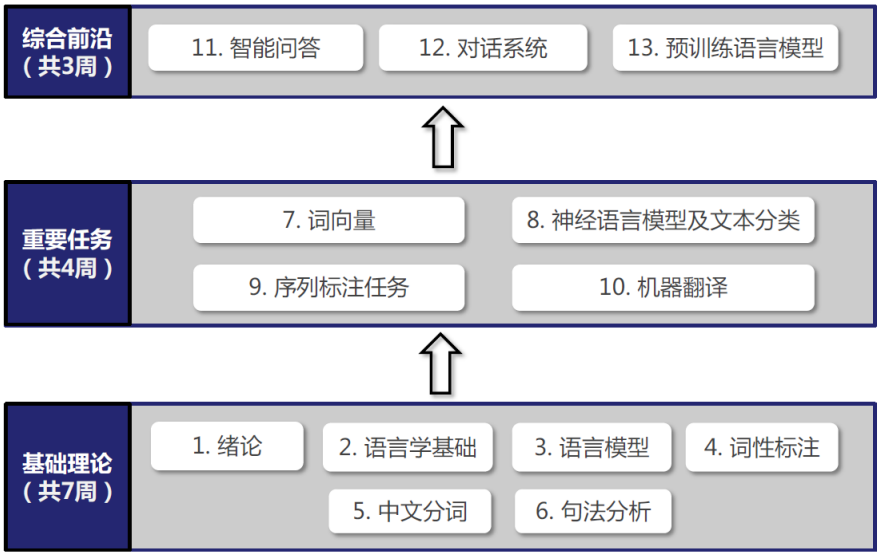
自然语言处理

All rights reserved by [CWHer](#)



目录

自然语言处理
- 目录
- 绪论
  - Chap1.1 绪论
    - 研究内容
      - 编码层
      - 语言学层
      - 知识与信息层
      - 系统与应用层
    - 发展历史
  - Chap1.2 语言学基础&NLP基础任务
    - 语言
    - 语言学
      - 语言单位
      - 语法
      - 语义
    - NLP基础任务
- 传统自然语言处理
  - Chap2.1 n-gram 语言模型
    - 数据预处理与标注
    - 语言模型
      - n-gram 语言模型
    - 评估
    - 平滑算法
    - 语言模型的应用
  - Chap2.2 词性标注及隐马尔科夫模型
    - 词性标注
    - 隐马尔科夫模型（HMM）
      - Viterbi算法
      - 前向算法&后向算法

有监督学习：最大似然参数估计
无监督学习：前向-后向算法
Chap2.3 英文分词与词元切分
问题
解决方案
Chap2.4 中文分词与条件随机场
中文分词
主要算法
最大匹配法
最短路径法
语言模型法
条件随机场（CRF）
CRF的重要算法
给定文本计算配分函数：前向与后向算法
给定文本推断分词方案：Viterbi算法
关于CRF的推论
有监督学习：最大似然参数估计
评价指标
Chap2.5 语法解析：构成式语法
构成式语法
基本概念
词汇化语法
语法间的等同关系
构成式语法的语法解析算法：CKY
概率化的构成式语法：PCFG
评价指标
Chap2.6 语法解析：依存式语法
依存式语法
依存式语法解析：shift-reduce parser
评价指标
基于深度学习的自然语言处理
Chap3.1 词向量
词的表示
词向量
基于共现频度的词向量
CBOW
Skip-gram
GloVe
词向量的评价
动态词向量
Chap3.2 神经网络语言模型
统计语言模型回顾
神经网络语言模型
前馈神经网络
循环神经网络
长短时记忆网络
门控循环单元GRU
Chap3.3 文本分类
文本分类
DNN
LSTM
CNN
FastText
RCNN
情感分类
文本匹配
DSSM
CLSM

基于交互的模型
Chap3.4 序列标注
LSTM + CRF
Chap3.5 机器翻译
基本翻译方法
直接转换法
基于规则的翻译方法
基于中间语言的翻译方法
基于事例的翻译方法
统计机器翻译
神经网络机器翻译
从LSTM/GRU到Attention Mechanism
集大成者：Transformer模型
预训练模型
综合型任务及学术前沿
Chap4.3 预训练语言模型
上下文相关词向量早期发展
CoVe
ELMo
典型预训练语言模型及其分类
BERT
GPT
BART
SupBART
XLNet: Permuted LM
UniLM: Prefix LM
典型预训练语言模型总结
预训练语言模型的发展和前景
模型自身结构的改进
优化预训练任务
预训练语料来源及其组成
微调方法

注：”常用工具“部分见课件，此处略去

绪论

Chap1.1 绪论

研究内容

编码层

计算机系统中对自然语言进行的底层处理、编码表征等

- 分词、分句
- 不同的表征方式
  - 离散向量：one-hot
  - 连续向量：TF-IDF、词向量
- 词干提取、词形还原
- 语言模型

语言学层

对语言本身的性质、功能、结构、运用等语言学知识进行解析和应用，关注“形式”、“结构”

- 词性标注、命名实体识别
- 句法分析、语义依存分析
- 语义角色标注、抽象语义表示

知识与信息层

融合外部世界知识进行知识层面的处理和应用，关注“内容”、“实质”

- 知识图谱
- 机器阅读理解
- 隐式情感分析
- 自然语言推理

系统与应用层

协调模块与组件构建具有完整语用功能的自然语言处理系统

- 搜索引擎
  - 文档检索：倒排索引，TF-IDF
  - 重要性排序：Page Rank

  倒排索引：提前遍历所有文档，储存出现的关键词，进行关键词匹配

  Q：如何选择关键词？

  词频（Term Frequency）

  词语在某个文档中出现的频率越大，则其关键程度越高

  $$tf_{i,j} = \frac{n_{i,j}}{total_j}$$

  逆向文档频率（Inverse Document Frequency）

  $$idf_i = \log \frac{|D|}{|\{j : w_i \in d_j\}|}$$

  即语料库中的文档总数/包含词语w_i的文档数目，越通用的词语，权重越低

  $$tf - idf_{i,j} = tf_{i,\ j} \times idf_i$$

  局限性：没有考虑位置信息与语义信息；简单的假设文档频率越大的单词越无用
- 机器翻译
- 自然语言生成
- 问答系统
- 对话系统

发展历史

理论算法

图灵测试
有限自动机和正则表达式

信息论与熵
经验主义

文法和句法结构
理性主义

20世纪50年代 — **基于规则的方法** — 20世纪70年代 — 21世纪初 — 现在

机器翻译： 基于规则的机器翻译初步成功
ALPAC报告指出其中泡沫

智能问答： 基于规则的数据库问答系统
问法、领域、规模受限

对话系统： 基于关键词匹配的闲聊机器人
不能“理解”用户意图

任务应用

理论算法

计算机硬件发展，数据存储规模增大
任务日益复杂，规则系统拓展性差

迅猛发展
占据主导

统计学习算法
经验主义

规则专家系统
理性主义

稳步推进
弊端凸显

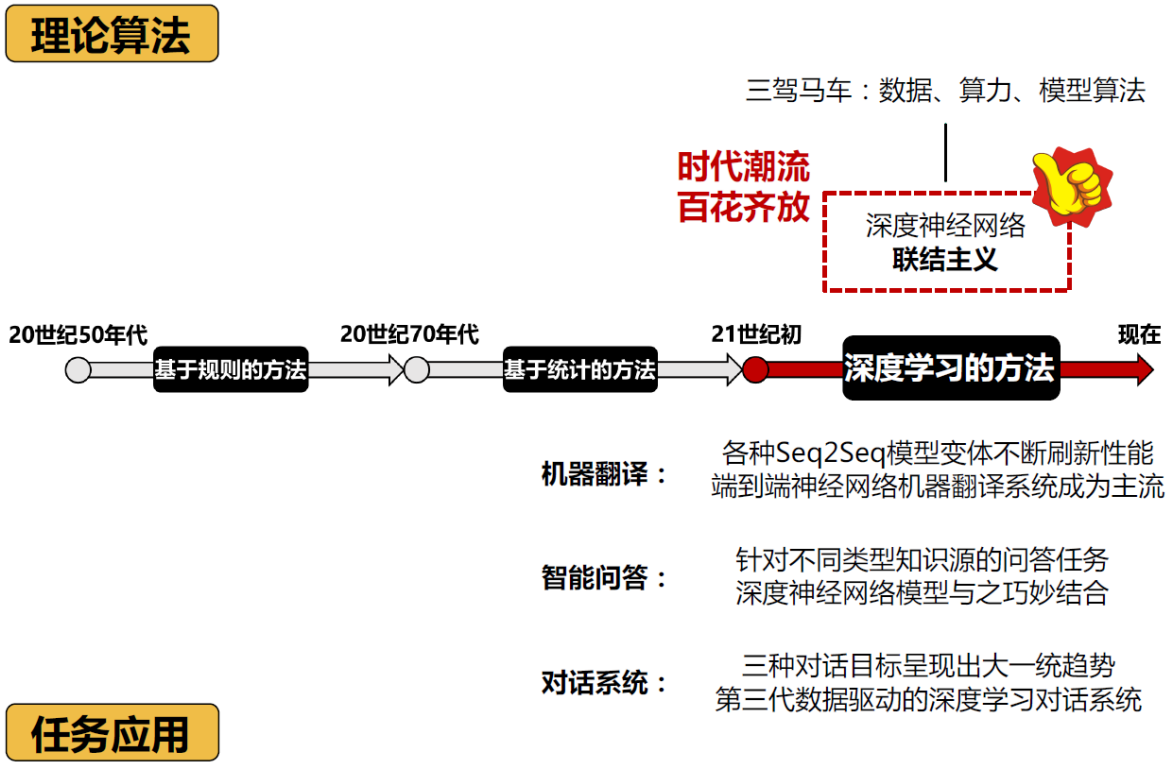
20世纪50年代 — **基于规则的方法** — 20世纪70年代 — **基于统计的方法** — 21世纪初 — 现在

机器翻译： 复苏发展并开始面向民用
统计机器翻译逐步取代传统规则系统

智能问答： 规则系统限制规模和通用性
问答从限定域知识库走向开放域信息检索

对话系统： 对话目标从情感聊天衍变出获取信息和完成任务
发展成为第二代数据驱动的统计对话系统

任务应用



Chap1.2 语言学基础&NLP基础任务

语言

- 语言的起源
- 语言系属分类

语言学

语言单位

- 词汇

  词 + 固定词语

  固定词语在造句中的作用相当于词，是词的等价物。
- 句子

  指构成语言的基本单位

  按照一定的语法规则组织，具有完整的意义
- 篇章

语法

指任意自然语言中句子、短语以及词等语法单位的语法结构与语法意义的规律

- 词法
  - 孤立语（e.g., 汉语）

    缺乏词形变化；虚词十分重要；复合词多，派生词少
  - 黏着语（e.g., 日语）

    词的尾部发生变化来表示语法意义；变词语素的每一种变化只表示一种语法意义；词根与变词语素不紧密，两者有很大独立性
  - 屈折语（e.g., 英语）

一般不允许无限地添加词根、词缀等规范词素；具有的屈折词素和词形屈折变化以及变格等语法范畴的变化；通过词形变化来表示词与词之前的关系；词尾和词干或词根结合十分紧密

- 句法

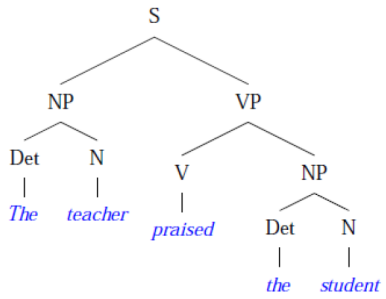
  支配句子结构，决定词、短语、从句等句子成分如何组成其上级成分

  大部分语言以主语开始，使用主谓宾或者主宾谓结构

  句法理论

  - 生成语法

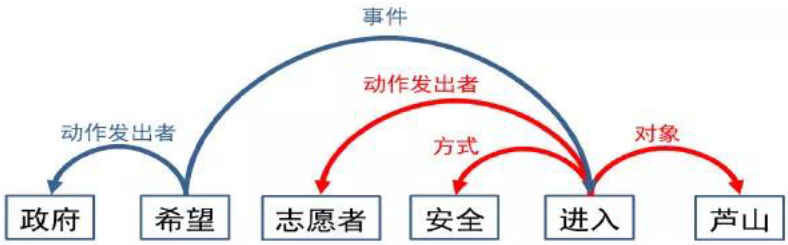
    - **深层结构**
      - S-> NP + VP
      - NP-> Det + N
      - VP-> V + NP
      - NP -> Det + N
    - **表层结构**
      - Det -> The
      - N -> Teacher
      - V -> Praise
      - Det -> the
      - N ->Student



  - 依存句法

    “谓语”中的动词是一个句子的中心，其他成分与动词直接或间接地产生联系

    基本假设：句法结构本质上包含词和词之间的依存（修饰）关系



- 篇章语法

  连贯性指篇章的语义关联，它存在于篇章的底层，是一种语义逻辑的联系。分析篇章的连贯性可以研究篇章中所表达的各种概念或论点之间的关系。

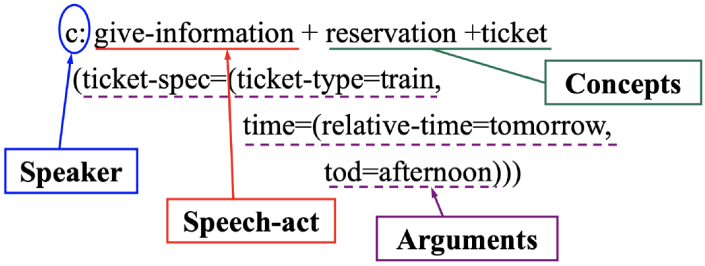
语义

找出语义表达的规律性、内在解释、不同语言在语义表达方面的个性以及共性

说话者的意图作为意义

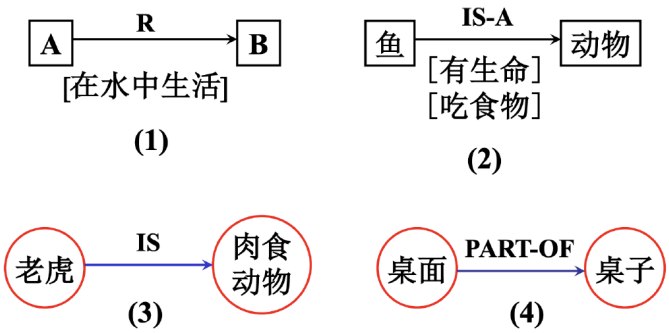
- **该理论**试图解释语言中一种被称为言语行为(Speech Acts) 的现象。 说话者把自己的话语当作行为，希望听者理 解、作出反应。这种意义被认为是独立于逻辑意义之外的。

例如：我想预订明天下午的火车票。



语义网络

- **语义网络通过由概念和语义关系组成的有向图 来表达知识、描述语义。**



NLP基础任务

- 词级别

  词是自然语言中能够独立运用的最小单位， 是自然语言处理的基本单位

  - 形态还原

    变化规则，不规则变化词表，特殊字符（年代、时间，货币等）

  - 分词

    词典切分：最大匹配，最少分词法

    基于统计的方法：语言模型，HMM，生成式方法

  - 词性标注
  - 词表示

    Word2Vec，Bert等

- 句子级别
  - 句子表示

    Bag-of-words，LSTM，Self-Attention等

  - 文本分类
  - 句子生成

- 篇章级别

- 篇章分析
- 机器阅读理解

  多项选择式：模型需要从给定的若干选项中选出正确答案

  区间答案式：答案限定是文章的一个子句，需要模型在文章中标明起止位置

  自由回答式：不限定模型生成答案的形式，允许自由生成语句

  完形填空式：除去若干关键词，需要模型填入正确单词或短语
- 篇章机器翻译

传统自然语言处理

Chap2.1 n-gram 语言模型

数据预处理与标注

- 词条化

  日语/黏着语：黏着语分割

  英语/屈折语：tokenization

  汉语/孤立语：汉语分词
- 去停用词

  停用词：通常指在各类文档中出现频繁，附带极少文本信息的助词、介词、连词、语气词等高频词
- 文本归一化

  转换训练文本，使得它变成纯语言的形式，同时移除很多和语言完全不相关的部分
  - 标点的移除
  - 日期，货币，数字的归一化
  - 缩写词的归一化

语言模型

给出词条序列合法性约束或序列概率统计特性的模型

- 基于规则的语言模型

  e.g., 语法网络（加权有限状态机）

  局限：很难手工去生成大量的语法规则；自然口语说话通常情况下是不合语法的
- 统计语言模型

  n-gram 语言模型，神经网络语言模型

n-gram 语言模型

近似地表示了在一个较短历史情况下的条件概率

$$P\left(w_k \mid w_1, \cdots, w_{k-1}\right) = P\left(w_k \mid w_{k-1}, \cdots, w_{k-n+1}\right)$$

即当前词的概率仅仅依赖于这个词之前的n - 1个前继词

常用模型: unigram (n = 1), bigram (n = 2), trigram (n = 3), 4-gram

<u>最大似然估计MLE</u>

数据：[w_1, ..., w_n]

模型：$P(w_k \mid W_{k-n+1}^{k-1})$

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{K}\sum_{k=1}^{K}\log P(w_k \mid W_{k-n+1}^{k-1}) \\ &= \frac{1}{K}\sum_{v\in\mathcal{V}}\sum_{y\in\mathcal{Y}} C(y,v)\log P(v \mid y)\end{aligned}$$

其中 $\mathcal{Y}$ 是所有可能的 n-gram 历史的集合，$C(y,v)$是训练集中(y,v)的数目

$$\hat{P}(v \mid y) = \arg\max_{P(v|y)} \sum_{v\in\mathcal{V}}\sum_{y\in\mathcal{Y}} C(y,v)\log P(v \mid y),\ \text{s.t.}\ \sum_{v\in\mathcal{V}} P(v \mid y) = 1$$

通过拉格朗日乘数法，可以解得

$$\hat{P}(v \mid y) = \frac{C(y,v)}{C(y)}$$

评估

信息

一个事件越不确定，就需要越多的信息来消除它的不确定性

$$I(x) = \log_2 P(x)$$

熵

熵是随机变量的信息的期望

$$\mathrm{H}[x] = -\sum P(x)\log_2 P(x)$$

条件熵

条件熵是条件分布$p(y \mid x)$的熵的期望。

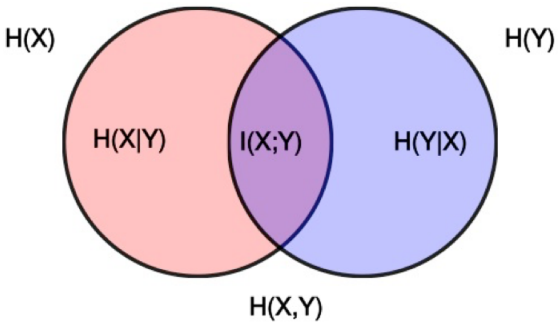
$$\begin{aligned}\mathrm{H}[y \mid x] &= \sum_{x'} P(x)\mathrm{H}\left(y \mid x = x'\right) \\ &= -\sum_{x} P(x)\sum_{y} P(y \mid x)\ln P(y \mid x)\end{aligned}$$

联合熵

$$\mathrm{H}[x,y] = \mathrm{H}[y \mid x] + \mathrm{H}[x]$$

互信息

$$\mathrm{I}[x,y] = \mathrm{H}[x] - \mathrm{H}[x \mid y] = \mathrm{H}[y] - \mathrm{H}[y \mid x]$$



KL散度

KL 散度广泛用于描述两个分布的差异，KL 散度是非对称的

$$
\begin{aligned}
\mathrm{KL}(p\|q) &= -\int p(x)\ln q(x)\mathrm{d}x - \left(-\int p(x)\ln p(x)\mathrm{d}x\right) \\
&= -\int p(x)\ln\left\{\frac{q(x)}{p(x)}\right\}\mathrm{d}x \\
\mathrm{KL}(p\|q) &\geq 0 \quad \mathrm{KL}(p\|q) \neq \mathrm{KL}(q\|p) \\
\mathrm{I}[x,y] &\equiv \mathrm{KL}(p(x,y)\|p(x)p(y)) \\
&= -\iint p(x,y)\ln\left(\frac{p(x)p(y)}{p(x,y)}\right)\mathrm{d}x\,\mathrm{d}y
\end{aligned}
$$

交叉熵

信息论角度，基于相同事件测度的两个概率分布p和q的交叉熵是指，当基于一个“非自然”（相对于“真实”分布p而言）的概率分布q进行编码时，在事件集合中唯一标识一个事件所需要的平均比特数

$$
\mathrm{H_c}(P,Q) = -\sum_x P(x)\log_2 Q(x)
$$

n-gram 模型$P(\cdot)$和实际的开发文本集合上的实际词分布$P_{\mathrm{data}}(\cdot)$之间的交叉熵被广泛地用作语言模型的评价指标

$$
\begin{aligned}
\mathrm{H}(P,Q) &= -\sum_{v\in\mathcal{V}}\sum_{y\in\mathcal{Y}} P_{\mathrm{data}}(v\mid y)\log_2 P(v\mid y) \\
&\approx -\frac{1}{K}\sum_{v\in\mathcal{V}}\sum_{y\in\mathcal{Y}} C(y,v)\log_2 P(v\mid y) \\
&\approx -\frac{1}{K}\sum_{k=1}^{K}\log_2 P\left(w_k\mid W_{k-n+1}^{k-1}\right) = -\mathcal{L}(\theta)
\end{aligned}
$$

困惑度 Perplexity

Perplexity 是语言模型在数据上的概率值倒数的几何平均

- 可以被理解或解释成分支度
- 越低的PPL说明越好的预测能力

$$
\begin{aligned}
\mathrm{PPL} &= 2^{-\mathcal{L}(\theta)} = 2^{-\frac{1}{K}\sum_{k=1}^{K}\log_2 P\left(w_k|W_{k-n+1}^{k-1}\right)} \\
&= \left(2^{-\sum_{k=1}^{K}\log_2 P\left(w_k|W_{k-n+1}^{k-1}\right)}\right)^{\frac{1}{K}} = \left(\prod_{k=1}^{K} 2^{-\log_2 P\left(w_k|W_{k-n+1}^{k-1}\right)}\right)^{\frac{1}{K}} \\
&= \left(\prod_{k=1}^{K}\frac{1}{P\left(w_k\mid W_{k-n+1}^{k-1}\right)}\right)^{\frac{1}{K}}
\end{aligned}
$$

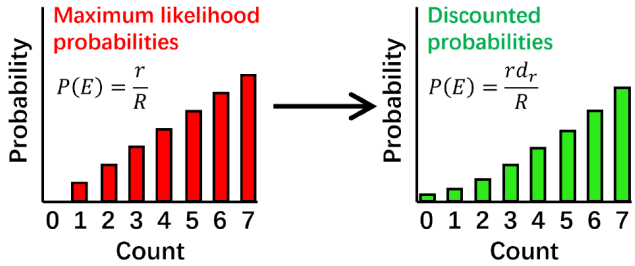
平滑算法

数据稀疏问题

- 零概率：导致无穷大的分支度PPL
- 低频次：导致不可靠的估计

解决方案

- 折扣法（Discounting）



为了把一些概率值重新分布到没有看到过的 n-gram 上，我们需要去减少那些在原本训练文本中的看到过的 n-gram 的频次（打折扣），从而满足概率和为 1 的要求。

$$\hat{P}_{\texttt{ML}}(w|y) = \frac{C(y,w)}{C(y)} \qquad \hat{P}_{\texttt{Discount}}(w|y) = d(y,w)\frac{C(y,w)}{C(y)}$$

其中 $d(y,w)$ 是一个折扣 (discount) 系数

Good Turing Discounting

基本的想法：将训练数据中出现次数为 $r+1$ 次的 n-gram 的概率重新分配给出现次数为 r 次的 n-gram。

$$r^* = (r+1)\frac{N_{r+1}}{N_r} \quad d(r) = \frac{(r+1)N_{r+1}}{rN_r}$$

其中 N_r 表示 **count of counts**, i.e. 在训练集合中出现次数是 r 次的 n-gram 的数量。因而，训练集中所能观测到的 n-grams 的总数是：

$$N = \sum_{r=0}^{\infty} rN_r = \sum_{r=1}^{\infty} rN_r$$

一个 n-gram 的概率可以定义为

$$P(v : C(v) = r) = \frac{r^*}{N}$$

- 在进行折扣算法之前，需要先对 N_r 做一些调整
- 分配给未出现过的 n-gram 的概率量是 $\frac{N_1}{N}$

Good Turing Discounting
示例

Given the following counts:

Symbol	A	B	C	D	E	F	G	H	I	J
Counts	1	1	0	2	1	2	2	0	1	3

Frequency table: (applying $r^* = (r+1)\frac{N_{r+1}}{N_r}$)

r	Symbols	N_r	rN_r	$P(x)$
0	C, H	2	0	0
1	A, B, E, I	4	4	4/13
2	D, F, G	3	6	6/13
3	J	1	3	3/13
Total			13	1

→

r^*	Symbols	N_r	rN_r	$P(x)$
2	C, H	2	4	4/13
3/2	A, B, E, I	4	6	6/13
1	D, F, G	3	3	3/13
1	J	1	0	0
Total			13	1

- 那些出现 r 次的 n-gram 的概率已被重新分配给那些出现了 $r+1$ 次的 n-gram。
- Good turing discounting 通常仅仅应用于低频次的词

- 回退及插值（Backing-off and interpolation）

Katz's Back-off Model

回退（Back-off）：在估计那些训练文本中从没见过或者见过频次很少的 n-gram 的概率时，使用某些条件下相对较短的历史。

$$P_{\text{bo}}(w_k|W_{k-n+1}^{k-1}) = \begin{cases} d(W_{k-n+1}^{k})\frac{C(W_{k-n+1}^{k})}{C(W_{k-n+1}^{k-1})} & C(W_{k-n+1}^{k}) > t \\ \alpha(W_{k-n+1}^{k-1})P_{\text{bo}}(w_k|W_{k-n+2}^{k-1}) & \texttt{otherwise} \end{cases}$$

- t 一般情况下置成零 (仅仅在 n-gram 完全没有出现的时候进行回退)
- d 和 α 分别是折扣因子 (discounting factor) 和回退权重 (back-off weights)
- Good-Turing 回退方法经常被作为 d 来使用
- 选取 α 使得

$$\sum_{w_k}^{V} P_{\text{bo}}(w_k|W_{k-n+1}^{k-1}) = 1$$

Jelinek-Mercer Smoothing (Interpolation)

Good-Turing: 未出现过的 n-grams 被赋予相同的频次

$$\left.\begin{array}{ll} C(\texttt{Be},\texttt{Happy}) & = 0 \\ C(\texttt{Be},\texttt{ZZZ}) & = 0 \end{array}\right\} \Rightarrow \quad P(\texttt{ZZZ}|\texttt{Be}) = P(\texttt{Happy}|\texttt{Be})$$

这个结果似乎有问题，因为 $P(\texttt{Happy}) > P(\texttt{ZZZ})$

解决方案：在 unigram 和 bigram 之间插值

$$P^*(w_i|w_{i-1}) = \lambda P(w_i|w_{i-1}) + (1-\lambda)P(w_i)$$

可以推广到高阶情况

$$P^*(w_i|W_{i-n+1}^{i-1}) = \lambda_n P(w_i|W_{i-n+1}^{i-1}) + (1-\lambda_n)P(w_i|W_{i-n+2}^{i-1})$$

λ_n 可以在预留出来的开发数据集合上学习得到。

语言模型插值

- 对同一个语言模型中的不同阶数 n-gram 的概率进行插值
- **对具有相同 n-gram 阶数的不同语言模型进行插值**

$$P(w_i|W_{i-n+1}^{i-1}) = \sum_{m=1}^{M} \lambda_m P_m(w_i|W_{i-n+1}^{i-1})$$

其中 λ_m 可以通过在一个预留的开发集合上通过 MLE 最大似然准则估计得到

- 对综合来自不同数据源的信息很有用
- 做语言模型自适应的快速方式

- 基于类的语言模型

基于类的语言模型 Class-based Language Model

- Class-based LM 使用参数绑定的方法来得到更加鲁棒的 n-gram 概率估计
- 将 n-gram 的历史信息归成不同的类别

$$P(w_i|W_{i-n+1}^{i-1}) = P(w_i|G_i)P(G_i|G_{i-n+1}^{i-1})$$

其中 $G(w)$ 表示映射词 w_i 为它的相关类或组的函数。

```
I (noun) buy (verb) a (article) book (noun).
He (noun) sold (verb) a (article) car (noun).
```

- 归类或分组工作一般在词层面来实现
- Class-based LMs 往往和 word-based LMs 进行插值来得到进一步的性能改进

语言模型的应用

常用策略

- First-Pass

  在识别解码过程中引入语言模型打分，作为选择最优解码路径的依据之一

- Second-Pass

  在解码得到若干可能结果序列后，使用语言模型对每个序列重打分，并选择得分最高的序列作为最终识别结果

Chap2.2 词性标注及隐马尔科夫模型

词性标注

- 词性与上下文相关
- 词性划分具有层次性

<u>解决方案</u>

- 基于规则的词性标注方法

  其基本思想是按兼类词搭配关系和上下文语境建造词类消歧规则

  人工提取 + 基于机器学习的规则自动提出

- 基于统计模型的词性标注方法

  其基本思想是：给定带有各自标注的词的序列，可以确定下一个词最可能的词性

- 基于统计方法与规则方法相结合的词性标注方法

  对那些被认为可疑的标注结果，采用规则方法进行歧义消解

- 基于深度学习的词性标注方法

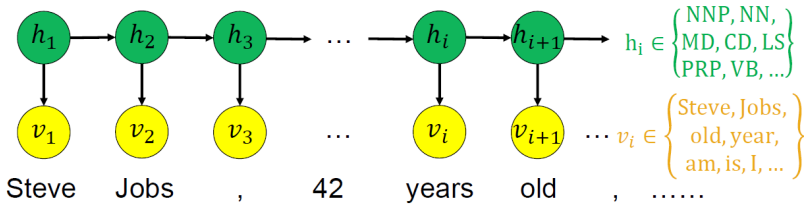
  可以当作序列标注的任务来做，常用方法包括 LSTM+CRF、BiLSTM+CRF 等

隐马尔科夫模型（HMM）

- 使用隐变量h_i表示单词背后的词性
- 将词性间的规则表示为概率分布
- 将词性到单词间的对应规则也表示为概率分布

目标：给定单词，找到最有可能的$h_1, \cdots, h_n$

隐马尔科夫模型



已知句子单词，预测每个单词的词性，变成了这样一个任务：

$$\underset{h_1,h_2,\ldots,h_i,\ldots,h_N}{\operatorname{argmax}} P(h_1, h_2, \ldots, h_i, \ldots, h_N \mid v_1, v_2, \ldots, v_i, \ldots, v_N)$$

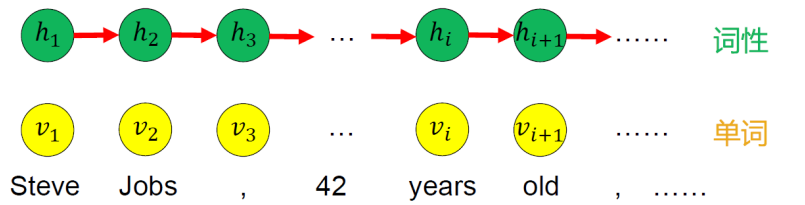
$$= \underset{h}{\operatorname{argmax}} P(h \mid v)$$

$$= \underset{h}{\operatorname{argmax}} \frac{P(v|h)P(h)}{P(v)} \qquad = \underset{h}{\operatorname{argmax}} \boxed{P(v|h) \cdot P(h)}$$

$P(v,\ h)$

分别求解$P(v \mid h), P(h)$

隐马尔科夫模型

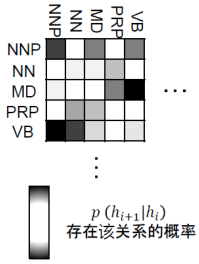


$$\boxed{P(h)}$$

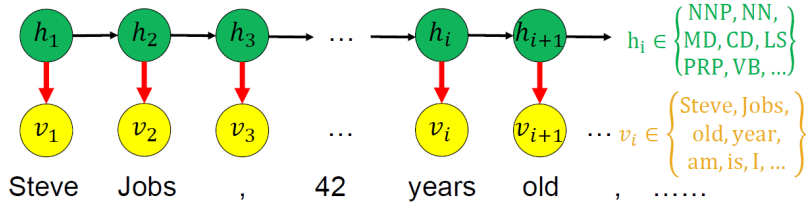
$$= P(h_1, h_2, \ldots, h_i, \ldots, h_N)$$

$$= P(h_1)P(h_2|h_1) \ldots P(h_{i+1}|h_i) \ldots P(h_N|h_{N-1})$$

$$= P(h_1) \prod_{i=1}^{N-1} P(h_{i+1}|h_i)$$



隐马尔科夫模型

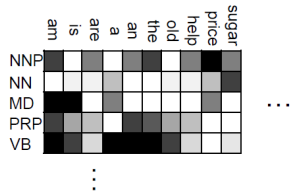


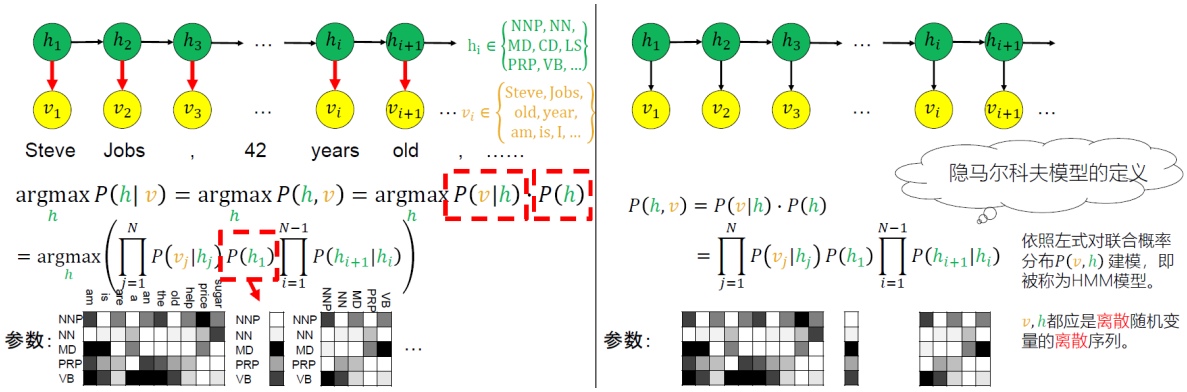
$$\boxed{P(v|h)}$$

$$= P(v_1, v_2, \ldots, v_i, \ldots, v_N \mid h_1, h_2, \ldots, h_i, \ldots, h_N)$$

$$= P(v_1|h_1)P(v_2|h_2) \ldots P(v_i|h_i) \ldots P(v_N|h_N)$$

$$= \prod_{j=1}^{N} P(v_j|h_j)$$





参数：初始矩阵，转移矩阵，发射矩阵

Viterbi算法

Idea：动态规划，$f[i][j]$表示前i个单词、结尾词性为j时的最优解

$$f[i][j] = \max_k \{f[i-1][k] \cdot t[k][j] \cdot e[j][v_i]\}$$

Viterbi算法给出的是

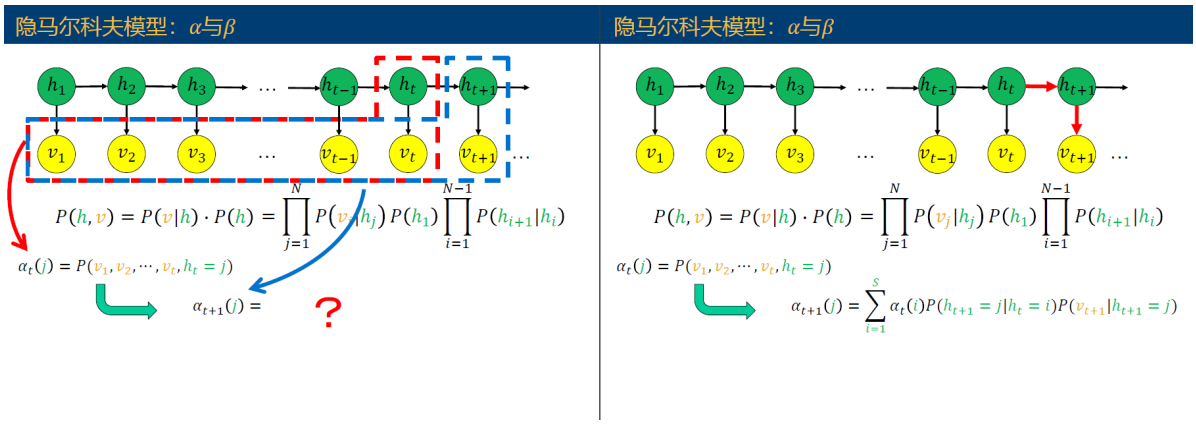
$$\operatorname{argmax}_h P(h \mid v) = \operatorname{argmax}_h P(h, v)$$

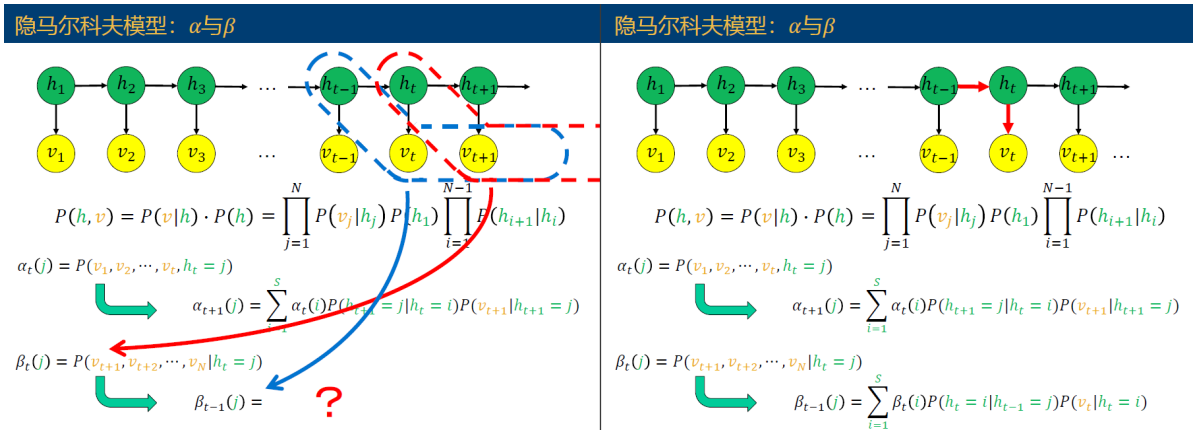
前向算法&后向算法

用于计算$P(v)$，本质为Variable Elimination

首先定义α, β方便计算

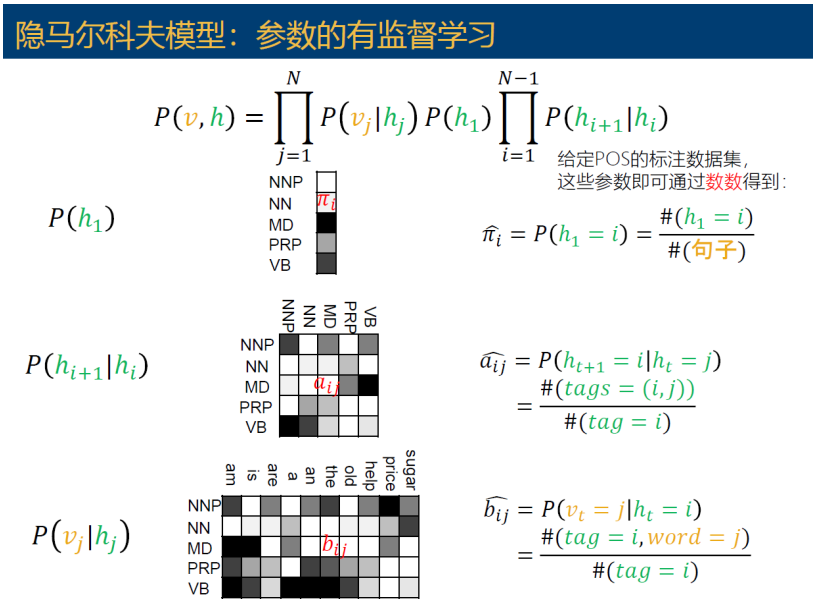
$$\alpha_t(j) = P(v_1, \cdots, v_t, h_t = j)$$
$$\beta_t(j) = P(v_{t+1}, \cdots, v_n \mid h_t = j)$$
$$P(v) = \sum_j \alpha_n(j) = \sum_j \beta_1(j) \cdot P(h_1 = j) \cdot P(v_1 \mid h_1 = j)$$





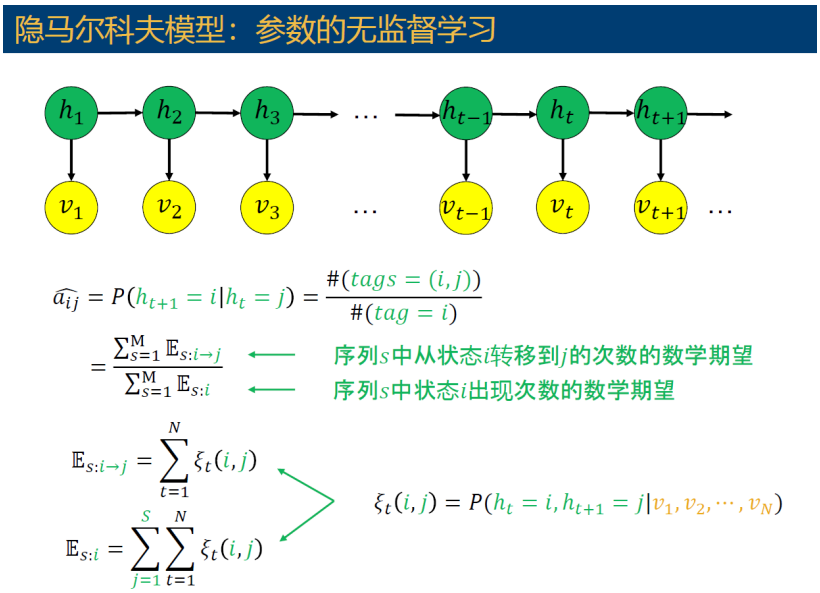
- 前向算法：使用α从前往后计算，其中$\alpha_1(j) = P(h_1 = j) \cdot P(v_1 \mid h_1 = j)$
- 后向算法：使用β从后往前计算，其中$\beta_n(j) = 1$

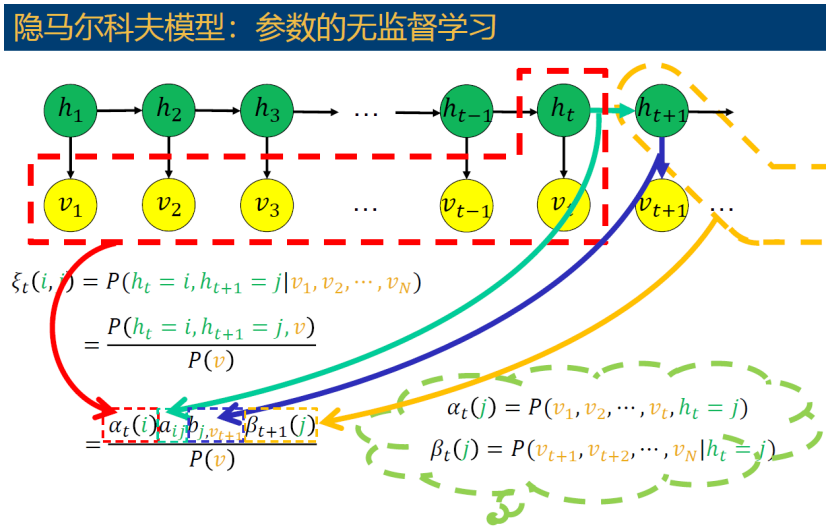
有监督学习：最大似然参数估计



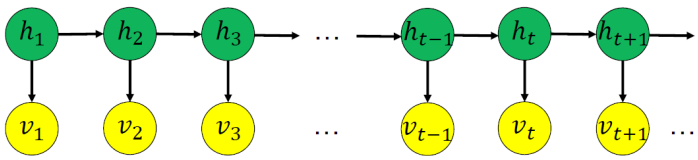
无监督学习：前向-后向算法

用于处理POS的无标注数据集，实际上准确率较低





隐马尔科夫模型：参数的无监督学习



$$\widehat{b_{ij}} = P(v_t = j | h_t = i) = \frac{\#(tag = i, word = j)}{\#(tag = i)}$$

$$= \frac{\sum_{s=1}^{M} \mathbb{E}_{s:i\to j}}{\sum_{s=1}^{M} \mathbb{E}_{s:i}}$$

$\longleftarrow$ 序列s中从状态i生成第j个单词的次数的数学期望

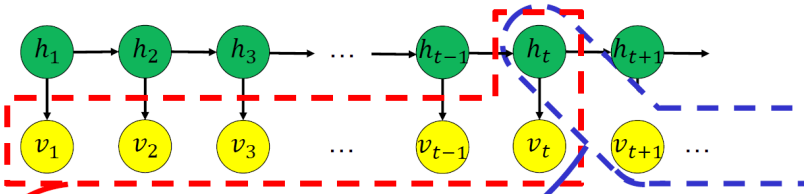
$\longleftarrow$ 序列s中状态i出现次数的数学期望

$$\mathbb{E}_{s:i\to j} = \sum_{v_t = j} \gamma_t(i)$$

$$\mathbb{E}_{s:i} = \sum_{t=1}^{N} \gamma_t(i)$$

$$\gamma_t(i) = P(h_t = i | v_1, v_2, \cdots, v_N)$$

隐马尔科夫模型：参数的无监督学习



$$\gamma_t(i) = P(h_t = i | v_1, v_2, \cdots, v_N)$$

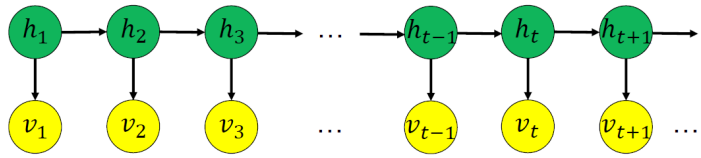
$$= \frac{P(h_t = i, v)}{P(v)}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{P(v)}$$

$$\alpha_t(j) = P(v_1, v_2, \cdots, v_t, h_t = j)$$

$$\beta_t(j) = P(v_{t+1}, v_{t+2}, \cdots, v_N | h_t = j)$$

隐马尔科夫模型：参数的无监督学习



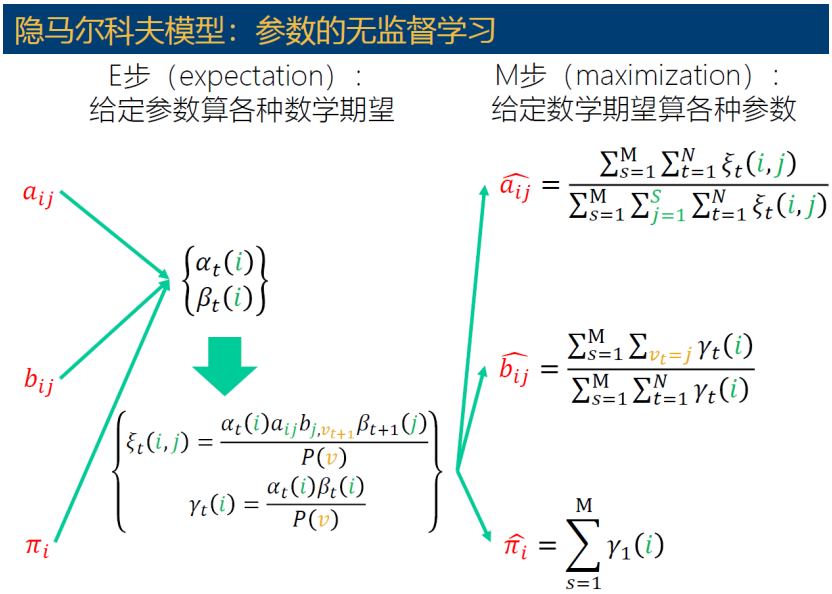
$$\hat{\pi}_i = P(h_1 = i) = \frac{\#(h_1 = i)}{\#(句子)}$$

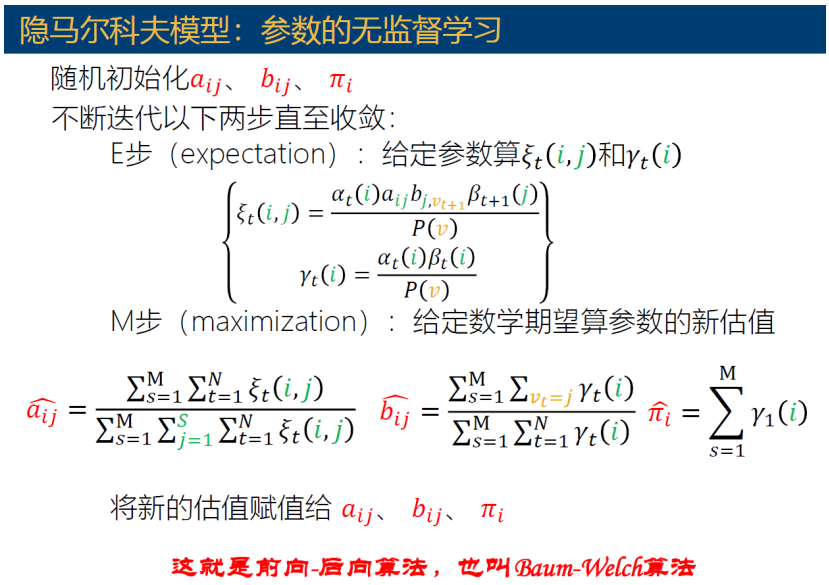
$$= \sum_{s=1}^{M} \mathbb{E}_{s:h_1=i}$$

$\longleftarrow$ 序列s中初始状态为i的概率

$$\mathbb{E}_{s:h_1=i} = \gamma_1(i)$$

$\longleftarrow$ $\gamma_t(i) = P(h_t = i | v_1, v_2, \cdots, v_N)$



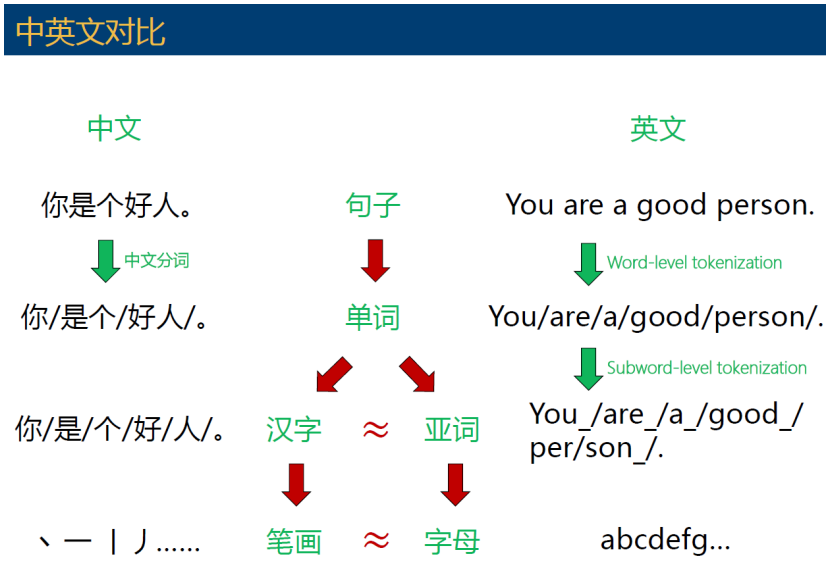


Chap2.3 英文分词与词元切分

问题

- 缩写词
  1. 将缩写后的词作为单独的一个新词（e.g., I'm）
  2. 将缩写后的词拆开（e.g., I / 'm）
- 数字、符号
  1. 将数字和符号按照标点切开（e.g., Fig. / 3）
  2. 不拆开（e.g., Fig. 3）

解决方案



- 词级别的词元切分

  使用正则表达式做匹配，不断修改正则表达式的数量和形式，使得在给定语料上的切分效果趋于完善

- 亚词级别的词元切分

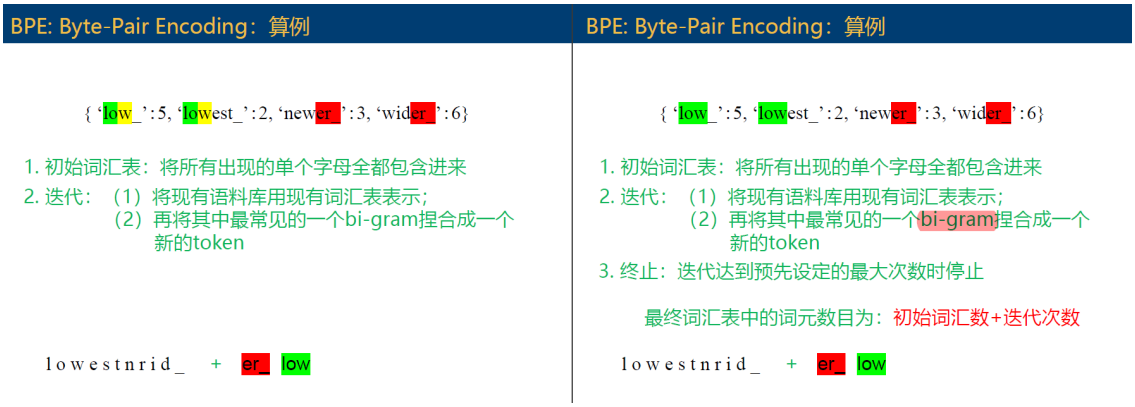
  对于英语等语言而言，词频一般呈现出长尾分布，词并不是一个特别合适的切分单元

  然而大量的单词之间存在互相联系，或是同一个词的不同屈折变化，或是共享部分词根

  亚词介于字母和词之间

  BPE: Byte-Pair Encoding

  - 该算法从数据中自底向上地组合字母来形成词元，词元数目精确可控
  - 通过适当调整参数，可以得到类似于“词根词缀”的英文亚词单元
  - 解决了未登录词的问题：无论设置的词元数量是多少，总是可以有办法使用有限的词元覆盖所有可能出现的单词
  - 算法复杂度低，可以在大规模语料上实用
  - 是目前绝大部分 NLP 模型采用的词元切分方法



Chap2.4 中文分词与条件随机场

中文分词

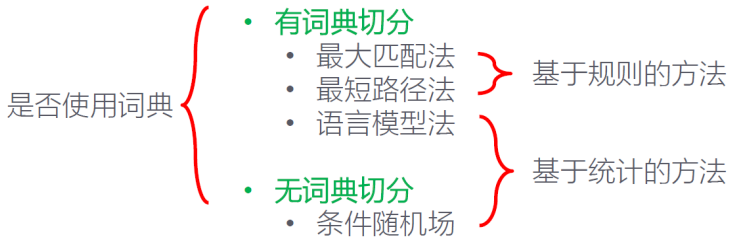
- 中文分词的规范
  - 合并原则
  - 切分原则
- 中文分词中的切分歧义

    - 交集型歧义（较多）
    - 组合型歧义
- 中文分词中的未登录词（较多）

  模型在训练阶段没有见过的单词

    - 人名、地名、组织机构名等
    - 新出现的词汇、术语、个别俗语等

主要算法

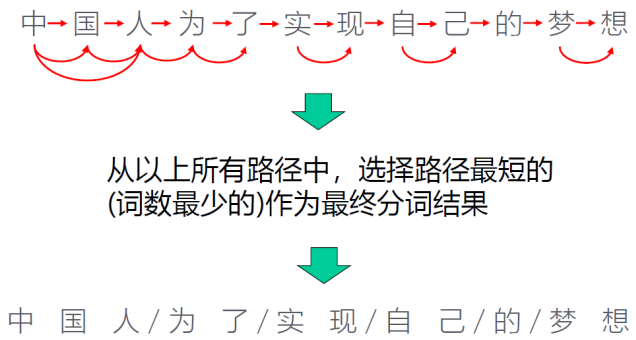


最大匹配法

贪心地寻找在词典中包含的长度最大的单词

- 正向最大匹配算法
- 逆向最大匹配算法

最短路径法



- 无法区分多条相同的最短路
- 字串长度较大和选取的最短路径数增大时，长度相同的路径数急剧增加

语言模型法

中文分词的基本算法：语言模型法

设句子$S=\{s_1,s_2,\cdots s_N\}$可切分为K个单词$W=\{w_1,w_2,\cdots w_K\}$，则此任务为找到最优的W，即W^*：

$$
\begin{aligned}
W^* &= \underset{W}{\operatorname{argmax}}\, P(W|S) \\
&= \underset{W}{\operatorname{argmax}}\, P(W)\underbrace{\boxed{P(S|W)}}_{=1} \\
&= \underset{W}{\operatorname{argmax}}\, P(W)
\end{aligned}
$$

对于给定的$P(W)$，其概率由预先训练好的语言模型给出。

枚举所有可能的W复杂度较大

束搜索 (beam search)

每层仅保留k个较优解的BFS

计算$\underset{W}{\operatorname{argmax}}(\cdot)$的方法：

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|S)$$

束搜索（beam search）：当穷举所有可能的W变得极多而无法一一计算时，利用束搜索从左至右地确定分词方案W^*。

他 是 研 究 生 物 化 学 的

他/是/研 究	他/是/研 究/生	×
	他/是/研 究/生 物	√
	他/是/研 究/生 物 化	√
他/是/研 究 生	他/是/研 究 生/物	×
	他/是/研 究 生/物 化	√
	他/是/研 究 生/物 化 学	×
他/是 研 究/生 物	他/是 研 究/生 物/化	×
	他/是 研 究/生 物/化 学	×
	他/是 研 究/生 物/化 学 的	×

- 训练语料的规模和覆盖领域不好把握
- 计算量较大

条件随机场（CRF）

中文分词的基本算法：条件随机场的问题设定

将分词看作是序列标注问题，依次对序列中每个字赋予一个标签，再根据标签合并同属于一个单词的字符：

- 词首(B)
- 词中(M)
- 词尾(E)
- 单独成词(S)

- 断字（I） 词的最后一个字
- 非断字（B） 此字之后不跟词分界符

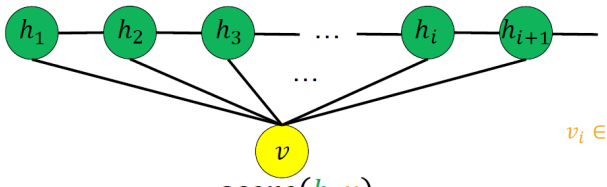
	中	国	人/为	了/实	现/自	己/的/梦	想
B	√	√	√	√	√	√	
I		√	√	√	√	√	√

中 国 人 为 了 实 现 自 己 的 梦 想

- 标签多，标错得不偿失
- 标签少，很难利用之前的标签信息

- HMM也可以用于处理分词，但无法加入语言规则（例如，人名有可能跟着动词）
- CRF不通过贝叶斯公式，而是直接对条件概率简单直接地进行拟合

中文分词的基本算法：CRF概念



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

$$P(h|v) = \frac{score(h,v)}{\sum_h score(h,v)} \qquad (s.t. \quad score(h,v) \geq 0)$$

$$score(h,v) = \exp\left(\sum_{k=1}^{K} w_k F_k(h,v)\right)$$

$$F_k(h,v) = F_k(h_1, h_2, \ldots, h_i, \ldots, h_N, v_1, v_2, \ldots, v_i, \ldots, v_N)$$

$$= \sum_{i=2}^{N} \boxed{f_k(h_i, h_{i-1}, v, i)}$$

与HMM不同，CRF中允许用户指定具体feature。这使得CRF可以引入各种各样的基于规则的feature

$f_k(h_i, h_{i-1}, v, i)$的选取：

- $f_k(h_i, h_{i-1}, v, i)$通常是值域为$\{0,1\}$的二值函数，在满足特征规定的条件后值为一，否则为零。
- $f_k(h_i, h_{i-1}, v, i)$的具体定义可以非常灵活。可以只与输入序列v有关，也可以与h_i, h_{i-1}和v的对应关系有关，甚至可以与feature所在的具体的位置i有关。

注：f_k对于h的约束必须相邻

$f_k(h_i, h_{i-1}, v, i)$的选取：

类型	特征	特征激活条件
一元特征	v_{i-1}	上一字符为某特定字符
	v_i	当前字符为某特定字符
	v_{i+1}	下一字符为某特定字符
二元特征	$\overline{v_{i-1}v_i}$	当前字符和上一字符为某特定串
	$\overline{v_i v_{i+1}}$	当前字符和下一字符为某特征串
跳跃特征	$\overline{v_{i-1} \blacksquare v_{i+1}}$	上一字符和下一字符为某特定串
三元特征	$\overline{v_{i-1}v_i v_{i+1}}$	上一字符、当前字符、下一字符为某特定串
……	……	……

$f_k(h_i, h_{i-1}, v, i)$的选取：

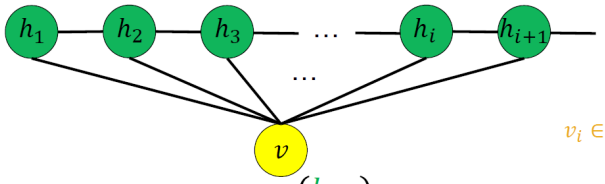
类型	特征	特征激活条件
状态转移	$\overline{h_{i-1}h_i}$	上一字符的标记 h_{i-1} 转移到当前字符的标记 h_i
一元状态特征	h_{i-1}	上一字符为某特定标记
	h_i	当前字符为某特定标记
混合特征	$\overline{v_{i-1}v_i} \cap h_i \cap (i=5)$	上一字符和当前字符为某特定串，且 当前字符的标记为h_i，且 当前所在位置为5
……	……	……

一般需要10^5~10^6的特征，通常由**特征模板**生成

CRF的重要算法

给定文本计算配分函数：前向与后向算法

条件随机场：前向算法与后向算法



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

$$P(h|v) = \frac{score(h, v)}{\sum_h score(h, v)} \qquad (score(h, v) \geq 0)$$

$$\sum_h score(h, v) = \boxed{\sum_h \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)} = Z(v)$$

$$Z(v) = \sum_h \exp\left(\sum_{k=1}^{K} w_k \sum_{i=2}^{N} f_k(h_i, h_{i-1}, v, i)\right)$$

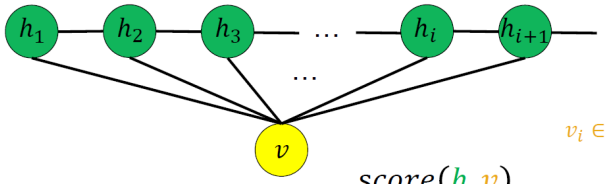
$$Z(v) = \sum_h \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right) = \sum_{h_1, h_2, \cdots, h_N} \exp\left(\sum_{i=2}^{N} \sum_{k=1}^{K} w_k f_k(h_i, h_{i-1}, v, i)\right)$$

$$= \sum_{h_N} \left\{ \exp\left(\sum_{k=1}^{K} w_k f_k(h_N, h_{N-1}, v, i)\right) \cdots \sum_{h_2} \left[\exp\left(\sum_{k=1}^{K} w_k f_k(h_3, h_2, v, i)\right) \sum_{h_1} \exp\left(\sum_{k=1}^{K} w_k f_k(h_2, h_1, v, 1)\right) \right] \right\}$$

$$= \sum_{h_N} \left\{ \exp\left(\sum_{k=1}^{K} w_k f_k(h_N, h_{N-1}, v, i)\right) \cdots \sum_{h_2} \left[\exp\left(\sum_{k=1}^{K} w_k f_k(h_3, h_2, v, i)\right) \sum_{h_1} \exp\left(\sum_{k=1}^{K} w_k f_k(h_2, h_1, v, 1)\right) \right] \right\}$$

给定文本推断分词方案：Viterbi算法

条件随机场： Viterbi算法



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

$$\begin{aligned} \underset{h}{\operatorname{argmax}}\, P(h|v) &= \underset{h}{\operatorname{argmax}} \frac{score(h, v)}{\sum_h score(h, v)} \qquad (score(h, v) \geq 0) \\ &= \underset{h}{\operatorname{argmax}} \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)}{\sum_h \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)} \\ &= \underset{h}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right) \end{aligned}$$

$$\underset{h}{\operatorname{argmax}}\, P(h|v) = \underset{h}{\operatorname{argmax}} \left[\exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)\right]$$

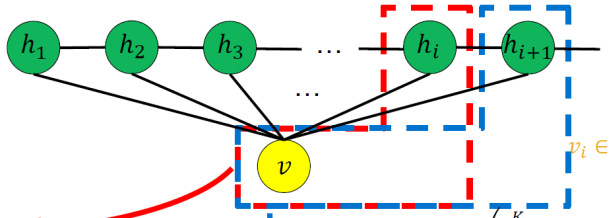
$$= \underset{h_1, h_2, \ldots, h_i, \ldots, h_N}{\operatorname{argmax}} \left[\begin{aligned} &\sum_{k=1}^{K} w_k f_k(h_2, h_1, v, 2) + \sum_{k=1}^{K} w_k f_k(h_3, h_2, v, 3) + \cdots \\ &+ \sum_{k=1}^{K} w_k f_k(h_N, h_{N-1}, v, N) \end{aligned} \right]$$

计算过程与HMM中的Viterbi算法类似

关于CRF的推论

α递推公式类似于边缘分布的计算

条件随机场：α



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

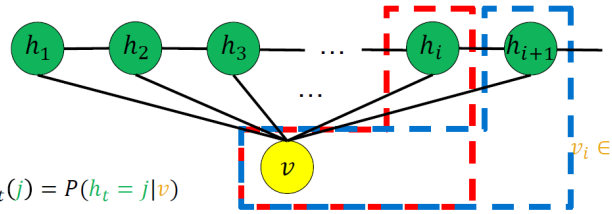
$$\alpha_t = P(h_1, h_2, \cdots, h_t | v) = \sum_{h_{t+1}, \cdots, h_N} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)$$

$$= \sum_{h_{t+1}} \left[\sum_{h_{t+2}, \cdots, h_N} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right) \right]$$

$$= \sum_{h_{t+1}} \alpha_{t+1}$$

$$\alpha_{t+1} = P(h_1, h_2, \cdots, h_t, h_{t+1} | v) = \sum_{h_{t+2}, \cdots, h_N} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} w_k F_k(h, v)\right)$$

条件随机场：γ



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

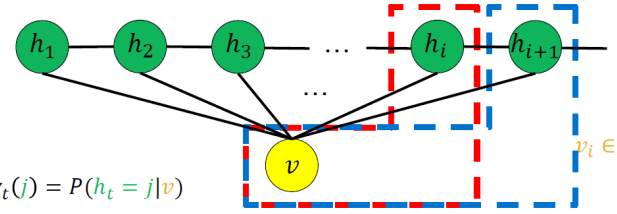
$$\gamma_t(j) = P(h_t = j | v)$$

$$\gamma_{t+1}(j) = \sum_{h_1, h_2, \cdots, h_t, h_{t+1} = j, h_{t+2}, \cdots, h_N} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{N} w_k f_k(h_i, h_{i-1}, v, i)\right)$$

$$= \sum_{h_1, h_2, \cdots, h_t, h_{t+1} = j} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{t+1} w_k f_k(h_i, h_{i-1}, v, i)\right)$$

$$= \sum_{m} \left\{ \sum_{h_1, h_2, \cdots, h_t = m, h_{t+1} = j} \frac{1}{Z(v)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{t+1} w_k f_k(h_i, h_{i-1}, v, i)\right) \right\}$$

条件随机场：γ



$h_i \in \begin{Bmatrix} B, M \\ E, S \end{Bmatrix}$

$v_i \in \begin{Bmatrix} 中, 国, 人, 为, \\ 了, 实, 现, 自, \\ 己, 的, 梦, 想, \ldots \end{Bmatrix}$

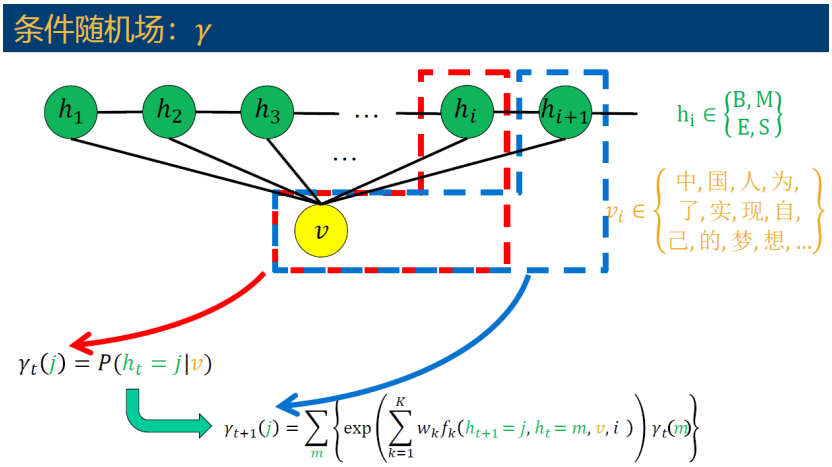
$$\gamma_t(j) = P(h_t = j | v)$$

$$\gamma_{t+1}(j) = \sum_{m} \left\{ \sum_{h_1, h_2, \cdots, h_t = m, h_{t+1} = j} \frac{1}{Z(v)} \exp\left[\sum_{k=1}^{K} \left(\sum_{i=2}^{t} w_k f_k(h_i, h_{i-1}, v, i)\right)\right] \exp\left(\sum_{k=1}^{K} w_k f_k(h_{t+1}, h_t, v, i)\right) \right\}$$

$$= \sum_{m} \left\{ \sum_{h_1, h_2, \cdots, h_t = m, h_{t+1} = j} \frac{1}{Z(v)} \exp\left[\sum_{k=1}^{K} \left(\sum_{i=2}^{t} w_k f_k(h_i, h_{i-1}, v, i)\right)\right] \exp\left(\sum_{k=1}^{K} w_k f_k(h_{t+1} = j, h_t = m, v, i)\right) \right\}$$

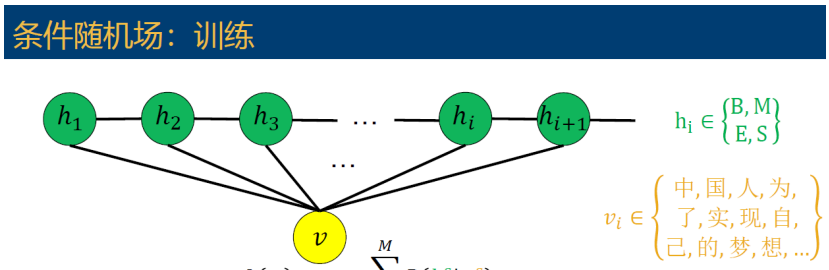
$$= \sum_{m} \left\{ \exp\left(\sum_{k=1}^{K} w_k f_k(h_{t+1} = j, h_t = m, v, i)\right) \sum_{h_1, h_2, \cdots, h_t = m} \frac{1}{Z(v)} \exp\left[\sum_{k=1}^{K} \left(\sum_{i=2}^{t} w_k f_k(h_i, h_{i-1}, v, i)\right)\right] \right\}$$

$$= \sum_{m} \left\{ \exp\left(\sum_{k=1}^{K} w_k f_k(h_{t+1} = j, h_t = m, v, i)\right) \gamma_t(m) \right\}$$



$$
\begin{aligned}
\gamma_{t+1}(j) &= P(h_{t+1} = j \mid v) \\
&= \sum_m P(h_{t+1} = j, h_t = m \mid v) \\
&= \sum_m P(h_t = m \mid v) \cdot P(h_{t+1} = j \mid h_t = m, v) \\
&= \sum_m \gamma_t(m) \cdot P(h_{t+1} = j \mid h_t = m, v)
\end{aligned}
$$

有监督学习：最大似然参数估计



$$
\begin{aligned}
\max_w L(w) &= \max_w \sum_{s=1}^{M} P(h^s|v^s) \\
&= \max_w \sum_{s=1}^{M} \frac{1}{Z(v_s)} \exp\left(\sum_{k=1}^{K} w_k F_k(h^s, v^s) \right) \\
&= \max_w \sum_{s=1}^{M} \frac{1}{Z(v_s)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{N} w_k f_k(h_i^s, h_{i-1}^s, v^s, i) \right)
\end{aligned}
$$

$$
\Leftrightarrow \quad \max_w [\log L(w)] = \max_w \left[\log \sum_{s=1}^{M} \frac{1}{Z(v_s)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{N} w_k f_k(h_i^s, h_{i-1}^s, v^s, i) \right) \right]
$$

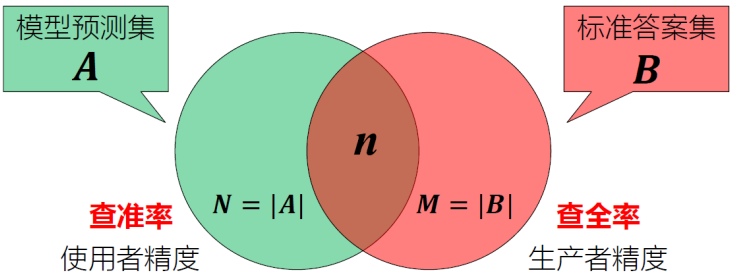
$$
\Leftrightarrow \quad \min_w [-\log L(w)] = \min_w \left[-\log \sum_{s=1}^{M} \frac{1}{Z(v_s)} \exp\left(\sum_{k=1}^{K} \sum_{i=2}^{N} w_k f_k(h_i^s, h_{i-1}^s, v^s, i) \right) \right]
$$

注：目标函数为凸函数，通过梯度下降即可求解

评价指标

中文分词的评价指标：查准率与查全率

假设系统输出N个结果，其中，正确的结果为n个，标准答案的个数为M个



查准率

使用者精度

准确率

Precision:

$$P = \frac{n}{N} \times 100\%$$

查全率

生产者精度

召回率

Recall:

$$R = \frac{n}{M} \times 100\%$$

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} \times 100\%$$

$$F_1 = \frac{2PR}{P+R}$$

Chap2.5 语法解析：构成式语法

构成式语法

基本概念

构成式语法（constituency grammar）简介

语法成分（constituent）

句子中的一组词，作为整体可以当做一个单独的语法单元

例如：名词性短语（NP），动词性短语（VP）……

语法规则（rules）

一组描述某个语法成分可以由什么组成的规则。

例如：

NP → Det Nominal（名词性短语可以由冠词加名词构成）

NP → ProperNoun（名词性短语可以专有名词构成）

词典（lexicon）

一组描述某个语法成分可以由什么词来构成的规则。

例如：

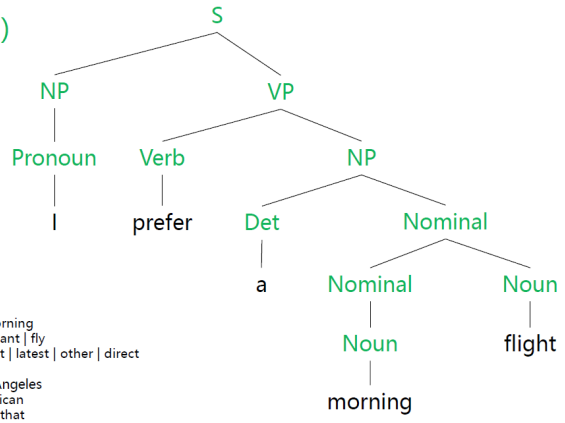
Det → a | an | the

Noun → flight | duck | paper

构成式语法（constituency grammar）简介

- 语法成分（constituent）
- 语法规则（rules）
- 词典（lexicon）

S, NP, VP, Pronoun, Proper-Noun, Det, Nominal, Verb, PP, Preposition



S → NP VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Nominal Noun
Nominal → Noun
PP → Preposition NP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP

Noun → flights | breeze | trip | morning
Verb → is | prefer | like | need | want | fly
Adjective → cheapest | non-stop | first | latest | other | direct
Pronoun → me | I | you | it
Proper-Noun → Alaska | Baltimore | Los Angeles | Chicago | United | American
Determiner → the | a | an | this | these | that
Preposition → from | to | on | near
Conjunction → and | or | but

构成式语法（constituency grammar）简介

语法成分（constituent）

S(句子)　Proper-Noun(专有名词)　Verb(动词)
NP(名词性短语)　Det(冠词)　PP(介词短语)
VP(动词性短语)　Nominal(名词性成分)　Preposition(介词)
Pronoun(代词)

N

语法规则（rules）

S → NP VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Nominal Noun
Nominal → Noun
PP → Preposition NP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP

R

词典（lexicon）

Noun → flights | breeze | trip | morning
Verb → is | prefer | like | need | want | fly
Adjective → cheapest | non-stop | first | latest | other | direct
Pronoun → me | I | you | it
Proper-Noun → Alaska | Baltimore | Los Angeles | Chicago | United | American
Determiner → the | a | an | this | these | that
Preposition → from | to | on | near
Conjunction → and | or | but

Σ

构成式语法（constituency grammar）简介

N　一组定义好的语法成分（constituent），并且只能作为语法树中的非叶子节点（non-terminal symbols）

S(句子)　Proper-Noun(专有名词)　Verb(动词)
NP(名词性短语)　Det(冠词)　PP(介词短语)
VP(动词性短语)　Nominal(名词性成分)　Preposition(介词)
Pronoun(代词)

Σ　词汇表，只能作为语法树中的叶子节点（terminal symbols）

flights | breeze | trip | morning | is | prefer | like | need | want | fly | cheapest | non-stop | first | latest | other | direct | Pronoun | me | I | you | it | Alaska | Baltimore | Los Angeles | Chicago | United | American | the | a | an | this | these | that | Preposition | from | to | on | near | and | or | but

R　语法规则，即$\alpha \rightarrow \beta$形式的规则。α与β均可代表由$N \cup \Sigma$中的元素构成的序列

S → NP VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Nominal Noun
Nominal → Noun
PP → Preposition NP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP

Noun → flights | breeze | trip | morning
Verb → is | prefer | like | need | want | fly
Adjective → cheapest | non-stop | first | latest | other | direct
Pronoun → me | I | you | it
Proper-Noun → Alaska | Baltimore | Los Angeles | Chicago | United | American
Determiner → the | a | an | this | these | that
Preposition → from | to | on | near
Conjunction → and | or | but

S　语法所规定的，每个句子的根节点

CSG与CFG

上下文相关语法
（Context-sensitive Grammar，CSG）

语法树的生成与上下文相关，即：语法规则中包含“→”号左边不是单个的N中元素情况

VP NP → VP Nominal
NP NP → NP Det Nominal
prefer Pronoun PP → *prefer* Pronoun to Verb Noun
prefer NP→ *prefer* Proper-Noun

约束R中规则的形式

上下文无关语法
（Context-free Grammar，CFG）

语法树的生成与上下文无关，即：每一条语法规则中，“→”号左边均为单个的N中元素情况

S → NP VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
PP → Preposition NP
VP → Verb
VP → Verb NP
VP → Verb NP PP

词汇化语法

存在大量的语法规则，且某个语法成分的展开方式与具体用词息息相关

Idea：以词汇为中心，对每个词定义他可以怎么用

组合范畴语法 CCG

词汇化语法（lexicalized grammar）与组合范畴语法（CCG）

组合范畴语法
(combinatory categorial grammar, CCG)

核心思想：词汇的类别（category）变得复杂；通过复杂的类别指导每个词该怎么用。

类别（category）

1）基础类别（atomic elements）

S(句子)	Proper-Noun(专有名词)	Verb(动词)
NP(名词性短语)	Det(冠词)	PP(介词短语)
VP(动词性短语)	Nominal(名词性成分)	Preposition(介词)
Pronoun(代词)	……	

2）由基础类别构成的函数，以及函数的函数

(PP/NP): 在该词的右边放上一个NP成分，那么该词与其右边的成分即可共同构成一个PP。

(S\Pronoun)：在该词的左边放上一个代词，那么该词与其左边的成分即可共同构成一个句子。

(S\NP)/NP：在该词的左边放上一个NP，然后在右边放上一个，那么该词与两边的成分共同构成一个句子。

to	you	I	say	Gov.	serve	people
(PP/NP)	NP	Pronoun	(S\Pronoun)	NP	(S\NP)/NP	NP
PP		S		S		

词汇化语法（lexicalized grammar）与组合范畴语法（CCG）

组合范畴语法
(combinatory categorial grammar, CCG)

核心思想：词汇的类别（category）变得复杂；通过复杂的类别指导每个词该怎么用。

词典（lexicon）

每个词的类别直接指定为以上形式的类别。这类复杂的类别模式已经暗含了词的使用方法

Flight:	N
Say:	Verb；(S\Pronoun)
Cancel:	(S\NP)/NP
……	

词汇化语法（lexicalized grammar）与组合范畴语法（CCG）

组合范畴语法
(combinatory categorial grammar, CCG)

核心思想：词汇的类别（category）变得复杂；通过复杂的类别指导每个词该怎么用。

组合规则（rules）

1）前向匹配与后向匹配(forward function application, backward function application)

$$X/Y\ Y \Rightarrow X$$
$$Y\ X\backslash Y \Rightarrow X$$

2）并联(conjunction)

$$X\ CONJ\ X \Rightarrow X$$

3）前向串联与后向串联(forward/backward composition)

$$X/Y\ Y/Z \Rightarrow X/Z$$
$$Y\backslash Z\ X/Y \Rightarrow X\backslash Z$$

4）类别升迁(type raising)

$$X \Rightarrow T/(T\backslash X)$$
$$X \Rightarrow T\backslash(T/X)$$

We	*flew*	*to*	*Geneva*	*and*	*drove*	*to*	*Chamonix*
NP	(S\NP)/PP	PP/NP	NP	CONJ	(S\NP)/PP	PP/NP	NP
	(S\NP)				(S\NP)		
	S\NP						

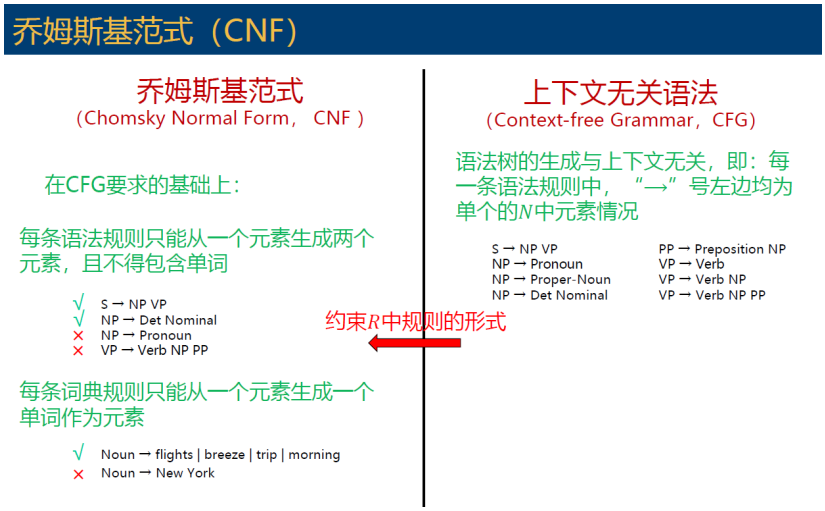
语法间的等同关系

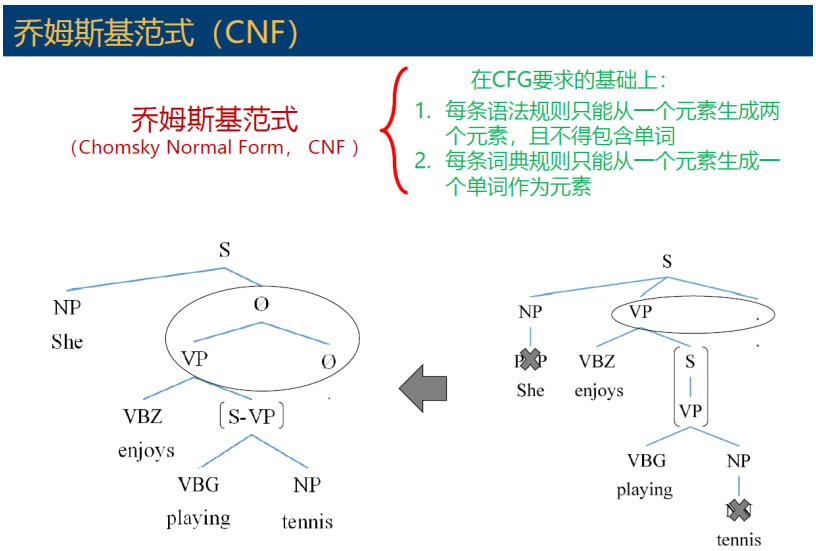
- 弱等价
  两个语法能够生成一模一样的合法语句集合，但是对于同一个句子，语法结构不一定相同
- 强等价

  两个语法能够生成一模一样的合法语句集合，且对于同一个句子，语法结构也相同

乔姆斯基范式 CNF

保证AST为二叉树

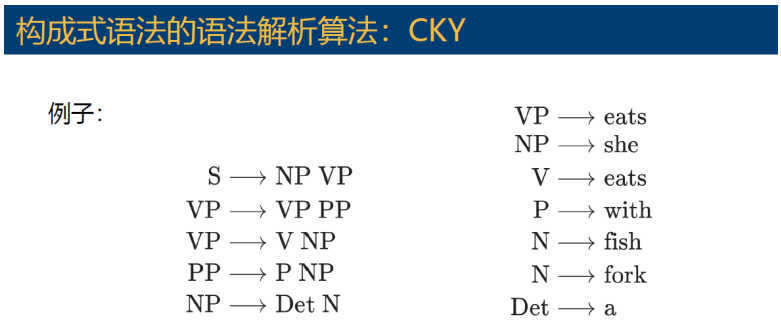




构成式语法的语法解析算法：CKY

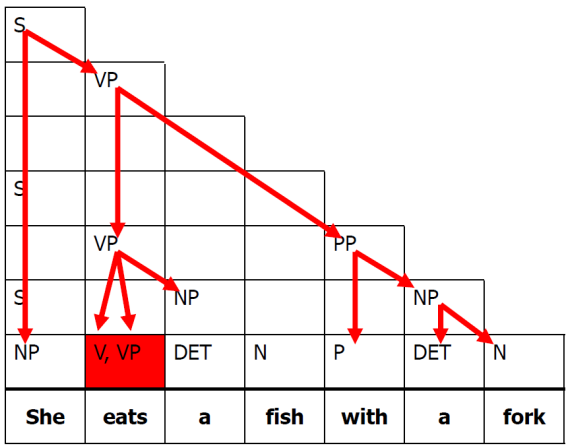
目标：给定一个定义好的CNF语法，对某一句话做自动语法分析

Idea：区间动态规划，$f[i][j]$记录i~j之间的单词所有可能构成的类别



构成式语法的语法解析算法：CKY

例子：



局限性：无法处理有歧义的情况

概率化的构成式语法：PCFG

概率化的上下文无关语法（PCFG）

概率化的上下文无关语法
（Probablistic CFG，PCFG）

在CFG要求的基础上：

对每一条生成规则赋予概率p，表示给定规则左边的constituent之后，这个constituent依照该条语法规则生成子树的条件概率。

即对于由A生成的规则：

$$A \longrightarrow \beta\ [p]$$

有

$$p = \mathrm{P}(\beta|A)$$

S → NP VP [1]
NP → Pronoun [0.3]
NP → Proper-Noun [0.3]
NP → Det Nominal [0.4]
PP → Preposition NP [1]
VP → Verb [0.2]
VP → Verb NP [0.2]
VP → Verb NP PP [0.6]

将R中规则赋予概率

上下文无关语法
（Context-free Grammar，CFG）

语法树的生成与上下文无关，即：每一条语法规则中，“→”号左边均为单个的N中元素情况

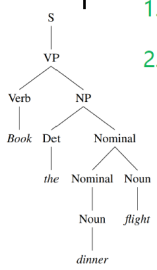
S → NP VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
PP → Preposition NP
VP → Verb
VP → Verb NP
VP → Verb NP PP

概率化的上下文无关语法（PCFG）

概率化的上下文无关语法
（Probablistic CFG，PCFG）

Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

1. 判断给定的语法树是否合法。
2. 给定句子，推断出他合法的语法树。
3. 给定句子，在所有合法的语法树中，给出最有可能的语法结构。
4. 像语言模型一样计算给定句子的概率。



上下文无关语法
（Context-free Grammar，CFG）

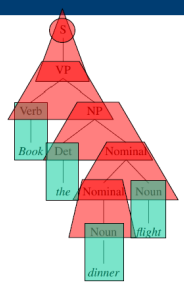
Rules			Rules		
S	→	VP	S	→	VP
VP	→	Verb NP	VP	→	Verb NP NP
NP	→	Det Nominal	NP	→	Det Nominal
Nominal	→	Nominal Noun	NP	→	Nominal
Nominal	→	Noun	Nominal	→	Noun
			Nominal	→	Noun
Verb	→	book	Verb	→	book
Det	→	the	Det	→	the
Noun	→	dinner	Noun	→	dinner
Noun	→	flight	Noun	→	flight

1. 判断给定的语法树是否合法。
2. 给定句子，推断出他合法的语法树。

概率化的上下文无关语法（PCFG）

概率化的上下文无关语法
（Probablistic CFG，PCFG）

Rules			P	Rules			P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

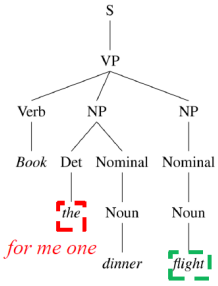


给定左边语法规则，计算右边句子(V)及其对应的语法树(T)出现的概率。

$$P(T,V) = \cancel{P(S)}P(S \to VP|S)P(VP \to Verb\ NP|VP)P(Nominal \to Det\ Nominal|NP)$$
$$P(Nominal \to Nominal\ Noun|Nominal)$$
$$P(Nominal \to Noun|Nominal)$$
$$P(Verb \to Book|Verb)P(Det \to The|Det)P(Noun \to dinner|Noun)$$
$$P(Noun \to flight|Noun)$$

$$P(S) = 1$$

概率化的上下文无关语法（PCFG）



N-gram只关注固定区间以内的单词，因而对flight的预测不受远端单词的影响；

但是基于树结构的语言模型，更改句子内任意单词，均会影响树结构本身，进而影响对flight的预测。

$$P(V) = \sum_{T\ s.t.\ y(T)=V} P(T,V)$$

基于树结构的语言模型

$$P(V) = P(V_1)P(V_2|V_1)P(V_3|V_2)$$
$$P(V_4|V_3)P(V_5|V_4)P(V_6|V_5)\cdots$$

N-gram语言模型

PCFG语法解析

给定一个句子，推断出其概率最大的语法树(T)即是这样一个优化问题：

$$\hat{T}(V) = \underset{T\ s.t.\ y(T)=V}{\operatorname{argmax}}\ P(T|V)$$

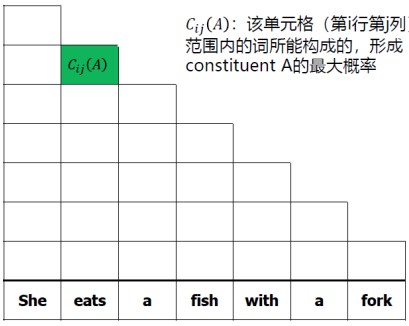
这里面$y(T) = V$代表语法树T生成的句子是V。

$$\hat{T}(V) = \underset{T\ s.t.\ y(T)=V}{\operatorname{argmax}}\ P(T|V) = \underset{T\ s.t.\ y(T)=V}{\operatorname{argmax}}\ \frac{P(T,V)}{P(V)}$$

$$= \underset{T\ s.t.\ y(T)=V}{\operatorname{argmax}}\ P(T,V) = \underset{T\ s.t.\ y(T)=V}{\operatorname{argmax}}\ P(T)$$

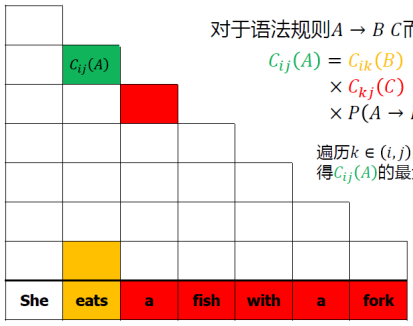
$$P(T,V) = P(T)$$

PCFG的语法解析：Probablistic CKY



PCFG的语法解析：Probablistic CKY

例子：

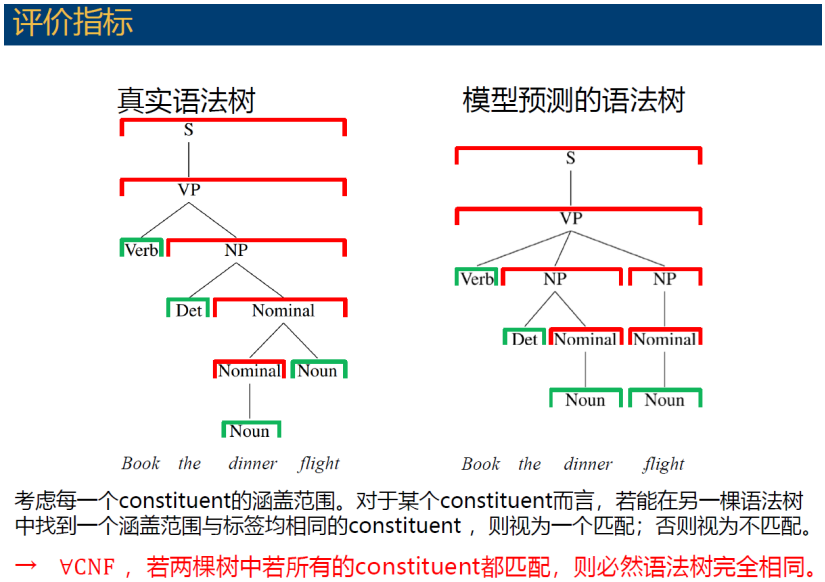


对于语法规则$A \to B\ C$而言：

$$C_{ij}(A) = C_{ik}(B) \times C_{kj}(C) \times P(A \to B\ C|A)$$

遍历$k \in (i,j)$即可求得$C_{ij}(A)$的最大值

评价指标



指标：Precision，Recall，F1

- Labeled F1

  考虑constituents涵盖范围以及标签，仅当两方面都正确时，才视作匹配

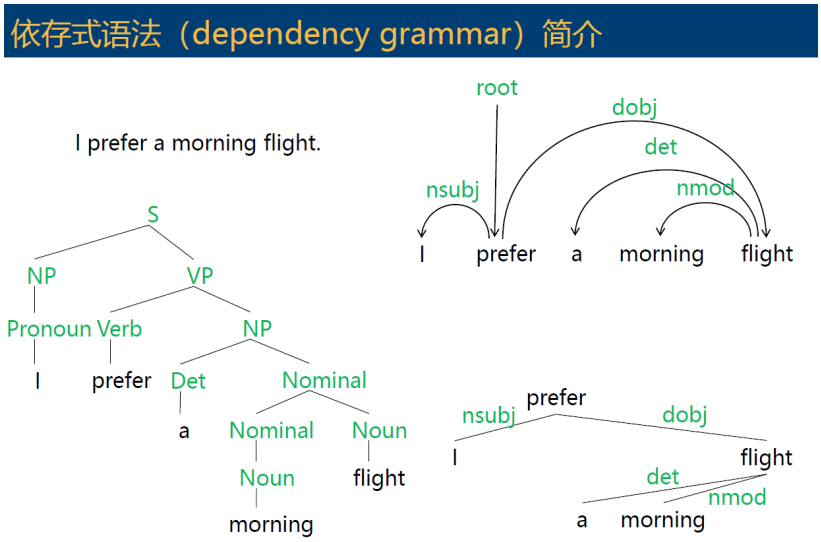
- Unlabeled F1

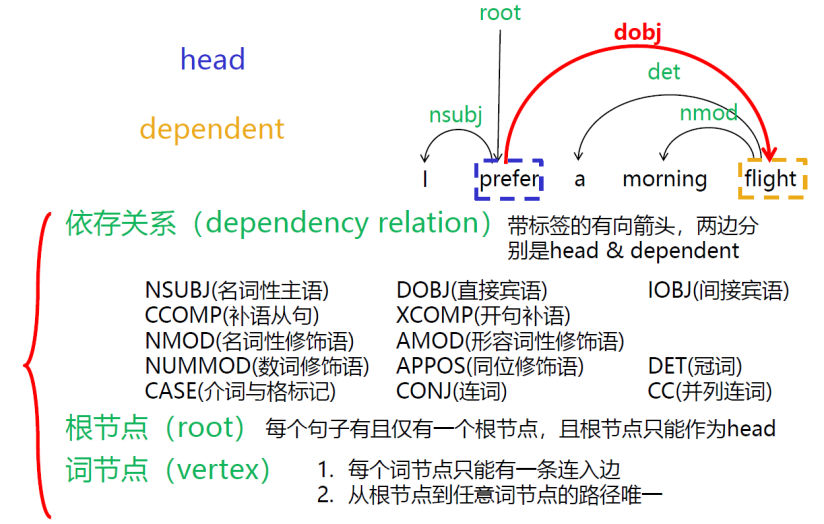
  只考虑constituents涵盖范围，即便标签预测错误，也视作是匹配

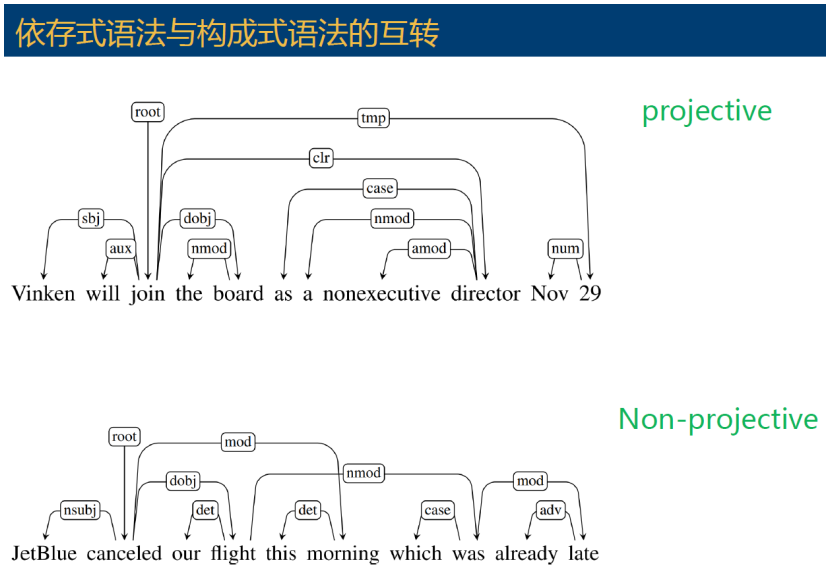
Chap2.6 语法解析：依存式语法

依存式语法

- Constituent的种类随着句子语序的变化指数型增长
- 词之间关系的类型不随语序的改变而改变







- 并非所有的依存式语法都可以转换为构成式语法

依存式语法解析：shift-reduce parser

1. 将句子中的单词一个个推入stack
2. 对stack顶部的两个元素进行连边，连边后的dependent从stack删除
3. 最终清空stack时，语法树就已标记完成

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

book	me	the	morning	flight

依存关系输出

parser → leftArc / rightArc / shift

堆栈（stack）

				root

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

me	the	morning	flight	

依存关系输出

parser → leftArc / rightArc / shift

堆栈（stack）

			root	book

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

the	morning	flight		

依存关系输出

parser → leftArc / rightArc / shift

堆栈（stack）

		root	book	me

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

the	morning	flight		

依存关系输出

book → me

parser → leftArc / rightArc / shift

堆栈（stack）

			root	book

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

依存关系输出

book → me

parser → leftArc / rightArc / shift

堆栈（stack）

root	book	the	morning	flight

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

依存关系输出

book → me

morning ← flight

parser → leftArc / rightArc / shift

堆栈（stack）

	root	book	the	flight

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

Book me the morning flight

剩余词列表（buffer）

依存关系输出

book → me

morning ← flight

the ← flight

book ← flight

root ← book

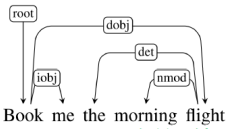
parser → leftArc / rightArc / shift

堆栈（stack）

				root

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式



剩余词列表（buffer）

依存关系输出

book → me

morning ← flight

the ← flight

book ← flight

root ← book

parser → leftArc / rightArc / shift

堆栈（stack）

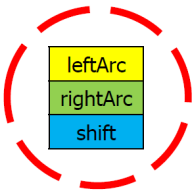
				root

使用shift-reduce parser做依存式语法解析

Shift-reduce parser的解析形式

1. leftArc　　将stack中最顶部两个词连接一条边。边的指向是从顶部第二个词指向最顶部的词。并且删除这条边的dependent。
2. rightArc　　将stack中最顶部两个词连接一条边。边的指向是从最顶部的词指向顶部第二个词。并且删除这条边的dependent。
3. shift　　将buffer中的下一个词推入stack中。

reduce

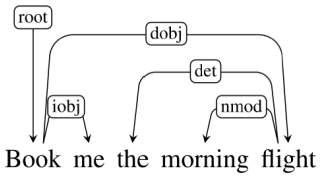


Parser可看做是一个分类器

- 读取当前时刻stack和buffer中的单词，输出三个类别的概率（若要预测具体arc的标签，则类别数不止三个）
- 分类器的feature，可按照CRF中定义feature模板的形式来定
- 分类器的训练，则依赖已经标记好的训练集

需要防止分类器出错

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	



- LeftArc & shift

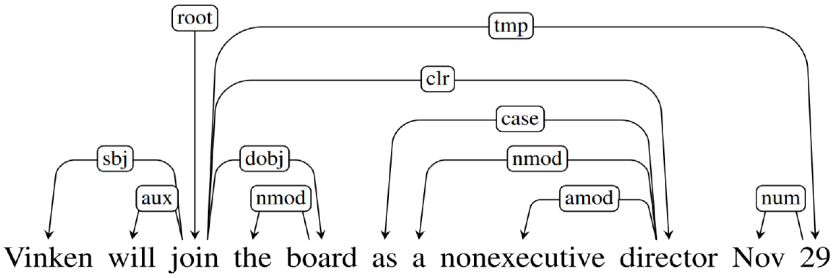
  只要分类器愿意，即可选择该操作

- RightArc

  存在限制条件：最顶部的词的所有dependents都已经被确定

评价指标

评价指标



每个单词有且仅有一次被作为dependent!
只要去数每个单词作为dependent时，他的head有没找对即可。

UAS (unlabeled attachment score):
每个单词的head被找对的比例。
LAS (labeled attachment score)
每个单词的head被找对，并且边的标签也对的比例。

基于深度学习的自然语言处理

Chap3.1 词向量

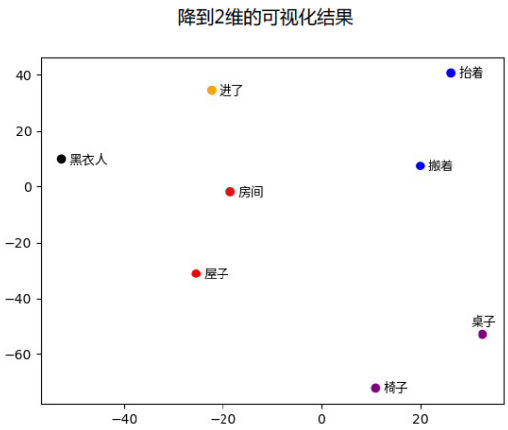
词的表示

- One-Hot表示

  词袋是对语句和短语的一种建模方式，即只记录短语中每个词的出现次数，而忽略词的排列顺序

  问题：维度爆炸，不能反映词与词之间的联系

- 分布式表示



$$
\begin{aligned}
\text{黑衣人} &\quad [-0.001, -0.085, \ldots, -0.030] \\
\text{搬着} &\quad [-0.064, 0.022, \ldots, 0.053] \\
\text{抬着} &\quad [-0.146, 0.018, \ldots, -0.059] \\
\text{桌子} &\quad [-0.057, 0.051, \ldots, -0.089] \\
\text{椅子} &\quad [-0.244, -0.048, \ldots, 0.101] \\
\text{进了} &\quad [-0.034, -0.018, \ldots, -0.012] \\
\text{屋子} &\quad [-0.045, -0.009, \ldots, 0.138] \\
\text{房间} &\quad [-0.149, -0.102, \ldots, 0.035]
\end{aligned}
$$

  - 具有聚类特性，能够通过距离反映词与词之间的关系
  - 每个维度代表了（某种）不同范畴上的语义，词的语义分散在不同的维度中
  - 构建比较复杂，但紧凑，可以降维

  欧几里得距离: $d(\boldsymbol{a}, \boldsymbol{b}) = \|\boldsymbol{a} - \boldsymbol{b}\|_2$，表示的是向量空间中两向量之间的直线段距离

  余弦距离（内积距离）: $d(\boldsymbol{a}, \boldsymbol{b}) = 1 - \cos\theta$，用于衡量两向量方向的一致性

词向量

基于共现频度的词向量

- 共现关系
  - 不同单词在同一语料或同一上下文中出现称为“共现”

1. *黑衣人拿了桌子上的书。*
2. *黑衣人搬着桌子进了屋子。*
3. *桌子上的书是小明的。*

	黑衣人	拿了	桌子	上	的	书	搬着	进了	屋子	是	小明
黑衣人	0	1	2	1	1	1	1	1	1	0	0
拿了	1	0	1	1	1	1	0	0	0	0	0
桌子	2	1	0	2	2	2	1	1	1	1	1
上	1	1	2	0	2	2	0	0	0	1	1
的	1	1	2	2	0	2	0	0	0	1	1
书	1	1	2	2	2	0	0	0	0	1	1
搬着	1	0	1	0	0	0	0	1	1	0	0
进了	1	0	1	0	0	0	1	0	1	0	0
屋子	1	0	1	0	0	0	1	1	0	0	0
是	0	0	1	1	1	1	0	0	0	0	1
小明	0	0	1	1	1	1	0	0	0	1	0

问题：共现矩阵比较稀疏，存储效率较低；未能达到降维的目的，维度依然等同于词表的大小

解决方案：特征值分解

1. 对共现矩阵做特征值分解

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{-1} \quad \boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$$

2. 得到的特征向量是稠密的连续向量，可以作为词的分布式表达

对之前得到的共现矩阵特征值分解，并将特征值按照绝对值从大到小排序，可以得到如下结果

$$\boldsymbol{U} = \begin{array}{c} \begin{bmatrix} 0.31 & 0.35 & 0.30 & 0 & 0 & 0.77 & -0.16 & 0 & 0 & 0 & -0.26 \\ 0.23 & -0.03 & 0 & 0 & 0 & -0.12 & 0.77 & 0 & 0 & 0 & -0.59 \\ 0.47 & 0.15 & -0.86 & 0 & 0 & 0.07 & -0.10 & 0 & 0 & 0 & 0.04 \\ 0.41 & -0.20 & 0.19 & 0.10 & -0.81 & -0.21 & -0.21 & 0 & 0 & 0 & -0.06 \\ 0.41 & -0.20 & 0.19 & -0.75 & 0.32 & -0.21 & -0.21 & 0 & 0 & 0 & -0.06 \\ 0.41 & -0.20 & 0.19 & 0.65 & 0.49 & -0.21 & -0.21 & 0 & 0 & 0 & -0.06 \\ 0.12 & 0.46 & 0.12 & 0 & 0 & -0.23 & 0.08 & 0.25 & -0.15 & 0.76 & 0.17 \\ 0.12 & 0.46 & 0.12 & 0 & 0 & -0.23 & 0.08 & 0.50 & -0.15 & 0.63 & 0.17 \\ 0.12 & 0.46 & 0.12 & 0 & 0 & -0.23 & 0.08 & -0.75 & 0.30 & -0.13 & 0.17 \\ 0.22 & -0.21 & 0.08 & 0 & 0 & 0.21 & 0.34 & -0.26 & -0.66 & -0.05 & 0.49 \\ 0.22 & -0.21 & 0.08 & 0 & 0 & 0.21 & 0.34 & 0.26 & 0.66 & 0.05 & 0.49 \end{bmatrix} \end{array} \begin{array}{l} \text{黑衣人} \\ \text{拿了} \\ \text{桌子} \\ \text{上} \\ \text{的} \\ \text{书} \\ \text{搬着} \\ \text{进了} \\ \text{屋子} \\ \text{是} \\ \text{小明} \end{array}$$

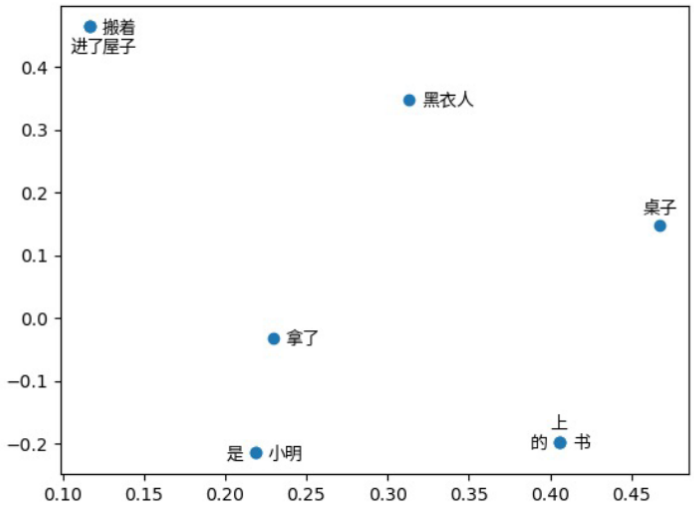
$$\boldsymbol{D} = \mathrm{diag}(8.7, 3.1, -2.6, -2, -2, -1.7, -1.2, -1, -1, -1, 0.7)$$

$\boldsymbol{U}$中每一行都对应了词表中一个词的词向量

3. 可以根据特征值绝对值的大小，舍弃较小的特征值对应的特征维度，实现词向量的降维

   每个特征向量u_i都代表了用于重建共现矩阵的一种特征，而其对应的特征值λ_i的绝对值则反映了该特征维度对重建共现矩阵的重要度

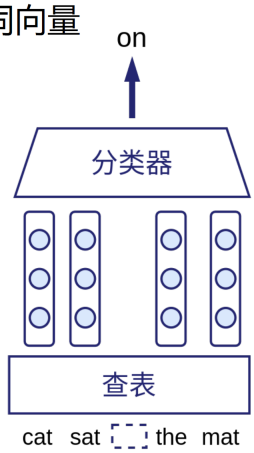
取特征值绝对值最显著的两个维度（即$\boldsymbol{U}$中最左侧的两列）可视化



问题：共现矩阵的统计时空开销很大；基于特征值分解的降维方法同样需要巨大的计算开销

CBOW

- Continuous Bag of Words (CBOW)
  - 利用统计模型从上下文预测当前词的条件概率，在这个过程中得到词向量



$$p^{(w_i)} = \hat{P}\left(w_i \mid w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}\right)$$
$$p^{(w_i)} = f\left(e_{i-k}, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{i+k}\right)$$
$$\mathcal{L} = -\sum_{e \in \mathcal{C}} \log p^{(w_i)}$$

- w_i表示词i，e_i代表词i的词向量，是<u>要学习的目标</u>
- $\mathcal{C}$是语料库，e代表输入样例
- f代表含有可学习参数的统计模型，$\mathcal{L}$是负对数似然

CBOW的实现

采用前向神经网络实现CBOW时

$$\boldsymbol{p} = \operatorname{softmax}(\boldsymbol{W}\boldsymbol{e})$$

- $\boldsymbol{W}$是神经网络的权重矩阵，其行数就是输出维度，也就是词表的大小，其列数就是输入维度
- e是由输入的上下文嵌入$e_{i-k}, \ldots, e_{i-1}, e_{i+1}, \ldots, e_{i+k}$计算得到的向量
  - 采用各输入词向量的拼接

$$\boldsymbol{e} = \left[e_{i-k}^\top, \ldots, e_{i-1}^\top, e_{i+1}^\top, \ldots, e_{i+k}^\top\right]^\top$$

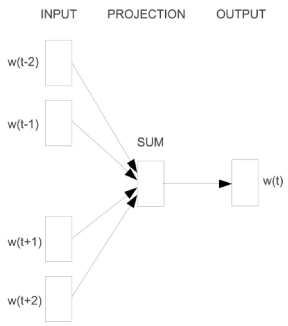
弊端在于会增大输入的维度

- 采用各输入词向量的加和

  - 为了降低输入维度，CBOW的实际实现采用词袋的方式生成 e ，即

$$e = \sum_{-k \le j \le k, j \ne 0} e_{i+j}$$

  - 代价则是损失了输入的上下文中的词序信息



CBOW

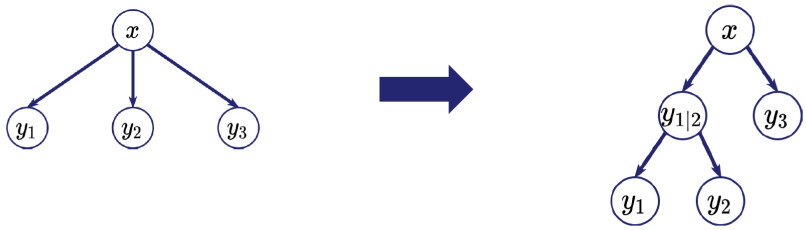
为了实现高效的训练，有两种降低输出维度的方法，其思想都是将多分类任务转化为多个二分类任务

- 分层柔性最大化

  增加计算次数，减少每次计算量

  e.g., 将100个单词分为10组，先预测属于哪组，再预测具体是哪个单词

  e.g., 将多分类问题转换为多个二分类问题，组成一棵二叉分类树（哈夫曼树最优）



用二进制串表示二叉分类树上的分类路径，用0表示走左分支，1表示走右分支

$$\hat{P}(y_1) = \mathrm{softmax}_1(y) = \frac{\exp({}^1y)}{\sum \exp({}^iy)}$$

$$\hat{P}(y_2) = \mathrm{softmax}_2(y)$$

$$\hat{P}(y_3) = \mathrm{softmax}_3(y)$$

$$\begin{aligned}\tilde{P}(y_1) &= \tilde{P}(00) \\ &= (1-\sigma(y_3))(1-\sigma(y_2)) \\ \tilde{P}(y_2) &= \tilde{P}(01) \\ &= (1-\sigma(y_3))\sigma(y_2) \\ \tilde{P}(y_3) &= \tilde{P}(1) \\ &= \sigma(y_3)\end{aligned}$$

- 负采样

  - 将多分类问题改造为正负例的二分类问题

$$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{W}\boldsymbol{e})$$

  - 其中 $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]^\intercal$ 的行数就等于输出的维度，即分类词表的大小，其每一行的向量 $\boldsymbol{w}_i$ 都对应了输出的一个维度
  - 若将 $\boldsymbol{w}_i$ 看作词 i 的索引向量，将 $\boldsymbol{w}_i^\intercal \boldsymbol{e}$ 看作两个向量的内积相似度，则该分类任务实质上就是选取相似度最高的词作为分类结果
  - 因此优化的目标实质上就是最大化正确分类的相似度 $\boldsymbol{w}_i^\intercal \boldsymbol{e}$ ，而最小化其他错误分类的相似度 $\boldsymbol{w}_j^\intercal \boldsymbol{e}$ （ $j \ne i$ ）

  仅挑选部分负样本进行计算

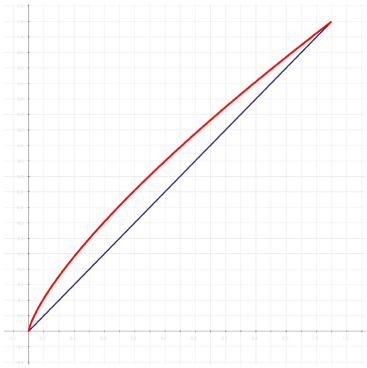
$$\mathcal{L} = -\sum_{e \in \mathcal{C}} \left[\log \sigma\left(\boldsymbol{w}_i^\top \boldsymbol{e}\right) - \sum_{j \neq i} \log \sigma\left(\boldsymbol{w}_j^\top \boldsymbol{e}\right) \right]$$

$$\mathcal{L} = -\sum_{e \in \mathcal{C}} \left[\log \sigma(\boldsymbol{w}_i^\intercal \boldsymbol{e}) - \sum_{j \sim P(w_j|\mathcal{C})} \log \sigma(\boldsymbol{w}_j^\intercal \boldsymbol{e}) \right]$$

- 当词表很大时，遍历词表中的负样例代价很大，因此需要采样
- 原论文中的采样函数：

$$P(w_i|\mathcal{C}) = \frac{\alpha_i^{\frac{3}{4}}}{\sum_j \alpha_j^{\frac{3}{4}}}$$

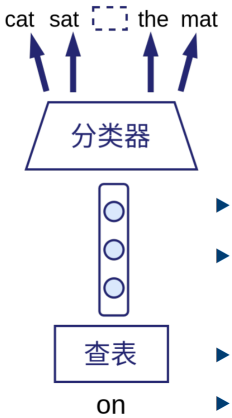
- 其中α_i 表示词i在语料库中出现频率
- 采用小于1的幂形式可以适当增加罕见词被采样到的概率，有助于改善训练的词向量的性能



$f(x) = x^{\frac{3}{4}}$函数图像

Skip-gram

- Continuous Skip-Gram (Skip-gram)
  - 利用统计模型预测当前词的上下文，在此过程中训练得到当前词的词向量



$$p_j^{(w_j)} = \hat{P}_j(w_j|w_i) \quad -k \leq j \leq k, j \neq 0$$

$$\boldsymbol{p}_j = f_j(e_i)$$

$$\mathcal{L} = -\sum_{e \in \mathcal{C}} \sum_{-k \leq j \leq k, j \neq 0} \log p_j^{(w_j)}$$

- w_i表示词 i ，e_i代表词i 的词向量
- $\mathcal{C}$ 是语料库，e代表输入样例

$$e_{i-k}, \ldots, e_{i-1}, e_i, e_{i+1}, \ldots, e_{i+k}$$

- f_j 代表第 j 个位置的分类器对应的等效函数
- $\mathcal{L}$是优化模型使用的损失函数

Skip-gram和CBOW具有对称的结构，二者在实现上也面临相似的问题，也可以通过相似的技术解决

GloVe

- GloVe (Global Vectors)

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{j=1}^{n} f(x_{ij}) \left(e_i^\intercal e_j + b_i + b_j - \log x_{ij}\right)^2$$

  - 式中，x_{ij}表示w_j在w_i的邻域内出现的次数

$$f(x) = \begin{cases} \left(\dfrac{x}{x_{\mathrm{TH}}}\right)^{\frac{3}{4}} & x < x_{\mathrm{TH}} \\ 1 & \end{cases}$$

  - 是用于平衡样本数量的权重函数，抑制过大的x_{ij}而适当提升较小的x_{ij}

推导过程

- 首先来看GloVe和Skip-gram的关系
- Skip-gram的损失函数

$$\mathcal{L} = -\sum_{e \in \mathcal{C}} \sum_{-k \le j \le k, j \ne 0} \log p_j^{(w_j)}$$

- 如果我们将 w_i 的邻域扩展为语料库全局，则会得到

$$\mathcal{L} = -\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \log \hat{P}(w_j | w_i)$$

  - Skip-gram中，

$$\hat{P}(w_j | w_i) = \frac{\exp(\boldsymbol{w}_j^\intercal e_i)}{\sum_l \exp(\boldsymbol{w}_l^\intercal e_i)}$$

$$\log \hat{P}(w_j | w_i) = \boldsymbol{w}_j^\intercal e_i - \log \sum_l \exp(\boldsymbol{w}_l^\intercal e_i)$$

  - 若将 $-\log \sum_l \exp(\boldsymbol{w}_l^\intercal e_i)$ 替换为 $b_i + b_j - \log x_{ij}$，

    就可以得到GloVe中的形式

思路

- GloVe的出发点
  - 词义的相关性会更好地反映在共现概率的差异上

	k=solid	k=gas	k=water	k=fashion
P(k|ice)	1.9e-4	6.6e-5	3.0e-3	1.7e-5
P(k|steam)	2.2e-5	7.8e-4	2.2e-3	1.8e-5
P(k|ice)/P(k|steam)	8.9	8.5e-2	1.36	0.96

$$\frac{P(w_k | w_i)}{P(w_k | w_j)} = \frac{P_{ik}}{P_{ij}}$$

- 若能让 $\log P_{ik} = e_i^\intercal e_k$
- 则可以具有很好的性质

$$(e_i - e_j)^\intercal e_k = \log \frac{P_{ik}}{P_{jk}}$$

- 可以用差向量的内积表示共现概率之比
- 此时 $e_i^\intercal e_k = \log P_{ik} = \log x_{ik} - \log x_i$
- 其中 $x_i = \sum_k x_{ik}$
- 则 $\log x_{ik} = e_i^\intercal e_k + \log x_i$
- 其中由于 $\log x_i$ 和 k 无关，训练中简单起见可以替换为偏置量 b_i，同时为了保持对称性，加上偏置量 b_k 就得到了损失函数中的式子

$$\log x_{ik} = e_i^\intercal e_k + b_i + b_k$$

词向量的评价

- **下游任务的性能**
  - 利用训练好的词向量表示词，用作下游任务的输入
    - 文段分类 - 识别文段的主题、风格、体裁等
    - 语句情感分类 - 判断感情的极性，或更具体的情感分类
    - 句子匹配 - 找到与原句语义最匹配、最相似的句子
    - 命名实体识别（NER） - 识别语句中的专有名词
    - 词类标注（POS tagging） - 标注句子中单词的词性
- **计算效率**
- **语言学评价**
  - 近义词
  - 类比构词
    - 词法关系 - $e(\text{shine}) + e(\text{raining}) - e(\text{rain}) = e(\text{shining})$
    - 语义关系 - $e(\text{king}) + e(\text{wife}) - e(\text{husband}) = e(\text{queen})$

动态词向量

用于处理多义词与语境建模

- 静态词向量的解决方式

  为同一词的不同含义视作不同的词，分配不同的词向量

  设计或训练专门表示上下文语境信息的向量

  增加了词表的大小

  增加了复杂度

  - 如何确定词的不同含义：自动聚类与专家知识的结合
  - 在使用中如何确定该用哪个词向量：上下文语义的识别
  - 考验技巧性的上下文向量设计方式
- 动态词向量

  使用模型根据上下文语境生成合乎语境契合当前词义的嵌入向量

  动态词向量通常利用预训练语言模型生成

Chap3.2 神经网络语言模型

统计语言模型回顾

基本任务：已知前文预测下一个字

- 输入: 单词序列$w_1, w_2, \cdots, w_{k-1}$
- 输出: 对下一个单词预贬的概率$P\left(w_k \mid w_1, \cdots, w_{k-1}\right)$

评价语言模型

$$
\begin{aligned}
\operatorname{PPL}(W) &= \left(\prod_{k=1}^{K} \frac{1}{P\left(w_k \mid w_1, \cdots, w_{k-1}\right)}\right)^{\frac{1}{K}} \\
&= 2^{\frac{1}{K} \sum_{k=1}^{K} \log P(w_k | w_1, \cdots, w_{k-1})} \\
&= 2^{H(p, \hat{p})}
\end{aligned}
$$

$$P\left(w_1, w_2, \cdots, w_k\right)=\prod_{k=1}^{K} P\left(w_k \mid w_1, \cdots, w_{k-1}\right)=\bar{P}^K$$

$$\mathrm{PPL}=\frac{1}{\bar{P}}$$

Perplexity越小，交叉熵越小，预测分布越接近真实分布

PPL的理解

- Weighted Average Branching Factor
  - 以0-9的数字预测为例

Bigram LM

完全随机

$\mathrm{PPL}=(10^{100})^{\frac{1}{100}}=10$

9后不会出现0，其他随机

$\mathrm{PPL} \approx (9^9 \times 10^{90})^{\frac{1}{99}}=9.90$

- 等价于需要构建多大的词表能使分布为均匀分布

$$\begin{aligned}\mathrm{PPL} &= (\mathrm{PPL}^K)^{\frac{1}{K}} \\ &= (\prod_{i=1}^{K} \frac{1}{1/\mathrm{PPL}})^{\frac{1}{K}}\end{aligned}$$

n-gram

统计语言模型回顾：n-gram

- 马尔科夫假设：每个字出现的概率仅与其前面出现的$n-1$个字有关。
- 基本思想：用出现频率代替出现概率
- 公式：

$$\begin{aligned}P(w_k|w_1,\cdots,w_{k-1}) &\approx P(w_k|w_{k-n+1},\cdots,w_{k-1}) \\ &= \frac{P(w_{k-n+1},\cdots,w_{k-1},w_k)}{P(w_{k-n+1},\cdots,w_{k-1})}\end{aligned}$$

- 实现：
  - 对数据集中的所有n-gram, (n-1)-gram计数并存储
  - $\frac{P(w_{k-n+1},\cdots,w_{k-1},w_k)}{P(w_{k-n+1},\cdots,w_{k-1})}=\frac{\mathrm{Count}(w_{k-n+1},\cdots,w_{k-1},w_k)}{\mathrm{Count}(w_{k-n+1},\cdots,w_{k-1})}$

- Storage Problem

  模型需要存储的数据规模较大

  不能建模远距依赖关系

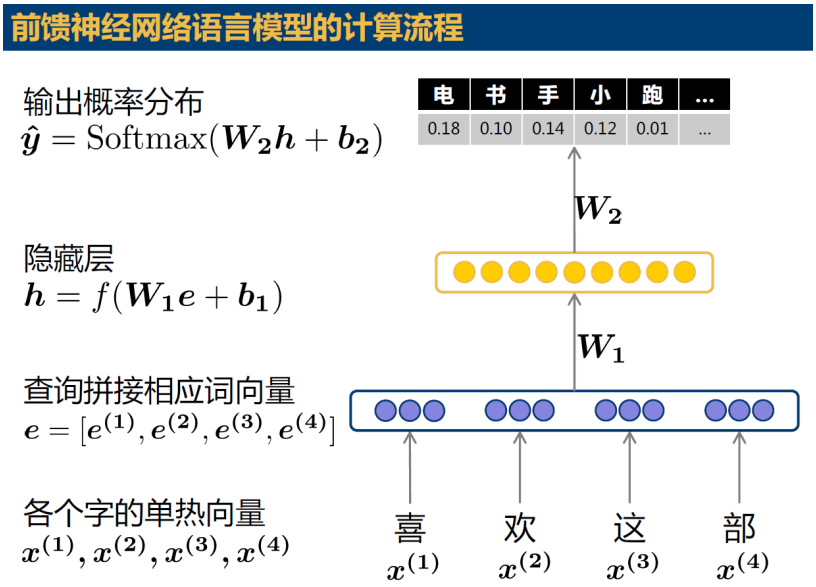
- Sparsity Problem

  低频次&零概率

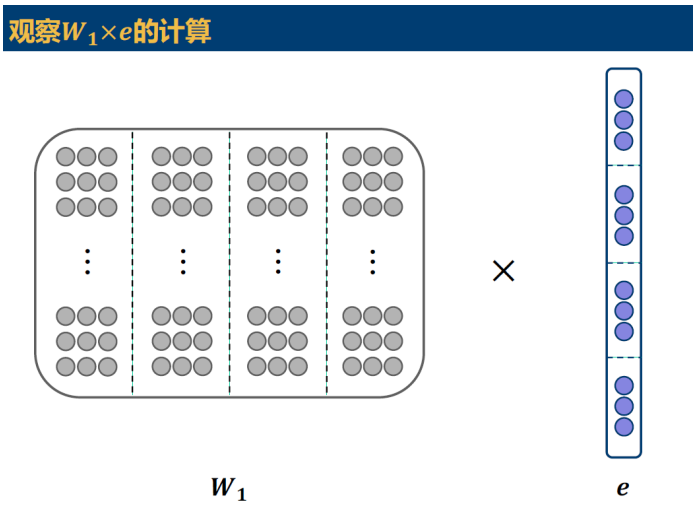
  Discounting & Backing-off

神经网络语言模型

前馈神经网络



- 权重矩阵的大小仍会随着n的增大而增大
- 仍不能很好的建模长时依赖



- 需要单独对每一个历史时刻的词学习参数
- 不同时刻间共通的处理要不断重复学习

循环神经网络

Idea: 在时间维度上大量使用共享参数

分解权重矩阵$\boldsymbol{W_1} \in \mathbb{R}^{d_h \times (n \times d)}$

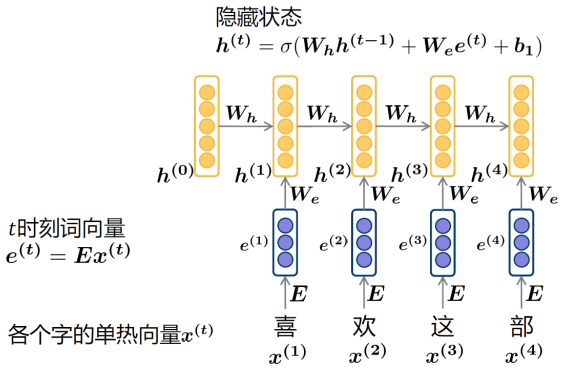
- 不同时刻间相同的处理

  $\boldsymbol{W_e} \in \mathbb{R}^{d_h \times d}$

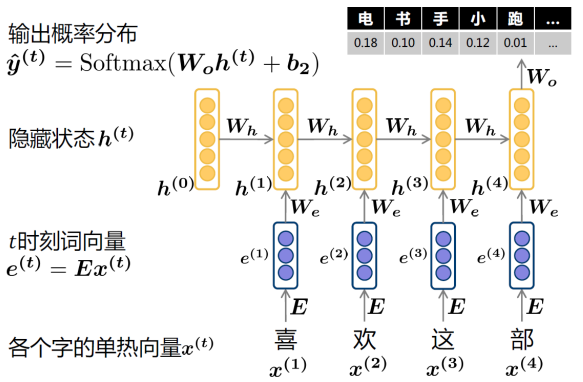
- 不同时刻间不同的处理

  假定已有信息向下一时刻传递时的处理是相同的，$\boldsymbol{W_h} \in \mathbb{R}^{d_h \times d_h}$

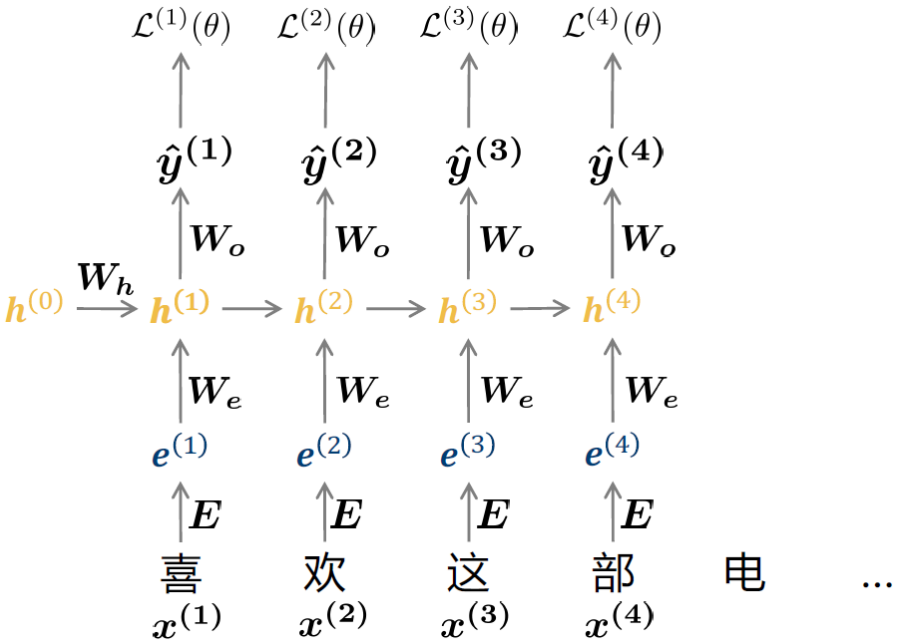
循环神经网络语言模型的计算流程



循环神经网络语言模型的计算流程



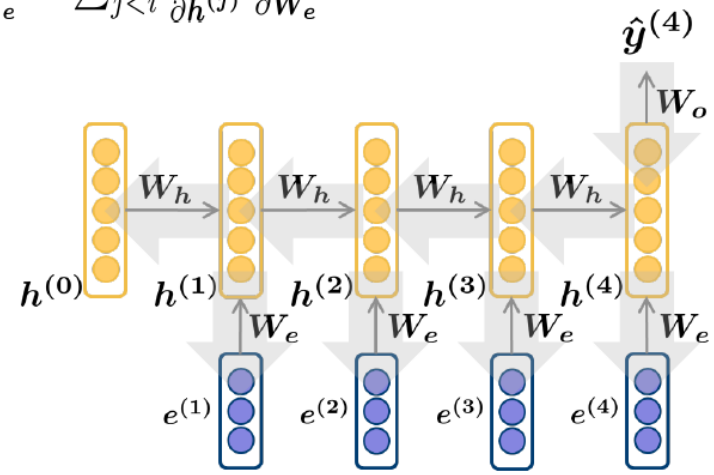
训练过程



$$\mathcal{L}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\log \hat{p}_{w_{t+1}}^{(t)}$$

循环神经网络的梯度计算

- 链式法则
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_h} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_h}$
  - $\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{W}_e} = \sum_{j<i} \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} \frac{\partial \boldsymbol{h}^{(j)}}{\partial \boldsymbol{W}_e}$



注：累加所有时间步的梯度

$$\frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial \mathcal{L}^{(i)}}{\partial \boldsymbol{h}^{(i)}} \prod_{j \le t < i} \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$

- 梯度消失问题

  并不能有效的建立远距依赖关系
- 梯度爆炸问题

  解决方法：clipping

Q：为什么会出现梯度消失问题？

A：时间维度上的深度神经网络；RNN对于历史处理的共享假设加剧了这个现象

Idea：向下一时刻传递信息时，考虑当前时刻的上下文情况

- RNN中$\boldsymbol{h}^{(t-1)}$的意义：记录历史信息，指导后续表达
- 当前时刻的输入w_t的意义：当前时刻进行的表达
- w_t对于$\boldsymbol{h}^{(t-1)}$的影响：完成了部分已有表达任务；为后续的表达添加了新的条件

长短时记忆网络

- 将长期记忆和短期记忆分离
  - **长期记忆**用来存储和传递整个表达过程的状态，每一时刻根据上下文情况更新，且不能被外界直接观察
  - **短期记忆**是根据当前上下文情况，从长期记忆中获取的，用以指导当前时刻的输出以及下一时刻长期记忆变化的部分，相当于当前时刻被“激活”的长期记忆
  - 长期记忆：单元状态$c^{(t)}$，短期记忆：隐藏状态$h^{(t)}$
- 使用门控制记忆的消除、传递与读写

  在使用时，将门与其作用的向量做逐元素的乘法

门是在每个时刻根据当前上下文动态生成的

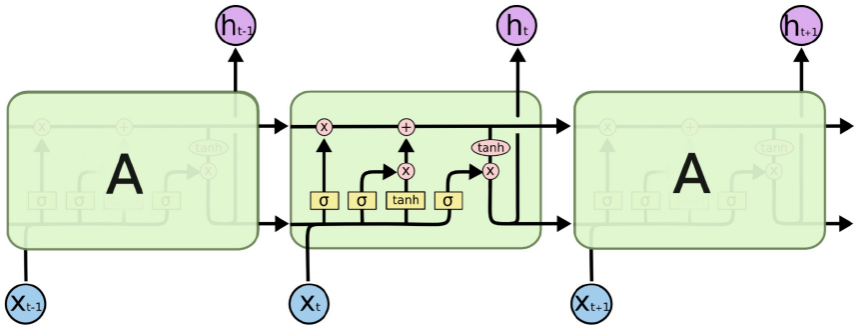
计算公式

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$和单元状态$\boldsymbol{c}^{(t)}$：
  - 遗忘门 $\boldsymbol{f}^{(t)}$：控制原单元状态中内容的保留与遗忘
  - 输入门 $\boldsymbol{i}^{(t)}$ ：控制当前时刻什么内容被写入单元状态
  - 输出门 $\boldsymbol{o}^{(t)}$ ：控制当前时刻什么内容从单元状态中被输出至隐藏状态

  - 遗忘门：$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_f[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_f)$
  - 输入门：$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_i[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_i)$
  - 输出门：$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_o[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_o)$

  - 候选单元状态：$\tilde{\boldsymbol{c}}^{(t)} = \tanh(\boldsymbol{W}_c[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_c)$
  - 单元状态：$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$
  - 隐藏状态：$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)})$

LSTM的流程图示



$$
\begin{aligned}
\boldsymbol{f}^{(t)} &= \sigma(\boldsymbol{W}_f[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_f) \\
\boldsymbol{i}^{(t)} &= \sigma(\boldsymbol{W}_i[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_i) \\
\boldsymbol{o}^{(t)} &= \sigma(\boldsymbol{W}_o[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{c}}^{(t)} &= \tanh(\boldsymbol{W}_c[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_c) \\
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{c}^{(t)})
\end{aligned}
$$

LSTM让网络能够更容易的保留远距的信息，从而极大的缓解了梯度消失的问题（LSTM只需要遗忘门为全1便可完全保留历史信息）

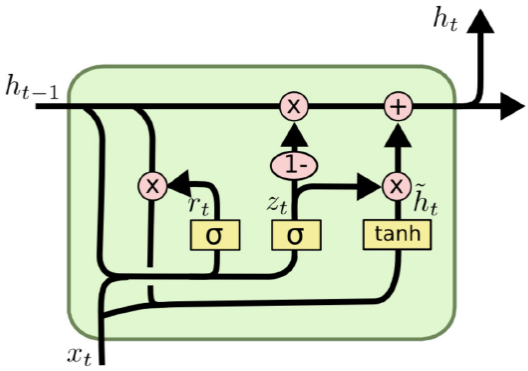
门控循环单元GRU

简化LSTM：GRU

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$（无单元状态）
  - 更新门 $\boldsymbol{z}^{(t)}$：控制原隐藏状态中内容的更新与保留
  - 重置门 $\boldsymbol{r}^{(t)}$：控制原隐藏状态中什么部分被用来和当前时刻的输入一起计算新的隐藏状态内容

- 在时间点t，输入的词向量为$\boldsymbol{x}^{(t)}$，计算其隐藏状态$\boldsymbol{h}^{(t)}$（无单元状态）
  - 更新门：$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{z}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{z}})$
  - 重置门：$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{r}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{r}})$

  - 候选状态：$\tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_{\boldsymbol{h}}[\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{h}})$
  - 隐藏状态：$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$



$$\boldsymbol{z}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{z}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{z}})$$
$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_{\boldsymbol{r}}[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{r}})$$
$$\tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_{\boldsymbol{h}}[\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}] + \boldsymbol{b}_{\boldsymbol{h}})$$
$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$$

Chap3.3 文本分类

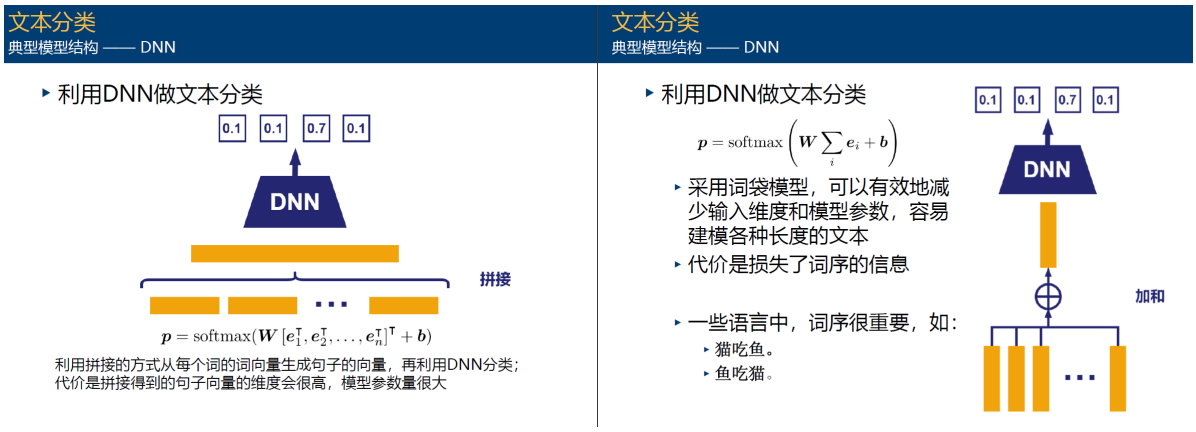
文本分类

输入：视具体应用而定的不同类型、不同长度的文本

输出：预先定义的分类标签

文本分类可以依照输入的篇幅、分类标签体系的不同、以及应用场景等分类

DNN



LSTM

文本分类

典型模型结构 —— LSTM

- 利用LSTM做文本分类

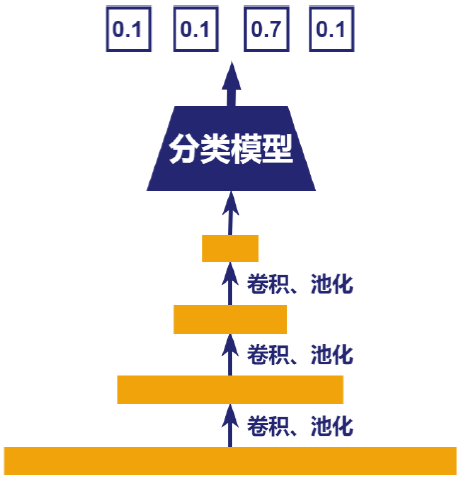
$$\boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]$$
$$\boldsymbol{E}' = \mathrm{LSTM}(\boldsymbol{E})$$
$$\boldsymbol{E}' = [\boldsymbol{e}'_1, \boldsymbol{e}'_2, \ldots, \boldsymbol{e}'_n]$$
$$\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{W}\boldsymbol{e}'_n + \boldsymbol{b})$$

  - LSTM的最后一位输出向量编码了整个词序列的信息，可以用于整段文本的分类
  - LSTM可以在保留顺序信息的情况下，以相对较少的参数量编码各种长度的文段

CNN

卷积层

- 利用CNN（一维卷积）编码文本训练并做文本分类



- 输出的特征图大小为$\frac{N-F}{S}+1$
- 卷积前使用补0可以保留特征图的分辨率，更好的利用边界信息，从而提升性能

CNN文本分类

一维序列上的卷积

- 一维CNN卷积
  - 卷积核以小窗口的形式在输入序列上移动，每个位置产生一个输出
  - 对窗口内的向量序列

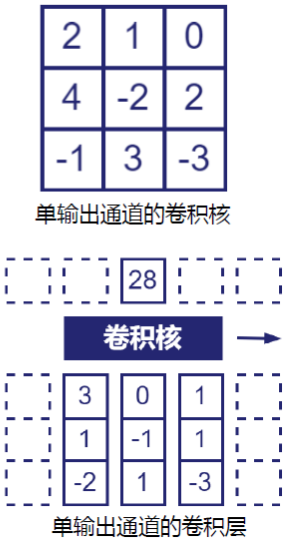
$$\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n],$$

$$\boldsymbol{a}_i = [a_{1i}, a_{2i}, \ldots, a_{di}]^{\intercal} \in \mathbb{R}^d$$

单输出通道的卷积核 $\boldsymbol{B} = (b_{ij})_{d \times n}$

卷积层运算为

$$\mathrm{Conv1d}(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i=1}^{d} \sum_{j=1}^{n} a_{ij} b_{ij}$$



CNN文本分类

理解卷积层运算

- 理解卷积层运算

$$\mathrm{Conv1d}(\boldsymbol{A}, \boldsymbol{B}) = \sum_{i=1}^{d} \sum_{j=1}^{n} a_{ij} b_{ij}$$

  - 卷积层可以看作是两个向量序列的相关运算，或者理解成两个向量的内积运算
  - 余弦相似度： $\cos\theta = \dfrac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|}$
  - 在相同的模长下，内积度量了两个向量的方向的相关性
  - 卷积层的输出值越大，说明窗口内的向量序列和卷积核越相似，每个通道的卷积核都可以看作特定的“模式提取器”，用来抽取输入序列中特定的局部模式，得到可以描述输入序列的某维度特征

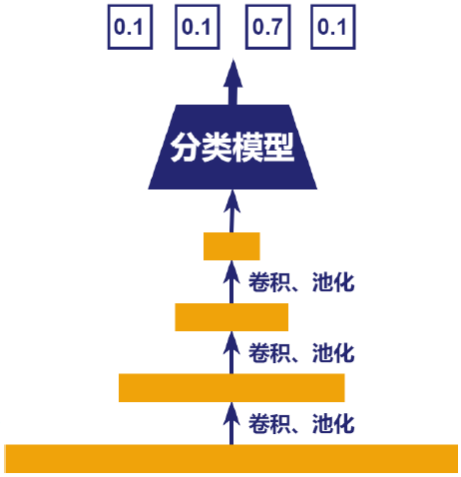
池化层

- 使特征表示更小、更容易管理
- 池化层一般filter size = stride，没有重叠
- 常见的池化操作：最大、平均、最小、随机、固定

CNN文本分类

由二维到一维

- 卷积核为$1 \times F_c \times D$
  - F_c为卷积窗口大小
  - D为特征向量维度
- 池化大小为$1 \times F_p$
  - F_p为池化窗口大小
- 卷积、池化操作后，原输入文本被编码为长度缩减的特征序列，可用LSTM或者DNN继续处理



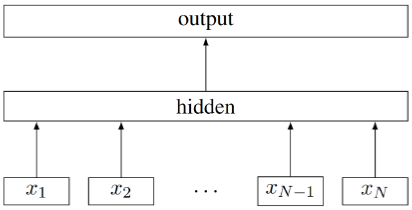
注：与文本上下文相关，相关范围与卷积核大小和层数等有关

FastText

文本分类

典型模型结构 —— FastText

- FastText
  - 与我们介绍过的词袋DNN模型的主要区别在于模型的输入不是单个词的词向量，而是n-gram词组的向量表示，从而在一定程度上保留了局部的词序信息

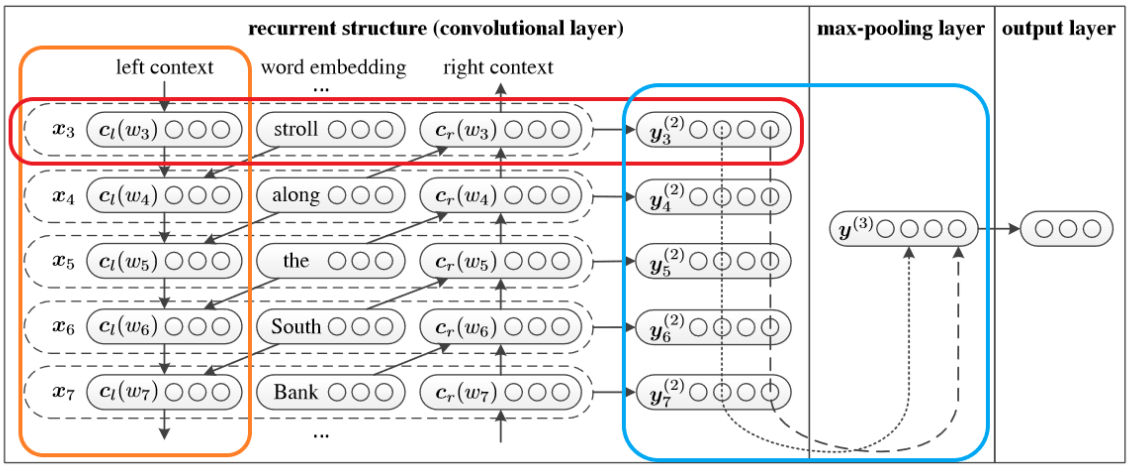


图自[Armand Joulin et al., 2017]

- 猫吃鱼。 ➡ (“[BOS]猫”, “猫吃”, “吃鱼”, “鱼[EOS]”)

RCNN

RCNN结合了CNN与RNN的计算特点，综合了CNN挖掘邻域特征和RNN建模时序关系的优势



- 使用RNN计算上下文信息
- 拼接当前时刻的左右上下文和词向量作为特征表示，随后经过全连接层生成语义向量

  注意，左右上下文中没有当前时刻的信息
- 全局最大值池化层

情感分类

极性分析用于判断一句话或一段言论的情感极性，多用于分析、筛选用户的评论、反馈以及网络舆论等

输入：用户的评论或社交媒体的发言等，通常比较短，仅包含若干句话

模型：通常需要基于极性词库，提取相关特征，再使用分类器分类；构建合适的极性词库也是重要的任务

输出：情感类别标签

- 简单的模型

  简单的情感分类可以利用词袋模型，通过统计文本中出现的正负极性词的比例来实现
- 更精细的模型

  更精细的情感分类模型还应该建模文本中的其他的句式、结构等特征，如：否定词、连接词、反问、转折、让步、假设、虚拟语气等

文本匹配

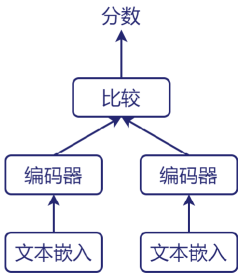
输入：两段文本内容

输出：两段文本是否匹配，或两段文本相似度的度量

应用：问答系统，机器阅读理解，文书检索

文本匹配
典型模型结构

- 基于表示的模型
  - 先利用编码器将输入映射到相同的向量空间，再比较得到的特征，计算相似度
  - 这种结构不要求采用相同的编码器：
    - 具有不同的语义身份，如一个是查询的关键词，一个是要匹配的文章
    - 具有不用的模态，如图像、视频、音频与文本做匹配



文本匹配
典型模型结构

- 编码器可以采用在文本分类中介绍过的各种编码器结构
- 比较阶段要计算两份输入的编码表示的相似度或距离：
  - 余弦相似度 $\cos\theta = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}$
  - 二范数（欧几里得距离）$\|\boldsymbol{a}-\boldsymbol{b}\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
  - 一范数（曼哈顿距离）$\|\boldsymbol{a}-\boldsymbol{b}\|_1 = \sum_i |a_i - b_i|$
- 损失函数的一般形式

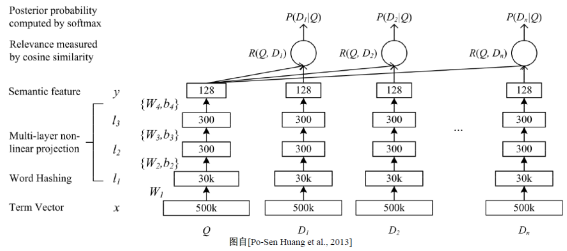
$$\mathcal{L} = \sum_{p^+} \left(l(p^+) - \sum_{p^- \in \mathcal{N}(p^+)} l(p^-) \right)$$

  - 其中p^+ 指数据集中的正样例，即匹配的样本对
  - p^- 为根据p^+构造的负样例，一般可以通过将匹配样本对中的某样本替换为随机样本得到
  - l 为单样本对上的损失，一般就是向量的距离或负的相似度

DSSM

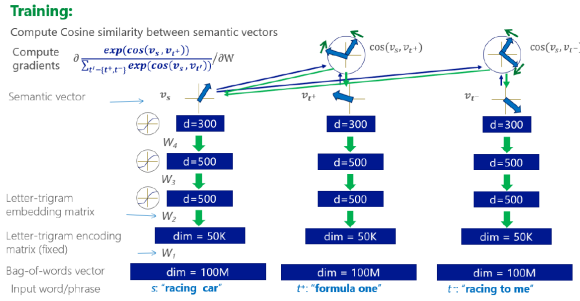
文本匹配
典型模型结构

- DSSM (Deep Structured Semantic Model)
  - 基于DNN提取的特征分别计算查询 Q 与各正负匹配样本的余弦相似度，然后根据余弦相似度计算后验概率
  - 通过这种方法可以学习到好的文本表示



图自[Po-Sen Huang et al., 2013]

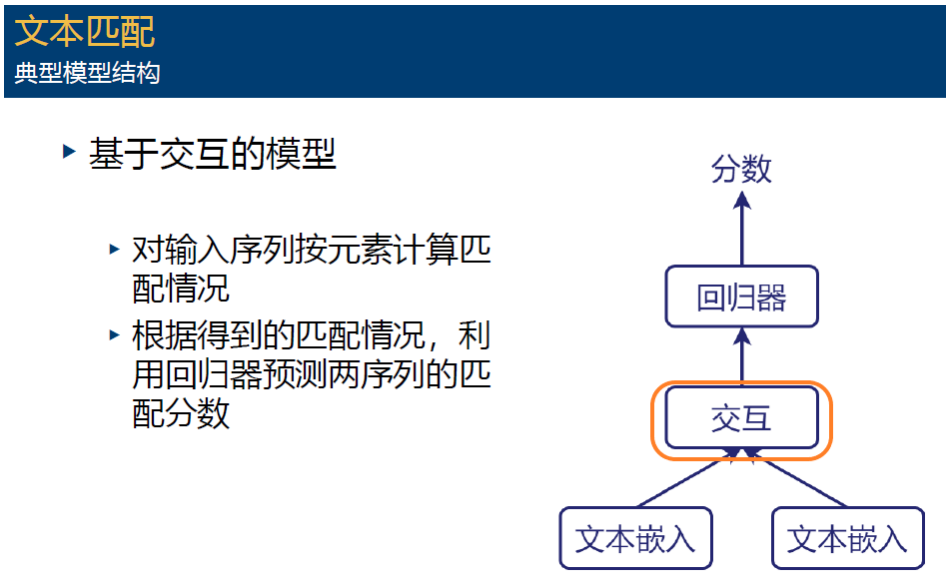
文本匹配
典型模型结构

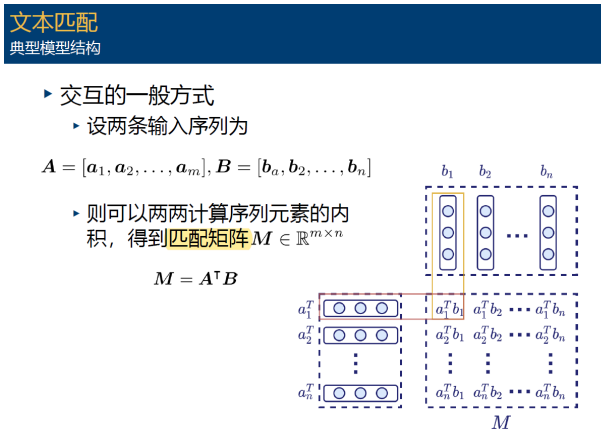


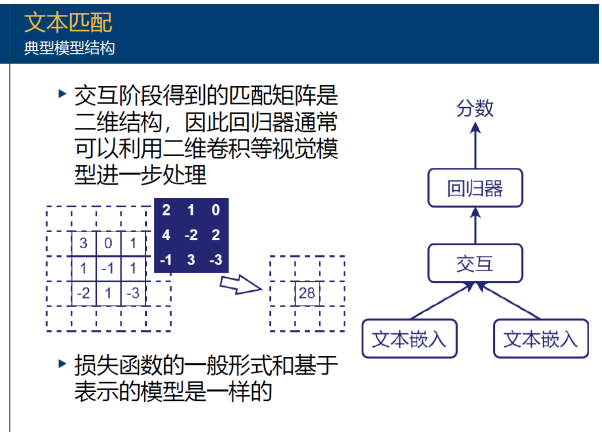
CLSM

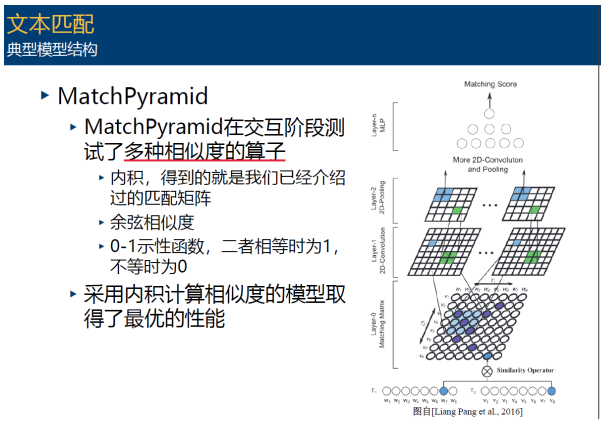
- 相比DSSM，采用CNN取代了DNN构造表示层，因此也被称为Convolutional DSSM
- 采用了和介绍过的FastText中类似的n-gram方案处理输入文本，有助于在一定程度上更多地保留语序信息

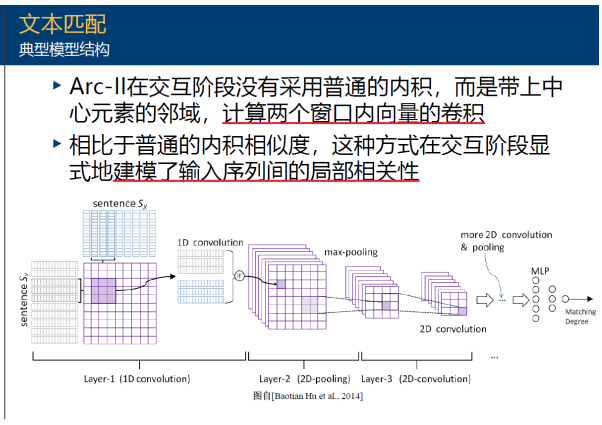
基于交互的模型







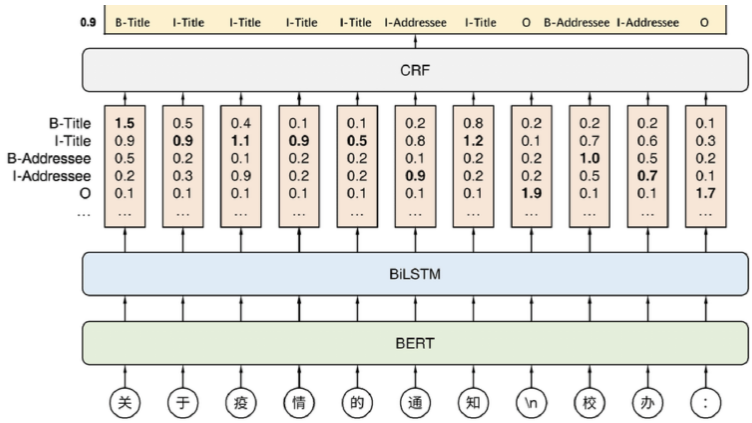




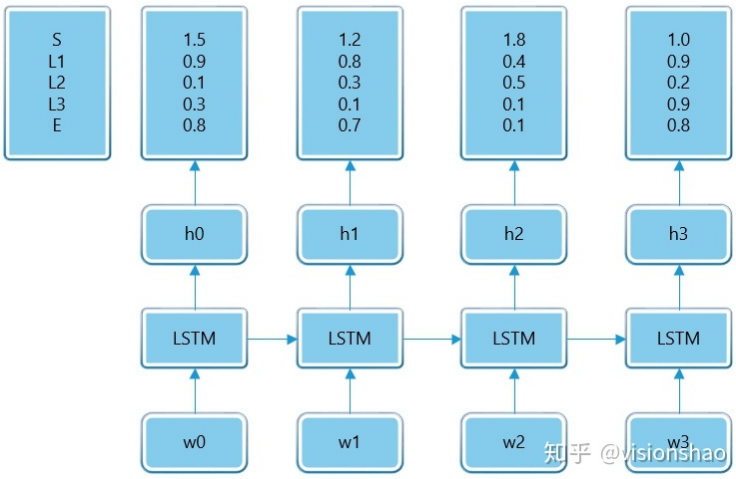
Chap3.4 序列标注

序列标注是NLP中最基础的任务，如分词、词性标注、命名实体识别、关键词抽取、语义角色标注、槽位抽取等实质上都属于序列标注的范畴

序列标注：深度学习框架



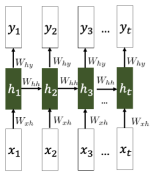
LSTM + CRF



训练

发射概率

- 发射(emission)概率e：时刻i处于状态h_i的条件下生成观测y_i的概率，在LSTM-CRF中，h_i是x_i一一对应的隐层状态，所以这个发射概率也可以理解成从序列x_i到标签y_i的概率。



$$e_i = p(y_i|x_i) = \frac{e^{score(y_i)}}{\sum_y e^{score(y)}}$$

词的标签分布概率

- 每个词x_i输入LSTM后，得到对应的发射概率e_i
- 整个序列$\mathbf{X}=\{x_1, x_2, x_3, \dots, x_n\}$输入，得到对应的发射矩阵E，其中

$$e_i = E[i] \in R^m$$

是一个m维的向量（y的种类数一共有m个）

- 其中第k类标签y_k的概率为：

$$e_i(y_k)$$

标签转移(transition)概率

- $\mathbf{Y}=\{y_1, y_2, y_3, \dots, y_n\}$是一个序列（sequence），彼此间有依赖关系
  - 动词后面一般不再出现动词，代词后面动词概率大
  - 两个命名实体一般不连续出现
  - 中文中时间一般出现在地点前面
  - 中文人名一般不超过4个字
  - .....
- **如果能构建这个转移概率?**

标签转移矩阵

- 我们将标签y_{i-1}转移到y_i的概率定义为标签转移概率T[i-1, i]
- 这样就构建了一个标签转移矩阵T。

	S	L1	L2	L3	E
S	0	0.7	0.8	0.2	0.3
L1	0	0.2	0.1	0.5	1.3
L2	0	0.4	1.4	1.5	0.1
L3	0	0.7	0.3	0.4	0.5
E	0	0	0	0	0

对于一个标签序列Y综合打分为：发射概率 + 转移概率（log前提下）

$$\text{score}(Y) = \sum_{i=1}^{n} e_i + \sum_{i=2}^{n} (T[i-1, i])$$

$$\log p(y \mid x) = \text{score}(y) - \log\left(\sum_{y} e^{\text{score}(y)}\right)$$

通过前向/后向算法计算配分函数

$$\log Z\left(y_t = i\right) = \log \sum_{y_t} e^{\text{score}(start \to y_t)}$$

$$\log Z\left(y_{t+1} = j\right) = \sum_{i} \left[\log Z\left(y_t = i\right) + \left(T[i,j] + E[t+1,j]\right)\right]$$

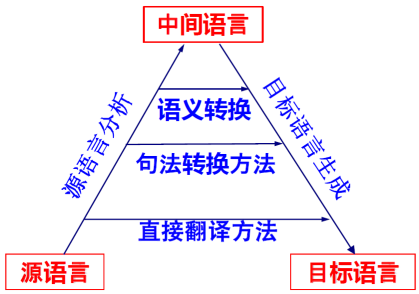
推理：使用Viterbi算法计算最可能的路径

Chap3.5 机器翻译

困难点

- 自然语言中普遍存在的歧义和未知现象
- 不同语言之间文化的差异

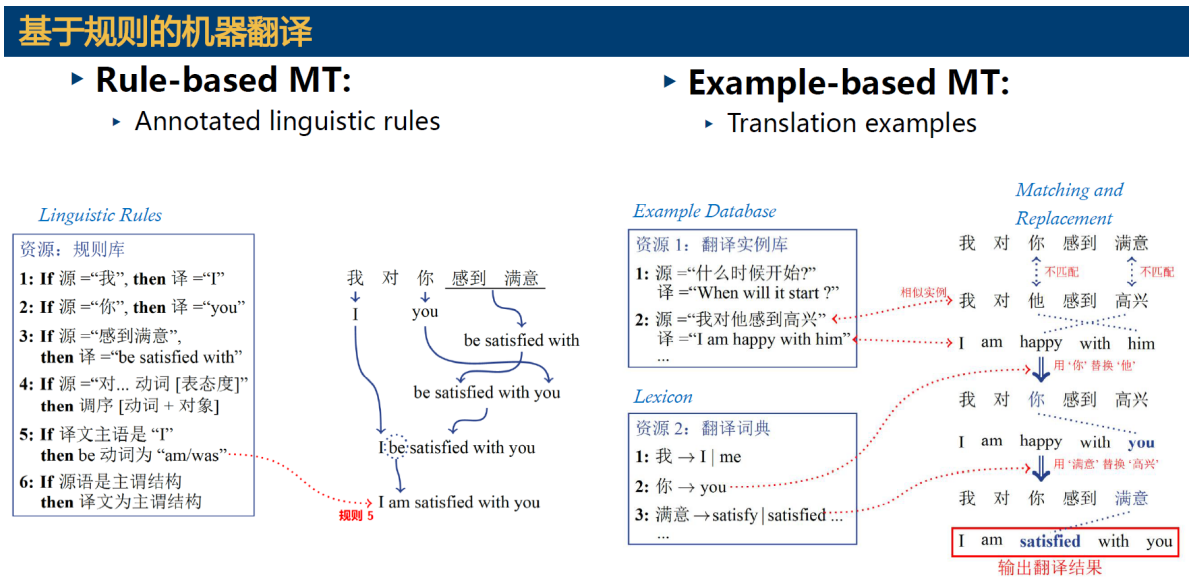
基本翻译方法



直接转换法

从源语言句子的表层出发，将单词、短语或句子直接置换成目标语言译文，必要时进行简单的词序调整。对原文句子的分析仅满足于特定译文生成的需要。这类翻译系统一般针对某一个特定的语言对，将分析与生成、语言数据、文法和规则与程序等都融合在一 起。

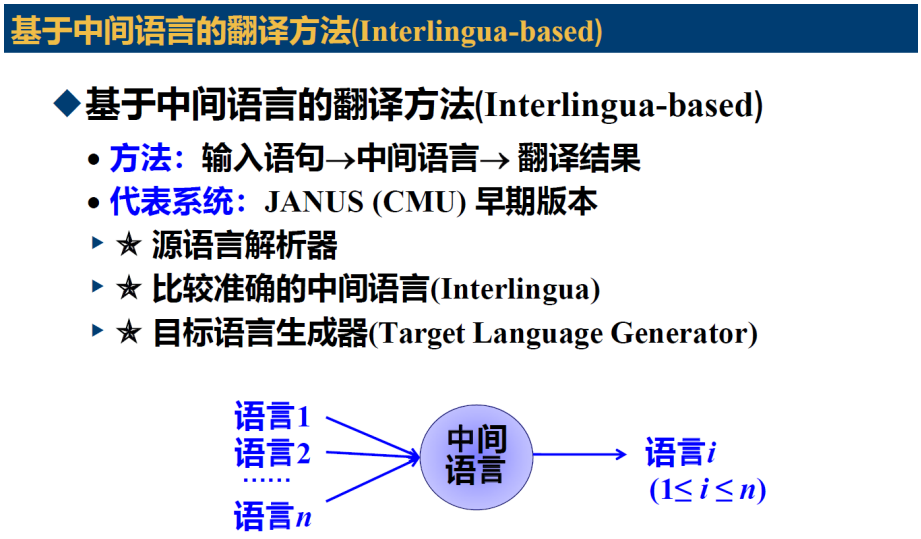
基于规则的翻译方法



- 优点：可以较好地保持原文的结构，产生的译文结构与源文的结构关系密切，尤其对于语言现象已知的或句法结构规范的源语言语句具有较强的处理能力和较好的翻译效果

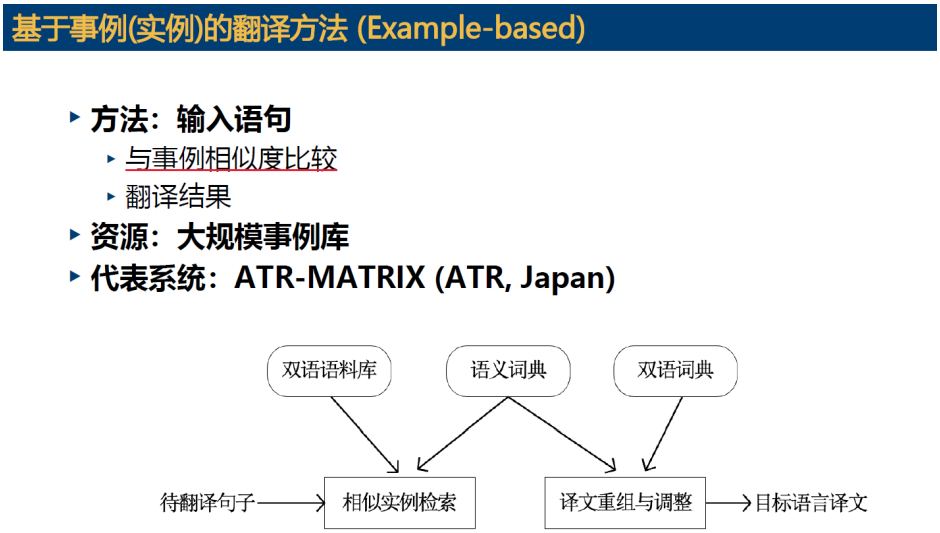
- 弱点：规则一般由人工编写，工作量大，主观性强，一致性难以保障，不利于系统扩充，对非规范语言现象缺乏相应的处理能力

基于中间语言的翻译方法



- 优点：中间语言的设计可以不考虑具体的翻译语言对，因此该方法尤其适合多语言之间的互译。
- 弱点：如何定义和设计中间语言的表达方式，以及如何维护并不是一件容易的事情，中间语言在语义表达的准确性、完整性等很多方面，都面临若干困难。

基于事例的翻译方法



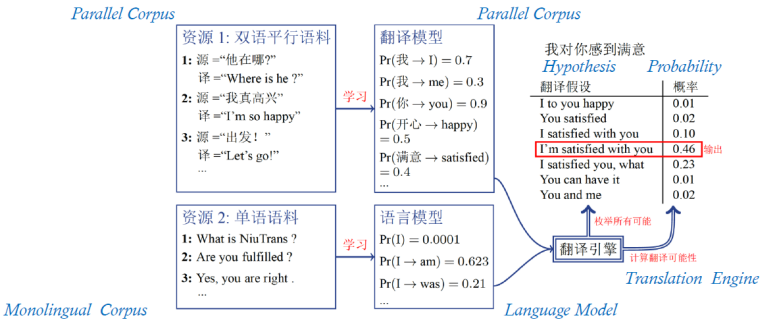
- 优点：不要求源语言句子必须符合语法规定，翻译机制一般不需要对源语言句子做深入分析。
- 弱点：两个不同的句子之间的相似性（包括结构相似性和语义相似性）往往难以把握，尤其在口语中，句子结构一般比较松散，成分冗余和成分省略都较严重。另外，系统往往难以处理事例库中没有记录的陌生的语言现象，而且当事例库达到一定规模时，其事例检索的效率较低。

统计机器翻译

噪声信道模型：一种语言T由于经过一个噪声信道而发生变形，从而在信道的另一端呈现为另一种语言S（信道意义上的输出，翻译意义上的源语言）。翻译问题实际上就是如何根据观察到的S，恢复最为可能的T问题。

基于统计的机器翻译

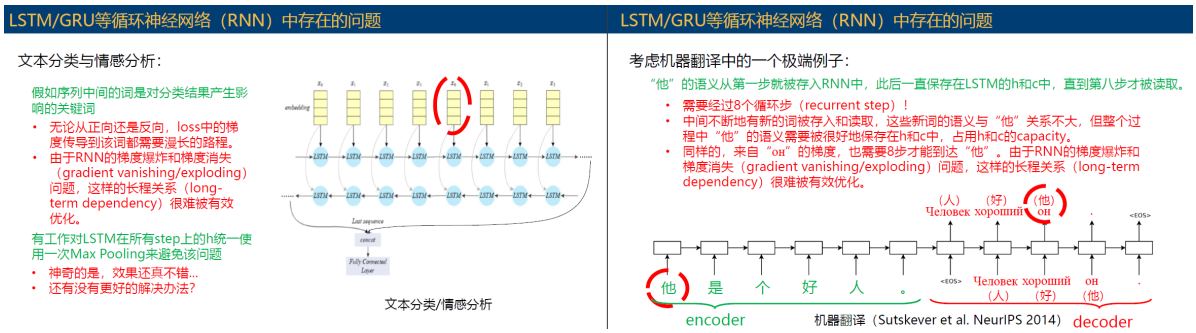
- **Statistical Machine Translation (SMT)**
  - Parallel corpus: sentence-level alignment.
  - Monolingual corpus: *n*-grams probability.
  - To learn the translation rules statistically.



BLEU score: Overlap with reference translations

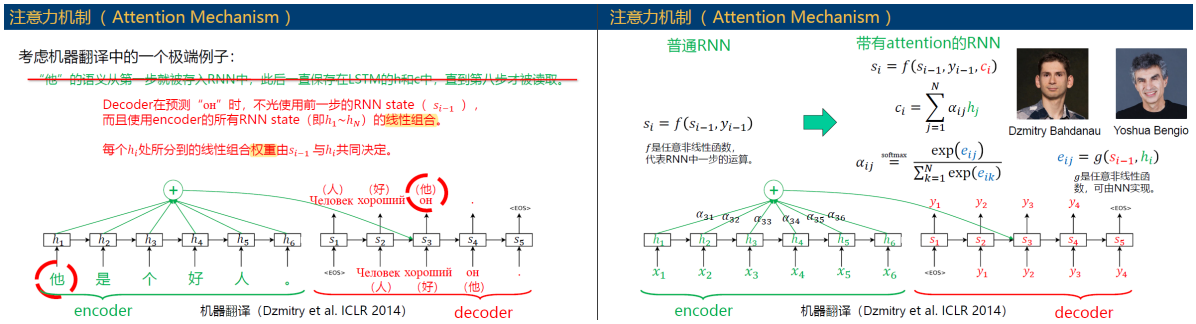
神经网络机器翻译

从LSTM/GRU到Attention Mechanism



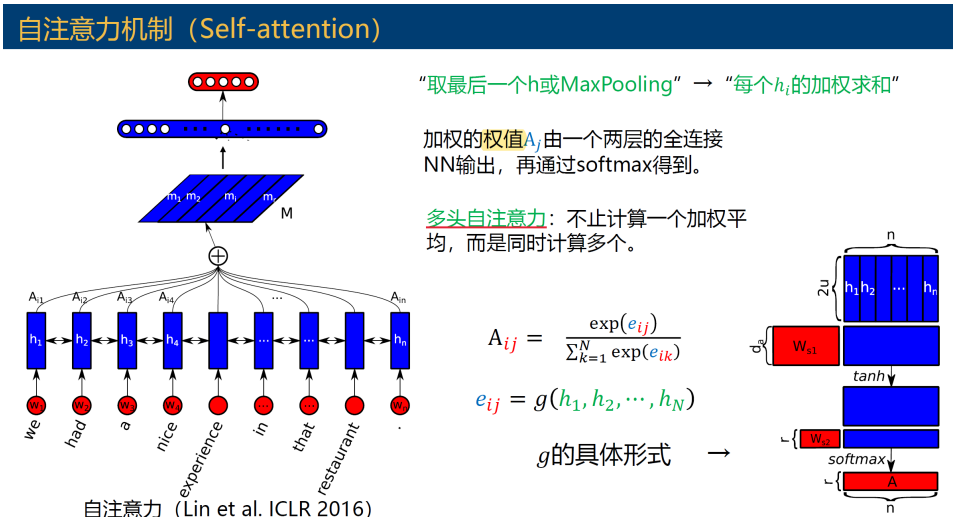
注意力机制

事实上，人在翻译长句子的某一个词的时候，更关心这个词对应的原文中相关的几个词，而并不关心别的上下文。



Self-Attention

Q：在没有decoder的时候，比如做文本分类、情感分析的时候，如何使用attention?



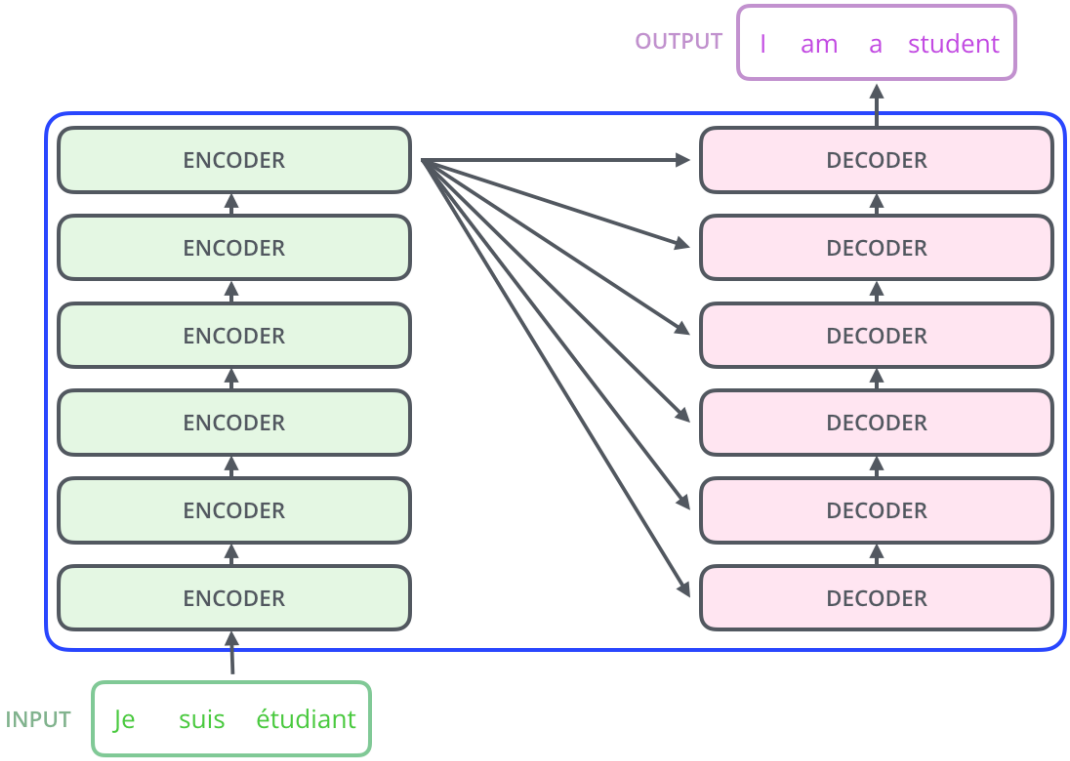
更进一步的，如果我们作以下三方面改变：

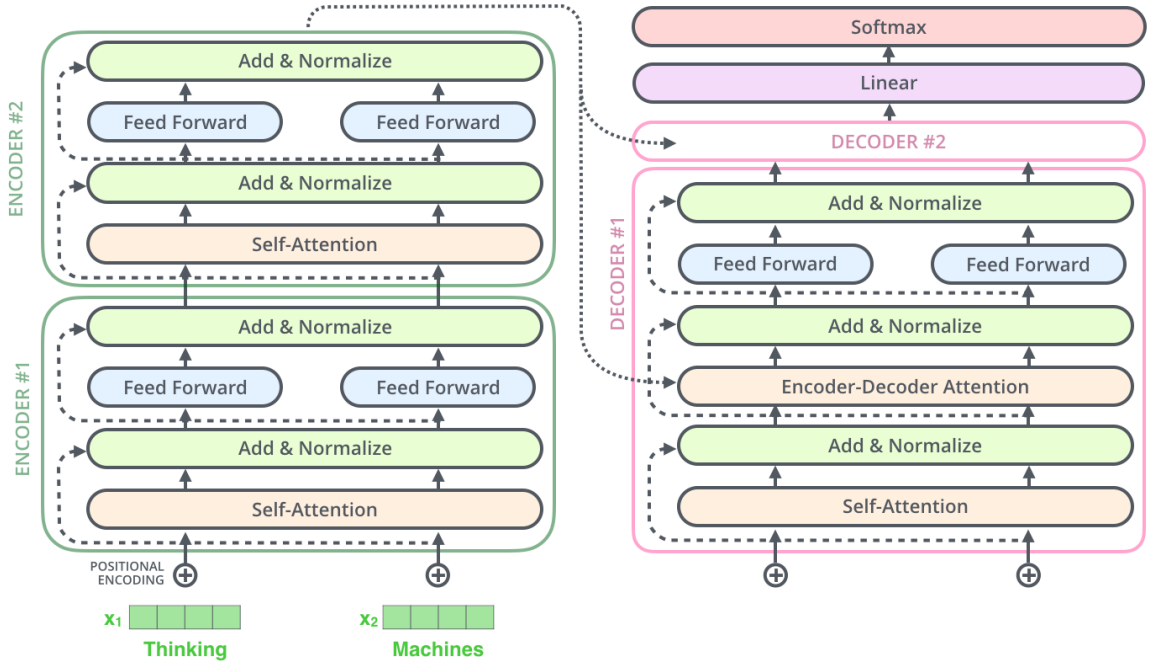
1. 干脆撤掉底下的RNN
2. 对每一个h计算一组多头自注意力机制所得到的向量集
3. 利用额外的positional embedding弥补撤掉RNN所引起的位置信息缺失

那么连RNN都可以不需要了，整个模型只使用了attention，我们就得到了Transformer

集大成者：Transformer模型

Tutorial: link

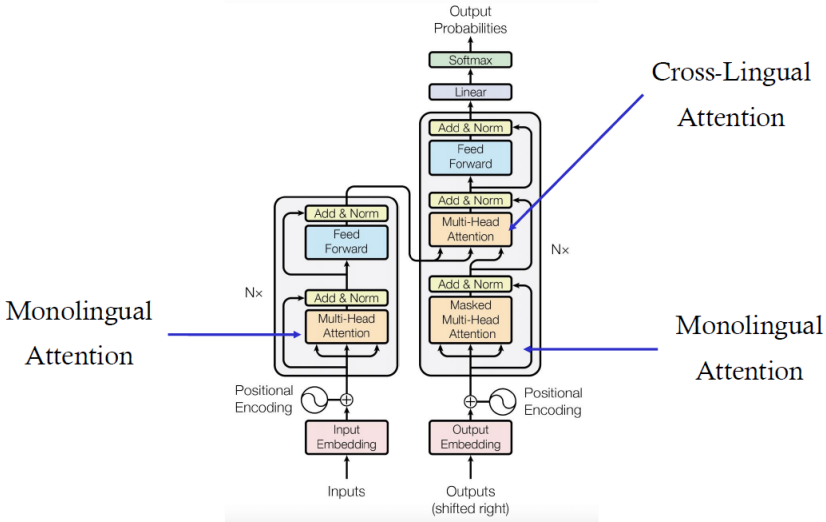




Self-Attention

Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \bullet k_1 = 112$	$q_1 \bullet k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12
Softmax X Value	v_1	v_2
Sum	z_1	z_2

There Are Two Attentions



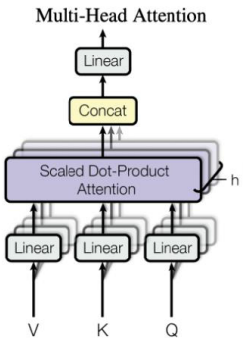
Multi-head

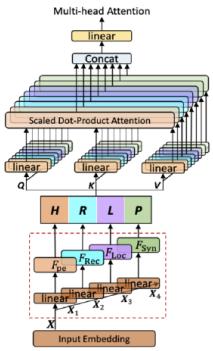
How to Enlarge the Diversity?

- Multi-head is to ensure the diversity for each representation vector

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

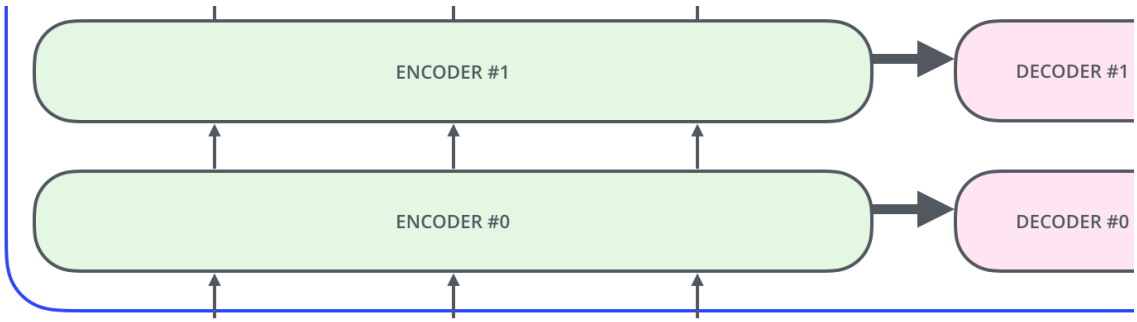
[Multi-head Structure]

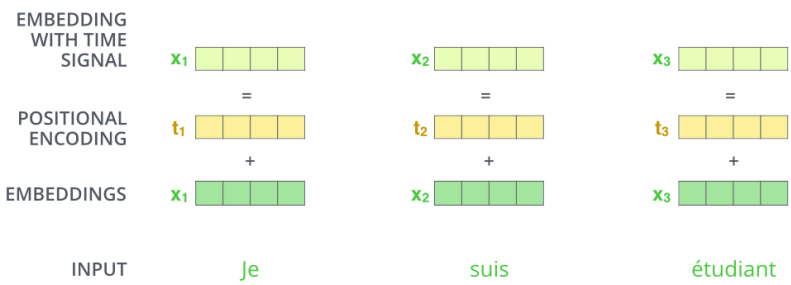




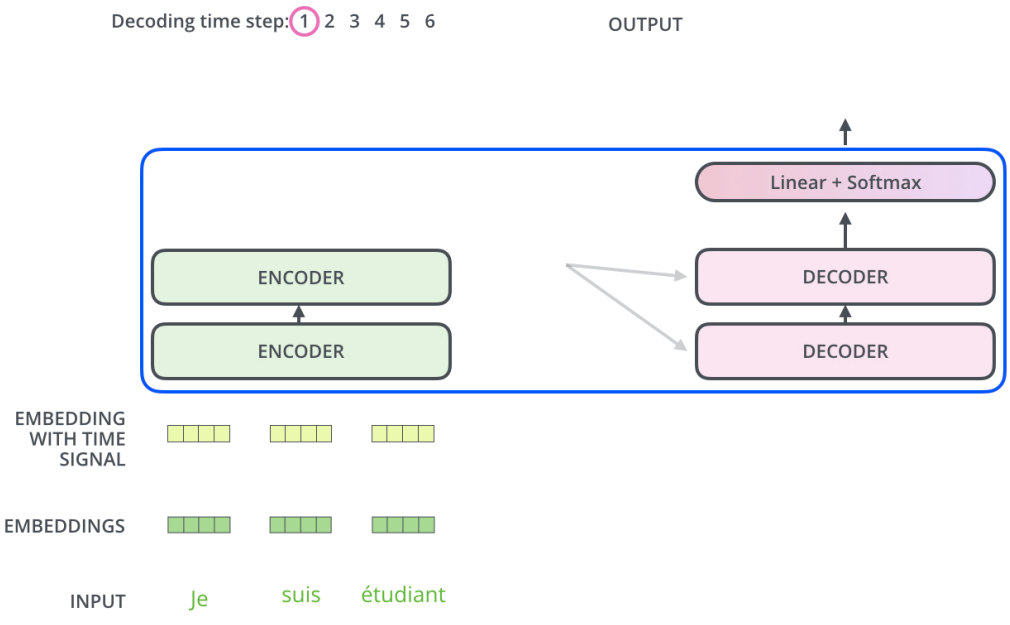
[Diverse Multi-head, Chen 2020]

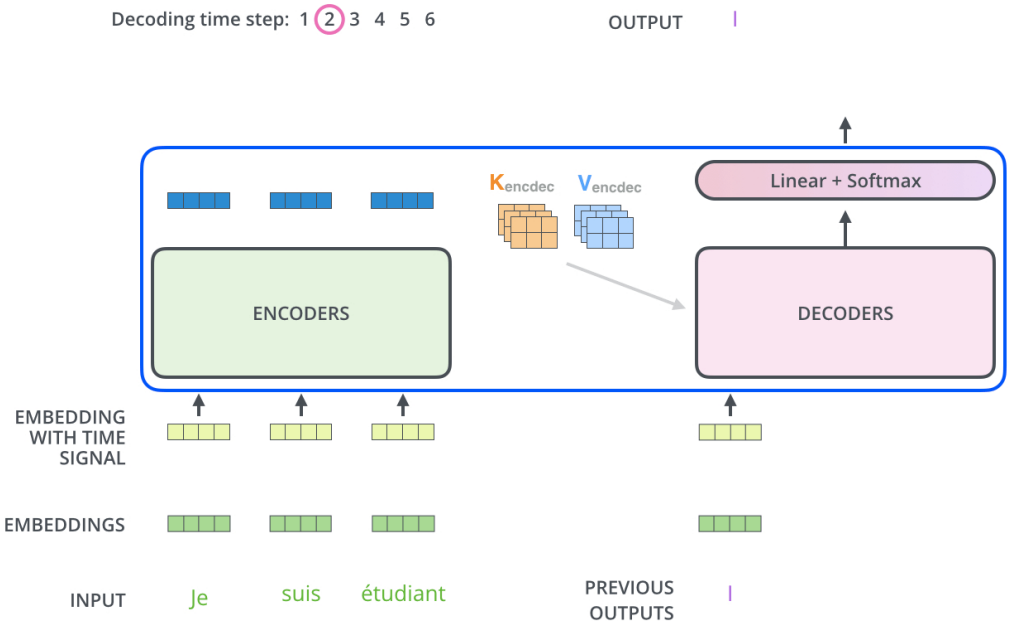
Position Encoding



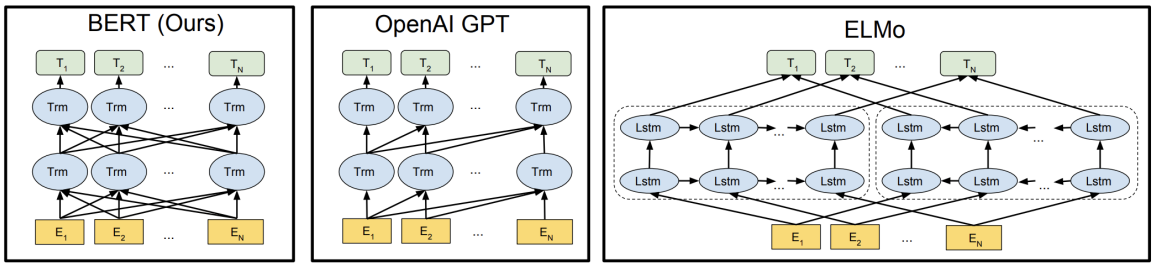


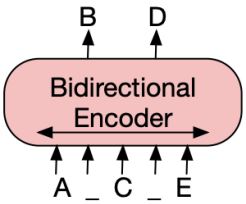
运行过程


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

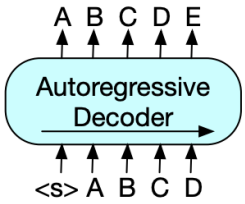

Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

预训练模型

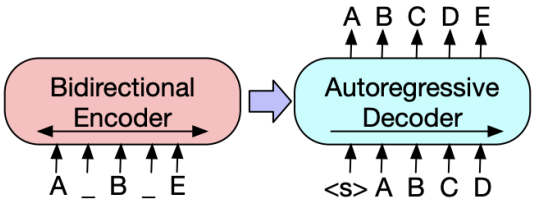

BERT (Ours)
OpenAI GPT
ELMo
Trm
Lstm



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

综合型任务及学术前沿

- [ ] 目前不在做NLP方向，暂时不打算review Chap4.1 智能问答&Chap4.2 对话系统

Chap4.3 预训练语言模型

词向量回顾

- One-Hot向量维度爆炸且不能反映词间关系
- 静态，上下文无关（e.g. 一行行行行）

  使用时需要引入复杂结构来建模上下文信息

  对于不同任务有时需要分别设计上下文建模模型

上下文相关词向量

- 预训练向量 ⇒ 预训练上下文建模模型
- 大规模数据预训练复杂模型 + 有限的领域特定数据微调简单模型

- 意义：
  - 极大的开发和揭示了大规模预训练的潜力
  - 极大的简化和统一了上下文信息建模的模型结构

上下文相关词向量早期发展

CoVe

上下文相关词向量的早期发展
Context Vector (CoVe)

- 第一次提出上下文相关词向量
- 受启发于CV中在ImageNet上预训练模型后在其他小数据集上使用

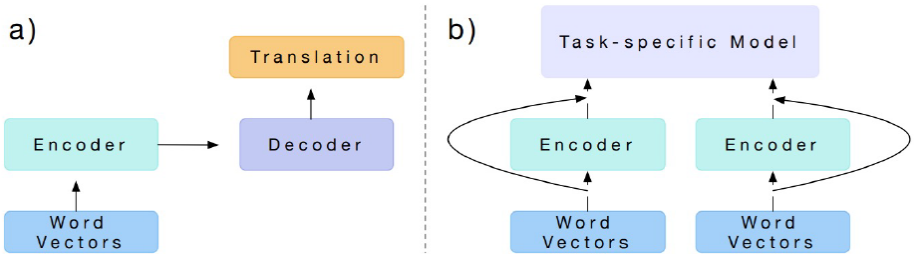


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

McCann, Bryan, et al. "Learned in Translation: Contextualized Word Vectors." *NIPS*. 2017.

Idea: 将机器翻译中encoder的输出作为词向量

问题：预训练过程使用的是有监督数据，需要大量额外人工标注

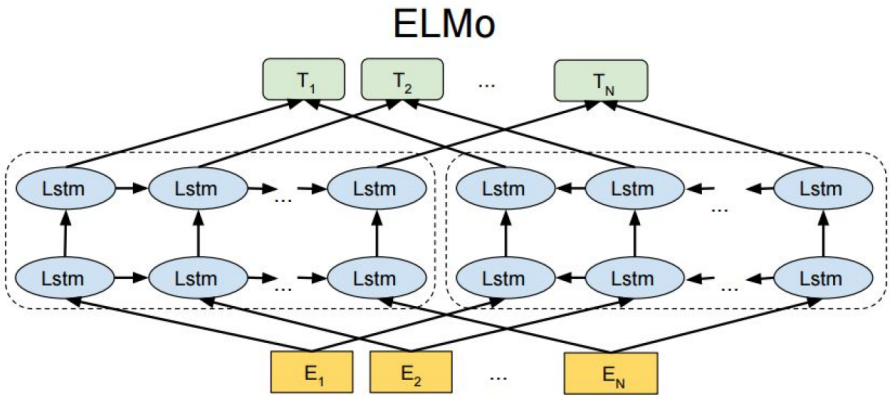
ELMo

神经网络语言模型的训练任务

- 序列到序列建模，整句输入网络，每个词元（Token）的位置预测下一个词元
- 可以自我监督，不需要额外的标注

上下文相关词向量的早期发展
Embeddings from Language Models (ELMo)

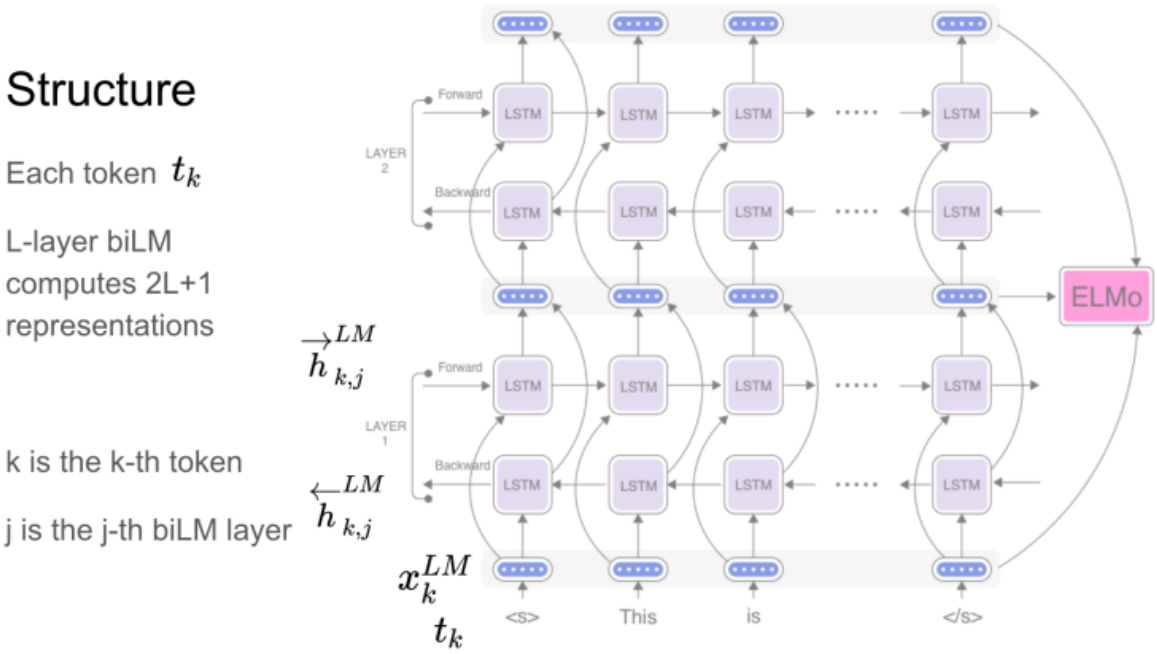
- 利用语言模型的自监督特性，不需要额外标注
- 使用双向语言模型，建模双向的依赖关系



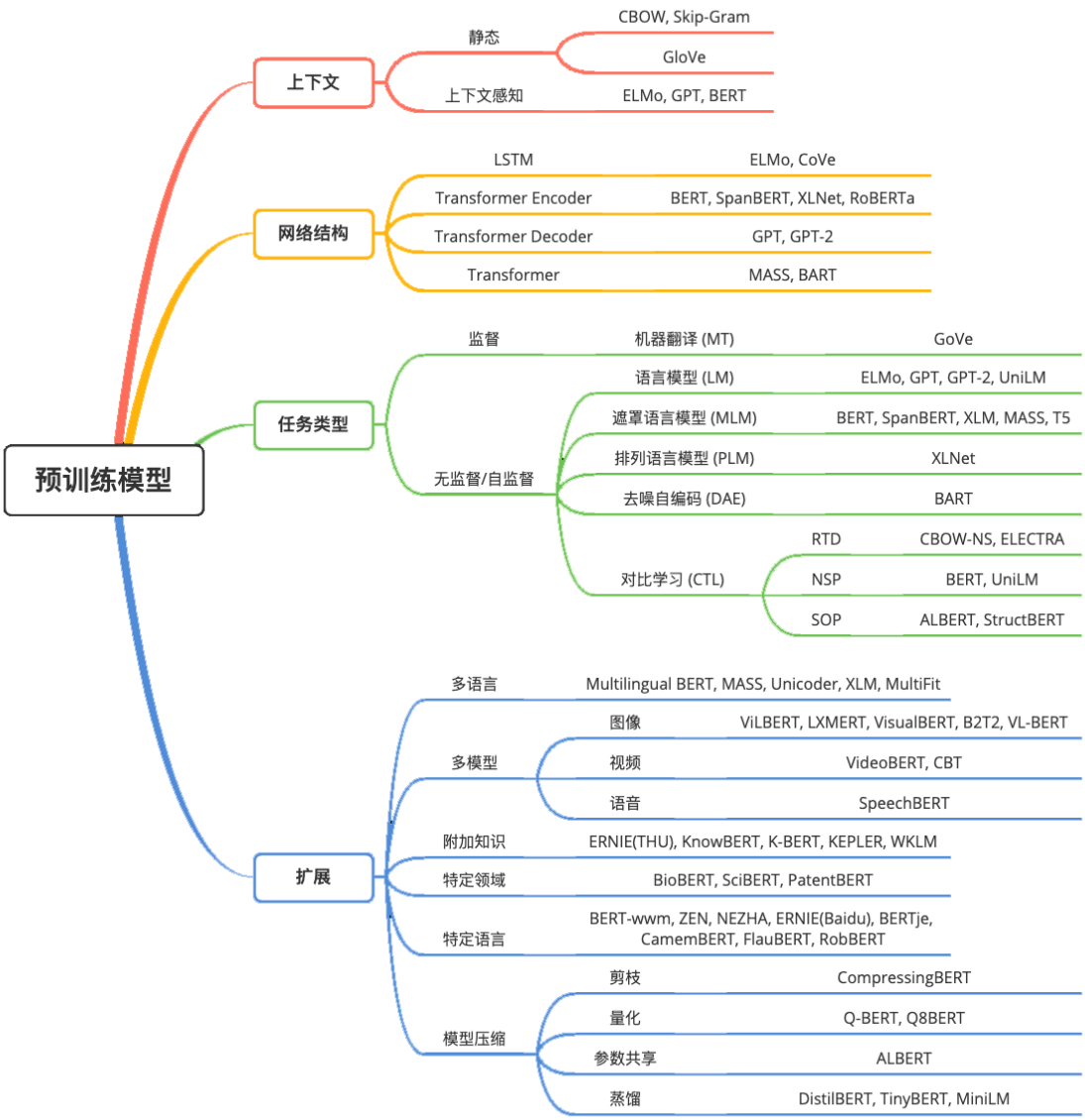
$$\sum_{k=1}^{N}\left(\log p\left(t_k \mid t_1,\ldots,t_{k-1};\Theta_x,\vec{\Theta}_{LSTM},\Theta_s\right)+\log p\left(t_k \mid t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s\right)\right)$$

- 实际上是对两个方向的语言模型取几何平均

- 加权平均所有中间层向量得到“动态词向量”



典型预训练语言模型及其分类

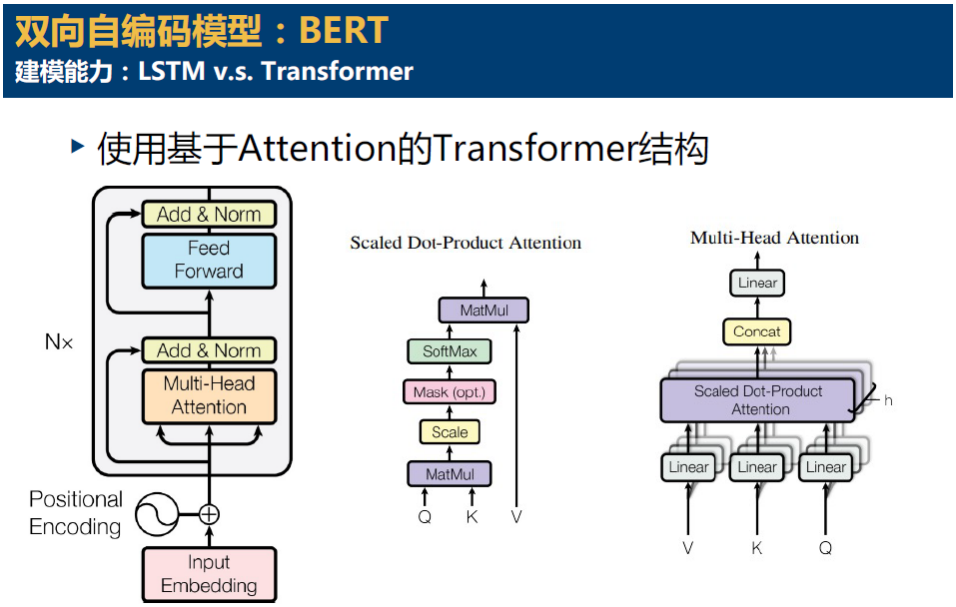


BERT

一个高效的预训练语言模型的要求：

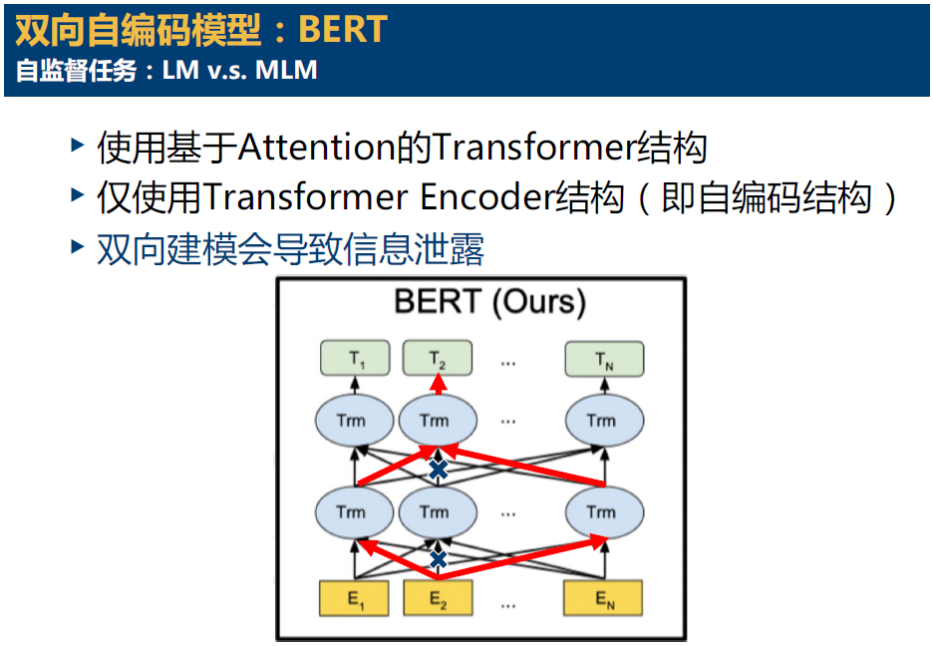
- 强大的建模能力

  ELMo中的Bi-LSTM的建模能力有限，会对历史信息进行压缩



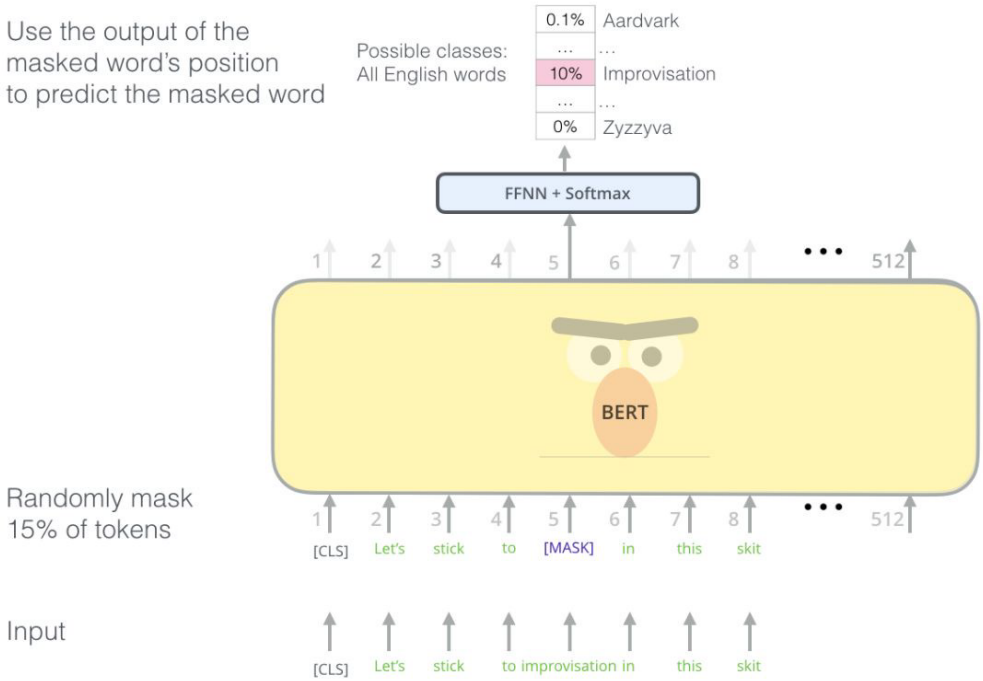
- 有效建模双向的上下文信息

  自回归模型（语言模型，单向性）难以有效建模双向上下文信息

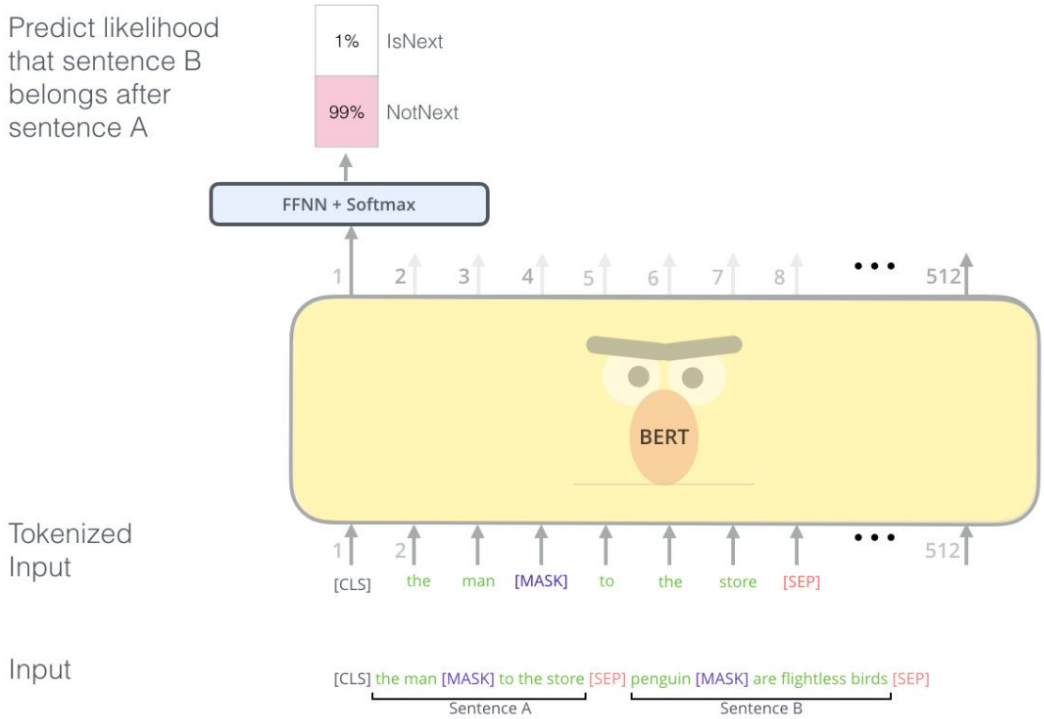


- 能够利用大规模无监督数据

  新的预训练任务：掩码语言模型（Masked Language Model, MLM）



另外，还引入了NSP（Next Sentence Prediction）任务来引入句间关系的建模



模型设计总结

- 使用Transformer Encoder的自编码模型进行建模
- 输入组成：
  - 词元嵌入+ 片段嵌入+ 位置嵌入
  - [CLS] & [SEP]
  - BPE（Token）

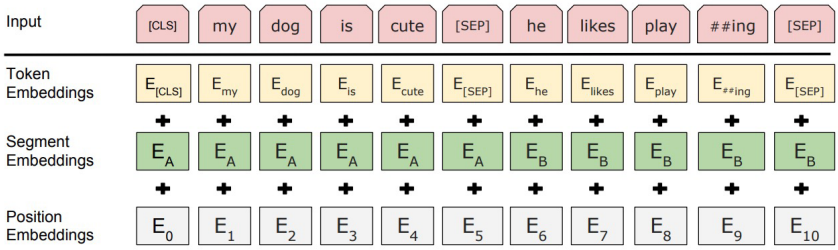
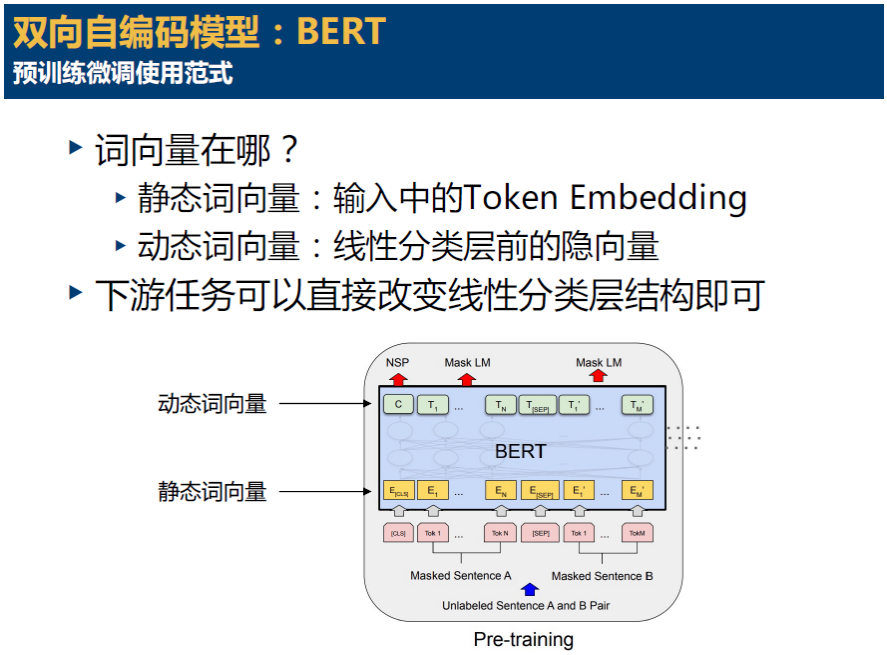


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- 预训练任务：
  - MLM：掩蔽15%的词元（10%保留、10%随机替换、80%[MASK]）
  - NSP：50%正例+50%负例

预训练微调使用范式



优缺点

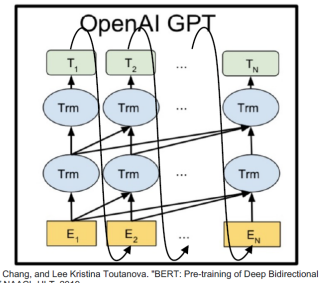
- 在自然语言理解任务上表现突出
- 难以解决自然语言生成问题
- [MASK]
  - 预训练和微调阶段的输入空间分布不匹配
  - 预训练过程收敛较慢，不能掩蔽过多词元
  - 假设被掩蔽词元是独立的，只能进行联合概率的<u>有偏估计</u>

$$p(\text{ New, York }) = p(\text{ New }) \times p(\text{ York } \mid \text{ New }) \neq p(\text{ New }) \times p(\text{ York })$$

GPT

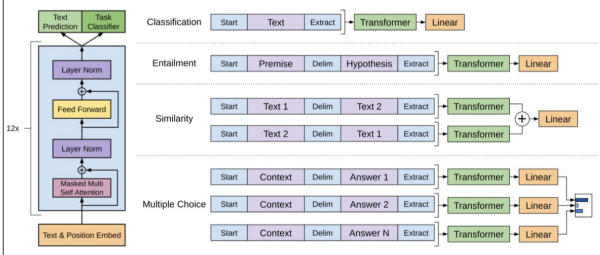
单向自回归模型：GPT
结构

- 直接使用语言模型任务进行预训练
- 采用类Transformer Decoder结构（即自回归结构）



Kenton, Jacob Devlin Ming-Wei Chang, and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Proceedings of NAACL-HLT. 2019.

单向自回归模型：GPT
下游任务使用方法



Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

优缺点

- 直接采用LM任务预训练，不存在[MASK]引入的偏差

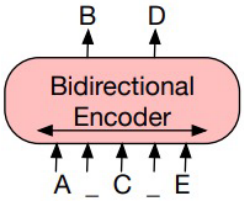
  在自然语言生成的任务上表现突出
- 由于LM任务的限制，只能建模上文信息，不能得到严格意义上的上下文相关词向量

  不适合自然语言理解相关任务

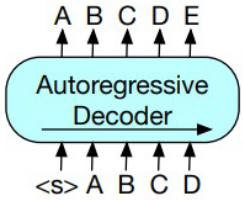
BART

- 采用完整的Transformer结构
- Decoder部分的Attention也包括Encoder的输出

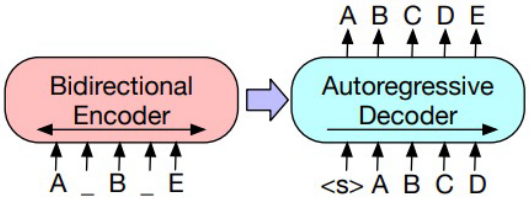
  采用MLM来解决Encoder端信息泄露问题



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

- 引入多种噪音来加强模型泛化性能

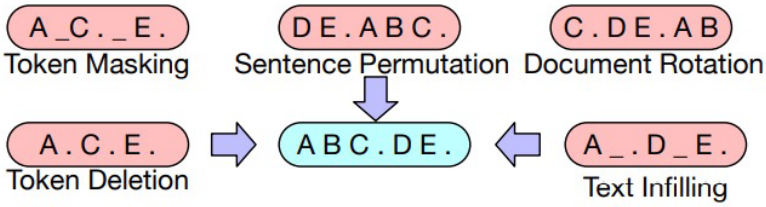
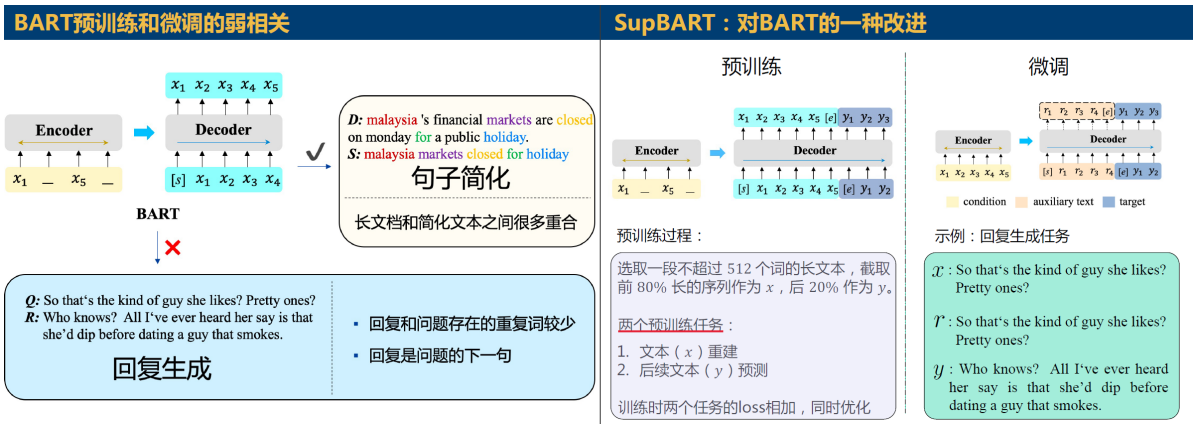


Figure 2: Transformations for noising the input that we experiment with. These transformations can be composed.

SupBART



XLNet: Permuted LM

- 自编码模型
  - 可以很好的建模双向上下文信息
  - 但[MASK]的引入：预训练-微调偏差；独立性假设

$$\max_{\theta} \log p_\theta(\overline{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta\left(x_t \mid \hat{\mathbf{x}}\right) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e\left(x_t\right)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e\left(x'\right)\right)}$$

- 自回归模型
  - 没有偏差和假设，严格按照联合概率公式展开
  - 但不能建模双向上下文信息

$$\max_{\theta} \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta\left(x_t \mid \mathbf{x}_{<t}\right) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e\left(x_t\right)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e\left(x'\right)\right)}$$

Q: 怎样同时利用自编码和自回归的优点且克服两者的缺点?

- 克服自编码模型的缺点：不能引入特殊标签，需要使用LM模式
- 克服自回归模型的缺点：将双向上下文均作为预测内容的条件

- 解决方法：Permutated LM
- 目标函数：

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T}\left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right].$$

  - 其中：
  - $\mathcal{Z}_T$指长度为T的序列的所有可能排列方式
  - z_t指排列$z \in \mathcal{Z}_T$中的第t个元素
  - $\boldsymbol{z}_{<t}$指排列$z \in \mathcal{Z}_T$中前$t-1$个元素

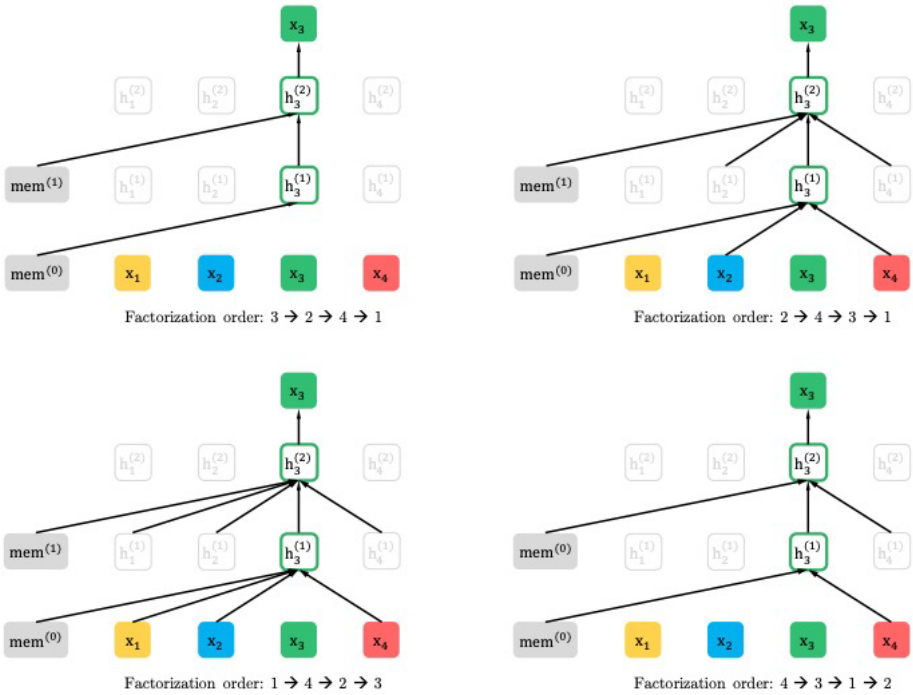


Figure 4: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence $\mathbf{x}$ but with different factorization orders.

Problem: assume we parameterize the next-token distribution $p_\theta\left(X_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}\right)$ using the standard Softmax formulation, i.e., $p_\theta\left(X_{z_t}=x \mid \mathbf{x}_{\mathbf{z}_{<t}}\right)=\frac{\exp\left(e(x)^\top h_\theta\left(\mathbf{x}_{\mathbf{z}}<t\right)\right)}{\sum_{x'} \exp\left(e\left(x'\right)^\top h_\theta\left(\mathbf{x}_{\mathbf{z}_{<t}}\right)\right)}$, where $h_\theta\left(\mathbf{x}_{\mathbf{z}_{<t}}\right)$ denotes the hidden representation of $\mathbf{x}_{\mathbf{z}_{<t}}$ produced by the shared Transformer network after proper masking. Now notice that the representation $h_\theta\left(\mathbf{x}_{\mathbf{z}_{<t}}\right)$ does not depend on which position it will predict, i.e., the value of z_t.

To avoid this problem, we propose to re-parameterize the next-token distribution to be target position aware:

$$p_\theta\left(X_{z_t}=x \mid \mathbf{x}_{z_{<t}}\right)=\frac{\exp\left(e(x)^\top g_\theta\left(\mathbf{x}_{\mathbf{z}_{<t}}, z_t\right)\right)}{\sum_{x'} \exp\left(e\left(x'\right)^\top g_\theta\left(\mathbf{x}_{\mathbf{z}_{<t}}, z_t\right)\right)}$$

引入Two-Stream Self-Attention

- 内容流：$h_{z_t}=h_\theta\left(\boldsymbol{x}_{\boldsymbol{z}_{\leq t}}\right)$

  $$h_{z_t}^{(m)} \leftarrow \text{ Attention }\left(\mathrm{Q}=h_{z_t}^{(m-1)}, \mathrm{KV}=\mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)} ; \theta\right), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t})$$

- 查询流：$g_{z_t}=g_\theta\left(\boldsymbol{x}_{\boldsymbol{z}_{<t}}, z_t\right)$

  $$g_{z_t}^{(m)} \leftarrow \text{Attention}\left(\mathrm{Q}=g_{z_t}^{(m-1)}, \mathrm{KV}=\mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)} ; \theta\right), \quad \left(\text{query stream: use } z_t \text{ but cannot see } x_{z_t}\right)$$

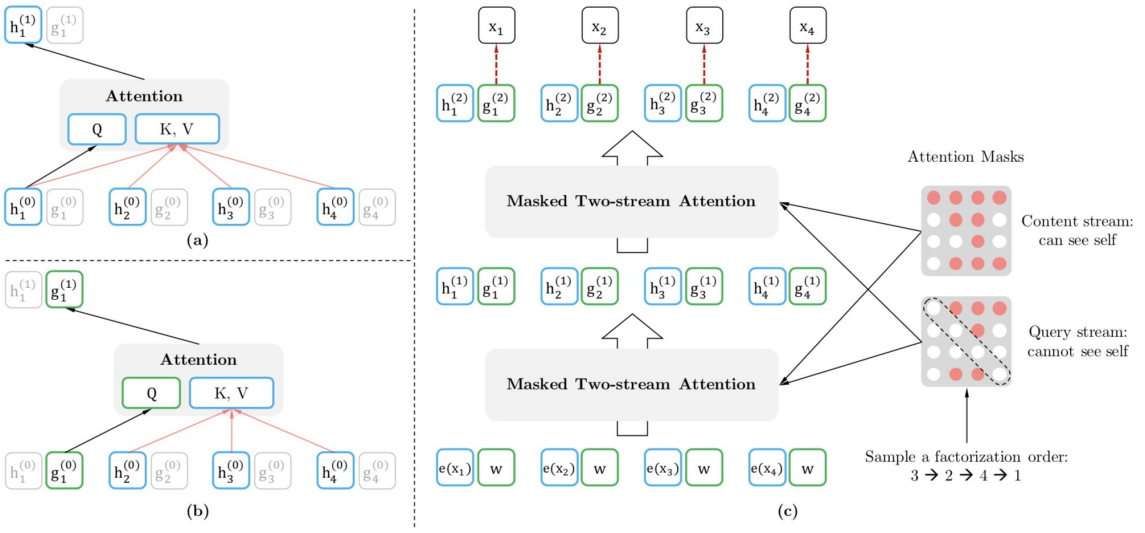


Figure 2: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t}. (c): Overview of the permutation language modeling training with two-stream attention.

优缺点

- 成功的结合了自编码和自回归两种模型的优点并避免了两者的缺点

  在很多任务上超越了GPT、BERT等模型

- 需要遍历所有可能排列，计算量巨大

  采样部分排列进行训练

- 收敛速度缓慢

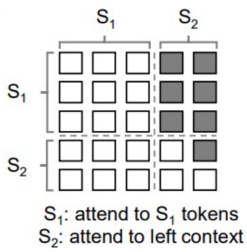
  引入超参数K，只考虑每个排列的最后1/K的单词作为优化目标（固定历史）

UniLM: Prefix LM

Q：是否有一个统一的(Unified)模型能够同时适应两种任务？

- 从实现角度考虑：两种模型的区别仅在掩码矩阵

- 从实现角度考虑：两种模型的区别仅在掩码矩阵
- 进一步观察：编解码模型也可以统一到这个框架中



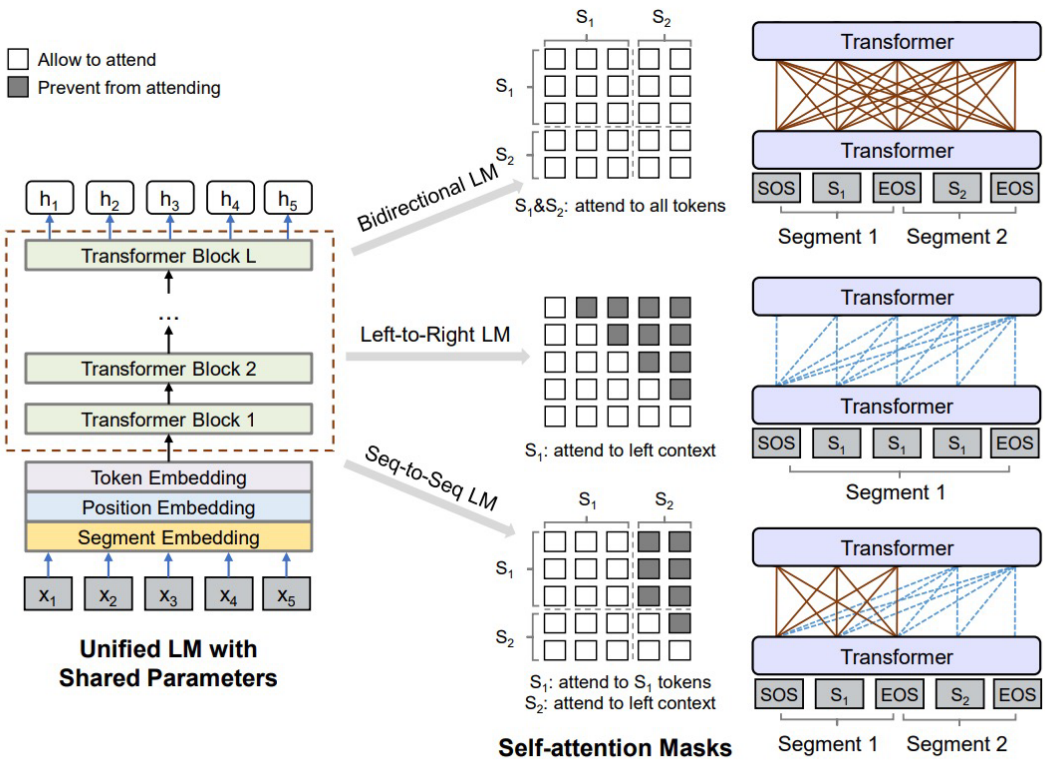
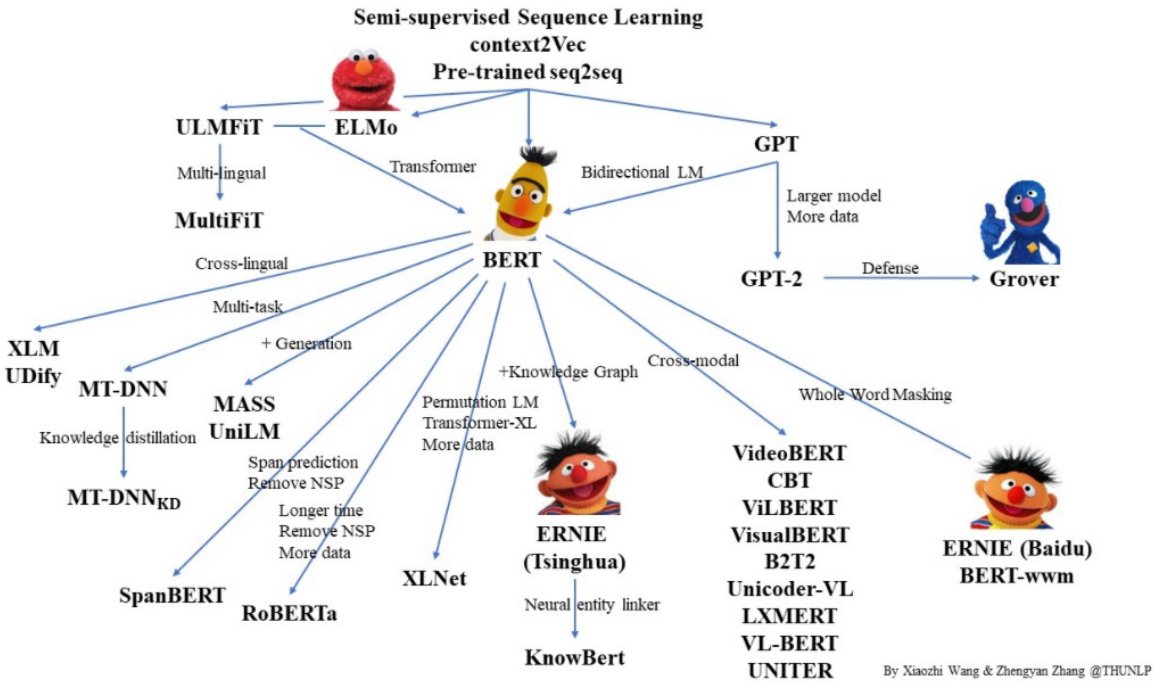


Figure 1: Overview of unified LM pre-training. The model parameters are shared across the LM objectives (i.e., bidirectional LM, unidirectional LM, and sequence-to-sequence LM). We use different self-attention masks to control the access to context for each word token. The right-to-left LM is similar to the left-to-right one, which is omitted in the figure for brevity.

- 使用BERT-large参数初始化
- 使用四个任务继续预训练
  - 单向LM：结合了MLM和LM两个任务，使用[MASK]词元来掩蔽目标词元，其余和语言模型相同
  - 双向LM：与BERT中MLM任务相同
  - 序列到序列LM：对于两句话的输入，前一句话使用双向LM任务，后一句话使用单向LM任务
  - NSP：和BERT中NSP任务相同

典型预训练语言模型总结



预训练语言模型的发展和前景

模型自身结构的改进

模型的压缩与简化

- ALBERT
  - 嵌入分解：将嵌入层分解为两个更小的矩阵的乘积
  - 参数共享：将所有层分成多个组，每组的参数完全共享

  极大的减少了参数量，但是计算量同时增加
- 知识蒸馏(+剪枝)：DistillBERT，TinyBERT等
- 量化：Q-BERT，Q8BERT等

Table 3: Comparison of Compressed PTMs

Method	Type	#Layer	Loss Function*	Speed Up	Params	Source PTM	GLUE‡
$\text{BERT}_{\text{BASE}}$ [36]	Baseline	12	$\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}}$		110M		79.6
$\text{BERT}_{\text{LARGE}}$ [36]		24	$\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}}$		340M		81.9
Q-BERT [156]	Quantization	12	HAWQ + GWQ	-		$\text{BERT}_{\text{BASE}}$	≈ 99% BERT°
Q8BERT [211]		12	DQ + QAT	-		$\text{BERT}_{\text{BASE}}$	≈ 99% BERT
ALBERT§ [93]	Param. Sharing	12	$\mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{SOP}}$	×5.6 ~ 0.3	12 ~ 235M		89.4 (ensemble)
DistilBERT [152]	Distillation	6	$\mathcal{L}_{\text{KD-CE}}+\text{Cos}_{\text{KD}}+\mathcal{L}_{\text{MLM}}$	×1.63	66M	$\text{BERT}_{\text{BASE}}$	77.0 (dev)
TinyBERT§ † [75]		4	$\text{MSE}_{\text{embed}}+\text{MSE}_{\text{attn}}+\text{MSE}_{\text{hidn}}+\mathcal{L}_{\text{KD-CE}}$	×9.4	14.5M	$\text{BERT}_{\text{BASE}}$	76.5
BERT-PKD [169]		3 ~ 6	$\mathcal{L}_{\text{KD-CE}}+\text{PT}_{\text{KD}}+\mathcal{L}_{\text{Task}}$	×3.73 ~ 1.64	45.7 ~ 67 M	$\text{BERT}_{\text{BASE}}$	76.0 ~ 80.6♯
PD [183]		6	$\mathcal{L}_{\text{KD-CE}}+\mathcal{L}_{\text{Task}}+\mathcal{L}_{\text{MLM}}$	×2.0	67.5M	$\text{BERT}_{\text{BASE}}$	81.2♯
MobileBERT§[172]		24	FMT+AT+PKT+ $\mathcal{L}_{\text{KD-CE}}+\mathcal{L}_{\text{MLM}}$	×4.0	25.3M	$\text{BERT}_{\text{LARGE}}$	79.7
MiniLM [194]		6	AT+AR	×1.99	66M	$\text{BERT}_{\text{BASE}}$	81.0♭
DualTrain§ †[216]		12	Dual Projection+$\mathcal{L}_{\text{MLM}}$	-	1.8 ~ 19.2M	$\text{BERT}_{\text{BASE}}$	75.8 ~ 81.9♮
BERT-of-Theseus [203]	Module Replacing	6	$\mathcal{L}_{\text{Task}}$	×1.94	66M	$\text{BERT}_{\text{BASE}}$	78.6

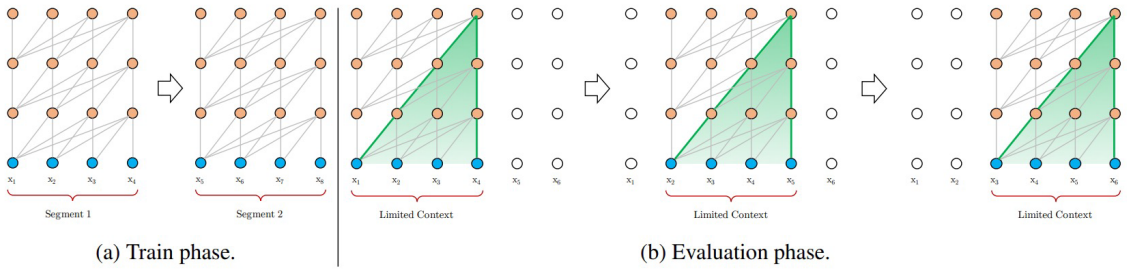
[1] The desing of this table is borrowed from [203, 150].
‡ The averaged score on 8 tasks (without WNLI) of GLUE benchmark (see Section 7.1). Here MNLI-m and MNLI-mm are regarded as two different tasks. 'dev' indicates the result is on dev set. 'ensemble' indicates the result is from the ensemble model.
* '$\mathcal{L}_{\text{MLM}}$ ', '$\mathcal{L}_{\text{NSP}}$', and '$\mathcal{L}_{\text{SOP}}$' indicate pre-training objective (see Section 3.1 and Table 1).'$\mathcal{L}_{\text{Task}}$' means task-specific loss.
'HAWQ', 'GWQ', 'DQ', and 'QAT' indicate Hessian AWare Quantization, Group-wise Quantization, Quantization-Aware Training, and Dynamically Quantized, respectively. 'KD' means knowledge distillation. 'FMT', 'AT', and 'PKT' mean Feature Map Transfer, Attention Transfer, and Progressive Knowledge Transfer, respectively. 'AR' means Self-Attention value relation.
§ The dimensionality of the hidden or embedding layers is reduced.
† Use a smaller vocabulary.
♭ Generally, the F1 score is usually used as the main metric of the QQP task. But MiniLM reports the accuracy, which is incomparable to other works.
° Result on MNLI and SST-2 only.
♯ Result on the other tasks except for STS-B and CoLA.
♮ Result on MRPC, MNLI, and SST-2 only.

知乎 @蘑菇先生

处理长文本问题

- 由于Transformer对于顺序的不敏感，需要特别通过Position Embedding来引入位置信息
- 使用时输入长度受限于预训练文本长度（一般为512）

<u>滑动窗口</u>



(a) Train phase. (b) Evaluation phase.

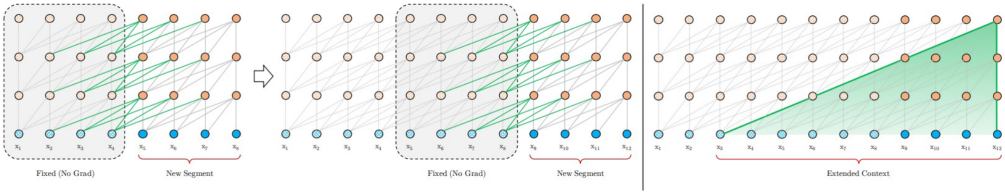
Figure 1: Illustration of the vanilla model with a segment length 4.

- 类似于固定窗口语言模型，限制了最大的依赖长度
- 有大量的重复计算，浪费计算资源

预训练语言模型的发展

模型自身结构的改进：处理长文本问题

- Transformer-XL
  - Segment-level Recurrent
    - 将上一个Segment计算时的隐状态缓存下来
    - 计算后一个Segment时使用缓存隐状态来增大上下文范围
    - 缓存隐状态不参与梯度反传
    - 问题：能建模的上下文范围扩大了多少？



优化预训练任务

- ALBERT：NSP -> SOP

  句子顺序预测（Sentence Order Prediction）

  预测两句话的顺序，排除原来可能有的主题等其他因素

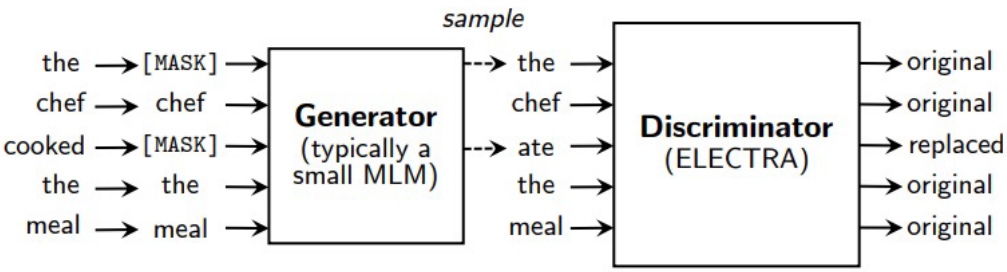
- Electra：+RTD

  替换词元检测（Replaced Token Detection）

RTD

- 掩蔽过多词元——破坏上下文信息
- 掩蔽过少词元——训练缓慢，难以收敛

受启发于GAN的思想——用一个鉴别器来检测每个词元是否是原句中的词元



理解生成器的作用：

- 对随机掩蔽的词元进行筛选，提高任务的难度
- 实验发现使用较小的生成器效果更好

  如果最终任务过难可能导致Electra不能进行有效学习

优势：

- 极大的提高了训练的效率

  每个位置均用于计算梯度

- 避免了因为[MASK]带来的预训练-微调偏差

预训练语料来源及其组成

多语言

- 多语言理解

  使用统一的模型对多种语言建模

  - mBERT：简单的使用多种语言语料进行预训练
  - XLM：在mBERT基础上，引入了机器翻译这个监督任务辅助
  - Unicoder：在XLM基础上，引入不同粒度的对齐任务（跨语言词语恢复，跨语言句义匹配判别，跨语言MLM），仍需要监督数据
- 多语言生成

  多语言的翻译和对齐

  - MASS：模型结构类似BART，使用多种语言的单语言语料进行训练
  - XNLG：2×2，两阶段预训练（encoder-MLM，decoder-DAE），单语言和跨语言两种语料

多模态

- 图像+文本：按照两个模态的交互位置分类
  - 双流（分别编码后交互再编码）

    ViBERT(MMM, MMA), LXMBERT(MLM, RoIFR, DLC, CMM, IQA)等
  - 单流（直接统一进行编码）

    VisualBERT(MLM, SIP), VL-BERT(MLM, MRoIC)等
- 视频+文本

  VideoBERT(MLM, MFM), UniVL(VTJ, CMLM, CMFM, VTA, LR)等
- 音频+文本

  SpeechBERT(After Initial Phonetic-Semantic Joint Embedding, MLM & MAL)等

领域特定与知识增强

领域特定语料预训练

- 面向专业领域

  BioBERT, SciBERT, ClinicalBERT等
- 面向特定下游任务：
  - 情绪分析：Label Aware MLM
  - 文本总结：Gap Sentence Generation
  - 不流畅语流检测：Noisy Words Detection

知识嵌入增强

- （词元级别）语言学知识

  例如词性标签、情感极性等；SentiLR、SenceBERT等
- （实体级别）知识图谱得到的实体嵌入

  例如将实体嵌入与文本中相关实体的词向量进行融合；ERNIE (THU)、KnowBERT、KEPLER等
- （句子级别）知识图谱中的关系三元组

  例如利用三元组拓展句子成树后进行编码；K-BERT等

微调方法

提升在特定下游任务上的性能

- 两阶段微调（Two-Stage）

  先在其他相同领域的数据上预微调
- 多任务学习（Multi Task）

  多个下游任务同时学习
- 解冻方法（Unfreezing）

  由少到多的微调参数，例如逐层解冻

In-Context Learning

统一所有的自然语言处理任务

GPT-2的核心思想

- 使用无监督的预训练模型不微调做下游任务
- 更多的无监督数据、更少的有监督数据

语言模型本质上是在建模$P\left(w_{k+1}, w_{k+2}, \cdots, w_N \mid w_1, w_2, \cdots, w_k\right)$，而有监督任务实际上实在建模$P\left(\text{Output} \mid \text{Input}\right)$

考虑到每个具体任务会使用不同的模型，可以将任务也作为条件

$$P(\text{ Output } \mid \text{ Input, Task })$$

因此有监督任务实际上是无监督语言模型的一种特例。只要模型容量足够大，可以直接通过无监督训练学习到有监督任务

- 例如在GPT-2的训练集中存在如下数据

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool].**

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose,**" which translates as, "**Lie lie and something will always remain.**"

Input → "Mentez mentez, il en restera toujours quelque chose,"

Output → "Lie lie and something will always remain."

Task → which translates as,

"I hate the word '**perfume,**"' Burr says. 'It's somewhat better in French: '**parfum.**'

If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

"**Brevet Sans Garantie Du Gouvernement**", translated to English: "**Patented without government warranty**".

- In-Context Learning
  - Zero-Shot场景

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:      ← task description
cheese =>                         ← prompt
```

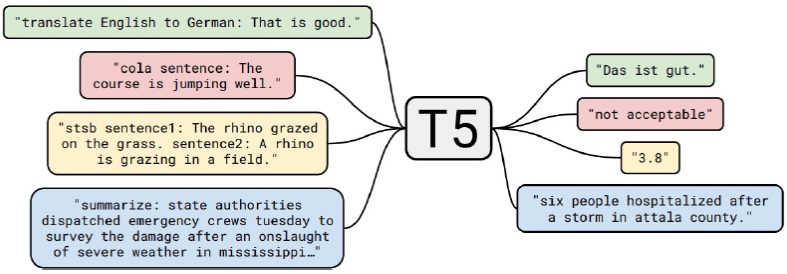
- In-Context Learning
  - Zero-Shot场景
  - One-Shot场景

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:      ← task description
sea otter => loutre de mer        ← example
cheese =>                         ← prompt
```

- 类似的思想：T5



- 进一步的发展：带提示的微调（Prompt）