自然语言处理: 概览部分

An Overview of Natural Language Processing*

* 主要参考: Prakash M Nadkarni *et al.*, Natural language processing: an introduction.

崔冠宇

2018202147

cuiguanyu@ruc.edu.cn

信息学院
中国人民大学

2020 年 5 月 7 日

中国人民大學
RENMIN UNIVERSITY OF CHINA

目录

自然语言处理涉及的子问题

早期："理性主义"——基于语法规则的分析

发展："经验主义"——基于概率统计的分析

低等级的任务

低等级的 NLP 任务（及一些主要困难）包括：

1. 句子边界检测 (sentence boundary detection)：如 Mr. 等不能认为是句子结束。
2. 分词 (tokenization)：如 10mg/天。
3. 词性标注 (part-of-speech assignment to individual words, POS tagging)：如多词性词的处理。
4. 合成词的语素分解 (morphological decomposition)。
5. 浅解析 (shallow parsing, chunking)：识别短语。
6. 基于具体问题的分段 (segmentation)：划分意群。

高等级的任务

高等级的 NLP 任务（及一些主要困难）包括：

1. 拼写 / 语法错误识别与纠正 (spelling/grammatical error identification and recovery)：大多需要交互式、实时的。
2. 命名实体识别 (named entity recognition, NER)：次序变化、词性 / 单复数等特性转换、同义 / 多义词都构成挑战。
3. 单词消歧义 (word sense disambiguation, WSD)。
4. 否定与不确定性识别 (negation and uncertainty identification)：有些否定是显式的，有些则是隐式的。
5. 关系提取 (relationship extraction)：涉及指代、部分整体、子集超集等关系。
6. 时序推理 (temporal inference)：推断时间先后。
7. 信息提取 (information extraction, IE)：1-6 的结合体。

早期尝试——词对词翻译

自然语言处理 (Natural language processing, **NLP**) 始于 1950 年代，当时是作为人工智能和语言学的交集出现的。

早期的简单尝试有词对词的英俄机器翻译，但是由于一词多义、比喻的存在，闹了不少笑话，比如：

- "The spirit is willing, but the flesh is weak."（心有余而力不足。）

被翻译成了——

- "The vodka is agreeable, but the meat is spoiled." (???)

仿佛是前几天大火的"狗屁不通文章生成器"的作品。

早期尝试——形式语言

乔姆斯基 (Avram N. Chomsky) 于 1956 年提出了著名的形式语言的四类文法[1]：(我们的《离散数学》书上也有介绍)

类别	对应语言	产生式规则限制
0-型文法	递归可枚举语言	——
1-型文法	上下文相关语言	$\alpha A\beta \rightarrow \alpha\gamma\beta(\gamma \neq \varepsilon)$
2-型文法	上下文无关语言	$A \rightarrow \gamma$
3-型文法	正则语言	$A \rightarrow aB; A \rightarrow a$

这也促使了 1963 年 Backus-Naur 范式 (**BNF**) 的出现。BNF 可以描述上下文无关语言，后被广泛用于表示编程语言的语法。

[1] **Chomsky N.** Three models for the description of language. *IRE Trans Inf Theory* 1956;**2**:113-24.

早期尝试——形式语言

后来在乔姆斯基的正则文法的基础上，Kleene[2]定义了用来做字符串匹配搜索正则表达式，正则表达式被 UNIX 上 Ken Thompson 编写的 *grep* 工具最先支持。

1970 年代，词法分析器 (lexical-analyser, **lexer**) 生成器及分析器 (**parser**) 生成器（例如著名的 *lex/yacc*）出现[3]。它们的功能是接受正则表达式或 BNF 来生成 lexer 和 parser。

尽管上下文无关文法 (context-free grammar, **CFG**) 在理论上表达能力不如自然语言，但在实践中还是经常使用。比如众多编程语言被设计成上下文无关文法的一种变体——**LALR(1)**[4]的。

CFG 的相关内容会由后面展示的同学讲解。

---

[2] **Kleene SC.** Representation of events in nerve nets and finite automata. In: Shannon C, McCarthy J, eds. *Automata Studies.* Princeton, NJ: Princeton University Press, 1956.

[3] 词法分析器会将文本转化为一系列的词 (**token**)，并交给解析器检验合法性。

[4] **L**ook-**A**head, **L**eft-to-right, **R**ightmost, **1** token. 指的是从左向右扫描，最多向前看一个词，自底向上构造表达式。

局限性

自然语言具有的巨大的体量与不严谨多歧义的本性，导致使用纯粹人工编制的规则时遇到了两个问题：

- 词性分类以及各种规则繁杂，而且可以不断扩充，导致难以维护；
- 对于一些语法不严格，高度省略的情景（如医院、电报）难以正确识别。

于是，1980 年代，学者们改变了研究方向，基于统计的自然语言处理开始受到关注。

全新的发展方向

基于统计的 NLP 的发展方向由 Klein 总结过[5]（这里仅摘取部分）：

- 简单的、鲁棒性的近似替代深度的分析；
- 使用概率的机器学习方法更加突出；
- 使用大量做好标记的文本（语料库，corpora）来训练模型。

这样一来，更宽的规则取代了详细但又繁杂的规则，概率也被用来消除歧义或是构建决策树等。基于统计的方向在实践上取得了很好的成效。

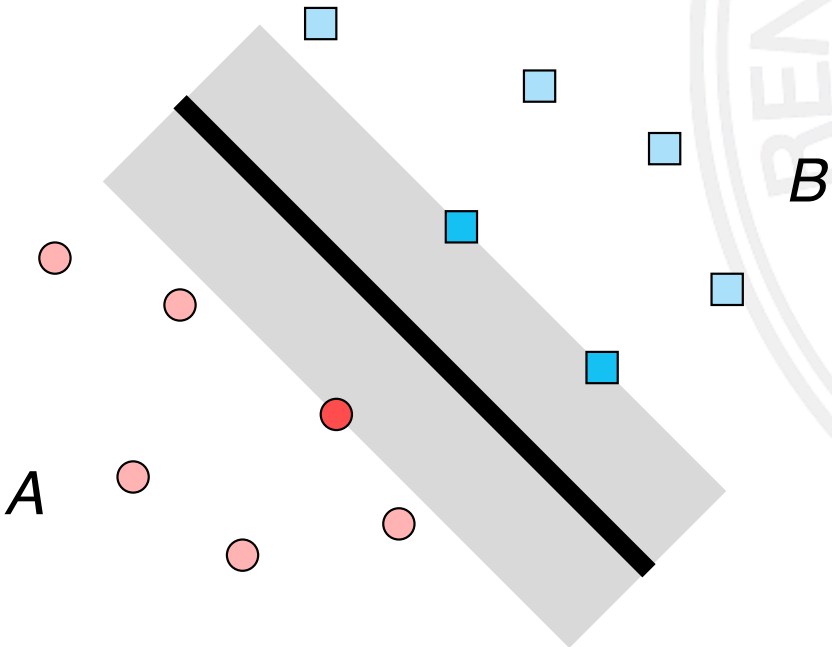
[5] **Klein D.** *CS 294-5: Statistical Natural Language Processing.* 2005. `http://www.cs.berkeley.edu/~klein/cs294-5` (accessed 2 Jun 2011).

数据驱动的方法：概览

部分机器学习的算法也被用在 NLP 相关的任务中，比如：

- 支持向量机 (support vector machine, **SVM**)

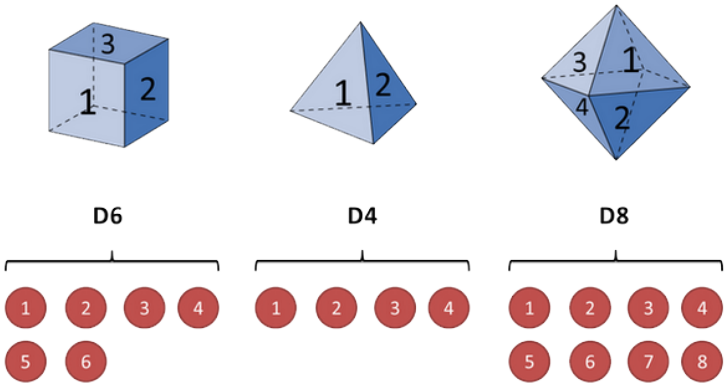
一种判别式学习方法。输入可能会经核函数 (kernel function) 处理，使得其能被线性分类。分类超平面 (hyperplane) 最大化输入与支持向量的距离，如下图所示：

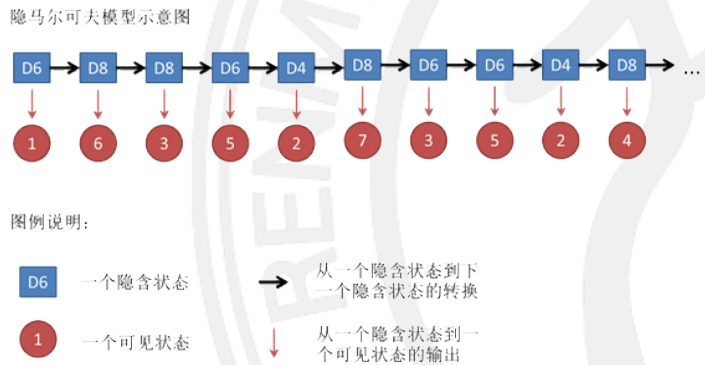


数据驱动的方法：概览

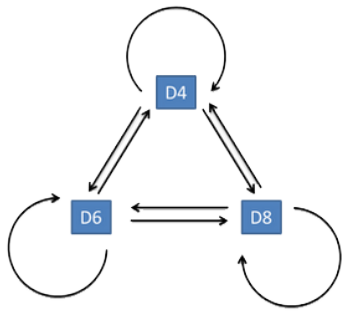
- 隐马尔可夫模型 (hidden Markov models, **HMMs**)

一种生成式学习方法。有多种状态，每次（随机）状态转换（随机）产生一个输出，但是我们只能观察输出，从而推断模型的内部状态转移情况。下图是《数据科学导论》课上的例子：





隐含状态转换关系示意图



数据驱动的方法：概览

HMM 有两条性质：

- 状态转换的概率仅仅取决于之前 N 个状态；
- 产生特定输出的概率仅仅取决于当前状态。

HMM 应用广泛，语音识别、生物信息（如多序列比对[6]和基因预测[7]）方面都有应用。更为细致的讲解请期待后面同学的展示。

- 除了这两种常见的模型外，还有条件随机场(conditional random fields, **CRFs**)、N-grams 等基于统计 / 概率的模型，这里不再赘述，有兴趣的同学请自行搜索。

---

[6] **Sonnhammer ELL,** Eddy SR, Birney E, *et al.* Pfam: Multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Res* 1998;**26**:320-2.

[7] **Lukashin A,** Borodovsky M. GeneMark.hmm: new solutions for gene finding. *Nucleic Acids Res* 1998;**26**:1107-15.

近年来的发展

近年来，人们逐渐开始引入深度学习来做自然语言处理研究，比如循环神经网络 (recurrent neural network, **RNN**) 已经是自然语言处理最常用的方法之一。这一点后面的同学会做详细介绍。

我的展示到此结束，谢谢大家！