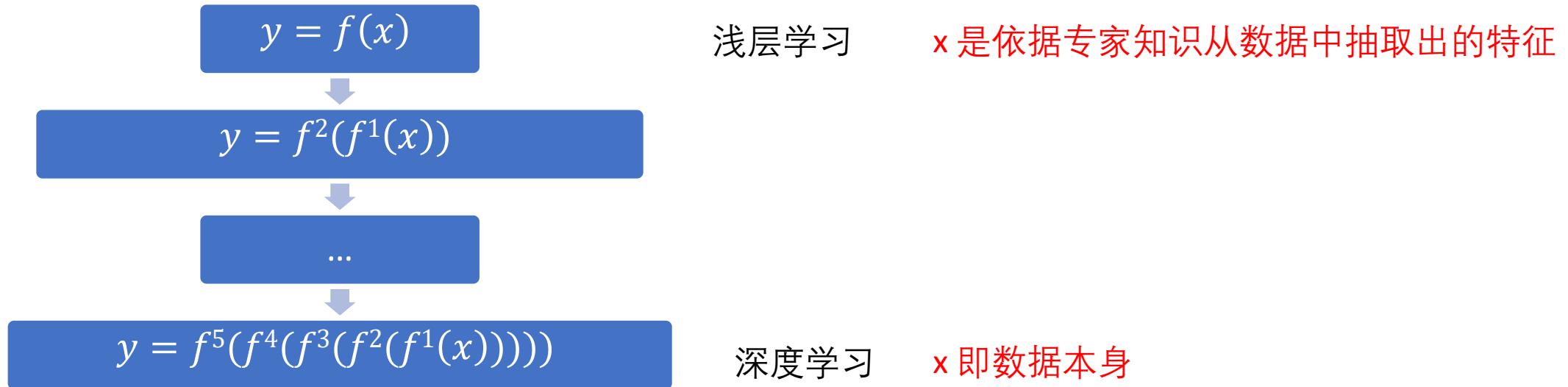
$$) = \text{“Cat”}$$

- Playing Go

$$f($$

$$) = \text{“5-5”}_{\text{(next move)}}$$

Machine Learning ≈ Looking for Function

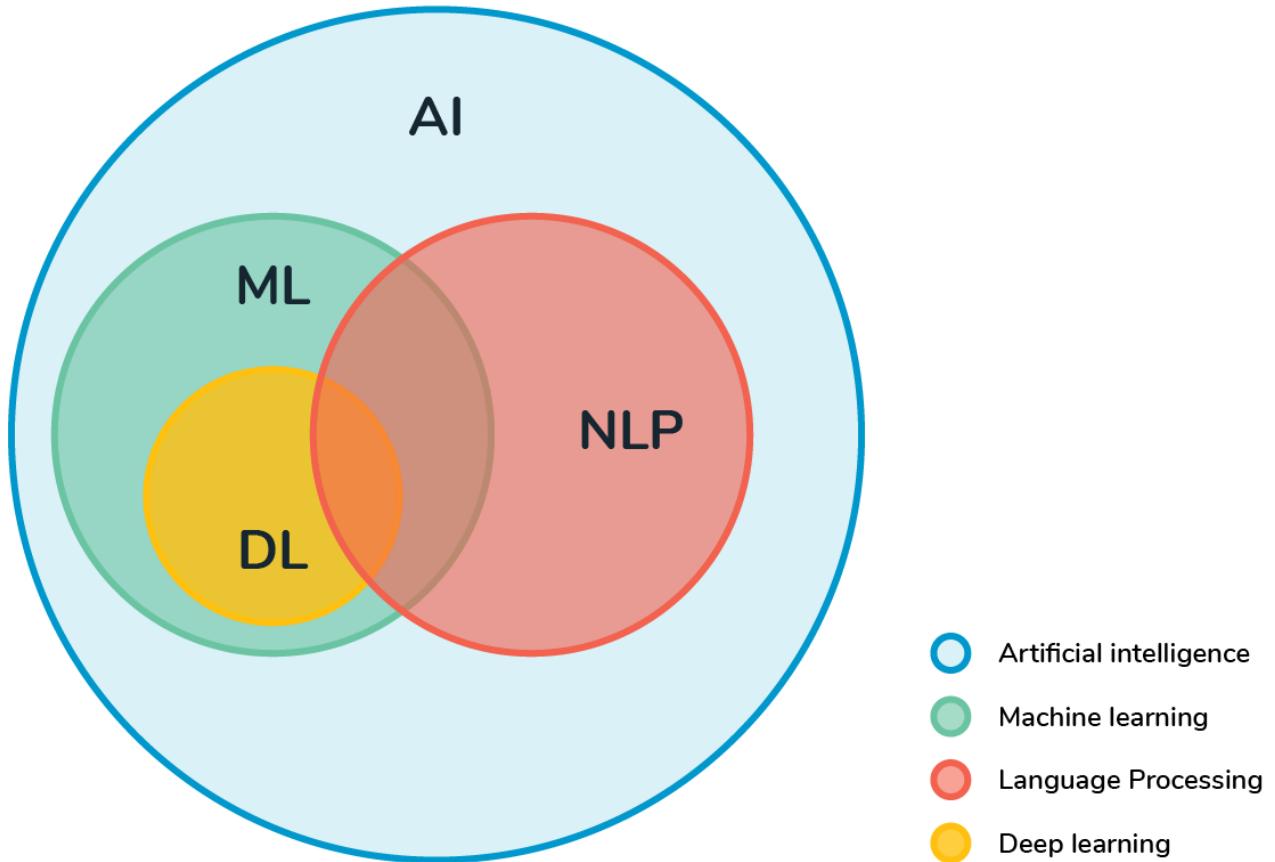


$f($  $) = \text{“你好”}$

$f($  $) = \text{“9”}$

$f($  $) = \text{“Hello!”}$

机器学习与NLP的关系



传统方法与深度学习

虽然深度学习在计算机视觉领域取得了耀眼的成绩，但在自然语言处理领域中的基础任务上发力并不大。

系统名称	算法模型	论文	准确率
TnT	隐马尔可夫模型	Brants (2000)	96.46%
Averaged Perceptron	平均感知机序列标注模型	Collins (2002)	97.11%
SVMTool	支持向量机序列标注模型	Giménez and Márquez (2004)	97.16%
Stanford Tagger 2.0	最大熵模型	Manning (2011))	97.29%
structReg	条件随机场	Sun(2014)	97.36%
Bi-LSTM-CRF	双向长短时记忆网络与CRF层	Huang et al. (2015)	97.55%
NLP4J	线性模型与动态特征提取	Choi (2016)	97.64%

词性标注准确率排行榜



传统方法与深度学习

传统方法和深度学习方法是基础和进阶的关系。无论是传统模型还是神经网络，都属于机器学习的范畴。掌握传统方法，不仅可以解决计算资源受限的工程问题，还可以为将来挑战深度学习打下坚实的基础。

系统名称	算法模型	论文	准确率(UAS)
MaltParser	支持向量机	Nivre (2006)	89.8%
MSTParser	最大生成树+MIRA	McDonald (2006)	91.4%
TurboParser	ILP	Martins (2013)	92.3%
C & M 2014	神经网络	Chen (2014)	92.0%
Weiss 2015	神经网络+结构化感知机	Weiss (2015)	94.0%
SyntaxNet	神经网络+CRF	Andor (2016)	94.6%
Deep Biaffine	深度Biaffine Attention	Dozat (2017)	95.7%

句法分析准确率排行榜



1.5 自然语言处理相关库

- 1. NLTK
- 2. jieba
- 3. HanLP
- 4. SnowNLP
- 5. LTP（哈工大）
- 6. Stanford CoreNLP
- 7. wordcloud
- 8. gensim
- 9. PaddleNLP
- 10. Huggingface transformers
-



1. NLTK

简介：

NLTK，全称Natural Language Toolkit，自然语言处理工具包，是NLP研究领域常用的一个Python库。这是一个开源项目，包含数据集、Python模块、教程等。

NLTK是构建Python程序以使用人类语言数据的领先平台。它为50多种语料库和词汇资源（如 WordNet）提供了易于使用的界面，还提供了一套用于分类，标记化，词干化，标记，解析和语义推理的文本处理库。NLTK是Python上著名的自然语言处理库自带语料库，具有词性分类库自带分类，分词，等等功能。

NLTK被称为“使用Python进行教学和计算语言学工作的绝佳工具”，以及“用自然语言进行游戏的神奇图书馆”。

官网链接：<https://www.nltk.org/>



1. NLTK

主要功能：

- 搜索文本
单词搜索；
相似词搜索；
相似关键词识别；
词汇分布图；
生成文本；
- 计数词汇

语言处理任务	NLTK模块	功能描述	
获取语料库	nltk.corpus	语料库和词典的标准化接口	
字符串处理	nltk.tokenize, nltk.stem	分词、句子分解、提取主干	
搭配研究	nltk.collocations	t-检验、卡方，点互信息	
词性标示符	nltk.tag	n-gram, backoff, Brill, HMM, TnT	
分类	nltk.classify, nltk.cluster	决策树，最大熵，朴素贝叶斯，EM，k-means	
分块	nltk.chunk	正则表达式, n-gram, 命名实体	
解析	nltk.parse	图标, 基于特征, 一致性, 概率性, 依赖项	
语义解释	nltk.sem, nltk.inference	图标, 基于特征, 一致性, 概率性, 依	
指标评测	nltk.metrics	精度, 召回率, 协议系数	
概率与估计	nltk.probability	频率分布, 平滑概率分布	
应用	nltk.app, nltk.chat	图形化的关键词排序, 分析器, WordNet查看器, 聊天	
语言学领域的工作	nltk.toolbox	处理SIL工具箱格式的数据	



1.NLTK

简单使用：

```
import nltk
from nltk import *

# 1. 分词
text = word_tokenize("They refuse to permit us to obtain the refuse permit.")
# 他们拒绝允许我们获得拒绝许可证
print(text)

['They', 'refuse', 'to', 'permit', 'us', 'to', 'obtain', 'the', 'refuse', 'permit', '.']

# 2. 标记词性
tagged = nltk.pos_tag(text)
tagged[0:11]

[('They', 'PRP'),
 ('refuse', 'VBP'),
 ('to', 'TO'),
 ('permit', 'VB'),
 ('us', 'PRP'),
 ('to', 'TO'),
 ('obtain', 'VB'),
 ('the', 'DT'),
 ('refuse', 'NN'),
 ('permit', 'NN'),
 ('.', '.')]
```

# 词性列表	
# CC	coordinatingconjunction 并列连词
# CD	cardinaldigit 纯数 基数
# DT	determiner 限定词(置于名词前起限定作用,如 the、some、my 等)
# EX	existentialthere (like:"there is"... think of it like "thereexists") 存在句;存现句
# FW	foreignword 外来语;外来词;外文原词
# IN	preposition/subordinating conjunction 介词/从属连词;主从连词;从属连接词
# JJ	adjective 'big' 形容词
# JJR	adjective, comparative 'bigger' (形容词或副词的) 比较级形式
# JJS	adjective, superlative 'biggest' (形容词或副词的) 最高级
# LS	listmarker 1)
# MD	modal (could, will) 形态的,形式的,语气的;情态的
# NN	noun, singular 'desk' 名词单数形式
# NNS	nounplural 'desks' 名词复数形式
# NNP	propernoun, singular 'Harrison' 专有名词
# NNPS	proper noun, plural 'Americans' 专有名词复数形式
# PDT	predeterminer 'all the kids' 前位限定词
# POS	possessiveending parent's 属有词 结束语
# PRP	personalpronoun I, he, she 人称代词
# PRP\$	possessive pronoun my, his, hers 物主代词
# RB	adverb very, silently, 副词 非常 静静地
# RBR	adverb,comparative better (形容词或副词的) 比较级形式
# RBS	adverb,superlative best (形容词或副词的) 最高级
# RP	particle give up 小品词(与动词构成短语动词的副词或介词)
# TO	to go 'to' the store.
# UH	interjection errrrrrrrrr 感叹词;感叹语
# VB	verb, baseform take 动词
# VBD	verb, pasttense took 动词 过去时;过去式
# VBG	verb,gerund/present participle taking 动词 动名词/现在分词
# VBN	verb, pastparticiple taken 动词 过去分词
# VBP	verb,sing. present, non-3d take 动词 现在
# VBZ	verb, 3rdperson sing. present takes 动词 第三人称
# WDT	wh-determiner which 限定词(置于名词前起限定作用,如 the、some、my 等)
# WP	wh-pronoun who, what 代词(代替名词或名词词组的单词)
# WP\$	possessivewh-pronoun whose 所有格;属有词
# WRB	wh-abverb where, when 副词



1.NLTK

简单使用：

```
# 3. 词汇统计
```

```
from nltk.text import Text
str = "Today 's weather is good,very windy and sunny,but we have a lot of classes in the afternoon,we have to play football tomorrow."
tokens = word_tokenize(str)
```

```
tokens[:5]
```

```
['Today', "'s", 'weather', 'is', 'good']
```

```
tokens = [word.lower() for word in tokens] # 转换成小写
```

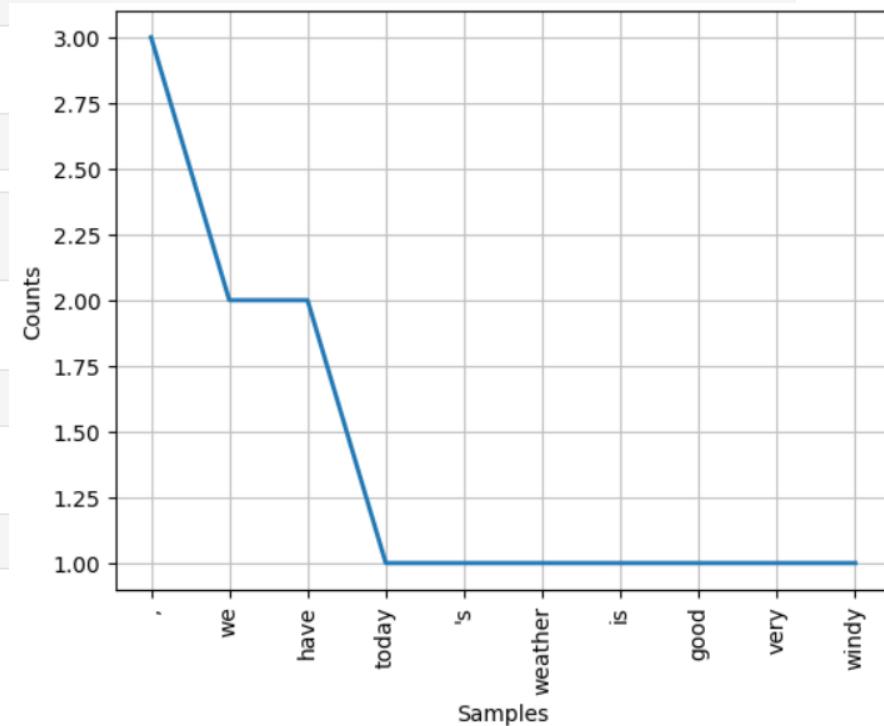
```
t = Text(tokens)
t.count("windy") # 计数某词出现频率
```

```
1
```

```
t.index("windy") # 查找某词索引
```

```
7
```

```
t.plot(10) # 前n个高频词的词频-词汇绘图
```





1.NLTK

简单使用：

```
# 4.停用词
from nltk.corpus import stopwords
stopwords.fileids() # 具体的语言
```

```
['arabic',
 'azerbaijani',
 'basque',
 'bengali',
 'catalan',
 'chinese',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'greek',
 'hebrew',
 'hinglish',
 'hungarian',
 'indonesian',
 'italian',
 'kazakh',
 'nepali',
 'norwegian',
 'portuguese',
 'romanian',
 'russian',
 'slovene',
 'spanish',
 'swedish',
 'tajik',
 'turkish']
```

```
test_words = [word.lower() for word in tokens] # tokens是上面的句子
test_words_set = set(test_words) # 集合
test_words_set.intersection(set(stopwords.words('english')))

# 在 "Today 's weather is good...we have to play football tomorrow." 中的停用词
```

```
{'a', 'and', 'but', 'have', 'in', 'is', 'of', 'the', 'to', 'very', 'we'}
```

```
filtered = [w for w in test_words_set if(w not in stopwords.words('english'))] # 过滤
filtered
```

```
['lot',
 ',',
 "'s",
 'football',
 'weather',
 'play',
 'windy',
 'classes',
 'today',
 'tomorrow',
 '.',
 'afternoon',
 'good',
 'sunny']
```



1.NLTK

简单使用：

```
# 5. 加载内置语料库
from nltk.corpus import brown
brown.categories()
```

回 ↑

```
['adventure',
'belles_lettres',
'editorial',
'fiction',
'government',
'hobbies',
'humor',
'learned',
'lore',
'mystery',
'news',
'religion',
'reviews',
'romance',
'science_fiction']
```

```
len(brown.sents())
```

57340

```
len(brown.words())
```

1161192

官网链接：<https://www.nltk.org/>



2.jieba

简介：

jieba是优秀的Python中文分词第三方库。Jieba库的分词原理：利用一个中文词库，确定汉字之间的关联概率，汉字间概率大的组成词组，形成分词结果。除了分词，用户还可以添加自定义的词组。

jieba分词具备以下优点：

- 社区活跃。Jieba在Github社区活跃度高，代表着该项目会持续更新，实际生产实践中遇到的问题能够在社区反馈并得到解决，适合长期使用。
- 功能丰富。Jieba其实并不是只有分词这一个功能，其是一个开源框架，提供了很多在分词之上的算法，如关键词提取、词性标注等。
- 提供多种编程语言实现。Jieba官方提供了Python、C++、Go、R、iOS等多平台多语言支持，不仅如此，还提供了很多热门社区项目的扩展插件，如ElasticSearch、solr、lucene等。在实际项目中，进行扩展十分容易。
- 使用简单。Jieba的API总体来说并不多，且需要进行的配置并不复杂，方便上手。

官网链接：<https://pypi.org/project/jieba/>



2.jieba

三种分词模式：

- **精确模式**：试图将句子最精确地切开，适合文本分析。
- **全模式**：把句子中所有可以成词的词语都扫描出来，速度非常快，但是不能解决歧义。
- **搜索引擎模式**：在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

```
import jieba

sent = '中文分词是文本处理不可或缺的一步！'

seg_list = jieba.cut(sent, cut_all=True)
print('全模式: ', '/ '.join(seg_list))

seg_list = jieba.cut(sent, cut_all=False)
print('精确模式: ', '/ '.join(seg_list))

seg_list = jieba.cut(sent)
print('默认精确模式: ', '/ '.join(seg_list))

seg_list = jieba.cut_for_search(sent)
print('搜索引擎模式', '/ '.join(seg_list))
```

全模式: 中文/ 分词/ 是/ 文本/ 文本处理/ 本处/ 处理/ 不可/ 不可或缺/ 或缺/ 的/ 一步/ !

精确模式: 中文/ 分词/ 是/ 文本处理/ 不可或缺/ 的/ 一步/ !

默认精确模式: 中文/ 分词/ 是/ 文本处理/ 不可或缺/ 的/ 一步/ !

搜索引擎模式 中文/ 分词/ 是/ 文本/ 本处/ 处理/ 文本处理/ 不可/ 或缺/ 不可或缺/ 的/ 一步/ !



2.jieba

简单使用：

```
# 词性分析
import jieba.posseg as posseg

text = "中文分词是文本处理不可或缺的一步!"
# generator形式形如pair('word', 'pos')的结果
seg = posseg.cut(text)
print([se for se in seg])

[pair('中文', 'nz'), pair('分词', 'n'), pair('是', 'v'), pair('文本处理', 'n'), pair('不可或缺', 'l'), pair('的', 'uj'), pair('一步', 'm'), pair('!', 'x')]
```

```
# 关键词抽取
import jieba.analyse as analyse
text = "中文分词是文本处理不可或缺的一步!"
# TF-IDF
tf_result = analyse.extract_tags(text, topK=5) # topK指定数量, 默认20
print('tf-idf:', tf_result)
# TextRank
tr_result = analyse.textrank(text, topK=5) # topK指定数量, 默认20
print('textrank:', tr_result)

tf-idf: ['文本处理', '分词', '不可或缺', '中文', '一步']
textrank: ['文本处理', '分词']
```



3. HanLP

简介：

HanLP是一系列模型与算法组成的NLP工具包，由大快搜索主导并完全开源，目前支持很多功能，项目主要是使用java语言开发的，也支持 python，目标是普及自然语言处理在生产环境中的应用。HanLP具备功能完善、性能高效、架构清晰、语料时新、可自定义的特点。

主要功能：

中文分词、词性标注、命名实体识别、依存句法分析、关键词提取、新词发现、短语提取、自动摘要、文本分类、拼音简繁.....

对于中文自然语言处理来说，HanLP功能强大，可以完成很多复杂的任务，因此HanLP也是一个非常重要自然语言处理工具包。

官网链接：<https://www.hanlp.com/>



3.HanLP

简单使用：

1.分词和词性标注

```
sentence = "我爱自然语言处理技术！"  
s_hanlp = HanLP.segment(sentence)  
for term in s_hanlp:  
    print(term.word, term.nature)
```

我 rr

爱 v

自然语言处理 nz

技术 n

！ w

2.依存句法分析

```
s_dep = HanLP.parseDependency(sentence)  
print(s_dep)
```

1	我	我	r	r	_	2	主谓关系	-	-
2	爱	爱	v	v	_	0	核心关系	-	-
3	自然语言处理	自然语言处理	v	v	_	4	定中关系	-	-
4	技术	技术	n	n	_	2	动宾关系	-	-
5	！	！	wp	w	_	2	标点符号	-	-



3.HanLP

简单使用：

3.关键词提取

```
document = u'''
```

自然语言处理是计算机科学领域与人工智能领域中的一个重要方向。

它研究能实现人与计算机之间用自然语言进行有效通信的各种理论和方法。

自然语言处理是一门融语言学、计算机科学、数学于一体的科学。

因此，这一领域的研究将涉及自然语言，即人们日常使用的语言，
所以它与语言学的研究有着密切的联系，但又有重要的区别。

自然语言处理并不是一般地研究自然语言，
而在于研制能有效地实现自然语言通信的计算机系统，
特别是其中的软件系统。因而它是计算机科学的一部分。

```
'''
```

```
doc_keyword = HanLP.extractKeyword(document, 3)
```

```
for word in doc_keyword:
```

```
    print(word)
```

研究

自然语言

自然语言处理

4.摘要抽取

```
doc_keysentence = HanLP.extractSummary(document, 3)
```

```
for key_sentence in doc_keysentence:
```

```
    print(key_sentence)
```

自然语言处理并不是一般地研究自然语言

自然语言处理是计算机科学领域与人工智能领域中的一个重要方向

它研究能实现人与计算机之间用自然语言进行有效通信的各种理论和方法

5.短语提取

```
phraseList = HanLP.extractPhrase(document, 3)
```

```
print(phraseList)
```

[计算机科学, 中的重要, 之间自然语言]

HanLP Python接口

- 安装Java
 - <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- 安装pyhanlp
 - pip install pyhanlp
 - 这个包依赖Java和Jpype。HanLP主项目采用Java开发，所以需安装要JDK和JRE。
 - 建议预先安装Anaconda，再安装pyhanlp



HanLP Python接口

- Anaconda环境下安装pyhanlp
 - conda install -c conda-forge jpype1
 - pip install pyhanlp

```
(hanlp) C:\Users\49260>hanlp
下载 https://file.hankcs.com/hanlp/hanlp-1.8.1-release.zip 到 f:\programfiles\anaconda3\envs\hanlp\lib\site-packages\pyhanlp\static\hanlp-1.8.1-release.zip
100% 1.8 MiB 138.7 KiB/s ETA: 0 s [=====]
下载 https://file.hankcs.com/hanlp/data-for-1.7.5.zip 到 f:\programfiles\anaconda3\envs\hanlp\lib\site-packages\pyhanlp\static\data-for-1.8.1.zip
100% 637.7 MiB 709.3 KiB/s ETA: 0 s [=====]
解压 data.zip...
找不到Java, 请安装JDK8: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
是否前往 https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html ? (y/n) y
```



HanLP Python 接口

• 测试

```
(hanlp) C:\Users\49260>hanlp segment --help
usage: hanlp segment [-h] [--tag | --no-tag] [-a ALGORITHM] [--config CONFIG]

optional arguments:
  -h, --help            show this help message and exit
  --tag                 show part-of-speech tags
  --no-tag              don't show part-of-speech tags
  -a ALGORITHM, --algorithm ALGORITHM
                        algorithm of segmentation e.g. perceptron
  --config CONFIG       path to hanlp.properties
```



HanLP Python接口

- 在初步体验HanLP后，来看看如何在python中调用HanLP的常用接口。

```
1  # -*- coding:utf-8 -*-
2  # Author: hankcs
3  # Date: 2019-03-21 21:22
4  # 《自然语言处理入门》1.6 开源工具
5  # 配套书籍: http://nlp.hankcs.com/book.php
6  # 讨论答疑: https://bbs.hankcs.com/
7  from pyhanlp import *
8
9
10 def main():
11     HanLP.Config.enableDebug()
12     # 为了避免你等得无聊，开启调试模式说点什么:-)
13     print(HanLP.segment("王国维和服务员"))
14
15
16 if __name__ == '__main__':
17     main()
```



HanLP Java接口

```
<dependency>
    <groupId>com.hankcs</groupId>
    <artifactId>hanlp</artifactId>
    <version>portable-1.7.5</version>
</dependency>
```



HanLP在线资料

- 官网: <http://hanlp.com/>
 - GitHub: <https://github.com/hankcs/HanLP>
 - 文档:
 - <https://github.com/hankcs/HanLP/blob/master/README.md>
 - <https://github.com/hankcs/HanLP/wiki>
 - 论坛: <https://bbs.hankcs.com/>
 - 博客: <http://www.hankcs.com/>





4.SnowNLP

简单使用：

```
from snownlp import SnowNLP
```

```
# 1. 中文分词
```

```
text = '希望本无所谓有，也无所谓无，这就像地上的路，其实地上本没有路，走的人多了，也便成了路。'
```

```
s = SnowNLP(text)
```

```
print(s.words)
```

```
# SnowNLP的分词是基于共19484行句子作为语料库来训练分词的，这些句子可能来自不同的几个方向，但不太全面，所以在某些词语的分解上有缺陷
```

```
['希望', '本', '无所谓', '有', '，', '也', '无所谓', '无', '，', '这', '就', '像地', '上', '的', '路', '，', '，', '其实地', '上本', '没', '有', '路', '，', '，', '走', '的', '人', '多', '了', '，', '也', '便', '成', '了', '路', '。']
```

```
#2. 情感分析
```

```
text1 = '这是我遇见的最棒的一家店，种类多，价格低，更喜欢的是服务质量很好'
```

```
text2 = '这是我遇到的最差的一家店，种类少，价格贵，更气人的是服务质量很差'
```

```
s1 = SnowNLP(text1)
```

```
s2 = SnowNLP(text2)
```

```
print(s1.sentiments)
```

```
print(s2.sentiments)
```

```
# 情感分析的结果是一个小数，越接近1，说明越偏向积极；越接近0，说明越偏向消极。
```

```
# SnowNLP的情感分析是基于自带的两个积极跟消极的语料文件来进行分析的（neg.txt、pos.txt）。
```

```
# 这两个语料文件是某平台的评论留言，主要有关于书本的、酒店的、电脑及配件的几个方向的评价留言。
```

```
# 由于语料文件比较片面，而且其中有些语句意向不准确，导致情感分析在某些场合中效果并不明显。
```

0.99509716108733

0.0033869983750457466



4. SnowNLP

简单使用：

```
# 3. 词性标注
text = '哪里有天才，我是把别人喝咖啡的工夫都用在了工作上了。'
s = SnowNLP(text)
print(list(s.tags))
```

```
[('哪里', 'r'), ('有', 'v'), ('天才', 'n'), ('', 'w'), ('我', 'r'), ('是', 'v'), ('把', 'p'), ('别人', 'r'), ('喝', 'v'), ('咖啡', 'n'), ('的', 'u'), ('工夫', 'n'), ('都', 'd'), ('用', 'v'), ('在', 'p'), ('了', 'u'), ('工作', 'vn'), ('上', 'f'), ('了', 'u'), ('。', 'w')]
```

```
# 4. 转换成拼音
text = '哪里有天才，我是把别人喝咖啡的工夫都用在了工作上了。'
s = SnowNLP(text)
print(s.pinyin)
```

```
['na', 'li', 'you', 'tian', 'cai', ' ', 'wo', 'shi', 'ba', 'bie', 'ren', 'he', 'ka', 'fei', 'de', 'gong', 'fu', 'dou', 'yong', 'zai', 'liao', 'gong', 'zuo', 'shang', 'liao', '。']
```

```
# 5. 繁体转简体
text = '希望本無所謂有，也無所謂無，這就像地上的路，其實地上本沒有路，走的人多了，也便成了路。'
s = SnowNLP(text)
print(s.han)
```

希望本无所谓有，也无所谓无，这就像地上的路，其实地上本没有路，走的人多了，也便成了路。



4.SnowNLP

简单使用：

```
# 6. 提取文本关键词
text = '随着顶层设计完成，全国政协按下信息化建设快进键：建设开通全国政协委员移动履职平台，开设主题议政群、全国政协书院等栏目，建设委员履职数据库，拓展网上委员履职'
s = SnowNLP(text)
print(s.keywords(3))
['全国', '政协', '远程']

# 7. 提取文本摘要
text = '随着顶层设计完成，全国政协按下信息化建设快进键：建设开通全国政协委员移动履职平台，开设主题议政群、全国政协书院等栏目，建设委员履职数据库，拓展网上委员履职'
s = SnowNLP(text)
print(s.summary(2))# 总结两条摘要
['全国政协按下信息化建设快进键：建设开通全国政协委员移动履职平台', '建成网络议政远程协商视频会议系统']

# 8. 文本相似度(BM25)
s = SnowNLP([['机器学习', '人工智能'],
            ['深度学习', '自然语言处理'],
            ['数据挖掘']])
artilc1 = ['自然语言处理']
print(s.sim(artilc1))
[0, 0.4686473612532025, 0]
```



5.LTP (哈工大)

简介：

哈工大语言技术平台Language Technology Platform (LTP) 是哈工大社会计算与信息检索研究中心历时十年开发的一整套中文语言处理系统。LTP制定了基于XML的语言处理结果表示，并在此基础上提供了一整套自底向上的丰富而且高效的中文语言处理模块(包括词法、句法、语义等6项中文处理核心技术)，以及基于动态链接库(Dynamic Link Library, DLL)的应用程序接口，可视化工具，并且能够以网络服务(Web Service)的形式进行使用。

主要功能：

对于中文文本进行分词、词性标注、句法分析、语义角色标注、情感分析.....

官网链接：<http://ltp.ai/>



5.LTP (哈工大)

简单使用：

1.分词

```
import pyltp
from pyltp import Segmentor#导入Segmentor库

math_path = "D:\ltp3.4.0\cws.model"#LTP分词模型库
segmentor = Segmentor()#实例化分词模块
segmentor.load(math_path)#加载分词库

words = segmentor.segment("中国是一个自由、和平的国家")
print(' '.join(words).split())#分割分词后的结果
分词结果: ['中国', '是', '一个', '自由', '、', '和平', '的', '国家']
```

2.词性标记

```
import pyltp
from pyltp import Postagger#导入Postagger库
math_path = "D:\ltp3.4.0\pos.model"#LTP词性标注模型库
postagger = Postagger() #实例化词性模块
postagger.load(math_path)#加载词性库
postags = postagger.postag(words)#这里的words是分词后的结果
print(' '.join(postags).split())#分割标注后的结果
标注结果: ['ns', 'v', 'm', 'a', 'wp', 'a', 'u', 'n']
```

3.命名实体识别

```
import pyltp
from pyltp import NamedEntityRecognizer#导入库NamedEntityRecognizer
math_path = "D:\ltp3.4.0\ner.model"#LTP命名实体识别模型库
recognizer = NamedEntityRecognizer() # 初始化实例
recognizer.load(math_path)#加载实体识别库
words = ['中国', '是', '一个', '自由', '、', '和平', '的', '国家']
postags = ['ns', 'v', 'm', 'a', 'wp', 'a', 'u', 'n']
netags = recognizer.recognize(words, postags) # 命名实体识别，这里的words是分词的结果
print(' '.join(netags).split())#分割识别后的结果
结果:
S-Ns      0      0      0      0      0      0      0
```



5.LTP (哈工大)

简单使用：

4.依存句法分析

```
import pyltp
from pyltp import Parser#导入库Parser
math_path = "D:\ltp3.4.0\parser.model"#LTP依存分析模型库
parser = Parser() # 初始化实例
parser.load(math_path)#加载依存分析库
words = ['中国', '是', '一个', '自由', '、', '和平', '的', '国家']
postags = ['ns', 'v', 'm', 'a', 'wp', 'a', 'u', 'n']
arcs = parser.parse(words, postags) # 句法分析, 这里的words是分词的结果

print ("\t".join("%d:%s" % (arc.head, arc.relation) for arc in arcs) ) # 依存分析,
结果:
2:SBV  0:HED  8:ATT  8:ATT  6:WP  4:COO  4:RAD  2:VOB
```

5.语义角色标注

```
import pyltp
from pyltp import SementicRoleLabeller # 导入库SementicRoleLabeller
math_path = "D:\ltp3.4.0\pisrl.model" # LTP语义角色标注模型库
labeller = SementicRoleLabeller() # 初始化实例
labeller.load(math_path) # 加载语义标注库
words = ['中国', '是', '一个', '自由', '、', '和平', '的', '国家']
postags = ['ns', 'v', 'm', 'a', 'wp', 'a', 'u', 'n']
roles = labeller.label(words, postags, arcs) # 语义角色标注,这里的words是分词结果, postags是词性标注结果, arcs是依存句法分析结果
```

打印结果

```
for role in roles:
    print(role.index, "".join(
        ["%s:(%d,%d)" % (arg.name, arg.range.start, arg.range.end) for arg in
        role.arguments]))
```



6. Stanford CoreNLP

简介：

CoreNLP项目是斯坦福大学开发的一套开源NLP工具包，包括词性标记器，命名实体识别器，解析器，情感分析，自举模式学习和开放式信息提取工具。

Stanford CoreNLP是用Java编写的，因此，**需要安装Java才能运行CoreNLP**。但是，可以通过命令行或其Web服务与CoreNLP交互；也可以使用Javascript，Python或其他语言编写自己的代码时使用CoreNLP。

它的语言支持广泛，目前支持阿拉伯语，中文，英文，法语，德语，西班牙语，做平行语料的对比非常方便。

主要功能：

除了从CoreNLP继承的功能外，StanfordNLP还包含将一串文本转换成句子和单词列表，生成这些单词的基本形式、它们的词类和形态学特征的工具，以及超过70种语言的句法结构。这个软件包采用高准确性的神经网络组件构建，这些组件支持用户使用自己的注释数据进行高效的训练和评估。

官网链接：<https://nlp.stanford.edu/software/>



6. Stanford CoreNLP

简单使用：

```
from stanfordcorenlp import StanfordCoreNLP

#默认是英文
nlp = StanfordCoreNLP(r'c:\Code\Jupyter Notebook\stanford-corenlp')
#如果要用其他语言，需要单独设置
nlp_ch = StanfordCoreNLP(r'c:\Code\Jupyter Notebook\stanford-
corenlp', lang='zh')
```

1. 分词

```
nlp.word_tokenize('This is an example of tokenziation.')
#结果: ['This', 'is', 'an', 'example', 'of', 'tokenziation', '.']

nlp_ch.word_tokenize('浙江大学有七个校区。')
#结果: ['浙江', '大学', '有', '七', '个', '校区', '。']
```

2. 词性标注

```
nlp.pos_tag('This is an example of tokenziation.')
#结果: [('This', 'DT'), ('is', 'VBZ'), ('an', 'DT'), ('example', 'NN'), ('of', 'IN'), ('tokenziation',
'NN'), ('.', '.')]

nlp_ch.pos_tag('浙江大学有七个校区。')
#结果: [('浙江', 'NR'), ('大学', 'NN'), ('有', 'VE'), ('七', 'CD'), ('个', 'M'), ('校区', 'NN'), ('。', ''),
'PU')]
```



6. Stanford CoreNLP

简单使用：

3. 句法成分分析

```
print(nlp.parse('This is an example of tokenization.'))  
'''
```

结果：

```
(ROOT  
(S  
(NP (DT This))  
(VP (VBZ is)  
(NP  
(NP (DT an) (NN example))  
(PP (IN of)  
(NP (NN tokenization))))  
(. .)))  
'''
```

```
print(nlp_ch.parse('浙江大学有七个校区。'))  
'''
```

结果：

```
(ROOT  
(IP  
(NP (NR 浙江) (NN 大学))  
(VP (VE 有)  
(NP  
(QP (CD 七)  
(CLP (M 个)))  
(NP (NN 校区))))  
(PU 。 )))  
'''
```

句法分析的标注实际上是由两个部分组成的，在最小的一层括号内实际上就是词性标注，而除此之外则是与句子结构相关的标注。



7.wordcloud

简介：

词云图，也叫文字云，是对文本中出现频率较高的“关键词”予以视觉化的展现，词云图过滤掉大量的低频低质的文本信息，使得浏览者只要一眼扫过文本就可领略文本的主旨。WordCloud 是一款python环境下的词云图工具包，同时支持python2和 python3，能通过代码的形式把关键词数据转换成直观且有趣的图文模式。

官网链接：<https://pypi.org/project/wordcloud/>



7.wordcloud

简单使用：

- w.generate(): 向WordCloud对象中加载文本txt

```
>>>w.generate("Python and WordCloud")
```

- w.to_file(filename): 将词云输出为图像文件, .png或.jpg格式

```
>>>w.to_file("outfile.png")
```

width: 指定词云对象生成图片的宽度,默认400像素

height: 指定词云对象生成图片的高度,默认200像素

min_font_size : 指定词云中字体的最小字号, 默认4号

max_font_size: 指定词云中字体的最大字号, 根据高度自动调节

font_step: 指定词云中字体字号的步进间隔, 默认为1

font_path: 指定文本文件的路径, 默认None

max_words: 指定词云显示的最大单词数量,默认200

stop_words: 指定词云的排除词列表, 即不显示的单词列表

mask: 指定词云形状, 默认为长方形, 需要引用imread()函数

background_color: 指定词云图片的背景颜色, 默认为黑色

```
w=wordcloud.WordCloud(width=600)
```

```
w=wordcloud.WordCloud(height=400)
```

```
w=wordcloud.WordCloud(min_font_size=10)
```

```
w=wordcloud.WordCloud(max_font_size=20)
```

```
w=wordcloud.WordCloud(font_step=2)
```

```
w=wordcloud.WordCloud(font_path="msyh.ttc")
```

```
w=wordcloud.WordCloud(max_words=20)
```

```
w=wordcloud.WordCloud(stop_words="Python")
```

```
from scipy.msc import imread
```

```
mk=imread("pic.png")
```

```
w=wordcloud.WordCloud(mask=mk)
```

```
w=wordcloud.WordCloud(background_color="white")
```

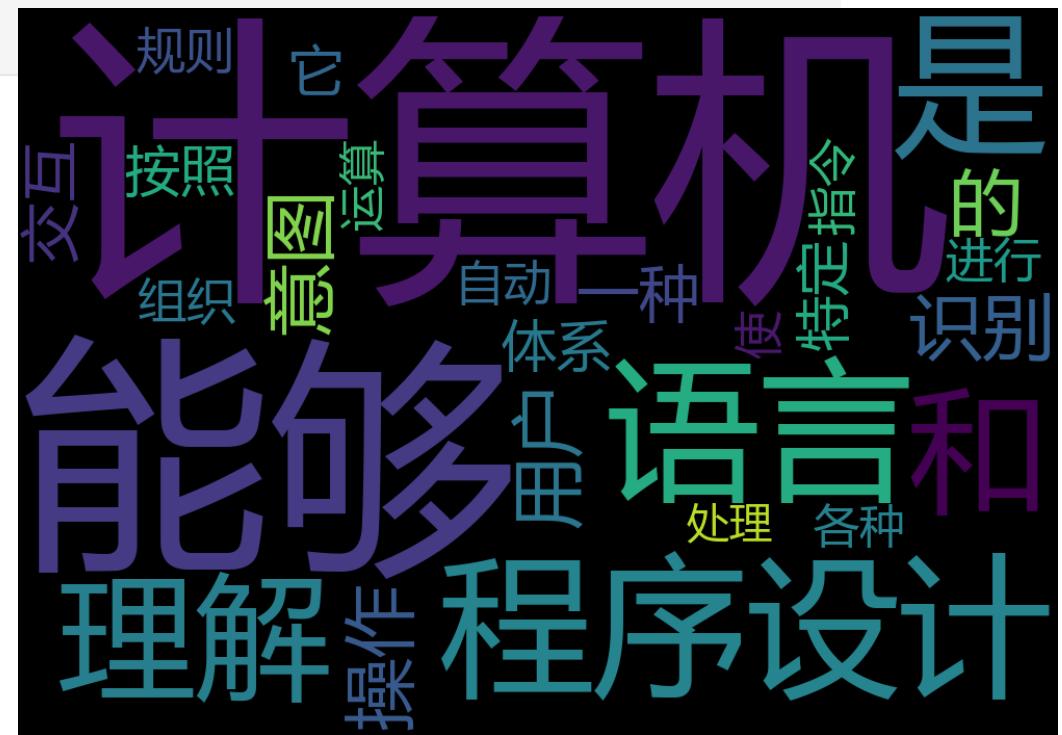


7.wordcloud

简单使用：

```
import jieba
import wordcloud
txt="程序设计语言是计算机能够理解和识别用户操作意图的一种交互体系，它按照特定规则组织计算机指令，使计算机能够自动进行各种运算处理。"
w=wordcloud.WordCloud(width=1000,font_path='msyh',height=700)
w.generate(" ".join(jieba.lcut(txt)))
w.to_file("computerlanguage.png")
```

<wordcloud.wordcloud.WordCloud at 0x1baeb5d62b0>





8.gensim

简介：

Gensim (generate similarity) 是一个简单高效的自然语言处理Python库，用于抽取文档的语义主题 (semantic topics)。Gensim的输入是原始的、无结构的数字文本（纯文本），内置的算法包括Word2Vec, FastText, 潜在语义分析 (Latent Semantic Analysis, LSA) , 潜在狄利克雷分布 (Latent Dirichlet Allocation, LDA) 等，通过计算训练语料中的统计共现模式自动发现文档的语义结构。这些算法都是非监督的，这意味着不需要人工输入——仅仅需要一组纯文本语料。一旦发现这些统计模式后，任何纯文本（句子、短语、单词）就能采用语义表示简洁地表达。

官网链接：<https://radimrehurek.com/gensim/>



8.gensim

优点：

- Memory independence：不需要一次性将整个训练语料读入内存，Gensim充分利用了Python内置的生成器（generator）和迭代器（iterator）用于流式数据处理，内存效率是Gensim设计目标之一。
- Memory sharing：训练好的模型可以持久化到硬盘，和重载到内存。多个进程之间可以共享相同的数据，减少了内存消耗。
- 多种向量空间算法的高效实现：包括Word2Vec, Doc2Vec, FastText, TF-IDF, LSA, LDA, 随机映射等。
- 支持多种数据结构。
- 基于语义表示的文档相似度查询。



8.gensim

核心概念：

- Document：在Gensim中，Document就是一个句子文本（即python3中的[str]），一个Document可以是一个段落，一本书的内容，一篇文章。
- Corpus：在Gensim中，Corpus是由Document组成的集合，一个Corpus往往包含许多个Document。Corpus一般可以扮演两种角色，一个是训练模型所要的输入。二是文件组织，此时它可充当相似咨询的索引（由语义相似性，聚类等产生的咨询）。
- Vector：在Gensim中，这里的向量主要指词袋向量。
- Model：在Gensim中，Document是由向量表示的，而Model就是在两个向量空间中做了一个转化。



8.gensim

简单使用：

```
from pprint import pprint # pretty-printer
from collections import defaultdict

# 一个由九个document组成的corpus
text_corpus = [
    "Human machine interface for lab abc computer applications",
    "A survey of user opinion of computer system response time",
    "The EPS user interface management system",
    "System and human system engineering testing of EPS",
    "Relation of user perceived response time to error measurement",
    "The generation of random binary unordered trees",
    "The intersection graph of paths in trees",
    "Graph minors IV Widths of trees and well quasi ordering",
    "Graph minors A survey",
]
```

```
# 利用Corpus产生它的向量空间

# remove common words and tokenize
stoplist = set('for a of the and to in'.split())
texts = [[word for word in document.lower().split() if word not in stoplist]
         for document in text_corpus]
# remove words that appear only once
frequency = defaultdict(int)
for text in texts:
    for token in text:
        frequency[token] += 1
texts = [[token for token in text if frequency[token] > 1]
         for text in texts]
pprint(texts)

# 把text_corpus中词频出超过1且非停用词的词给抽出来了，这个操作相当于做了一个词的过滤

[['human', 'interface', 'computer'],
 ['survey', 'user', 'computer', 'system', 'response', 'time'],
 ['eps', 'user', 'interface', 'system'],
 ['system', 'human', 'system', 'eps'],
 ['user', 'response', 'time'],
 ['trees'],
 ['graph', 'trees'],
 ['graph', 'minors', 'trees'],
 ['graph', 'minors', 'survey']]
```



8.gensim

简单使用：

```
# 为了得到这些文本的向量空间模型，需要使用 gensim.corpora.Dictionary 这个方法改建 Dictionary类

from gensim import corpora
dictionary = corpora.Dictionary(texts)
print(dictionary)

Dictionary(12 unique tokens: ['computer', 'human', 'interface', 'response', 'survey']...)

# 使用 dictionary.docbow方法得到该语料的向量空间模型

corpus = [dictionary.doc2bow(text) for text in texts]
corpus # corpus对应的向量

[[ (0, 1), (1, 1), (2, 1)],
 [(0, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1)],
 [(2, 1), (5, 1), (7, 1), (8, 1)],
 [(1, 1), (5, 2), (8, 1)],
 [(3, 1), (6, 1), (7, 1)],
 [(9, 1)],
 [(9, 1), (10, 1)],
 [(9, 1), (10, 1), (11, 1)],
 [(4, 1), (10, 1), (11, 1)]]
```



9.PaddleNLP

简介：

PaddleNLP 是飞桨自然语言处理开发库，具备易用的文本领域API，多场景的应用示例、和高性能分布式训练 三大特点，旨在提升飞桨开发者文本领域建模效率，旨在提升开发者在文本领域的开发效率，并提供丰富的NLP应用示例。

主要功能：

- 自然语言理解任务：中文分词、词性标注、命名实体识别、文本纠错、句法分析、情感分析、文本相似度.....
- 自然语言生成任务：生成式问答、智能写诗、开放域对话、文本翻译、自动对联.....

官网链接：<https://www.paddlepaddle.org.cn/paddle/paddlenlp>



9. PaddleNLP

优点：

- 易用的文本领域API

提供丰富的产业级预置任务能力 Taskflow 和全流程的文本领域API：支持丰富中文数据集加载的 Dataset API，可灵活高效地完成数据预处理的 Data API，预置60+预训练词向量的 Embedding API，提供100+预训练模型的 Transformer API 等，可大幅提升NLP任务建模的效率。

- 多场景的应用示例

覆盖从学术到产业级的NLP应用示例，涵盖NLP基础技术、NLP系统应用以及相关拓展应用。全面基于飞桨核心框架2.0全新API体系开发，为开发者提供飞桨文本领域的最佳实践。

- 高性能分布式训练

基于飞桨核心框架领先的自动混合精度优化策略，结合分布式Fleet API，支持4D混合并行策略，可高效地完成大规模预训练模型训练。



9. PaddleNLP

简单使用：

PaddleNLP数据处理流程

1. 加载数据集（内置数据集或者自定义数据集，数据集返回原始数据）。
2. 定义 `trans_func()`，包括 `tokenize`, `token to id` 等操作，并传入数据集的 `map()` 方法，将原始数据转为 `feature`。
3. 根据上一步数据处理的结果定义 `batchify` 方法和 `BatchSampler`。
4. 定义 `DataLoader`，传入 `BatchSampler` 和 `batchify_fn()`。



10.Huggingface transformers

简介：

Huggingface Transformers 是基于一个开源基于 transformer 模型结构提供的预训练语言库，它支持 Pytorch, Tensorflow2.0，并且支持两个框架的相互转换。框架支持了最新的各种NLP预训练语言模型，使用者可以很快速的进行模型的调用，并且支持模型further pretraining 和 下游任务fine-tuning。

该库是使用 BERT 等预训练模型的最常用的库，甚至超过了google等开源的源代码。它的设计原则保证了它支持各种不同的预训练模型，并且有统一的合理的规范。使用者可以很方便的进行模型的下载，以及使用。同时，它支持用户自己上传自己的预训练模型到Model Hub中，提供其他用户使用。对于NLP从业者，可以使用这个库，很方便地进行自然语言理解（NLU）和自然语言生成（NLG）任务的SOTA模型使用。

官网链接：<https://huggingface.co/>



10. Huggingface transformers

优点：

- 简单，能够快速上手。
- 适合于各类人群 - NLP研究员，NLP应用人员，教育工作者等。
- NLU/NLG SOTA 模型支持。
- 减少预训练成本，提供了30+预训练模型，100+语言 - 支持Pytorch与Tensorflow2.0 转换。



10. Huggingface transformers

各种任务的代表模型：

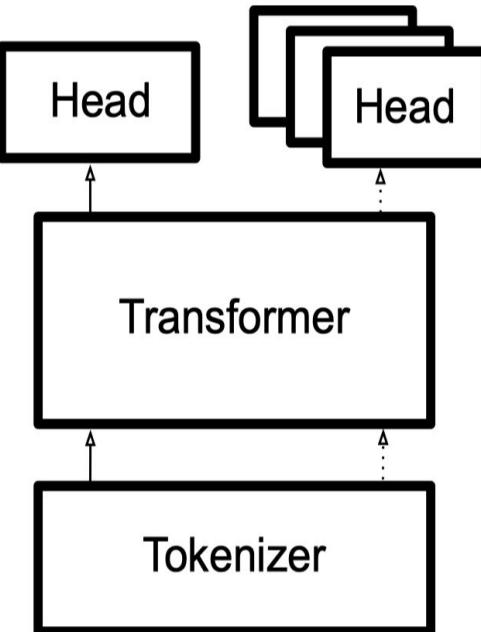
Model	Examples	Tasks
Encoder 编码器模型	ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa	Sentence classification, named entity recognition, extractive question answering 适合需要理解完整句子的任 务，例如句子分类、命名实体 识别（以及更一般的单词分 类）和提取式问答
Decoder 解码器模型	CTRL, GPT, GPT-2, Transformer XL	Text generation 解码器模型的预训练通常围绕 预测句子中的下一个单词。这 些模型最适合涉及文本生成的 任务
Encoder-decoder 序列到序列模型	BART, T5, Marian, mBART	Summarization, translation, generative question answering 序列到序列模型最适合围绕根 据给定输入生成新句子的任 务，例如摘要、翻译或生成式 问答。



10. Huggingface transformers

简单使用：

transformers 库包含了机器学习相关的主要三个部分：数据处理process data, 模型应用 apply a model, 和做出预测make predictions。分别对应的如下三个模块：Tokenizer, Transformers, 以及 Head。





10. Huggingface transformers

- Tokenizers 分词器，支持不同的分词。主要作用是将输入进行分词化后，并转化为相应模型需要的embedding。Tokenizer 类支持从预训练模型中进行加载或者直接手动配置。这些类存储了 token 到 id 的字典，并且可以对输入进行分词，和decode。
huggingface transformers 已经提供了如下图的相关tokenizer 分词器。用户也可以很轻松的对tokenizer 里的特殊字符进行更换，例如CLS/SEP。或者是对Tokenizer模型的字典进行大小修改等。Tokenizer 提供了很多有用的方法，例如padding, truncating，用户可以很方便的对其进行使用。
- Transformer transformers 指的是各种基于transformer结构的预训练语言模型，例如 BERT, GPT等。它将输入的sparse的序列，转化为上下文感知的的 contextual embedding。
- encoder 模型的计算图通常就是对模型输入进行一系列的 self-attention 操作，然后得到最后的encoder的输出。通常情况下，每个模型都是在一个文件中被定义完成的，这样方便用户进行更改和拓展。

主流NLP工具比较

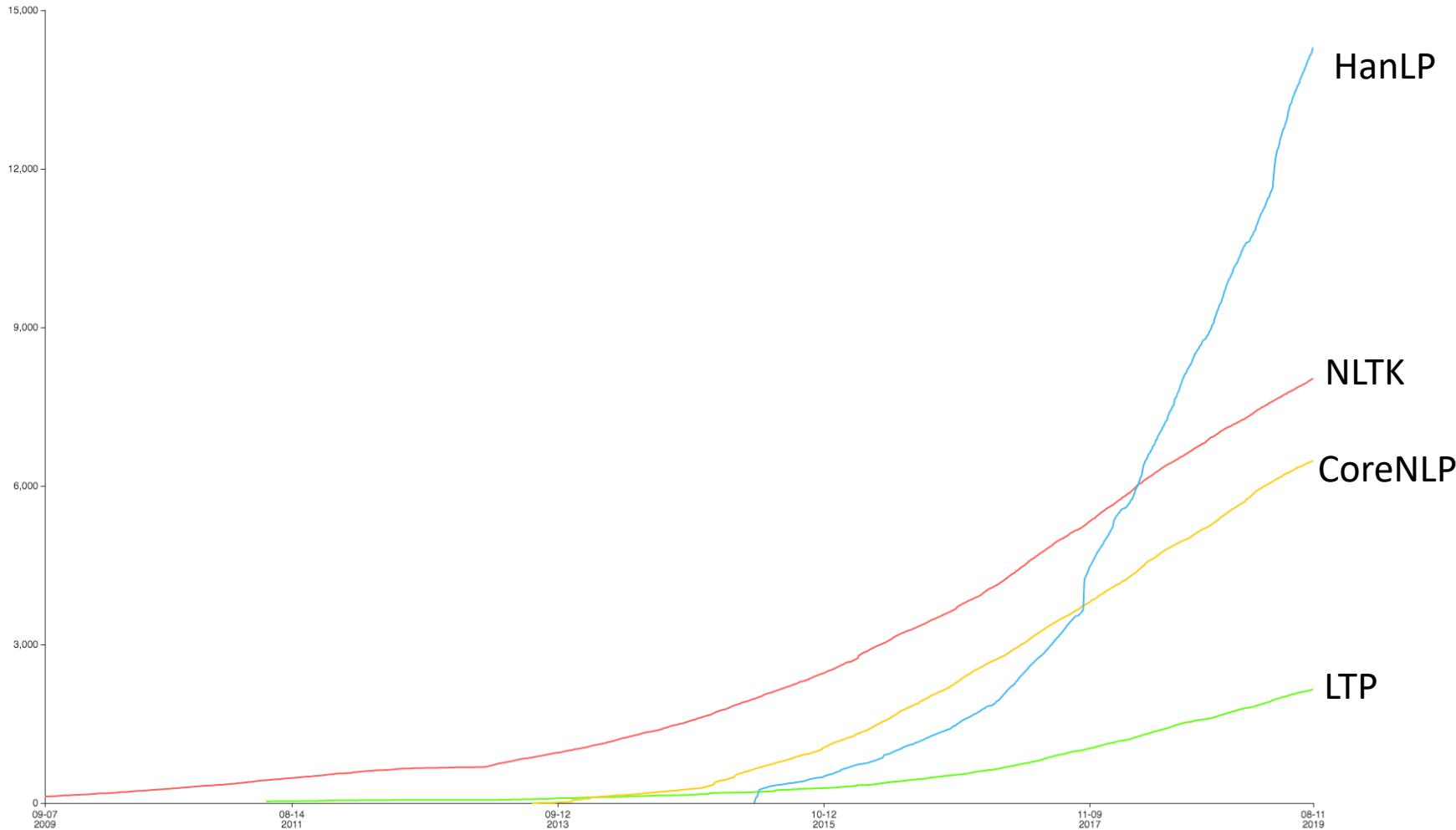
指标	NLTK	CoreNLP	LTP	HanLP
功能	词法分析、句法分析、语义分析、文本分类等	词法分析、句法分析、语义分析、关系抽取、指代消解、文本分类等	词法分析、句法分析、语义分析	词法分析、句法分析、关键词提取、文本分类等
语言	Python	Java等	C++、Python等	Java、Python等
速度	慢	较慢	较快	特别快
内存	占用多	占用特别多	省内存	省内存
精度	较准	准	准	较准
插件	无	Lucence	无	Lucence和Hadoop等
社区	8000	6400	2100	143 100
授权	Apache-2.0	Apache-2.0	商用付费	Apache-2.0



主流NLP工具比较

主流自然语言工具包star数量

HanLP NLTK CoreNLP LTP



[《自然语言处理入门》](#)



1. 6 NLP开发环境搭建

- Anaconda
- Jupyter Notebook
- Pycharm



Thank you!

I UGUK XODI