数据科学方法
Data Science Methodology
自然语言处理(NLP)

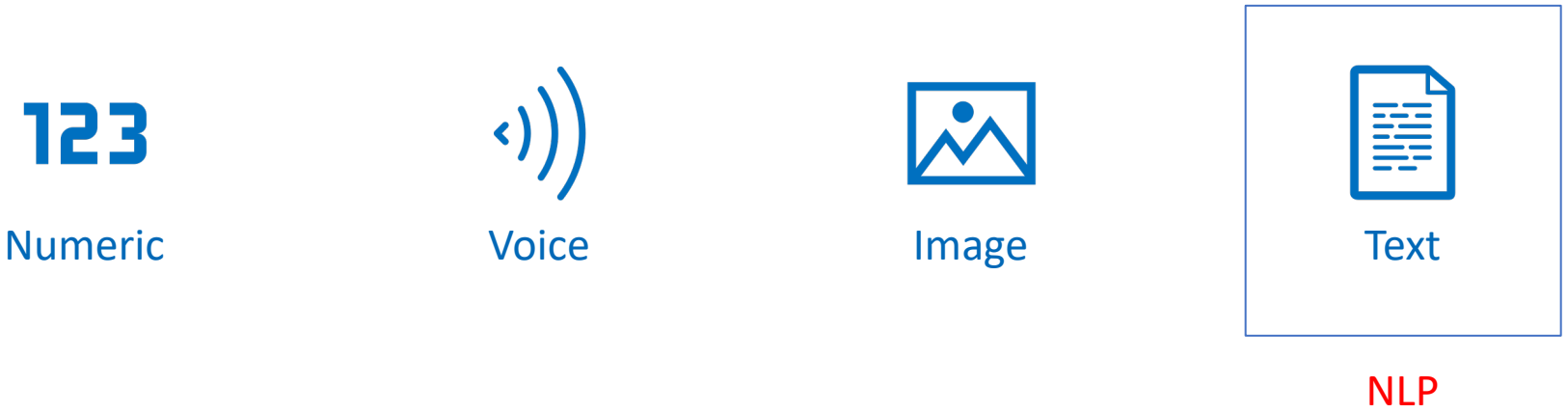
何铁科 (*hetieke.ml*)

南京大学智能软件工程实验室

www.iselab.cn

What is NLP?

常见的数据类型有哪些？



在数据科学领域，值类型、标称型或序数型均可采用离散化方法对数据进行预处理，而对于语音、图像等非结构化数据，则有专门的方法处理。而在NLP(Natural Language Processing)，即自然语言处理领域，其主要为能将自然语言，如英语、法语、德语、汉语等语言，转化为合理的可利用数据科学方法或统计学相关方法处理的数据形式。

自然语言：英语，汉语，法语，拉丁语等各类语言，程序语言，如Java、C++等不属于自然语言

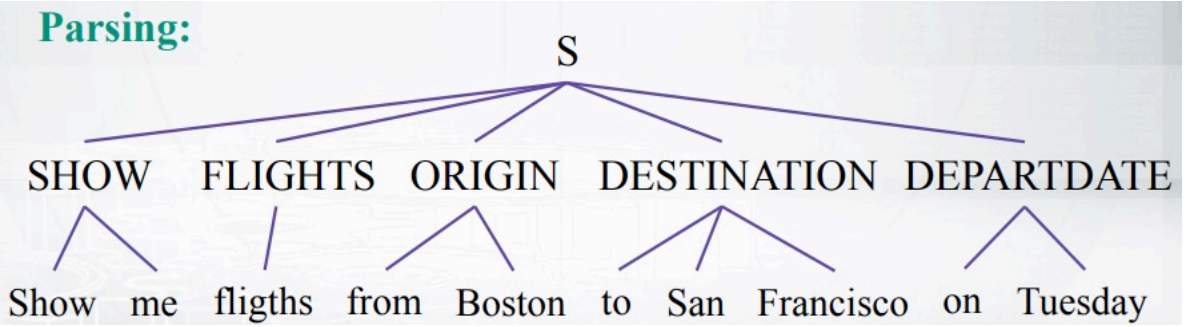
目的：如何有效地获取和利用以自然语言形式出现的信息?

Features in NLP

基于规则的特征：1) 固定的语法结构或词性分布规则，如<ORIGIN DESTINATION>表示源位置及目标位置，使用该规则可匹配位置变更信息；2) 固定的谓词逻辑规则，如<NO NO>规则，表示双重否定的规则，其最终的结果为肯定；3) 通用的前缀、后缀的规则特征，如ir, ed, un等前缀，后缀。

基于统计的特征：1) 基于大规模预料的特征，如频率特征；2) 基于词的编辑距离衡量的词相似性度量；

基于语言结构的特征：1) 词性特征；2) 命名实体特征等；



她 是 与 剧院 公司 的 一 颗 星。
她 是 剧团 的 明星。
删除4次：~~与~~ ~~公司~~ ~~一~~ ~~颗~~
替换2次: 剧院 → 剧团 星 → 明星
距离为6

ec24a9b0a4f3db4edefbecace6fcf2ee90532ac4

http://finance.china.com.cn/news/20170824/4365141.shtml

新疆食药监局：2批次食品抽检不合格 涉西域华新公司等　　中国网财经8月24日讯
格情况的通告》(2017年第86号)，涉及新疆维吾尔自治区两家食品生产企业。　　一、
国家食品药品监督管理总局组织的抽检中，标称乌鲁木齐市西域华新网络技术有限公司(以下
商标：西域关农；规格型号：250 g/袋；生产日期：2016-10-07；质量等级：/),经检验
经新疆维吾尔自治区食品药品监督管理局(以下简称新疆食药监局)执法人员现场检查，丰疆物
司。经查该批次产品已全部销售完毕。　　(三)目前丰疆物语公司已启动不合格产品
称乌鲁木齐丝路亚心商贸有限责任公司出品的新疆乌鲁木齐市兴华腾工贸有限公司西山加工厂生产的
(以下简称丝路亚心商贸)出品的新疆乌鲁木齐市兴华腾工贸有限公司西山加工厂(以下简称兴华
经检验霉菌项目不符合食品安全国家标准(霉菌检出值为130 CFU/g，标准值为≤25 CF
亚心商贸私自分装销售，并非委托兴华腾工贸生产加工。乌市食药监局已向杭州市市场
不合格产品进行召回，目前已召回不合格产品12袋，共计3公斤。

Features of Natural Languages

其他可用特征：

1. 词频特征；
2. 词的共现频率（见N-Gram）；
3. 虚词个数；
4. 词性特征；
5. 句子的句法、文法特征；
6. 句子长度；
7. 词编辑距离特征等；
8. 词情感特征（否定、肯定等）；

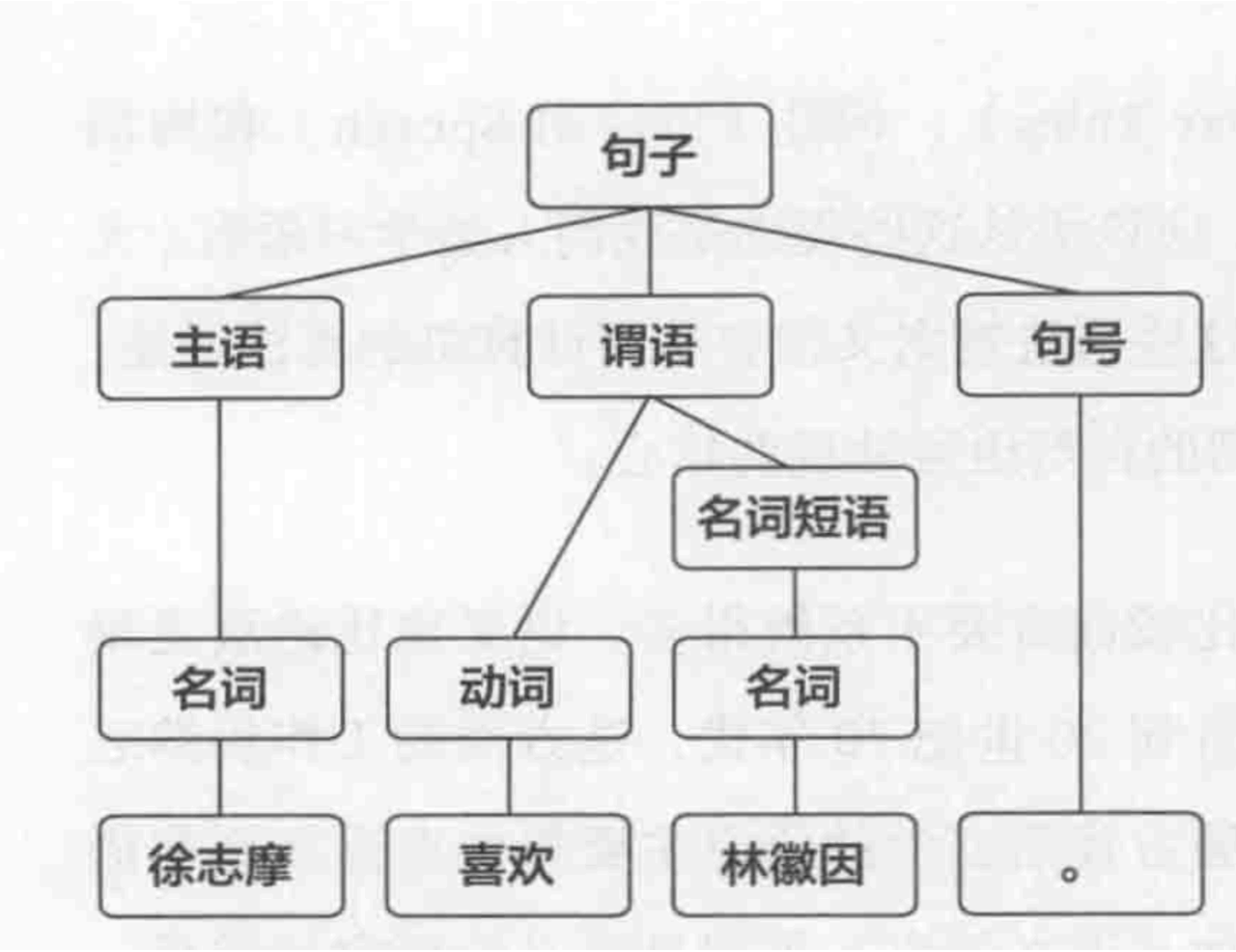
不同的任务可用不同的特征，特征的选取需根据具体的问题来选择。常见的NLP任务有：

- ✓ 垃圾邮件、短信识别；
- ✓ 文本相似性判断，用于文本聚类或信息检索；
- ✓ 文本情感分析；
- ✓ 文本质量评估；
- ✓ 文本主题提取；

From Rules to Statistics

语言学家对语言的规则等进行了总结和整理，便于人们学习，因此在学习语言时，需要了解语法规则(Grammar Rules)、词性(POS)和构词法(Morphologic)，这些都是**基于规则的自然语言处理**。

在基于规则的自然语言处理方法中，通常结合语法树对句子特征进行提取。



How to Extract Features：Word Segmentation

分词，即利用计算机技术将连续的字序列按照一定的规范重新组合成词序列的过程，其通常见于中文文本处理，由于英文文本可直接采用空格作为分隔符对词进行切分。

例：汽水不如果汁好喝

分词结果：汽水/ 不如/ 果汁/ 好喝/

问题：如何准确地获得分词序列？

Regular Match? 通常在处理文本时，我们会采用正则表达式以提取需要的部分，所以正则能否用于分词？

一个可行的操作：词典匹配，将词典中的词按照从长到短在文本中遍历匹配，直至所有的词都遍历之后。

Word Segmentation

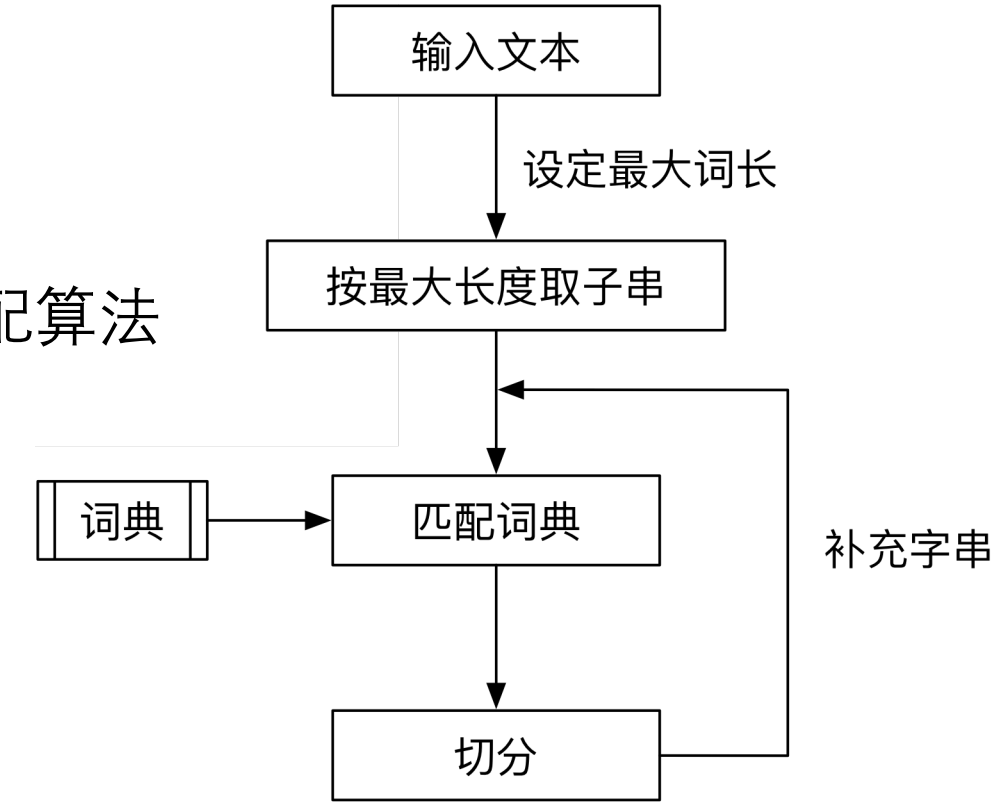
现有的分词算法可分为三大类：基于字符串匹配的分词方法、基于理解的分词方法和基于统计的分词方法。

基于字符串匹配的分词方法： 1）正向最大匹配法（由左到右的方向）；2）逆向最大匹配法（由右到左的方向）；3）最少切分（使每一句中切出的词数最小）；4）双向最大匹配法（进行由左到右、由右到左两次扫描）

基于字符匹配的算法为一种“规则分词”算法，其主要思想在于通过维护词典，在切分语句时，将语句的每个字符串与词表中的词逐一进行匹配，找到则切分，否则不予切分。

过程如下：

最大匹配算法



Word Segmentation：Maximum Matching

例子: 将句子 “今天来了许多新同事 ”
分词。 设最大词长为5。

今天来了许
今天来了
今天来
今天 ====》 得到一个词 – **今天**

来了许多新
来了许多
来了许
来了
来 ====》 得到一个词 – **来**

了许多新同
了许多新
了许多
了许
了 ====》 得到一个词 – **了**

许多新同事
许多新同
许多新
许多 ====》得到一个词 – **许多**

新同事
新同
新 ====》得到一个词 – **新**
同事 ====》得到一个词 – **同事**

最后正向最大匹配的结果是：/今天/来/了/许多/新/同事/

Word Segmentation： Reverse Maximum Matching

在上一节中，已经见过正向最大匹配算法，现在看下面一个例子。

输入例句：S1="计算语言学课程有意思" ；
定义：最大词长MaxLen = 5；S2= " "；分隔符 = “/”；
假设存在词表：…，计算语言学，课程，意思，…；

最大逆向匹配分词算法过程如下：
（1）S2=“”；S1不为空，从S1右边取出候选子串W=“课程有意思”；
（2）查词表，W不在词表中，将W最左边一个字去掉，得到W=“程有意思”；
（3）查词表，W不在词表中，将W最左边一个字去掉，得到W=“有意思”；
（4）查词表，W不在词表中，将W最左边一个字去掉，得到W=“意思”
（5）查词表， “意思”在词表中，将W加入到S2中，S2=" 意思/"，并将W从S1中去掉，此时S1="计算语言学课程有"；
（6）S1不为空，于是从S1左边取出候选子串W="言学课程有"；
（7）查词表，W不在词表中，将W最左边一个字去掉，得到W="学课程有"；
（8）查词表，W不在词表中，将W最左边一个字去掉，得到W="课程有"；
（9）查词表，W不在词表中，将W最左边一个字去掉，得到W="程有"；
（10）查词表，W不在词表中，将W最左边一个字去掉，得到W="有"，这W是单字，将W加入到S2中，S2=“ /有 /意思” ，并将W从S1中去掉，此时S1="计算语言学课程"；
（11）S1不为空，于是从S1左边取出候选子串W="语言学课程"；
（12）查词表，W不在词表中，将W最左边一个字去掉，得到W="言学课程"；
（13）查词表，W不在词表中，将W最左边一个字去掉，得到W="学课程"；
（14）查词表，W不在词表中，将W最左边一个字去掉，得到W="课程"；
（15）查词表， “意思”在词表中，将W加入到S2中，S2=“ 课程/ 有/ 意思/”，并将W从S1中去掉，此时S1="计算语言学"；
（16）S1不为空，于是从S1左边取出候选子串W="计算语言学"；
（17）查词表， “计算语言学”在词表中，将W加入到S2中，S2=“计算语言学/ 课程/ 有/ 意思/”，并将W从S1中去掉，此时S1=""；
（18）S1为空，输出S2作为分词结果，分词过程结束。

Word Segmentation： Bidirectional Maximum Matching

MM与RMM的比较：

MM从左向右依次按最长匹配；

RMM从右至左依次按最长匹配；

逆向最大匹配法比正向最大匹配法的误差要小。

统计结果表明，

单纯使用正向最大匹配的错误率为 1/169,

单纯使用逆向最大匹配的错误率为 1/245。

改进策略？

双向匹配，增强准确性。

双向匹配：将正向最大匹配法与逆向最大匹配法组合。先根据标点对文档进行粗切分，把文档分解成若干个句子，然后再对这些句子用正向最大匹配法和逆向最大匹配法进行扫描切分。如果两种分词方法得到的匹配结果相同，则认为分词正确，否则，按最小集处理（最少切分）。

Word Segmentation： Optimal Matching

基于规则的字符串匹配算法，都需要借助词典对文本内容进行匹配，因此字典大小，字典质量都会影响算法效率。基于正向匹配和逆向匹配算法，提出了最佳匹配算法来提升分词效率。

实现方向：在词典中按词频的大小顺序排列词条，以求缩短对分词词典的检索时间，达到最佳效果，从而降低分词的时间复杂度，加快分词速度。

缺点：

- 其空间复杂度有所增加

优点：

- 对提高分词精度没有影响，分词处理的时间复杂度有所降低。

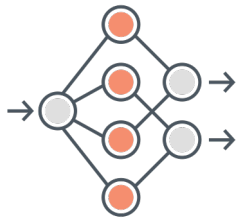
Word Segmentation Based on Understanding

基于理解的方法又称基于人工智能的分词方法，其基本思想就是在分词的同时进行句法、语义分析，利用句法信息和语义信息来处理歧义现象。它通常包括三个部分：分词子系统、句法语义子系统和总控部分。通常有如下几种方法：

- 专家系统分词法。
- 神经网络分词法。
- 神经网络专家系统集成式分词法。

专家系统分词

神经网络分词



集成式

Word Segmentation Based on Statistic

基于统计的方法思想为：词是稳定的组合，因此在上下文中，相邻的字同时出现的次数越多，就越有可能构成一个词。因此字与字相邻出现的概率或频率能较好反映成词的可信度。可以对训练文本中相邻出现的各个字的组合的频度进行统计，计算它们之间的互现信息。互现信息体现了汉字之间结合关系的紧密程度。当紧密程度高于某一个阈值时，便可以认为此字组可能构成了一个词。该方法又称为无字典分词。

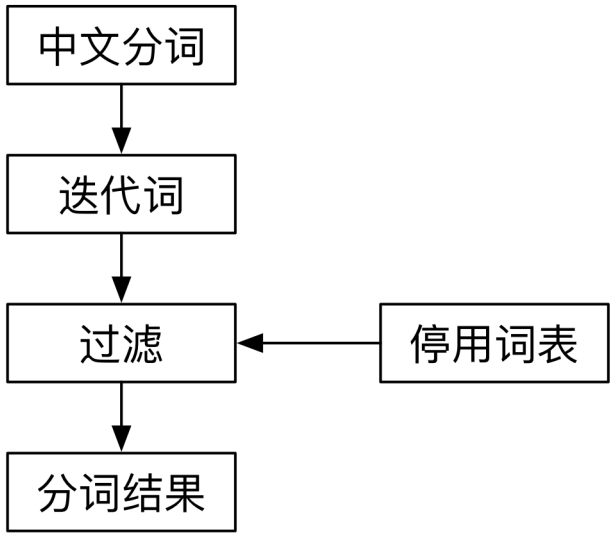
主要统计模型：N 元文法模型、隐Markov 模型和最大熵模型等。在实际应用中一般是将其与基于词典的分词方法结合起来，既发挥匹配分词切分速度快、效率高的特点，又利用了无词典分词结合上下文识别生词、自动消除歧义的优点。

Stop-Words Removal

什么是停用词？

- 功能词：‘the’、‘is’、‘at’、‘which’、‘on‘、的、是等；
- 词汇词：'want‘等

过程：



去停用词是在自然语言处理中常用的过程之一，其目的在于将句子中的无意义的词过滤，因此其过程可如左图所示。

Named Entity Recognition

NER，即命名实体识别，所谓的命名实体就是**人名**、**机构名**、**地名**以及其他所有以名称为标识的实体，也包括**日期**、**货币**等。

注：命名实体的定义与其处理的问题有关，如在医学领域，药物名、疾病名也是命名实体。

问题1：如何确定汉语的实体边界？

问题2：如何将可能的实体，划分到正确的类别中？

NER Based on Rules / Sequence Tagging

借助语言学专家，对自然语言中命名实体出现的规则进行总结，并编写模板，使用模板匹配实体。

从图中的标注中，可总结的规则：

实体 = <若干地名> + <若干其他成分> + <若干特征词>

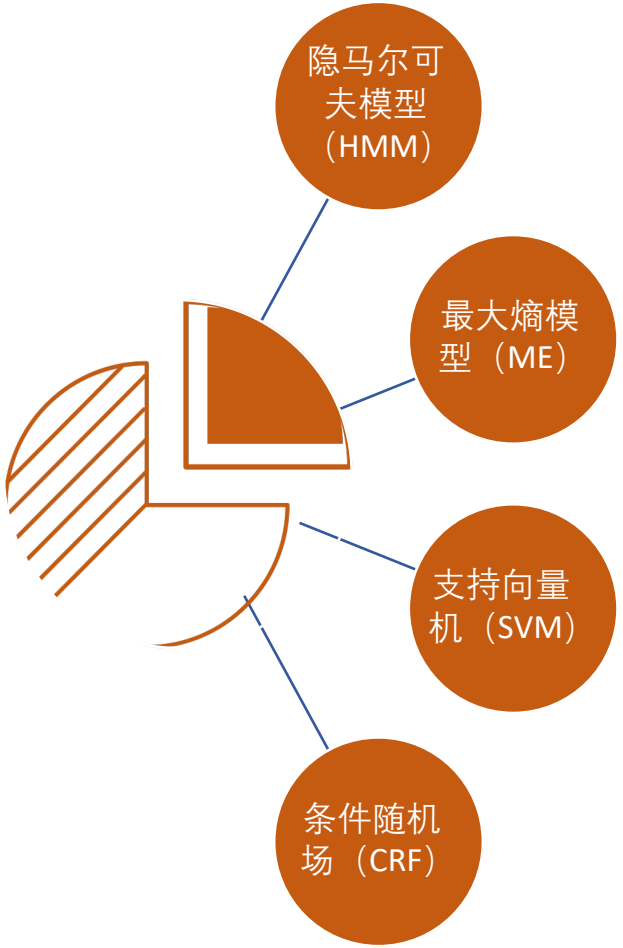
当含有地名，并含有诸如公司、工厂、XX局等关键字时，通常可匹配为一个命名实体。

NER Based on Statistic

基于规则的实体提取弊端：

- 仅适用于规律确定的实体；
- 需要领域专家制定规则；
- 规则不全面，无法最大限度识别实体。

基于规则 优化为 基于统计



优点：
避免了规则的构造和提取；利用统计规律，更具通用性；统计模型更加容易实现自动化。

Part of Speech Tagging

词性（POS）：指以词的特点作为划分词类的根据。词类是一个语言学术语，是一种语言中词的语法分类，是以语法特征（包括句法功能和形态变化）为主要依据、兼顾词汇意义对词进行划分的结果。

方法：
- ✓ 规则匹配
- ✓ 隐马尔可夫
- ✓ 条件随机场

同NER任务比较

相同点：都是类别识别；都是序列标注任务，部分方法可共用；

不同点：POS标注粒度更细，NER会标注词组短语

Other Processes

其他处理过程，如：

- 词根化（针对非汉语，如英语）；
- 去时态化（针对非汉语，如英语）；
- 语义角色标注（SRL）；
- CFG（上下文无关文法）；
- 句法树等。

Markov Chain

定义：马尔可夫链（Markov Chain, MC）是概率论和数理统计中具有马尔可夫性质（Markov property）且存在于离散的指数集（index set）和状态空间（state space）内的随机过程（stochastic process）。

特点：时间、状态都离散。

在马尔可夫链中，随机变量的状态转换不具备记忆性，即当前的状态仅与前一刻的状态有关。以数学表达式表示则为：

$$p(X_{t+1}|X_t, \dots, X_1) = p(X_{t+1}|X_t)$$

在马尔可夫的推广中，可将该规则推广到n阶，称为n-阶马尔可夫链，传统的马尔可夫链称为1-阶马尔可夫。

$$p(X_{t+1}|X_t, \dots, X_1) = p(X_{t+1}|X_t, \dots, X_k), 1 < k < t$$

Markov Chain

在马尔科夫链中，通常可用于从当前的状态，计算未来状态的概率。如计算天气，设第一天的下雨、晴天、多云概率为p，则第一天的概率分布为$p = (p_1 \quad p_2 \quad p_3), p \in \mathbb{R}^{1\times3}$，根据历史数据，可以观测出从某个天气到另一个天气的转化概率，如第一天为晴天，后一天为晴天的概率记为$P_{晴晴}$。由此，我们可以得到观测的转移概率矩阵，记为P。

$$P = \begin{pmatrix} P_{11} & P_{12} & P_{1n} \\ P_{21} & P_{22} & P_{2n} \\ P_{n1} & P_{n2} & P_{nn} \end{pmatrix}, P \in \mathbb{R}^{3\times3}$$

如上式，P_{ij}表示从状态i转化为状态j的概率，则可计算第二天的各天气概率为：$\mathrm{Q}_1 = p\times P, Q_1 \in \mathbb{R}^{1\times3}$，同理可得，$Q_2 = Q_1\times P$。

Markov Chain in N-Gram Model

在基于统计的分词中，N-Gram模型是一个较为常用且简单的模型，其中N-Gram模型关注的主要特征为词的共现特征，即共现频率高的多个字其更有可能组成一个词，如“人”和“民”，其通常一起出现，被称为共现，因此“人民”可作为一个词的切分。因此N-Gram需要能够计算词的共现频率，来判断该词能不能作为整体切分。

对于一个句子，我们通常用概率模型来衡量该句子的可能性：

$$P(S) = P(w_1, w_2, \dots, w_n)$$

对于句子来说，$P(w_1, w_2, \dots, w_n)$无法直接计算，需要使用条件概率对该表达式进行变换：$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \dots P(w_n|w_1, w_2, \dots, w_{n-1})$。

如何计算$P(w_n|w_1, w_2, \dots, w_{n-1})$？是否有必要？

N-Gram Model: unigram, bigram, trigram

回顾马尔可夫链：当前随机变量的状态概率分布仅与上一个状态有关。n-阶马尔可夫表示了什么？

$P(w_n|w_1, w_2, \ldots, w_{n-1})$表达的含义是什么？

基于马尔可夫假设得到的模型：

- ✓ unigram（只考虑相邻的1个词共现）；
- ✓ bigram（只考虑相邻的2个词共现）；
- ✓ trigram（只考虑相邻的3个词共现）；

在分词问题中，可将$P(S)$同理建模为$P(W)$，表示在一个序列中，W表示m个字符作为一个切分的概率。

由于引入了马尔可夫假设，则在计算$P(W)$时会更加简单，并且转移概率(转移概率见“马尔可夫链”)更容易从语料中得到。

Hidden Markov Model

隐马尔可夫模型（Hidden Markov Model，HMM）是统计模型，它用来描述一个含有隐含未知参数的马尔可夫过程。其难点是从可观察的参数中确定该过程的隐含参数。

需要了解的定义：

✓ 可见状态链（能够从实验或已有的数据中统计分析观测得出的序列）；

✓ 隐含状态（无法从观测得出但将影响观测结果的序列，即隐含参数）；

✓ 转换概率（从一个状态到另一个状态的可能性）；

适用的问题：

✓ 给定HMM求一个观察序列的概率（评估）

✓ 搜索最有可能生成一个观察序列的隐含状态序列（解码）

✓ 给定观察序列生成一个HMM（学习）

Hidden Markov Model

例：假设我手里有三个不同的骰子。第一个骰子是我们平常见的骰子（称这个骰子为D6），6个面，每个面（1-6）出现的概率是1/6。第二个骰子是个四面体（称这个骰子为D4），每个面（1-4）出现的概率是1/4。第三个骰子有八个面（称这个骰子为D8），每个面（1-8）出现的概率是1/8。开始掷骰子，我们先从三个骰子里挑一个，挑到每一个骰子的概率都是1/3。然后掷骰子，得到一个数字，1-8中的一个。不停的重复上述过程，我们会得到一串数字，每个数字都是1-8中的一个。如可能得到这么一串数字（掷骰子10次）：1 6 3 5 2 7 3 5 2 4，若需要求这10次投掷的过程中，选用骰子的序列，便是一个通过可见状态链求解隐含状态链的过程。

其可能的隐含状态链为：D6 D8 D8 D6 D4 D8 D6 D6 D4 D8

在NLP领域，需要将一段序列进行标注，如命名实体识别等，便是在已有的可观测序列（字词序列）中搜索到一个可能的隐含状态链（实体类别或词性类别）。

Similarities between Word and Word

距离定义：在一个空间内，用于衡量两个物体或位置间相距多远的描述。
距离的计算：**p阶闵可夫斯基距离**

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$

词的编辑距离：由于词由字符组成，而字符无法进行数值计算，因此闵可夫斯基无法使用。词的编辑距离，描述了一个词变为另一个词所需的操作次数，从某种程度而言，可以作为词于词之间的相似性。
词编辑距离存在的问题：
对于中文无较好支持。由于英文由字母组成，其编辑距离可由字母变换计算，而中文无法实现。
问题：词如何使用闵可夫斯基距离计算？

Similarities between Word and Word

问题：除范数以外，是否还有其他的相似性度量方法？

余弦相似度：

$$similarity = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

Tanimoto系数（广义Jaccard相似系数）：

$$T(A,B) = \frac{A \cdot B}{||A||^2 + ||B||^2 - A \cdot B} = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2} + \sqrt{\sum B_i^2} - \sum A_i B_i}$$

Word Vector: Transform Words to Computable Items

词向量化技术（在深度学习领域通常称为词嵌入），能够将自然语言中的词转化为可供计算的数学表示（通常是向量）。若考虑向量的维度为m，则可以使用m范数距离计算向量之间的距离，表示词间距离。

使用ID编码的词：每个词对应一个唯一的数值型ID。
缺点：数值型计算无法反映词的真实距离

常用词向量技术：One-Hot、词袋模型（BoW）

One-Hot优点：

✓ 解决了词到数值空间的映射问题；
✓ 编码不同，但是计算得到的结果相同；

BoW优点：

✓ 考虑了词频特征，能够关注频繁的项；
✓ 解决了One-Hot “一视同仁” 的弊端；

Word Vector: One-Hot Encoding

One-Hot方法：将所有的词使用编码的方式构造一个词典，词典中词的数量作为One-Hot编码的维度数，每个维度表示一个词，每个词在其对应的维度上出现时，该维度值为1.

例：给定一个词典，大小为5，有词“你”，“我”，“他”。
你：[1, 0, 0, 0, 0],
我：[0, 1, 0, 0, 0],
他：[0, 0, 1, 0, 0]

问题，由于词典的构建不同，那么在编码时，为1的位也不同，是否同直接编码一样产生距离计算结果不同的问题？

假设使用1-范数距离来衡量词与词之间的远近，则“你”“我”的距离为2。此时更换编码顺序，将“你”编码为[0, 0, 0, 1, 0]，此时得到的距离依然为2。因此可以避免前面所介绍的问题。

缺点：
- ✗ 没有考虑词的频率，使得任意两个词之间的距离相同；
- ✗ 矩阵稀疏，维度可能太高，需要降维处理；
- ✗ 没有考虑上下文，词被认为是独立的。

Bag of Word (BoW)

BoW方法思想：句子（或文档）由词组成，词出现的次数能够反映该句子的部分含义。

BoW方法：对所有的词构造词典，形成ID与词的映射。以词典大小作为向量维数，并统计句子（或文档）中每个词的出现频率，将词的频率填充到向量中对应的维度上得到该句子（或文档）向量。

1. 构造词典，每个词给定唯一编码，如1:John，获得词典，大小为10；
2. 统计词典中每个词在句子中出现的次数，如s_1中movies次数为2；
3. 将不同词出现的次数，填充到对应的维度上，得到最终向量。

例：

有两个句子分别为$s_1 =$ “$John\ likes\ to\ watch\ movies, Mary\ likes\ movies\ too$” 与 $s_2 =$ “$John\ also\ likes\ to\ watch\ football\ games$”，其BoW向量为？

	also	football	games	john	likes	mary	movies	to	too	watch
s1 = [	0,	0,	0,	1,	2,	1,	2,	1,	1,	1]
s2 = [	1,	1,	1,	1,	1,	0,	0,	1,	0,	1]

Bag of Word (BoW)

优点：

✓ 考虑了词频特征，能够更好地反映词在句子或文档中的分布情况；

✓ 基于统计的特征提取，通用性更好；

✓ 简单易用，有一定可用性和有效性；

缺点：

✗ 不考虑词序特征、文法、句法特征，有信息丢失；

✗ 矩阵（向量）稀疏，运算量大；

Term Frequency and TF-IDF

词频(Term Frequency, TF)定义：某个词在语料库中出现的频率，用以表示语料库中各词的分布情况。

$$TF(w) = \frac{\text{单词}w\text{在文章中出现的次数}}{\text{文章的单词总数}}$$

BoW模型是利用TF特征构建的一种向量化模型。

除词频外，还有另外一个频率，即逆文本频率(Inverse Document Frequency, IDF)：

$$IDF(w) = \log\left(\frac{\text{语料库中文档的总数}}{\text{包含词}w\text{的文档数}+1}\right)$$

TF-IDF

NLP领域的一个定理：

LAW: Zipf's Law, given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.

根据上述定理，我们可以导出TF-IDF定义：

$$TF - IDF(w) = TF(w) \times IDF(w)$$

解决的问题：相比于BoW，其词的分布特征更加准确，TF-IDF值反映了该词在该文档中的重要程度。

TF-IDF

TF-IDF能够反映文档中词对文档的贡献度，那么其能应用在哪些领域呢？

1. 文本挖掘领域（Text Mining）；
2. 信息检索领域（Information Retrieval）

TF-IDF的问题：

✗ 忽略了位置信息，不能完全地进行语义建模；

✗ 假设词是独立的，没有考虑上下文信息；

✗ 无法处理在文章中出现次数少，但是确是文档内容核心的情况；

Topic Model Based on TF-IDF

主题模型（Topic Model）：一种以非监督学习的方式对文集的隐含语义结构（latent semantic structure）进行聚类（clustering）的统计模型。

TF-IDF在主题模型中的应用：

计算文档中各词的TF-IDF值，选取TF-IDF高的词作为文档的主题表示。

方法：

1. 构建词典；
2. 统计各词在文档、语料库中的TF值及IDF值；
3. 计算TF-IDF值，并倒排序；
4. 选取TF-IDF值高的项，并在词典中查找对应词，作为主题词。

Topic Model Based on TF-IDF

例：

假设100篇文档有10000个词，研究某篇500词且”出现了20次，那么他们的TF都是20/500=章，每篇都出现了“而且”，因此它的IDF就出现了10篇，那么它的IDF就是log10=1,他的T且”更加重要。

如何可视化主题信息？

词云

Extract Features from Natural Language: Term-Document Matrix

前面我们提到，通过计算TF-IDF的值可以提取出每个词对文章的贡献度，但其反映了词的特征，如何提取整个文本的特征呢？

词项-文档矩阵（Term-Document Matrix）：通过计算TF-IDF值，来构建矩阵，矩阵中的每个元素值代表了相应行上的词项对应于相应列上的文档的权重，即这个词对于这篇文章来说的重要程度。

基本过程：

1. 读入文档；
2. 分词；
3. 建立字典(得到ID:词的映射，ID将作为矩阵行的索引)；
4. 计算TF-IDF值；
5. 得到Term-Document Matrix

Extract Features from Natural Language: Co-occurrence Matrix

前面我们提到，TF-IDF和One-Hot编码均有一个弊端：认为词是独立存在，不考虑词的位置信息及上

反观N-Gram模

其前一个词的

一种模型，其

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

https://blog.csdn.net/ha[illegible]335156

Extract Features from Natural Language: Co-occurrence Matrix

例：

设语料库如为：”I like deep learning.” , “I like NLP.” , “I enjoy flying.”，利用bigram求出共现矩阵。

设置窗口大小为1，分别滑动，判断每个bigram组合的出现频率，如”I like”为2, “I enjoy”为1，”like deep”为1，依次滑动窗口，求出所有bigram项的频率。则有如图所示的**共现矩阵**。

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

https://blog.csdn.net/ha[illegible]335156

Word Vector Based on Co-occurrence Matrix

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

https://blog.csdn.net/ha[illegible]335156

基于行（或列）的词向量，可以发现，对于like和enjoy，其欧式距离或1-范数距离较小，说明两词含义相近，符合预期，可用于比较词与词之间的相似度。

从矩阵中可以发现，每一列（或每一行）都可提取到一个向量表示，该向量表示有何意义？

从意义上考虑，每一行（或每一列）都表示了某个词同其他各词共同出现的频率。该频率信息表示了词与词的共现特征，则建模了该词的语义环境，因此其可用于作为该词语义向量。

Further Topic Models : LDA、LSA

隐含狄利克雷分布（Latent Dirichlet Allocation, LDA）：

一种主题模型（topic model），它可以将文档集中每篇文档的主题按照概率分布的形式给出。

LDA的理论依据：

1. 一个函数：gamma函数
2. 四个分布：二项分布、多项分布、beta分布、Dirichlet分布
3. 一个概念和一个理念：共轭先验和贝叶斯框架
4. 两个模型：pLSA、LDA、
5. 一个采样：Gibbs采样

Further Topic Models : LDA、LSA

隐含语义分析（Latent Semantic Analysis, LSA）：

Scott Deerwester, Susan T. Dumais等人在1990年提出来的一种新的索引和检索方法。

该方法和传统向量空间模型(vector space model)一样使用向量来表示词(terms)和文档(documents)，并通过向量间的关系(如夹角)来判断词及文档间的关系；

原理核心：

使用奇异值分解（SVD）结合降维来去除噪音，从而提取隐含语义。

1. 分析文档集合，建立Term-Document矩阵。
2. 对Term-Document矩阵进行奇异值分解。
3. 对SVD分解后的矩阵进行降维，也就是奇异值分解一节所提到的低阶近似。
4. 使用降维后的矩阵构建潜在语义空间，或重建Term-Document矩阵。

Application in Information Retrieval Domain

主题模型能够提取文档中的主题词，当获得主题词后，便可利用相似性度量方法，来计算输入与语料中文章的匹配程度，即信息检索。

信息检索（Information Retrieval）：信息按一定的方式进行加工、整理、组织并存储起来，再根据信息用户特定的需要将相关信息准确的查找出来的过程。

通过主题模型，可获得文档的语义特征，如LDA可获得一个概率分布，利用该特征结合距离度量办法，如KL距离等，可计算出目标查询文本与文档之间的语义相似度，并对其做排序，获得最终的检索结果。此外信息检索还有更多的模型，如布尔模型等，信息检索模型详细信息，可根据需要自行检索相关文献。

Measurement of Information

信息熵：用于衡量信息的不确定性。对于一个随机变量，其计算方式如下：

$$H(X) = -\sum_{x \in X} P(x) log P(x)$$

由信息熵可以引出KL散度的指标，其也是用来衡量距离的一种手段，一般地，其被用来衡量两个分布之间的差异（详见数据热力学）。

在自然语言中，可使用KL来衡量距离。

在之前，我们已经介绍过使用条件概率来建模和评估句子和自然语言的质量，而在信息领域，则可以通过熵，使用**困惑度(perplexity)**来衡量NLP的质量：

$$PP(S) = P(w_1, w_2 \dots w_n)^{-\frac{1}{N}}$$

在主题模型调参时，也使用困惑度来衡量主题提取的质量。

TF-IDF in Scrawler

TF-IDF模型可基于统计特征提取出文档中的关键词，从而达到主题提取的目的，而主题模型可利用在信息检索领域，因此TF-IDF也可用于网络爬虫。

搜索引擎的网络爬虫主要是通过爬取网页，并从网页中提取关键信息，建立索引，从而能够更好地建立搜索引擎的页面相关度信息。

而TF-IDF可为信息检索提供关键词信息，通过词的相关性分析，从而实现查询结果的相关匹配。

在前面的过程中，TF-IDF基于Zipf‘s定理，而从信息论的角度来看，其可从信息量的角度来推导该关系：

$$I(w) = -P(w)\log P(w)$$

$$= -\frac{TF(w)}{N}\log\frac{TF(w)}{N}$$

$$= \frac{TF(w)}{N}\log\frac{N}{TF(w)}$$

N表示语料库大小，对该计算式进一步优化，可得 $TF-IDF = I(w) - TF(w)\log\frac{M}{c(w)}$，$M$为词个数 $c(w)$为词出现的频度

Maximum Entropy Model

最大熵模型(Maximum Entropy Model, MEM):在满足所有已知条件的情况下，保留最大的不确定性，即保持熵最大。

例：对于一个骰子，其每个面朝上的概率为多少？若4朝上的概率为$\frac{1}{4}$，则其他面朝上的概率为多少？

$\frac{1}{6}$、$\frac{3}{20}$

在NLP中，假设已知拼音"Wang-Xiao-Bo"，如何确定"王小波"还是"王晓波"？通过使用上下文信息，可以确定两种可能的结果，通过主题才可以从两者中确定一种，因此可以结合最大熵模型：

$$P(w_3|w_1,w_2,s)$$

$$=\frac{1}{Z(w_1,w_2,s)}e^{\lambda_1(w_1,w_2,w_3)+\lambda_3(s,w_3)}$$

其中w_3表示要预测的词，w_1,w_2表示上下文，s表示主题，Z为调和因子，保证概率相加为1。最大熵模型通过指数函数来建模，并通过大量的预料进行训练得到参数λ,Z。

Co-occurrence in Input

在现代输入法中，通常会给出联想输入，即输入一个词后，给出可能的下一个词的联想结果。**利用共现数据，可以方便地实现输入预测**。

例：在已有的预料中，可以统计出，“我”,”的”共现概率大，可以快速的给出在输入“我”后应联想出“的”。而倘若在”我”之前输入了”要”，则应考虑”要我”,”的”和”去”的共同特征，可根据条件概率模型，来计算P(“的”|“要我”)和P(“去”|“要我”)的概率大小，从而对预测结果进行排序。

数据科学方法
Data Science Methodology
NLP实践

何铁科 (*hetieke.ml*)

南京大学智能软件工程实验室

www.iselab.cn

Toolkits in NLP

1. NLTK(Natural Language Toolkit)
2. Spacy
3. Scikit-learn
4. Gensim
5. Stanford CoreNLP(支持中文)
6. jieba(支持中文)
7. 哈工大LTP(支持中文)
8. 中科院大学NLPIR(支持中文)
9. 清华THULAC(支持中文)
10. 北京大学pkuseg(支持中文)

NLTK可以执行诸如分词，词形还原，词干提取，解析，词性标注等任务。该库包含的工具可用于几乎所有NLP任务；

Spacy支持NLTK同样的任务；

Scikit-Learn是机器学习提供了一个包罗万象的工具库，其中包含了一些用于文本预处理的工具，并非专业NLP工具；

Gensim实现了常用的主题模型，文档相似度计算等算法，也可构建TF-IDF等特征；

Stanford CoreNLP为Java编写并且提供Python API的NLP工具，可执行分词，NER，POS等常见任务，并且有中文训练预料支持，**NLTK中可与CoreNLP整合**；

其他中文分词工具，效果各不相同，但基本任务相同，均可进行分词、NER、POS，以及进行文法分析等。

Word Segmentation with NLTK

1.输入一个段落，分成句子（Punkt句子分割器）

```
import nltk
import nltk.data
def splitSentence(paragraph):
  tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
  sentences = tokenizer.tokenize(paragraph)
  return sentences
if __name__ == '__main__':
  print splitSentence("My name is Tom. I am a boy. I like soccer!")
```

结果为['My name is Tom.', 'I am a boy.', 'I like soccer!']

2.输入一个句子，分成词组

```
from nltk.tokenize
import WordPunctTokenizer
def wordtokenizer(sentence): #分段
  words = WordPunctTokenizer().tokenize(sentence)
  return words
if __name__ == '__main__':
  print wordtokenizer("My name is Tom.")
```

结果为['My', 'name', 'is', 'Tom', '.']

POS with NLTK

NLTK(3.2.5)中提供了一些已经标注好词性的文本语料，通过下面代码可以查看：

```
import nltk
nltk.corpus.brown.tagged_words()

outputs:
[(u'The', u'AT'), (u'Fulton', u'NP-TL'), ...]
```

支持的语料：

```
nltk.corpus.sinica_treebank.tagged_words()
nltk.corpus.indian.tagged_words()
nltk.corpus.mac_morpho.tagged_words()
...
```

对文本进行POS：

```
text = nltk.word_tokenize('They refuse to permit us to obtain the refuse permit')
nltk.pos_tag(text, tagset='universal')

outputs:
[('They', u'PRON'), ('refuse', u'VERB'), ('to', u'PRT'), ('permit', u'VERB'), ('us', u'PRON'), ('to',
u'PRT'), ('obtain', u'VERB'), ('the', u'DET'), ('refuse', u'NOUN'), ('permit', u'NOUN')]
```

POS with NLTK Based on Regular

```
patterns = [
  (r'.*ing$', 'VBG'),
  (r'.*ed$', 'VBD'),
  (r'.*es$', 'VBZ'),
  (r'.*ould$', 'MD'),
  (r'.*\'s$', 'NN$'),
  (r'.*s$', 'NNS'),
  (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),
  (r'.*', 'NN’)
]
regexp_tagger = nltk.RegexpTagger(patterns)
regexp_tagger.tag(brown_sents[3])
outputs:
[(u'``', 'NN'), (u'Only', 'NN'), (u'a', 'NN'), (u'relative', 'NN'), (u'handful', 'NN'),
(u'of', 'NN'), (u'such', 'NN'), (u'reports', 'NNS'), (u'was', 'NNS'), (u'received', 'VBD'),
(u"''", 'NN'), (u',', 'NN'), (u'the', 'NN'), (u'jury', 'NN'), (u'said', 'NN'), (u',', 'NN'),
(u'``', 'NN'), (u'considering', 'VBG'), (u'the', 'NN'), (u'widespread', 'NN'), (u'interest',
'NN'), (u'in', 'NN'), (u'the', 'NN'), (u'election', 'NN'), (u',', 'NN'), (u'the', 'NN'),
(u'number', 'NN'), (u'of', 'NN'), (u'voters', 'NNS'), (u'and', 'NN'), (u'the', 'NN'),
(u'size', 'NN'), (u'of', 'NN'), (u'this', 'NNS'), (u'city', 'NN'), (u"''", 'NN'), (u'.',
'NN')]
```

POS with NLTK Based on N-Gram

```
unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
unigram_tagger.evaluate(brown_tagged_sents)

outputs:
0.9349006503968017

print "\n".join([cate + "\t" +
str(unigram_tagger.evaluate(brown.tagged_sents(categories=cate))) for cate in
brown.categories()[:10]])

outputs:
adventure       0.787891898128
belles_lettres  0.798707075842
editorial       0.813940653204
fiction         0.799147295877
government      0.807778427485
hobbies         0.771327949481

t0 = nltk.DefaultTagger('NN')
t1 = nltk.UnigramTagger(brown_tagged_sents, backoff=t0)
t2 = nltk.BigramTagger(brown_tagged_sents, backoff=t1)
t2.evaluate(brown_tagged_sents)

outputs:
0.9730592517453309
```

NER with NLTK

```
sentence = "Let's meet tomorrow at 9 pm";
tokens = nltk.word_tokenize(sentence)
pos_tags = nltk.pos_tag(tokens)
print nltk.ne_chunk(pos_tags, binary=True)

tokenized_sentences = [nltk.word_tokenize(sentence)
sentences and use nltk's Named Entity Chunker
tagged_sentences = [nltk.pos_tag(sentence) for sent
ne_chunked_sents = [nltk.ne_chunk(tagged) for tagge
named entities
named_entities = []
for ne_tagged_sentence in ne_chunked_sents: for tag
extract only chunks having NE labels
  if hasattr(tagged_tree, 'label'):
  entity_name = ' '.join(c[0] for c in tagged_tree.
  entity_type = tagged_tree.label() # get NE categ
  named_entities.append((entity_name, entity_type))
  named_entities = list(set(named_entities))
```

	Entity Name	Entity Type
0	FIFA	ORGANIZATION
1	Central America	ORGANIZATION
2	Belgium	GPE
3	Caribbean	LOCATION
4	Asia	GPE
5	France	GPE
6	Oceania	GPE
7	Germany	GPE
8	South America	GPE
9	Denmark	GPE
10	Zürich	GPE
11	Africa	PERSON
12	Sweden	GPE
13	Netherlands	GPE
14	Spain	GPE
15	Switzerland	GPE
16	North	GPE
17	Europe	GPE

Lemmatization and Stemming support by NLTK

词形还原（lemmatization），是把一个词汇还原为一般形式（能表达完整语义），方法较为复杂；而**词干提取（stemming）**是抽取词的词干或词根形式（不一定能够表达完整语义），方法较为简单。

```
# Porter Stemmer基于Porter词干提取算法
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()
porter_stemmer.stem('leaves')

Outputs:
leav

from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize('leaves')

Outputs:
leaf
```

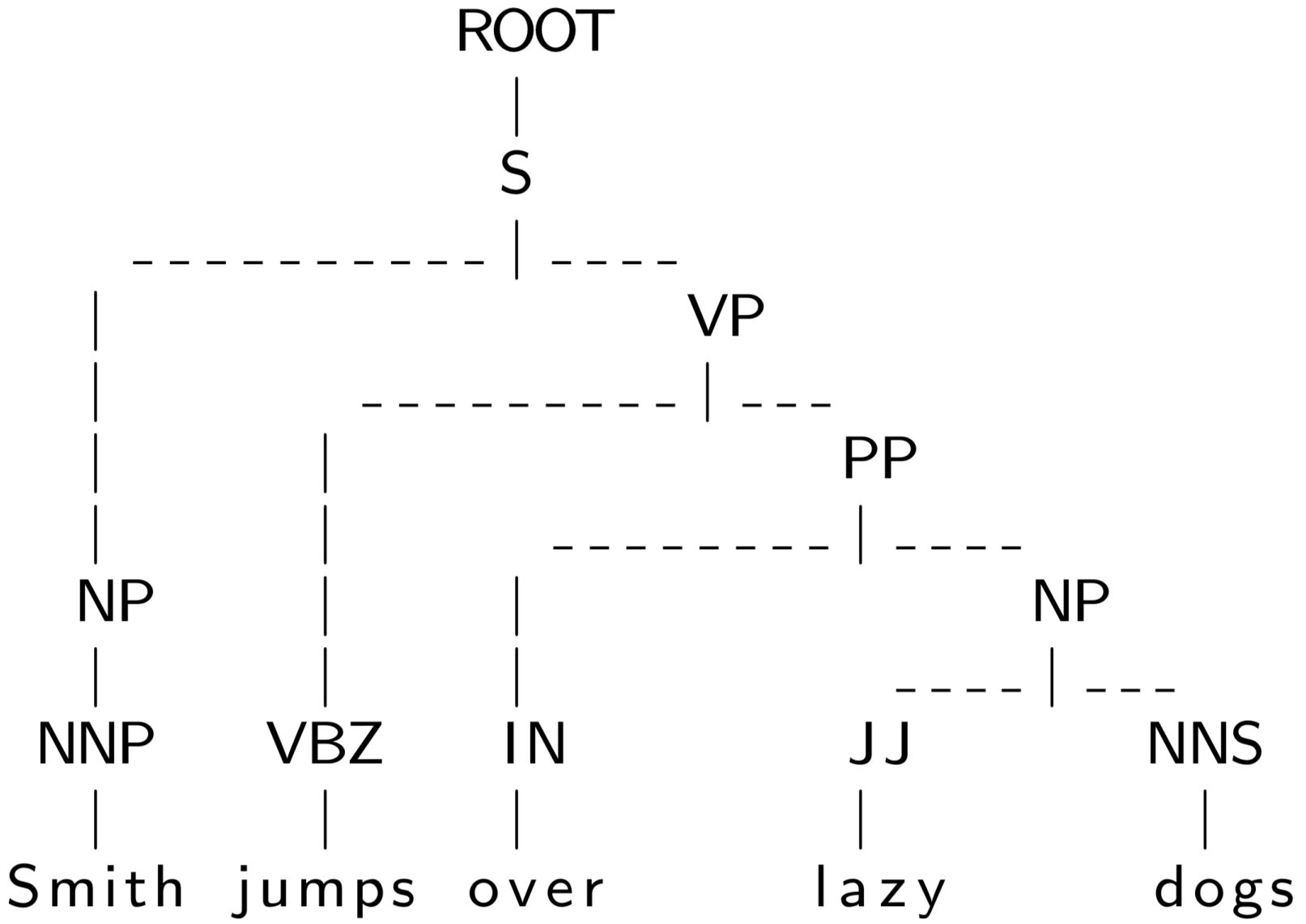
Grammar Parsing in NLTK

NLTK中还提供了一定的文法分
可结合CoreNLP提供的接口，

```
from nltk.parse.stanford
chi_parser = StanfordPars
parser.jar",r"E:\tools\st
models.jar",r"E:\tools\st
sent = u'北海 已 成为 中国
print(list(chi_parser.par
```

此外在CoreNLP中还提供使
独启动并在NLTK中进行网络
消耗，如：

```
from nltk.parse.corenlp i
parser = CoreNLPParser(ur
parse , = parser.raw_pars
print(parse)
```



Gensim Practice

Gensim是一个强大的NLP工具，其提供了TF-IDF、词袋模型、主题模型等一系列NLP相关模型及方法。

```
from gensim import corpora, models, similarities
from pprint import pprint


def GenDictandCorpus():
    documents = ["Human machine interface for lab abc computer applications",
                 "A survey of user opinion of computer system response time",
                 "The EPS user interface management system",
                 ...]

    texts = [[word for word in document.lower().split()] for document in documents]

    # 词典
    dictionary = corpora.Dictionary(texts)
    # 词库，以(词，词频)方式存贮
    corpus = [dictionary.doc2bow(text) for text in texts]
    print(dictionary)
    print(corpus)
    return dictionary, corpus
```

Gensim Practice

有了相应的词频特征后，便可计算出相应词的的TF-IDF值：

```
def Tfidf():
    dictionary, corpus = GenDictandCorpus()

    # initialize a model
    tfidf = models.TfidfModel(corpus)
    # print(tfidf)

    # Transforming vectors
    # 此时，tfidf被视为一个只读对象，可以用于将任何向量从旧表示（词频）转换为新表示（TfIdf实值权重）
    doc_bow = [(0, 1), (1, 1)]
    # 使用模型tfidf，将doc_bow(由词,词频)表示转换成(词,tfidf)表示
    # print(tfidf[doc_bow])

    # 转换整个词库
    corpus_tfidf = tfidf[corpus]
    for doc in corpus_tfidf:
        print(doc)

    return corpus_tfidf
```

Gensim Practice

在gensim中，还提供了主题模型相关的模型：

```
def LDA():
    dictionary, corpus = GenDictandCorpus()
    ldamodel = models.LdaModel(corpus, id2word=dictionary, num_topics=2)

    ldamodel.print_topics()
    pprint(ldamodel.print_topics())

# 潜在语义索引(Latent Semantic Indexing,以下简称LSI)，有的文章也叫Latent Semantic
Analysis（LSA）
# LSI是基于奇异值分解（SVD）的方法来得到文本的主题的
def LSI():
    dictionary, corpus = GenDictandCorpus()
    corpus_tfidf = Tfidf(corpus)
    lsi = models.LsiModel(corpus_tfidf, id2word=dictionary, num_topics=2)
    corpus_lsi = lsi[corpus_tfidf]
    # 在这里实际执行了bow-> tfidf和tfidf-> lsi转换
    for doc in corpus_lsi:
        print(doc)
```

Gensim Practice

在gensim中，内置了一个距离计算的方法，其被所有的模型所共用，用于在已有的模型（矩阵形式）中构建相似性索引，以便于通过模型快速查找相似性。

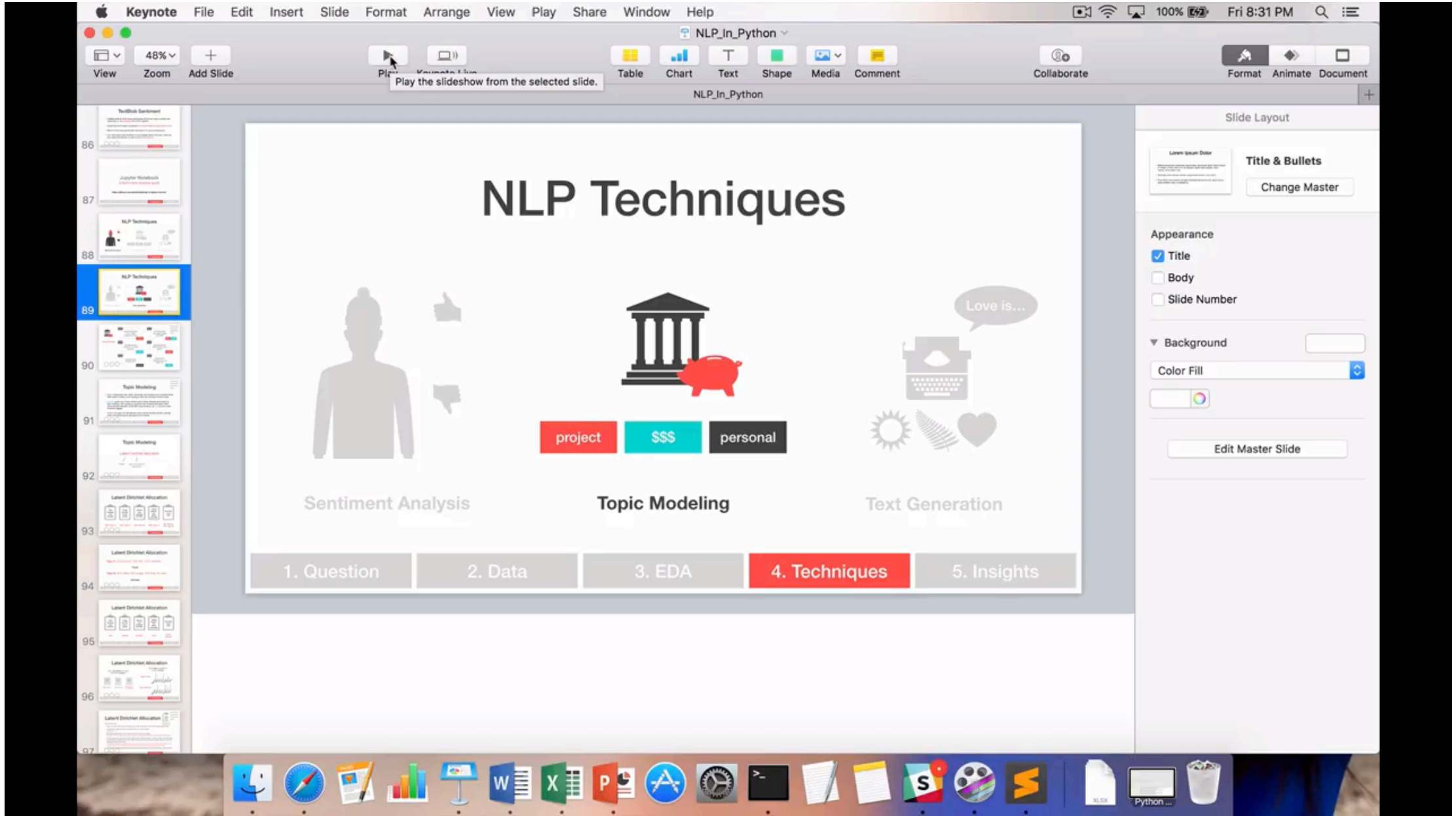
```
from gensim import corpora, models, similarities

index = similarities.MatrixSimilarity(corpus_tfidf) #把所有评论做成索引
sims = index[test_tfidf] #利用索引计算每一条评论和商品描述之间的相似度

lsi = models.LsiModel(corpus_tfidf)
corpus_lsi = lsi[corpus_tfidf]
similarity_lsi = similarities.Similarity('Similarity-LSI-
index', corpus_lsi, num_features=400,num_best=2)

query_lsi = lsi[‘test query’]
sims = similarity_lsi[query]
```

LDA Details


Keynote File Edit Insert Slide Format Arrange View Play Share Window Help
Fri 8:31 PM
NLP_In_Python
Play the slideshow from the selected slide.
NLP_In_Python
NLP Techniques
project
$$$
personal
Love is...
Sentiment Analysis
Topic Modeling
Text Generation
1. Question
2. Data
3. EDA
4. Techniques
5. Insights
Slide Layout
Title & Bullets
Change Master
Appearance
Title
Body
Slide Number
Background
Color Fill
Edit Master Slide

Neural Network Language Model