深度学习-自然语言处理

史春奇

2017 年

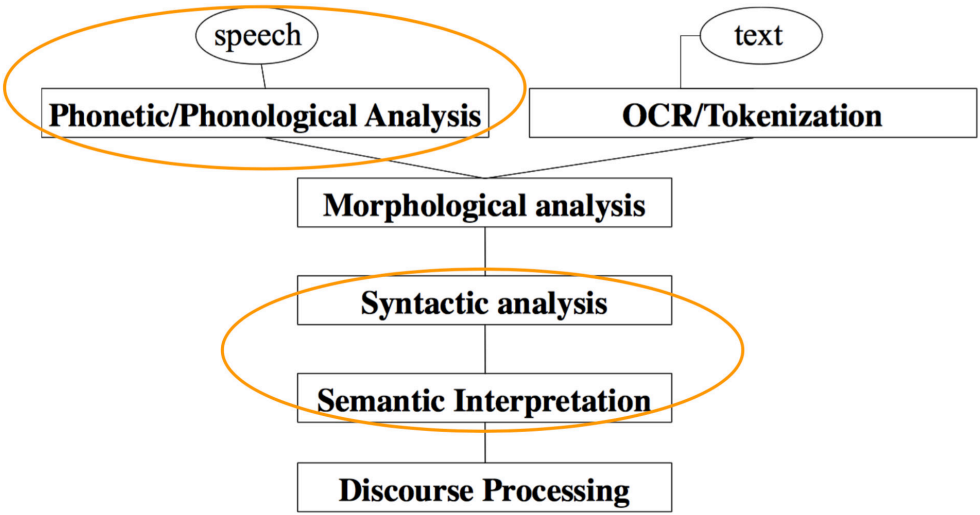
目录

第三部分-自然语言处理

- 深度自然语言处理的由来
  - 谁发明了神经网络语言模型？
  - 谁最早应用 RNN 到语言模型？
  - 词嵌入是深度模型么？
- 词嵌入的理解
  - 同样是无监督，词嵌入为什么优于主题模型？
  - 词类比是如何可以做到的？
  - 上下文和上文有什么区别？
- 神经网络机器翻译
  - 编码器和解码器为什么能做机器翻译？
  - 为什么要使用双向 RNN？
  - 什么是软（硬）注意力机制？

1 自然语言处理简介

- NLP 范围

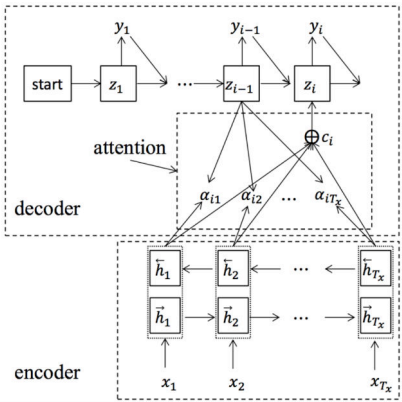


- NLP 范围

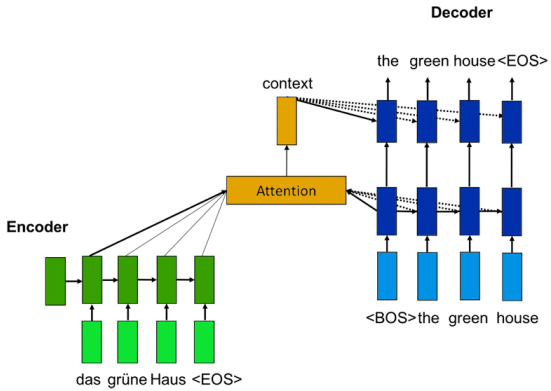
1. 情感分析 Sentiment Analysis
2. 问答 Question Answering
3. 机器翻译 Machine Translation
    - 最有代表性的自然语言处理

1.1 Seq2Seq 模型

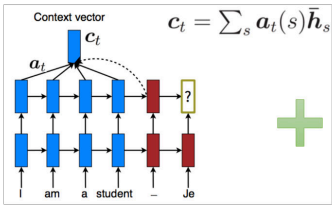
- Seq2Seq 机器翻译模型

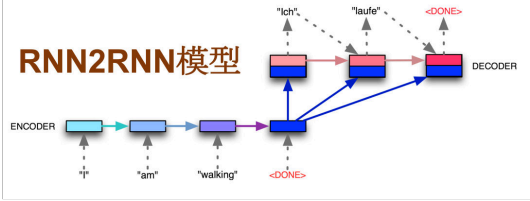


- Seq2Seq 机器翻译模型

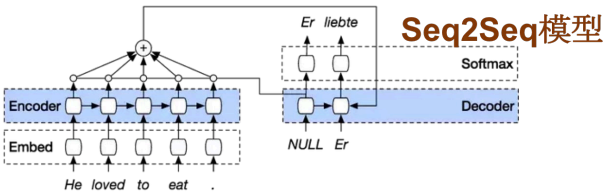


- Seq2Seq 机器翻译模型: RNN + Attention

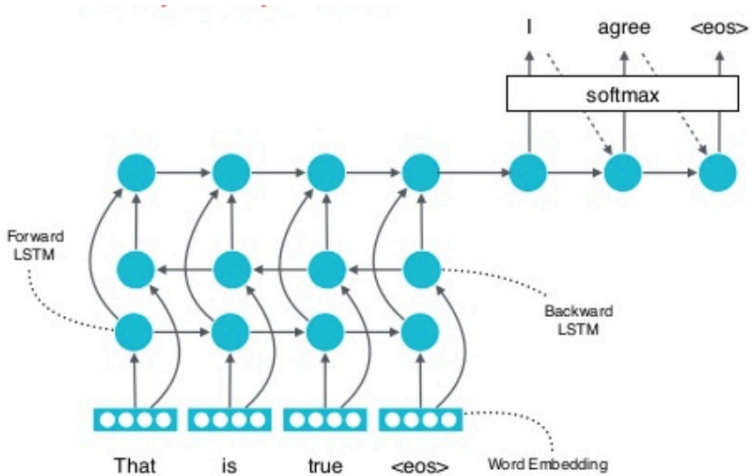




Attention模型

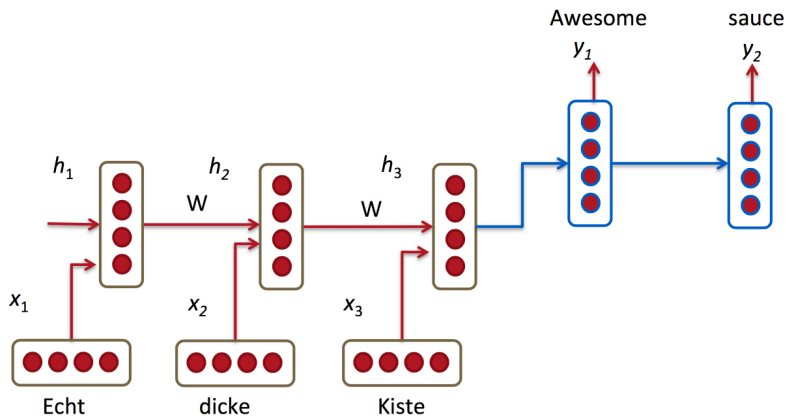


- Seq2Seq:Bidirectional RNN + Attention

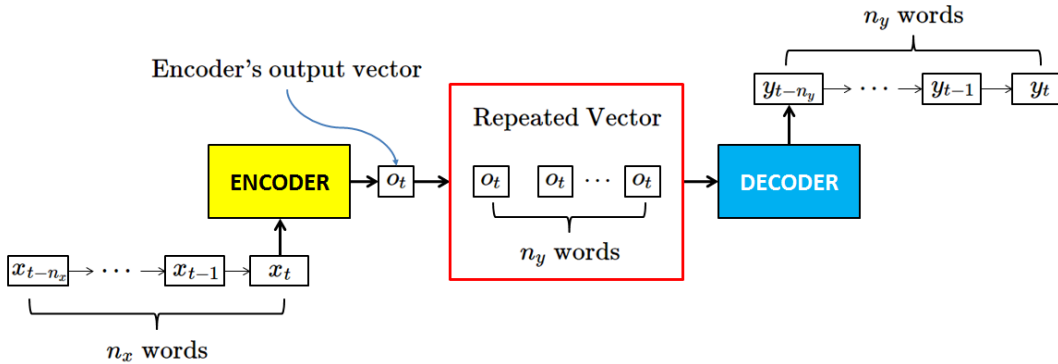


1.2 RNN Encoder-Decoder 模型

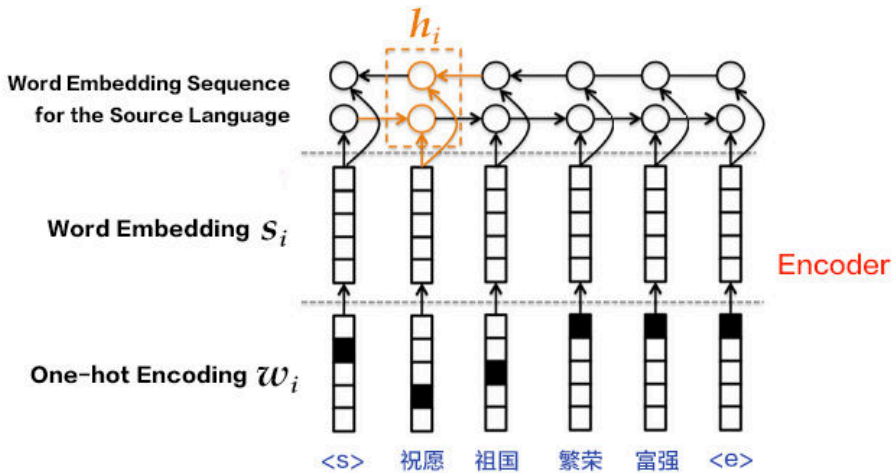
- RNN 模型



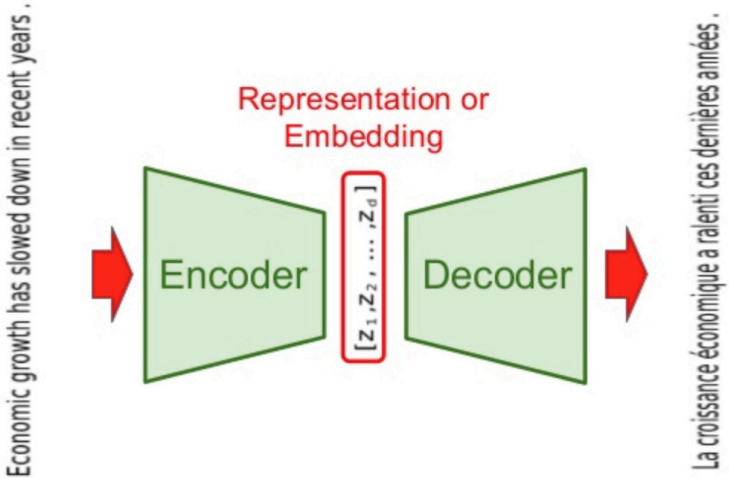
- RNN Encoder-Decoder



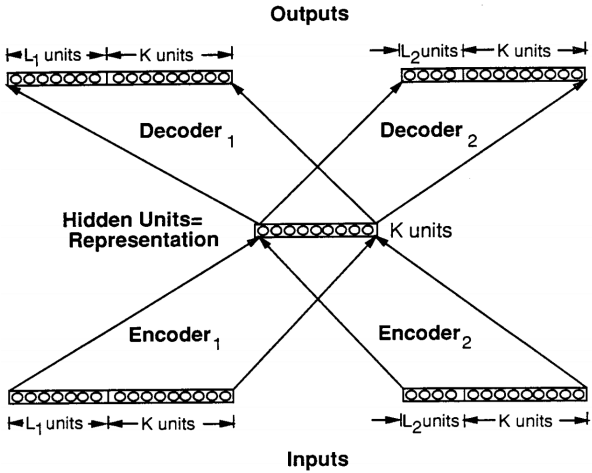
- Bidirectional RNN Encoder



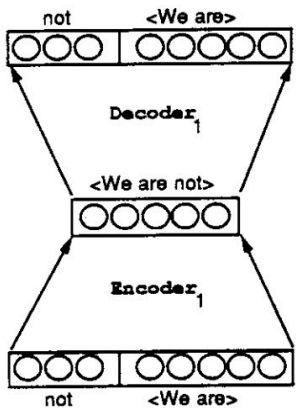
1.3 Encoder-Decoder 模型

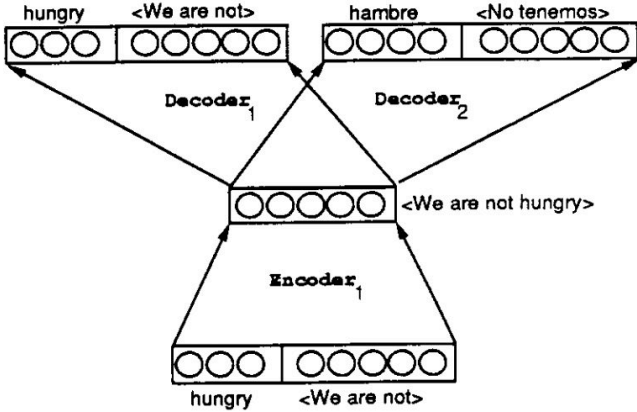


- Dual-Ported Encoder-Decoder



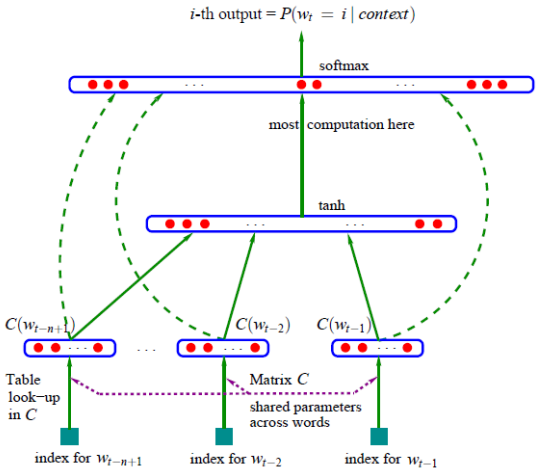
- Contextual Encoder-Decoder



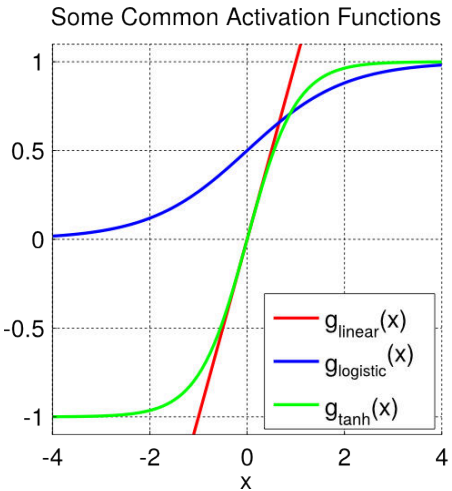


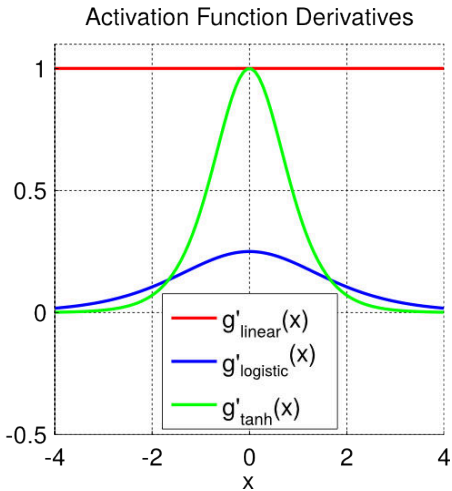
1.4 NNLM 模型

- 神经网络语言模型 (Bengio)



- Tanh 在 0 附近近似 y=x





1.5 Log-Linear 模型

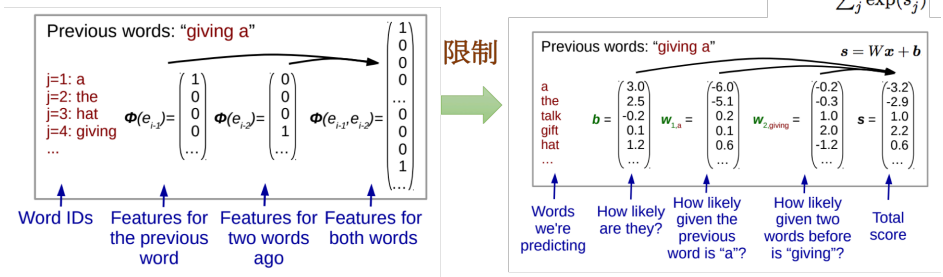
- Log-Linear 模型

$$P(y \mid x, \mathbf{W}) = \frac{e^{\Sigma_k \mathbf{W}_k \phi_k(x,y)}}{\Sigma_{y' \in \mathcal{Y}} e^{\Sigma_k \mathbf{W}_k \phi_k(x,y')}}$$

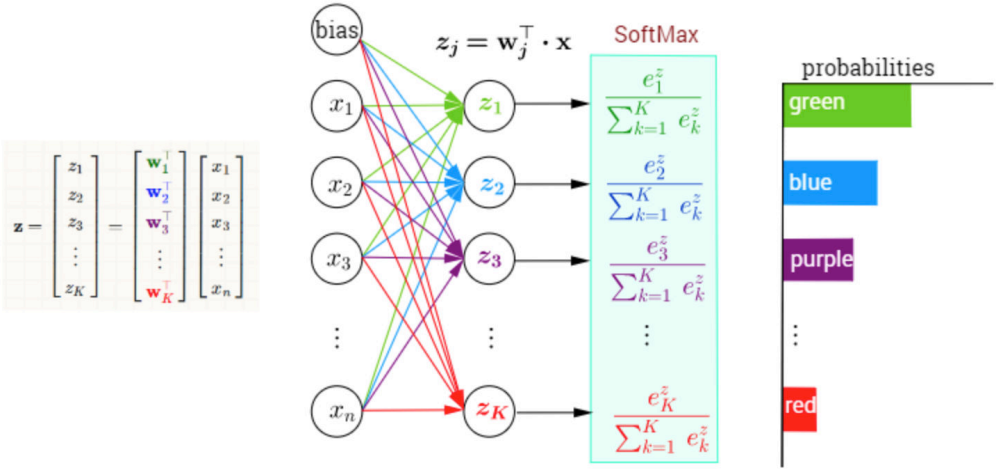
概率

$$\boldsymbol{p} = \text{softmax}(\boldsymbol{s})$$

$$p_j = \frac{\exp(s_j)}{\sum_{\tilde{j}} \exp(s_{\tilde{j}})}$$

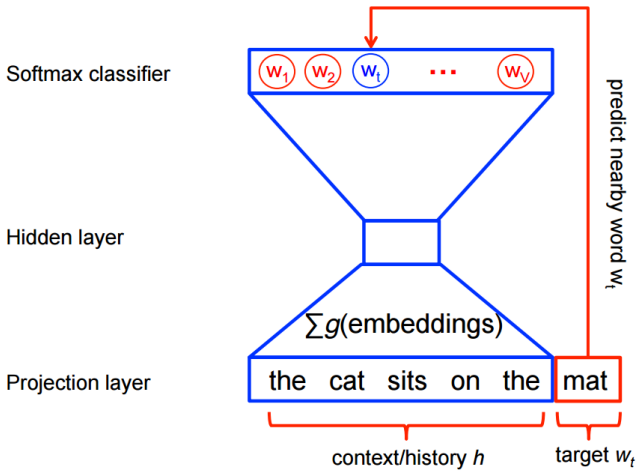


- Softmax 神经网络模型

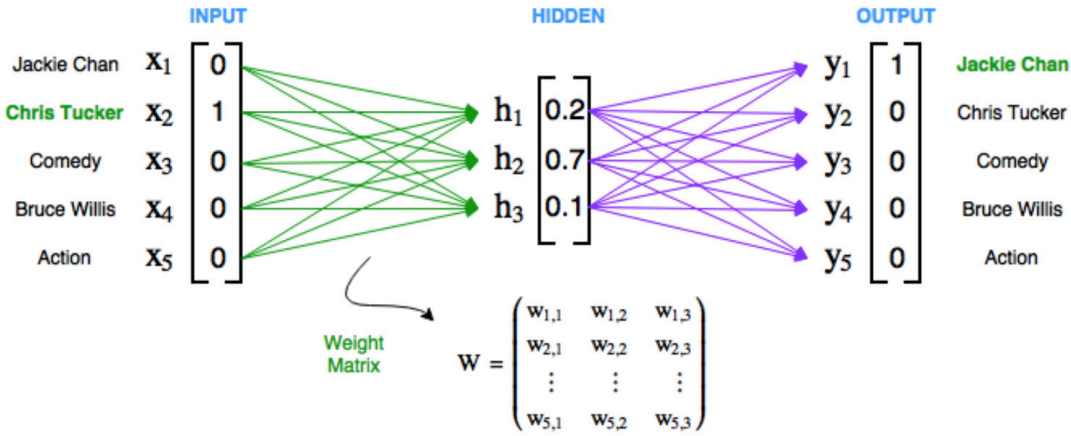


1.6 嵌入词模型

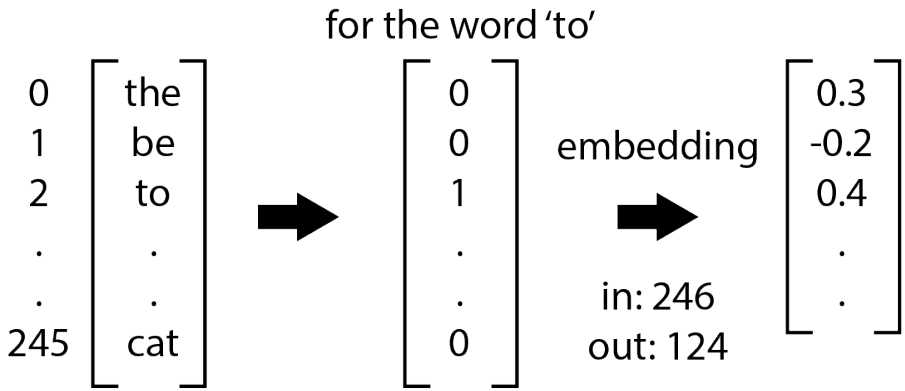
- Softmax 神经网络模型



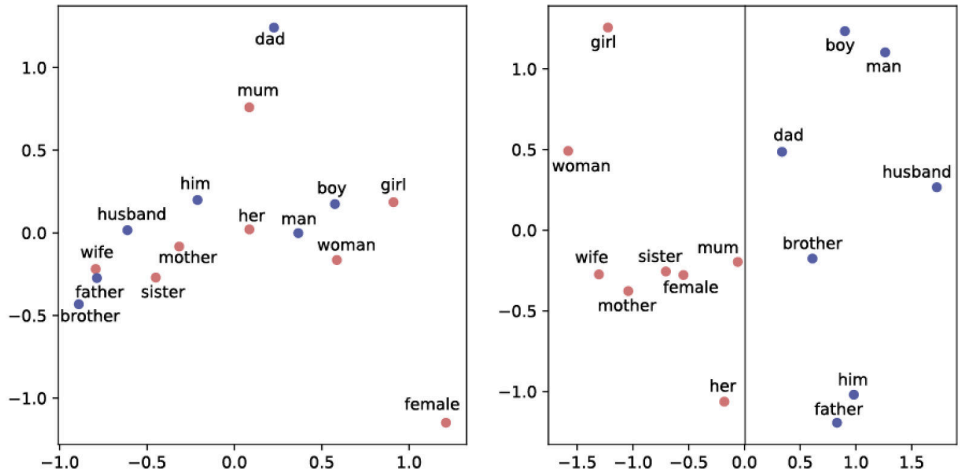
- 词嵌入模型


INPUT
HIDDEN
OUTPUT
Jackie Chan
Chris Tucker
Comedy
Bruce Willis
Action
Jackie Chan
Chris Tucker
Comedy
Bruce Willis
Action
Weight Matrix

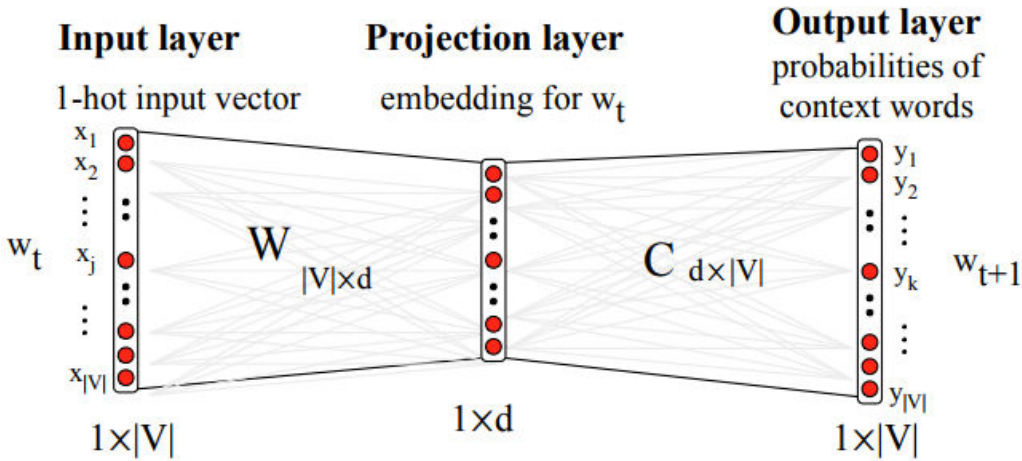
- 词嵌入模型


for the word 'to'
0
1
2
.
.
245
the
be
to
.
.
cat
0
0
1
.
.
0
embedding
in: 246
out: 124
0.3
-0.2
0.4
.
.

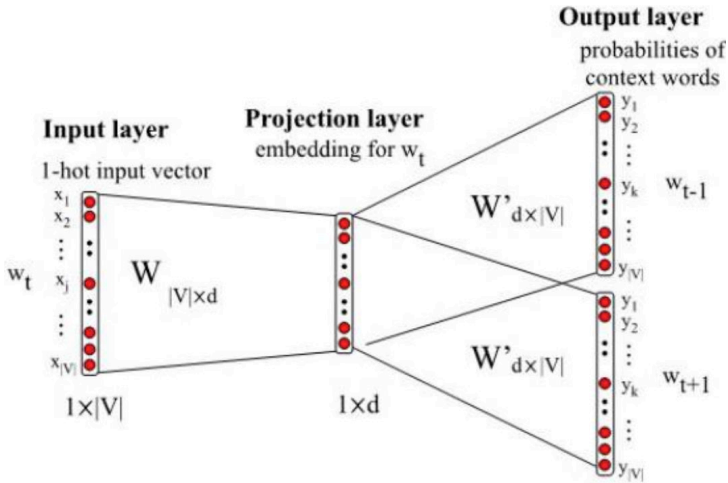
- 词嵌入模型例子


dad
mum
him
boy
girl
husband
her
man
woman
wife
mother
father
sister
brother
female
girl
boy
man
dad
woman
husband
wife
sister
mum
brother
female
mother
her
him
father

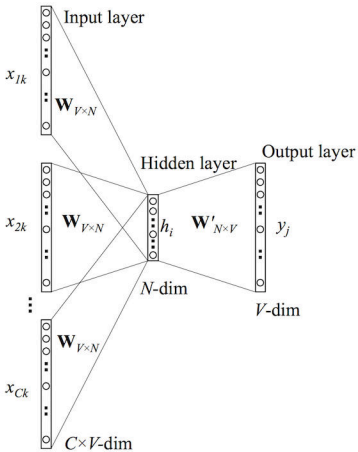
- 词嵌入模型例子


Input layer
1-hot input vector
Projection layer
embedding for w_t
Output layer
probabilities of context words
w_t
x_1
x_2
x_j
$x_{|V|}$
$W_{|V|\times d}$
$C_{d\times|V|}$
y_1
y_2
y_k
$y_{|V|}$
w_{t+1}
$1\times|V|$
$1\times d$
$1\times|V|$

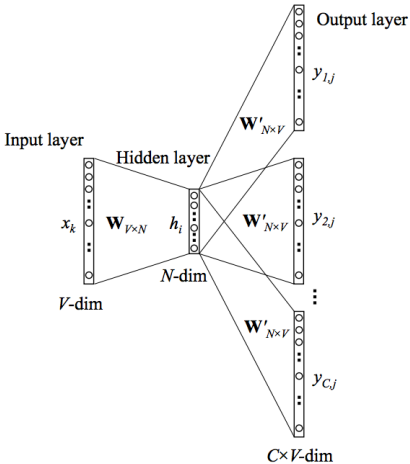
- 词嵌入模型例子


Output layer
probabilities of context words
Input layer
1-hot input vector
Projection layer
embedding for w_t
w_t
$W_{|V|\times d}$
$W'_{d\times|V|}$
$W'_{d\times|V|}$
w_{t-1}
w_{t+1}
$1\times|V|$
$1\times d$

- 连续词袋 CBoW 模型

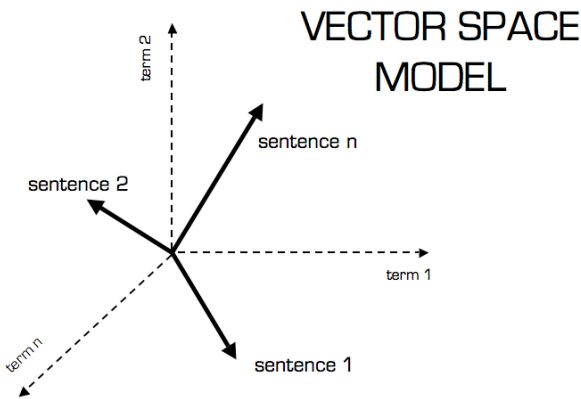

Input layer
x_{1k}
$\mathbf{W}_{V\times N}$
Hidden layer
Output layer
x_{2k}
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
y_j
N-dim
V-dim
$\mathbf{W}_{V\times N}$
x_{Ck}
C×V-dim

- Skip-Gram 模型


Output layer
$y_{1,j}$
$\mathbf{W}'_{N\times V}$
Input layer
Hidden layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{2,j}$
N-dim
V-dim
$\mathbf{W}'_{N\times V}$
$y_{C,j}$
$C\times V$-dim

1.7 向量空间模型

- 向量空间 Vector Space Model: 句子



- TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

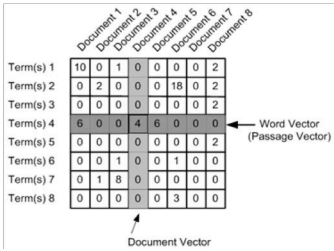
TF-IDF

Term x within document y

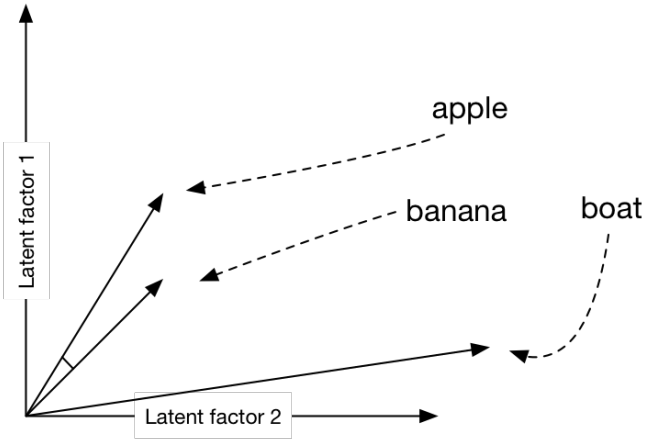
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents



- 向量空间 Vector Space Model: 词



- 共现矩阵 Co-occurrence Matrix

car engine hood tires truck trunk

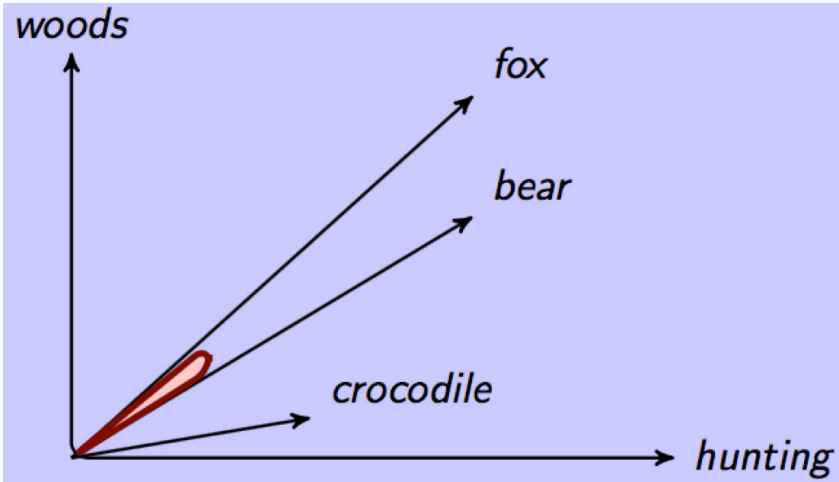
car emissions hood make model trunk

Chomsky corpus noun parsing tagging wonderful

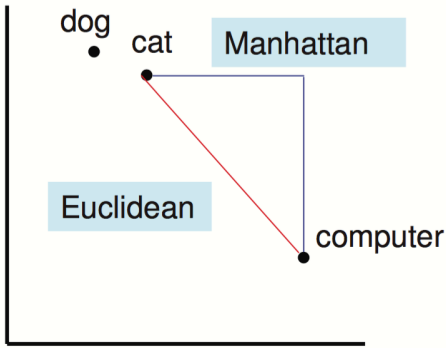
共现矩阵

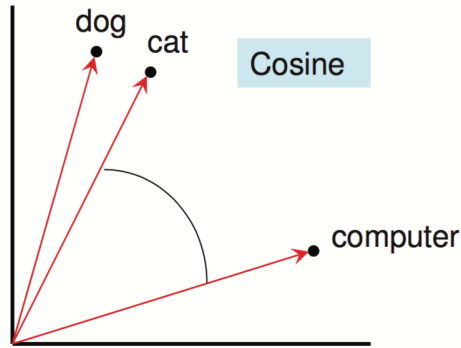
	car	Chomsky	corpus	emissions	engine	hood	make	model	noun	parsing	tagging	tires	truck	trunk	wonderful
car	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0
hood	1	0	0	1	0	0	1	1	0	0	0	0	0	1	0
Chomsky	0	0	1	0	0	0	0	0	1	1	1	0	0	0	1

- Cosine 距离



- 其他常用距离：欧式距离和曼哈顿距离





- 距离计算公式

The *cosine* of the angle between two vectors **x** and **y** is:

$$cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| \cdot ||\mathbf{y}||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}}$$

The *Euclidean distance* of two vectors **x** and **y** is:

$$||\mathbf{x} - \mathbf{y}|| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

1.8 词袋 BoW 模型

- BoW 模型: 基于词典序列

"Represent each document which the bag of words it contains"

d1 : Mary loves Movies, Cinema and Art
d2 : John went to the Football game

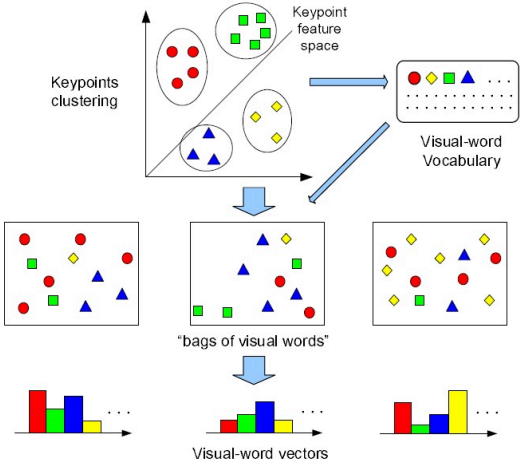
Class 1 : Arts
Class 2 : Sports

d3 : Robert went for the Movie Delicatessen

Class : Arts

	Mary	Loves	Movies	Cinema	Art	John	Went	to	the	Delicatessen	Robert	Football	Game	and	for
d1	1	1	1	1	1									1	
d2						1	1	1	1			1	1		
d3			1				1		1		1				1

- BoW 模型


Keypoint
feature
space
Keypoints
clustering
Visual-word
Vocabulary
"bags of visual words"
Visual-word vectors

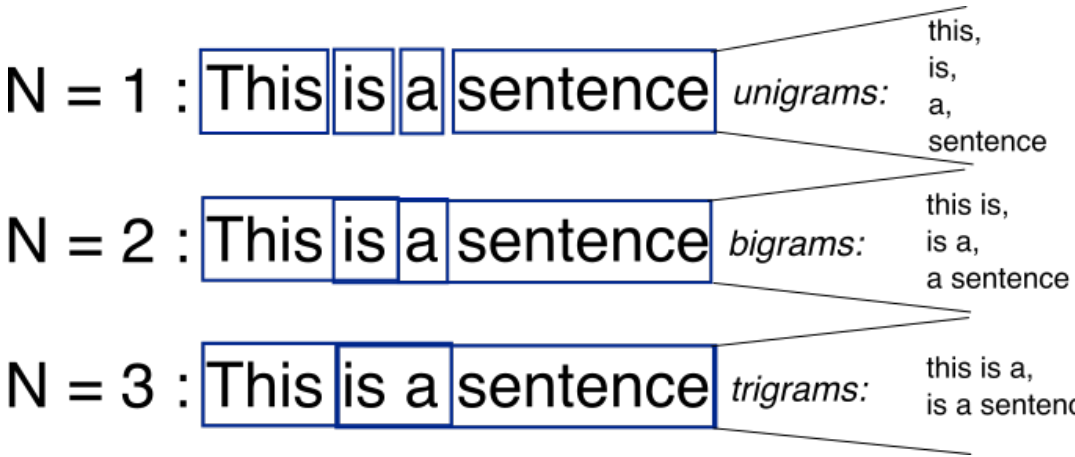
- 1-Hot-Encoding

Original data:	
id	Color
1	White
2	Red
3	Black
4	Purple
5	Gold

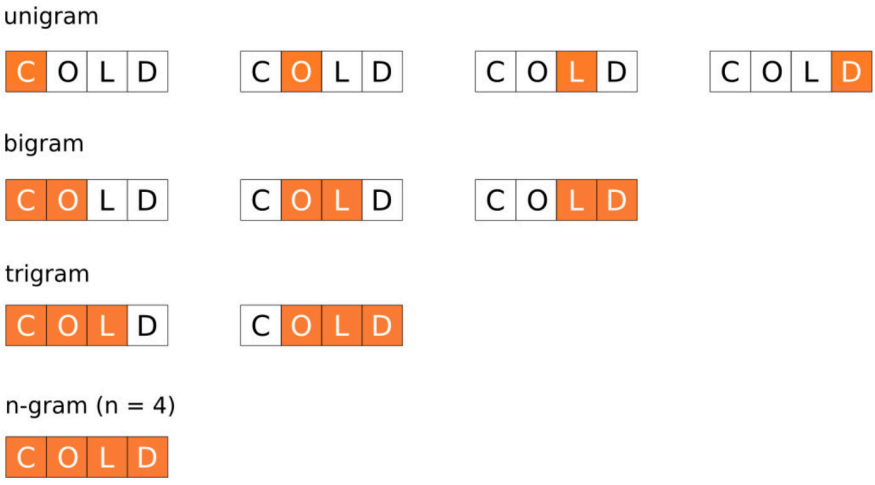
One-hot encoding format:					
id	White	Red	Black	Purple	Gold
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

1.9 N-Gram 模型

- N-Gram



- N-Gram


unigram
C O L D
C O L D
C O L D
C O L D
bigram
C O L D
C O L D
C O L D
trigram
C O L D
C O L D
n-gram (n = 4)
C O L D

- 全概率公式展开

$$P(Girl \mid Flower) = \frac{\Sigma\, count(Flower\, Girl)}{\Sigma\, count(Girl)}$$

$P(W) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n)$

$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)\ldots P(x_n \mid x_1, \ldots, x_{n-1})$

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water | its) × P(is | its water)

× P(so | its water is) × P(transparent | its water is so)

- 马尔可夫近似

1阶Markov假设 （bigram） :

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

2阶Markov假设 (trigram) :

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

$$P(Girl \mid Flower) = \frac{\Sigma\, count(Flower\, Girl)}{\Sigma\, count(Girl)}$$

- 概率估算问题

P(|E| = 3, e_1="she", e_2="went", e_3="home") =
P(e_1="she")
* P(e_2="went" | e_1="she")
* P(e_3="home" | e_1="she", e_2="went")
* P(e_4="</s>" | e_1="she", e_2="went", e_3="home")

$$P(E) = \prod_{t=1}^{T+1} P(e_t \mid e_1^{t-1})$$

高阶 **N-Gram** 数据稀疏，不存在问题！

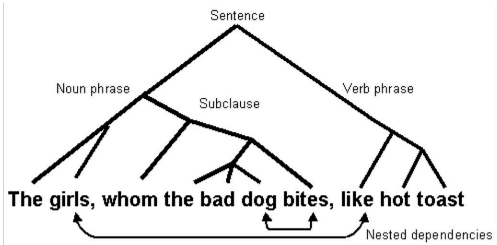
Smoothing 技术：线性插值低阶N-Gram

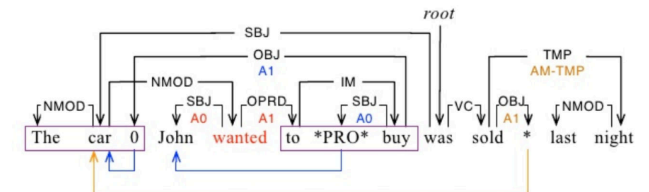
$$P(e_t \mid e_{t-1}) = (1-\alpha)P_{\mathrm{ML}}(e_t \mid e_{t-1}) + \alpha P_{\mathrm{ML}}(e_t)$$

Modified Kneser-Ney smoothing

$$P(e_t \mid e_{t-m+1}^{t-1}) = (1-\alpha_m)P_{\mathrm{ML}}(e_t \mid e_{t-m+1}^{t-1}) + \alpha_m P(e_t \mid e_{t-m+2}^{t-1})$$

- 长距离依赖问题


Sentence
Noun phrase
Subclause
Verb phrase
The girls, whom the bad dog bites, like hot toast
Nested dependencies


root
SBJ
OBJ
A1
TMP
AM-TMP
NMOD
IM
NMOD
SBJ
A0
OPRD
A1
SBJ
A0
VC
OBJ
A1
NMOD
The car 0 John wanted to *PRO* buy was sold * last night

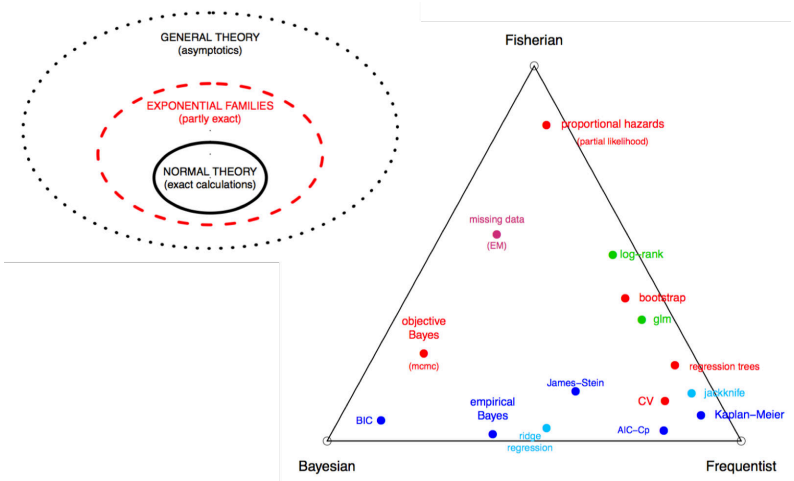
2 数学基础

2.1 概率统计

- 三大学派

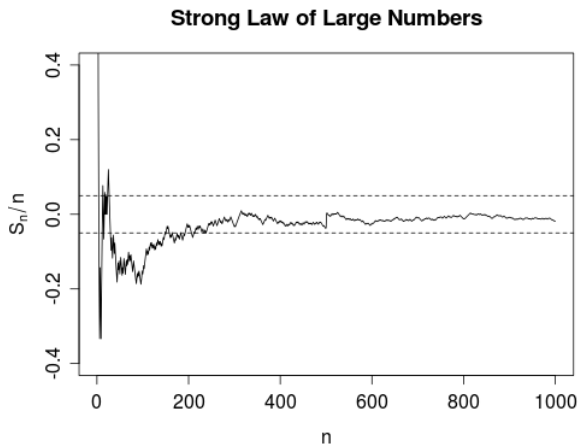
1. 频率派基于计数，通过概率极限建立概率分布
2. Fisher 派基于经验，通过最大似然估计建立概率分布
3. 贝叶斯派基于推理，通过最大熵和贝叶斯公式建立概率分布

- 分布、方法、学派

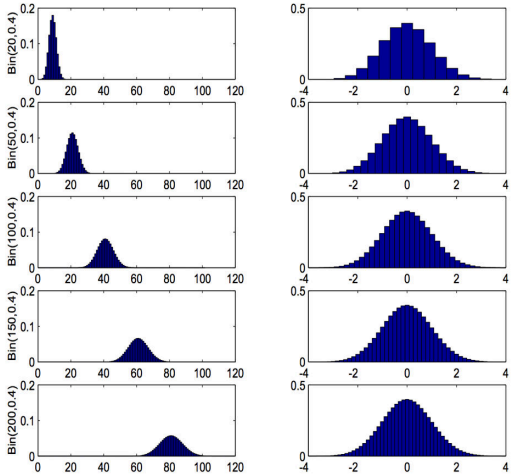

GENERAL THEORY
(asymptotics)
EXPONENTIAL FAMILIES
(partly exact)
NORMAL THEORY
(exact calculations)
Fisherian
proportional hazards
(partial likelihood)
missing data
(EM)
log-rank
bootstrap
glm
objective
Bayes
(mcmc)
regression trees
James–Stein
jackknife
CV
empirical
Bayes
Kaplan–Meier
BIC
AIC–Cp
ridge
regression
Bayesian
Frequentist

2.2 频率派

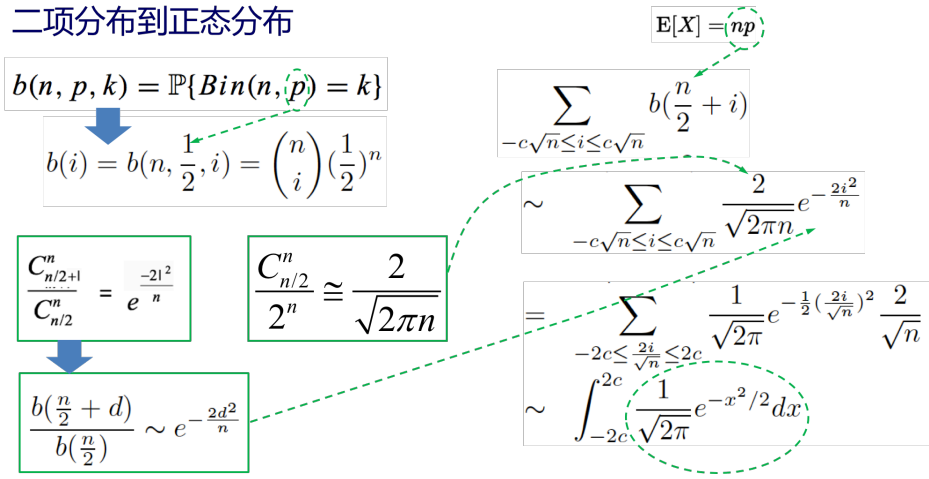
- 大数定理和按概率收敛



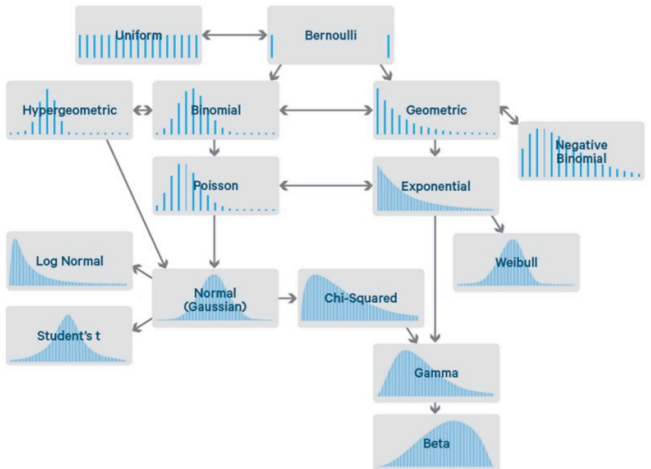
- 二项分布的大数极限是正态分布



- 二项分布的大数极限是正态分布

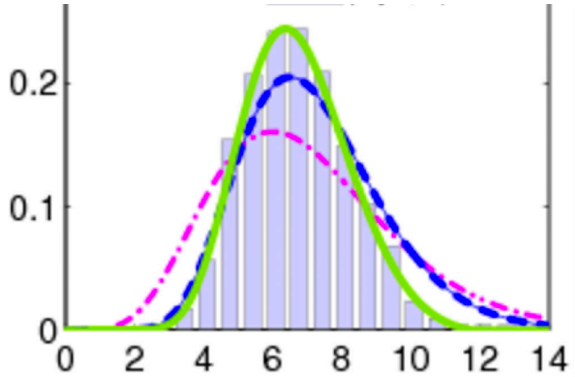


- 正态分布到指数族分布


Uniform
Bernoulli
Hypergeometric
Binomial
Geometric
Negative Binomial
Poisson
Exponential
Log Normal
Normal (Gaussian)
Chi-Squared
Weibull
Student's t
Gamma
Beta

2.3 Fisher 派

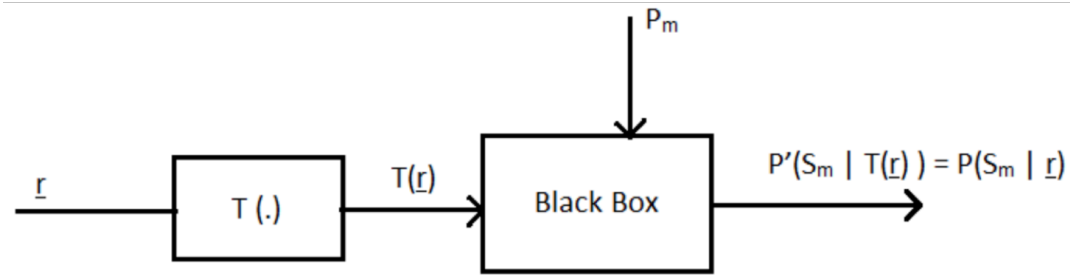
- 最大似然估计和经验



- 奥卡姆剃刀原理

- 充分统计量

$$L(x_1, x_2, \ldots x_n \mid \vartheta) = g(\vartheta, T(x_1, x_2, \ldots x_n)) \cdot h(x_1, x_2, \ldots x_n)$$



- Fisher-Neyman 分解

Fisher–Neyman factorization theorem

$f_\theta(x) = h(x)\, g_\theta(T(x))$

$$
\begin{aligned}
f_{X_1^n}(x_1^n) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\theta)^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\sum_{i=1}^{n} \frac{(x_i-\theta)^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\sum_{i=1}^{n} \frac{((x_i-\overline{x})-(\theta-\overline{x}))^2}{2\sigma^2}\right) \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}(x_i-\overline{x})^2 + \sum_{i=1}^{n}(\theta-\overline{x})^2 - 2\sum_{i=1}^{n}(x_i-\overline{x})(\theta-\overline{x})\right)\right) \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}(x_i-\overline{x})^2 + n(\theta-\overline{x})^2\right)\right) \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\overline{x})^2\right) \exp\left(-\frac{n}{2\sigma^2}(\theta-\overline{x})^2\right)
\end{aligned}
$$

- 最大似然估计到正态分布

$x_1, \cdots, x_n$ n次观测值

$e_i = x_i - \theta$ n次观测误差

假定误差的密度分布函数为 $f(e)$

$$\frac{d}{dx}\left(\mathbf{e}^x\right) = \mathbf{e}^x \qquad \frac{d}{dx}\left(a^x\right) = a^x \ln a$$

$$\frac{d}{dx}\left(\ln x\right) = \frac{1}{x} \qquad \frac{d}{dx}\left(\log_a x\right) = \frac{1}{x \ln a}$$

$$\begin{aligned} L(\theta) &= L(\theta; x_1, \cdots, x_n) \\ &= f(e_1) \cdots f(e_n) \\ &= f(x_1 - \theta) \cdots f(x_n - \theta) \end{aligned}$$

$$\frac{d \log L(\theta)}{d\theta} = 0$$

$$\hat{\theta} = \bar{x}$$

$$\sum_{i=1}^{n} \frac{f'(x_i - \theta)}{f(x_i - \theta)} = 0$$

- 最大似然估计到正态分布

$$\sum_{i=1}^{n}\frac{f'(x_i-\theta)}{f(x_i-\theta)}=0 \quad \hat{\theta}=\bar{x}$$

$$f'(x)=e^{x^2}.(x^2)'=2xe^{x^2}$$

$$\boxed{f'(x)=2xe^{x^2}}$$

$$g(x)=\frac{f'(x)}{f(x)} \Rightarrow \sum_{i=1}^{n}g(x_i-\theta)=0 \Rightarrow \sum_{i=1}^{n}g(x_i-\bar{x})=0$$

标准正态分布：

$$\varphi(x)=\frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}}$$

$n=2$

$$g(x_1-\bar{x})+g(x_2-\bar{x})=0$$

$$x_1-\bar{x}=-(x_2-\bar{x})$$

$$g(-x)=-g(x)$$

$n=m+1 \quad x_1=\cdots=x_m=-x, x_{m+1}=mx \quad \bar{x}=0$

$$\sum_{i=1}^{n}g(x_i-\bar{x})=mg(-x)+g(mx)$$

$$g(mx)=mg(x) \Rightarrow g(x)=cx \Rightarrow f(x)=Me^{cx^2}$$

- 改变充分统计量到指数族分布

Distribution	Parameter(s) θ	Natural parameter(s) η	Base measure $h(x)$	Sufficient statistic $T(x)$
Bernoulli distribution	p	$\ln\frac{p}{1-p}$ • This is the logit function.	1	x
Poisson distribution	λ	$\ln\lambda$	$\frac{1}{x!}$	x
exponential distribution	λ	$-\lambda$	1	x
normal distribution	μ,σ^2	$\begin{bmatrix}\frac{\mu}{\sigma^2}\\ -\frac{1}{2\sigma^2}\end{bmatrix}$	$\frac{1}{\sqrt{2\pi}}$	$\begin{bmatrix}x\\ x^2\end{bmatrix}$
gamma distribution	α,β	$\begin{bmatrix}\alpha-1\\ -\beta\end{bmatrix}$	1	$\begin{bmatrix}\ln x\\ x\end{bmatrix}$
	k,θ	$\begin{bmatrix}k-1\\ -\frac{1}{\theta}\end{bmatrix}$		

$$f_X(x \mid \theta) = h(x)g(\theta)\exp(\eta(\theta)\cdot T(x))$$

2.4 贝叶斯派

- 最大相互熵的求解：满足分布积分为 1 和一阶均值矩估计

1. 初始观察分布 $m(x)$：这个可以是随意的观察情况，不一定必须是一个分布函数。
2. $\sum_{x\in S} f(x) = 1,\ f(x) \geq 0$：满足分布函数的情况，这里仅仅考虑离散的情况。
3. $\sum_{x\in S} t_j(x) f(x) = \mathbb{E}\{t_j(x)\} = \mu_j$，其中 $j \in J$：$t_j(x)$ 是在结合上的一个测量函数，并且我们知道测量值的期望是 μ_j。

目标：$argmax_{f(x)} RE(f(x)\|m(x))$：我们希望找一个函数满足上述的限制条件，并且尽可能与初始观察分布的相对熵最大。

$$RE(f(x)\|m(x)) = -\sum_{x\in S} f(x) \log \frac{f(x)}{m(x)}$$

- 应用拉格朗日乘数法:

$$L(f) = -\sum_{x\in S} f(x)\log\frac{f(x)}{m(x)} + \lambda(\sum_{x\in S} f(x) - 1) + \sum_{j\in J}\theta_j(\sum_{x\in S} t_j(x)f(x) - \mu_j)$$

令 $L(f)$ 对 f 的导数为 0

$$\begin{aligned} 0 = \frac{\partial L(f)}{\partial f} &= -(\log\frac{f(x)}{m(x)} + 1) + \lambda + \sum_{j\in J}\theta_j t_j(x) \\ &= -\log f(x) + \log m(x) - 1 + \lambda + \sum_{j\in J}\theta_j t_j(x) \end{aligned}$$

从而得到

$$f(x) = m(x)\exp[\lambda - 1 + \sum_{j\in J}\theta_j t_j(x)] \tag{1}$$

- 接下来应用概率求和为 1 的限制条件：

$$
\begin{aligned}
1 &= \sum_{x \in S} f(x) = \sum_{x \in S} m(x) \exp\{\lambda - 1 + \sum_{j \in J} \theta_j t_j(x)\} \\
&= e^{\lambda - 1} \sum_{x \in S} m(x) \exp\{\sum_{j \in J} \theta_j t_j(x)\}
\end{aligned}
$$

于是有

$$
1 - \lambda = \log \left(\sum_{x \in S} m(x) \exp[\sum_{j \in J} \theta_j t_j(x)] \right)
$$

- 代入优化表达式化简：

我们将上式的右边定义为 $b(\vec{\theta})$，其中 $\vec{\theta}=(\theta_1,\dots,\theta_{|J|})$，$\vec{t}(x)=(t_1(x),\dots,t_{|J|}(x))$：

$$\begin{aligned} b(\vec{\theta}) &= \log\left(\sum_{x\in S} m(x)\exp[\sum_{j\in J}\theta_j t_j(x)]\right) \\ &= \log\left(\mathbb{E}(e^{\vec{t}(x)^T\vec{\theta}})\right) \\ &= 1-\lambda \end{aligned}$$

在(1)式中替换 $1-\lambda$ 得到 $f(x)$ 的如下形式：

$$\begin{aligned} f(x) &= m(x)\exp[-b(\vec{\theta})+\textstyle\sum_{j\in J}\theta_j t_j(x)] \\ &= m(x)\exp[\vec{t}(x)^T\vec{\theta}-b(\vec{\theta})] \end{aligned}$$

- 最大相互熵推导出概率分布

1. 指数族概率分布
   - 这(2)就是指数族概率分布形式
2. 正态分布
   - 只用了 1 阶矩估计：均值限制
   - 如果用了 2 阶矩估计：方差限制，就得到我们要的正态分布了！
3. 频率派，费希尔派，到贝叶斯派，要求的事实观测越来越少，要求的经验也越来越少。

- 以少测多，见微知著

  - 从频率出发生成概率基本概念
  - 从经验出发总结概率基本工具
  - 从推理出发奠定概率基本原理

2.5 K 分类最大相互熵推导 Softmax

- 限制下求分布设 $\sigma(\boldsymbol{x})_v$ 表示 $\boldsymbol{x}$ 属于第 v 个分类的概率，共有 K 个分类。

1. 概率分布 $\sigma(\boldsymbol{x})_v$ 的限制条件：

$$\begin{aligned} \sigma(\boldsymbol{x})_v &\geq 0 \\ \sum_{v=1}^{K} \sigma(\boldsymbol{x})_v &= 1 \end{aligned}$$

2. 分布一致限制：分布 $\sigma(\boldsymbol{x})_v$ 满足训练集 (Training Set) 中数据的要求，即 $\sigma(\boldsymbol{x})_v$ 与训练集的数据分布一致：

$$\sum_{i=1}^{n} \sigma(\boldsymbol{x}(i))_v \boldsymbol{x}(i)_j = \sum_{i=1}^{n} \mathbb{I}_v(y(i)) \boldsymbol{x}(i)_j$$

   $\mathbb{I}_u(y(i))$ 为指示函数，当 $y(i) = u$ 时值为 1，当 $y(i) \neq u$ 时值为 0。

3. 最大熵目标在每一个分类 u 里，任意一个特征 j 在属于该分类的训练数据 $\boldsymbol{x}(i)$ 上的求和，等于所训练的模型分配给特征 j 的概率质量 (Probability Mass) 之和 ($\boldsymbol{x}(i)$ 属

于分类 u 的概率，在全部训练数据上求和)。这表明 $\sigma(\boldsymbol{x}(i))_v$ 是对训练集的指标函数 $\mathbb{I}_u(y(i))$ 的一个很好的近似。

$\sigma(\boldsymbol{x}(i))_v$ 的熵定义为：

$$-\sum_{v=1}^{K}\sum_{i=1}^{n}\sigma(\boldsymbol{x}(i))_v\log(\sigma(\boldsymbol{x}(i))_v)$$

根据 1，2，3 条件求最优化问题，可以使用拉格朗日乘数法求解：

$$\begin{aligned}L = & \sum_{j=1}^{m}\sum_{v=1}^{K}\lambda_{v,j}(\sum_{i=1}^{n}\sigma(\boldsymbol{x}(i))_v\boldsymbol{x}(i)_j-\sum_{i=1}^{n}\mathbb{I}_v(y(i))\boldsymbol{x}(i)_j)\\ & +\sum_{i=1}^{n}\beta_i(\sum_{v=1}^{K}\sigma(\boldsymbol{x})_v-1)\\ & -\sum_{v=1}^{K}\sum_{i=1}^{n}\sigma(\boldsymbol{x}(i))_v\log(\sigma(\boldsymbol{x}(i))_v)\end{aligned}$$

式中的 L 对 $\sigma(\boldsymbol{x}(i))_v$ 求偏导得到：

$$\frac{\partial L}{\partial \sigma(\boldsymbol{x}(i))_v} = \lambda_v \boldsymbol{x}(i) + \beta_i - \log(\sigma(\boldsymbol{x}(i))_v) - 1$$

令上式等于 0：

$$\lambda_v \boldsymbol{x}(i) + \beta_i - \log(\sigma(\boldsymbol{x}(i))_v) - 1 = 0$$

解方程得到：

$$\sigma(\boldsymbol{x}(i))_v = e^{\lambda_v \boldsymbol{x}(i) + \beta_i - 1}$$

根据限制条件，概率 $\sigma(\boldsymbol{x}(i))_v$ 求和等于 1：

$$\sum_{v=1}^{k} e^{\lambda_v \boldsymbol{x}(i) + \beta_i - 1} = 1$$

于是得到：

$$e^{\beta_i - 1} = \frac{1}{\sum_{v=1}^{k} e^{\lambda_v \boldsymbol{x}(i) - 1}}$$

把上式中的 $e^{\beta_i - 1}$ 代入：

$$\sigma(\boldsymbol{x}(i)) = \frac{e^{\lambda_u \boldsymbol{x}(i)}}{\sum_{v=1}^{k} e^{\lambda_v \boldsymbol{x}(i)}}$$

即

$$\sigma(\boldsymbol{x}) = \frac{e^{\lambda_u \boldsymbol{x}}}{\sum_{v=1}^{k} e^{\lambda_v \boldsymbol{x}}}$$

再对 Softmax 函数的每个分量做线性扩展，那么我们就得到 Log-Linear 模型

$$p(y|x;\vec{w}) = \frac{\exp\left(c + \sum_{j=1}^{|\vec{w}|} w_j f_j(x,y)\right)}{Z(x,\vec{w})},$$

$$Z(x,\vec{w}) = \sum_{y' \in Y} \exp\left(c + \sum_{j=1}^{|\vec{w}|} w_j f_j(x,y')\right)$$

如果写成向量的形式就变成了：

$$p(y|x;\vec{w}) = \frac{\exp\left(\vec{w}^T \vec{f}(x,y)\right)}{Z(x,\vec{w})}$$

对比一下 Softmax 函数，我们把 Softmax 的每个输入自变量变成了线性：

$$v(x,y) = \vec{w}^T \vec{f}(x,y) \Rightarrow$$

$$p(y|x;\vec{w}) = \sigma(v(x,y))_{y\in Y} = \frac{e^{v(x,y)}}{\sum_{y'\in Y} e^{v(x,y')}}$$

3 词嵌入模型

3.1 词-文上下文 Word-Document

- 向量空间模型：文档模型

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

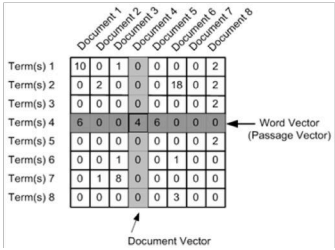
TF-IDF

Term x within document y

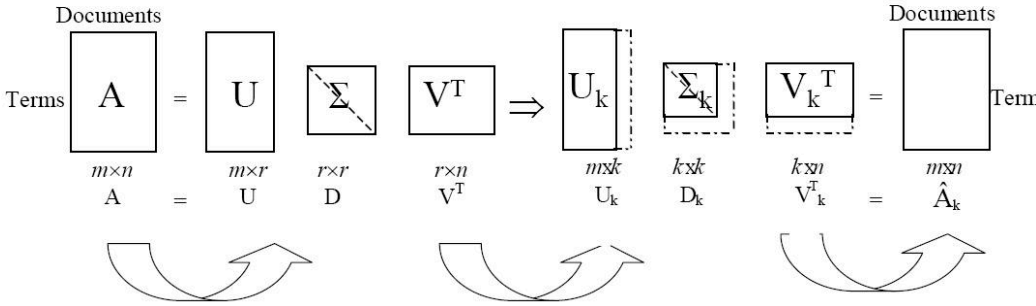
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

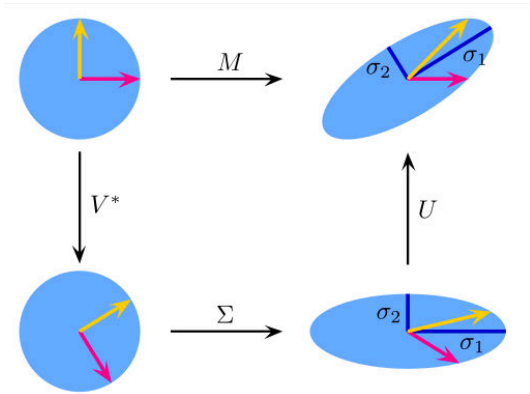
N = total number of documents



- 向量空间矩阵的 SVD


Documents
Terms
A
=
U
Σ
V^T
⇒
U_k
Σ_k
V_k^T
=
Documents
Terms
m×n
m×r
r×r
r×n
m×k
k×k
k×n
m×n
A
=
U
D
V^T
U_k
D_k
V^T_k
=
Â_k

- Singular Value Decomposition （SVD）



$$M = U \cdot \Sigma \cdot V^*$$

- 隐性语义索引 Latent Semantic Indexing（LSI）

A

M	D_1	D_2	D_3	D_4	D_5	D_6	⋯	D_n
T_1	0.00060	0.00012	0.00003	0.00003	0.00333	0.00048	⋯	a_{1n}
T_2	0	0	0	0	0	0	⋯	a_{2n}
T_3	0	2.98862	0	0	0	1.49431	⋯	a_{3n}
T_4	0	0	0	13.32555	0	0	⋯	a_{4n}
T_5	0	0	0	0	0	0	⋯	a_{5n}
T_6	1.03442	1.03442	0	0	0	3.10326	⋯	a_{6n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T_m	a_{m1}	a_{m2}	a_{m3}	a_{m4}	a_{m5}	a_{m6}	⋯	a_{mn}

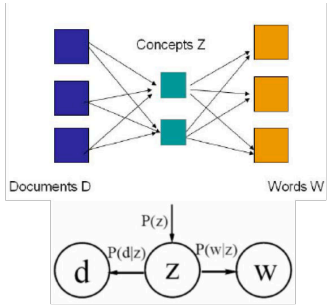
B

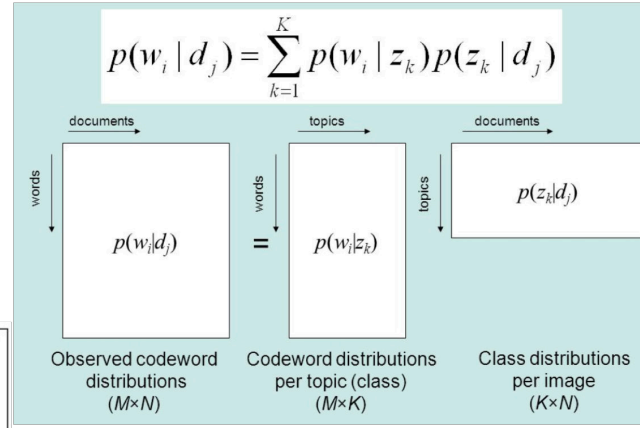
$$U = \begin{array}{c} U_k \\ \begin{array}{c|cccc} & C_1 & C_2 & C_3 & \cdots \; C_m \\ T_1 & a_{11} & a_{12} & a_{13} & \cdots \; a_{1m} \\ T_2 & a_{21} & a_{22} & a_{23} & \cdots \; a_{2m} \\ T_3 & a_{31} & a_{32} & a_{33} & \cdots \; a_{3m} \\ T_4 & a_{41} & a_{42} & a_{43} & \cdots \; a_{4m} \\ T_5 & a_{51} & a_{52} & a_{53} & \cdots \; a_{5m} \\ T_6 & a_{61} & a_{62} & a_{63} & \cdots \; a_{6m} \\ \vdots & \vdots & \vdots & \vdots & \ddots \; \vdots \\ T_m & a_{m1} & a_{m2} & a_{m3} & \cdots \; a_{mm} \end{array} \end{array}$$

$$\Sigma = \begin{array}{c} \Sigma_k \\ \begin{array}{c|ccccc} & D_1 & D_2 & D_3 & \cdots & D_n \\ T_1 & a_{11} & 0 & 0 & \cdots & 0 \\ T_2 & 0 & a_{22} & 0 & \cdots & 0 \\ T_3 & 0 & 0 & a_{33} & \cdots & 0 \\ T_4 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ T_m & 0 & 0 & 0 & \cdots & a_{mm} \end{array} \end{array}$$

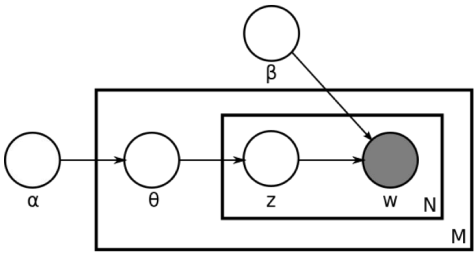
$$V^T = \begin{array}{c} V_k^T \\ \begin{array}{c|ccccc} & D_1 & D_2 & D_3 & \cdots & D_n \\ C_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ C_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ C_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ C_4 & a_{41} & a_{42} & a_{43} & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_n & a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{array} \end{array}$$

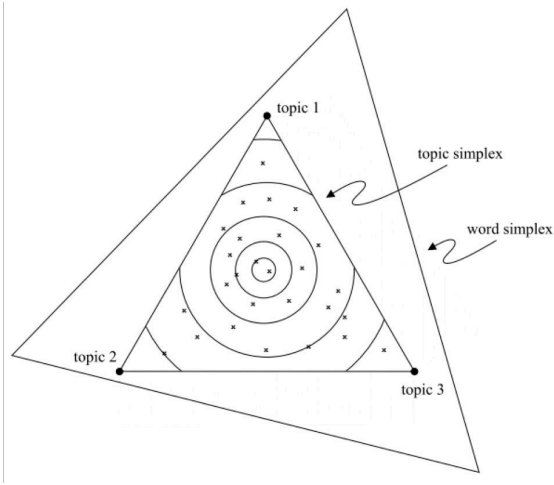
- pLSA：probabilistic Latent Semantic Analytics



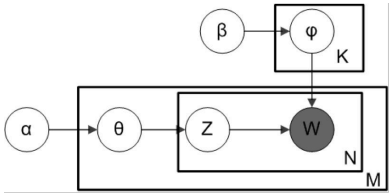


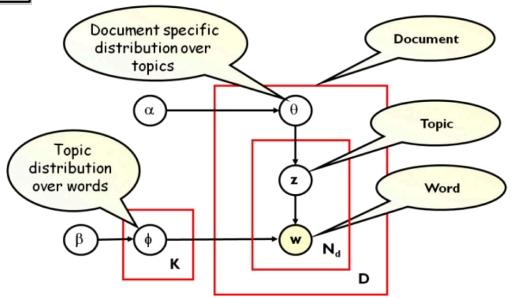
- LDA：Latent Dirichlet allocation





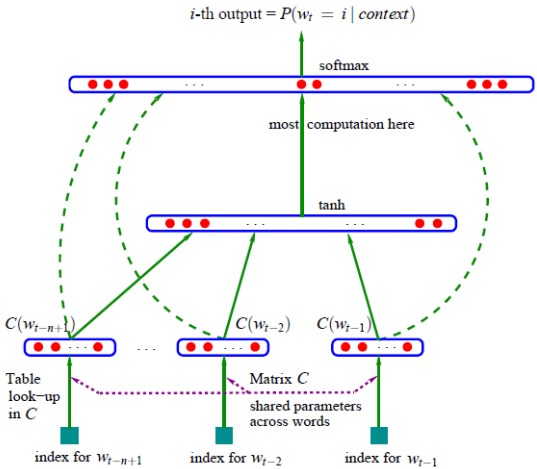
- LDA：Latent Dirichlet allocation



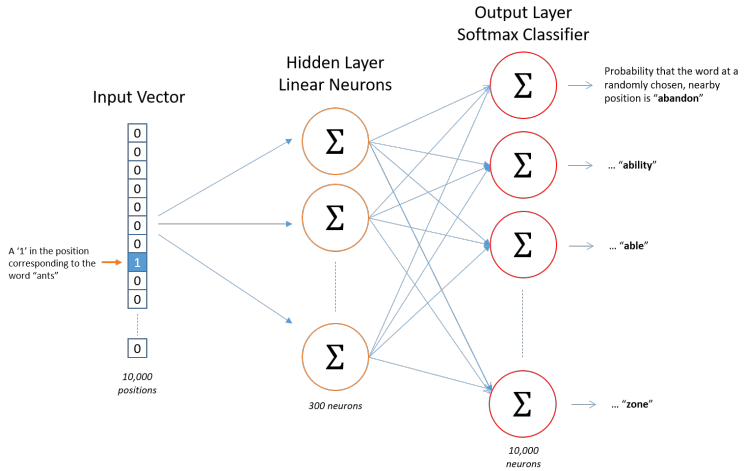


3.2 词上下文 Word-Neighboring Word

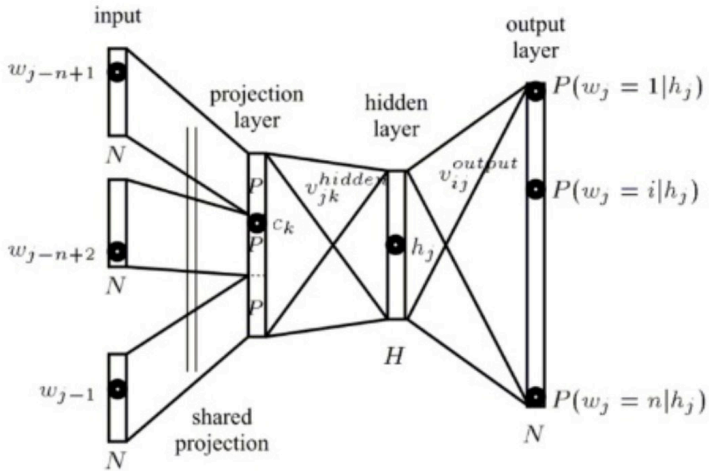
- Neural Net Language Model 2003



- Softmax Network



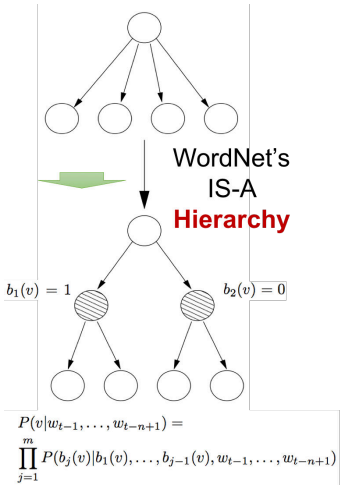
- NNLM 2006: 3 层模型

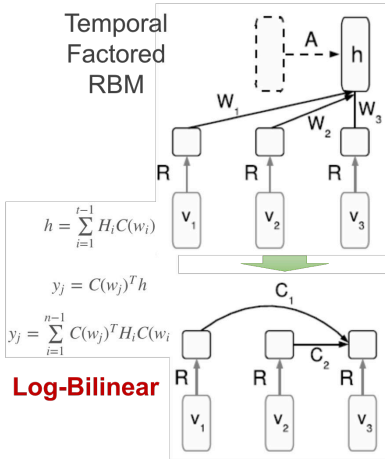


- NNLM 小结

1. 开山之作：2001 年发在 NIPS，2003 同名发到 JMLR
2. 三层的神经网络来构建语言模型
3. 一词多义有待解决，9 年后 Huang 模型搞定
4. 降低参数个数，比如用 RNN, Mikolov 搞定
5. 提出计算量问题，开启 Softmax 近似计算

- M&H 的 Hierarchical Log-BiLinear (HLBL) 模型 2008

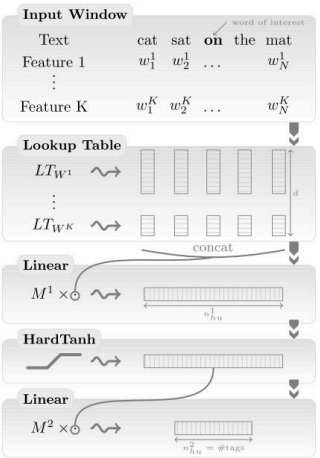




- M&H HLBL 小结

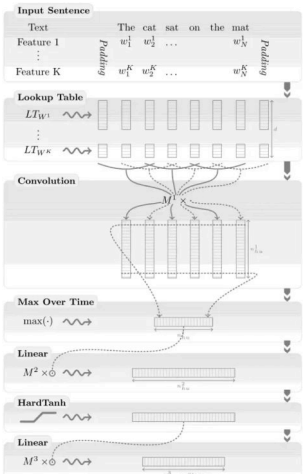
1. Softmax 层变成平衡二叉树输出，训练速度提升
2. Factored RBM(Temporal Factored RBM) 简化到 Log-BiLinear
   - 相邻词向量矩阵对应的词向量表示，通过链接矩阵和当前词关联，而关联关系使用能量函数表示
3. Log-BiLinear 合并隐藏层和输出成，简化网络，训练速度提升 (Constrastiv Divergence 算法)
4. Hierarchical Log-BiLinear 限制对上下文 Context 的关系矩阵为对角矩阵
5. 启发式，随机初始化二叉树 (Bagging 思想)

- Collobert and Weston (SENNA) 模型 2008：Window 窗口



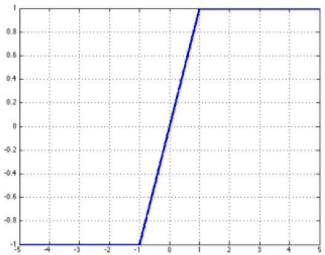
HardTanh 取代了 Tanh

- Collobert and Weston (SENNA) 模型 2008：Sentence 句子



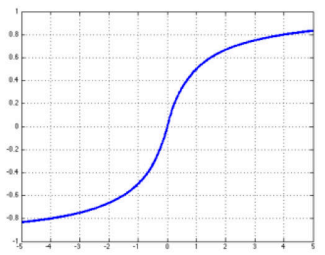
CNN 压缩句子

- HardTanh



$$\text{hardtanh}(z) = \begin{cases} -1 & : z < -1 \\ z & : -1 \le z \le 1 \\ 1 & : z > 1 \end{cases}$$

$$\text{hardtanh}'(z) = \begin{cases} 1 & : -1 \le z \le 1 \\ 0 & : \text{otherwise} \end{cases}$$



$$\text{softsign}(z) = \frac{z}{1+|z|}$$

$$\text{softsign}'(z) = \frac{\text{sgn}(z)}{(1+z)^2}$$

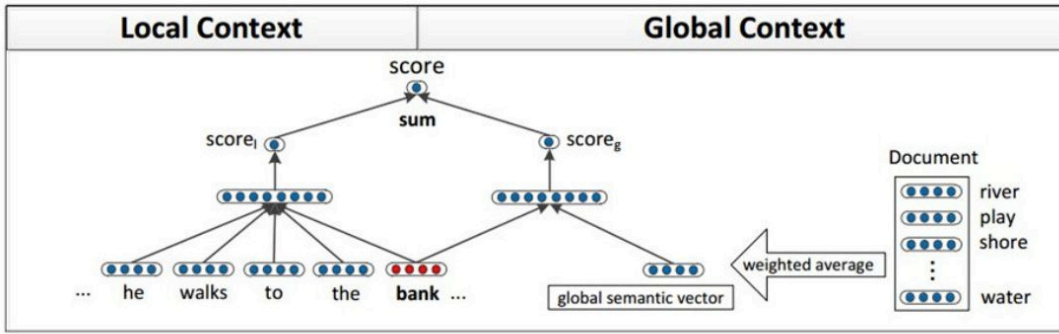
- C&W SENNA 小结

1. 专注到 NLP 应用：词性标注、命名实体识别、短语识别、语义角色标注
2. 近似计算：使用 pair-wise 的方法训练词向量

$$\sum_{x\in\mathfrak{X}}\sum_{w\in\mathfrak{D}}\max\{0,1-f(x)+f(x^{(w)})\}$$

3. 输出层只有一个节点，而不是 NNLM 的 |V| 个节点，降低计算复杂度
4. 大写开头的单词和小写单词当作同一个词处理
5. 维基百科英文语料和路透社语料中一共训练了 7 周

- Eric Huang's Model

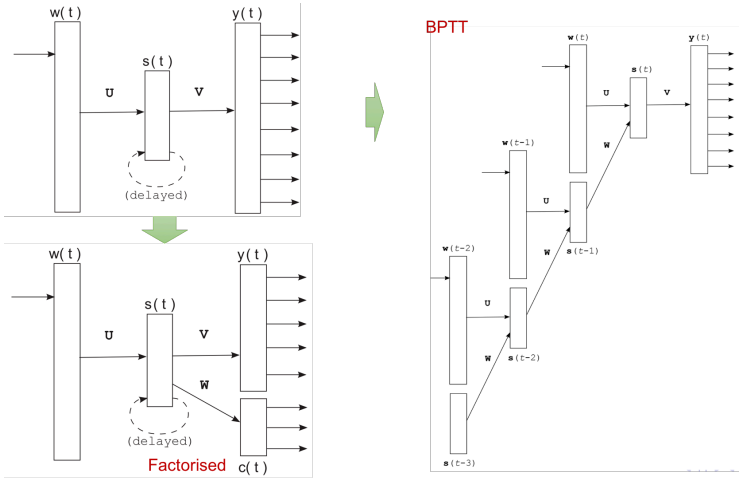


- Clustering word base on context and retrain the model

- Eric Huang's Model 小结

1. 基于 C&W 的基础的改进 (pair-wise 目标函数)
2. 使用全文信息辅助已有的局部信息
   - 全局信息：文章中所有词的词向量求个加权平均 (权重是词的 idf)
3. 使用多个词向量来表示多义词
   - 词的上下文向量：各 5 个词 (共 10 个词) 的词向量做加权平均
   - 对上下文向量做 k-means 聚类，根据聚类结果给每个词打上标签
   - 重新训练词向量

- RNN-LM：Mikolov 2010



- RNN-LM 模型

$$x(t) = w(t) + s(t-1) \tag{1}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{2}$$

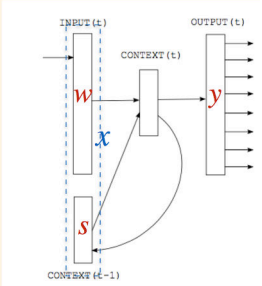
$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right) \tag{3}$$

where $f(z)$ is sigmoid activation function:

$$f(z) = \frac{1}{1+e^{-z}} \tag{4}$$

and $g(z)$ is softmax function:

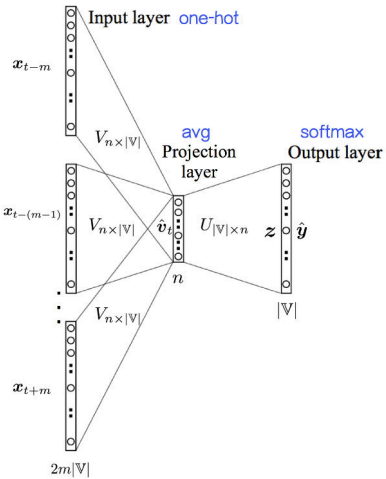
$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{5}$$



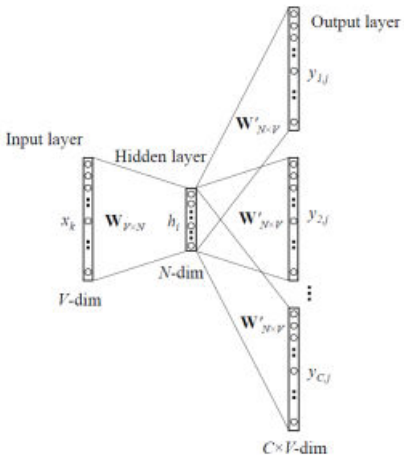
- RNN-LM 小结

1. RNN 可以解决 Long-Term Dependency 问题
2. RNN 是深度网络可以发现更多词与词的 Pattern
3. RNN 让训练变得复杂，训练困难
4. RNN-LM 采用分组计算 Softmax
5. RNN-LM 隐藏层采用了 Sigmoid，均值不在 0 上下

- CBOW (Continuous Bag of words)：给定周围词，预测中心词



- Skip-Gram：给定词，预测周围词 (随机选一个周围词)



- CBOW

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

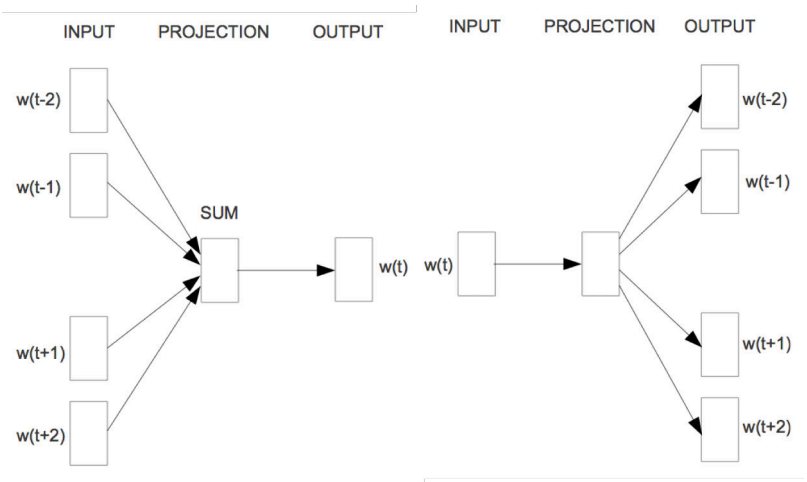
$$p(w_t \mid w_{t-1}, \cdots, w_{t-n+1}) = \frac{\exp(h^\top v'_{w_t})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

- Skip-Gram

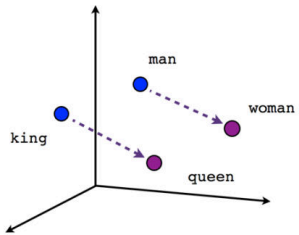
$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \le j \le n, \neq 0} \log p(w_{t+j} \mid w_t)$$

$$p(w_{t+j} \mid w_t) = \frac{\exp(h^\top v'_{w_{t+j}})}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

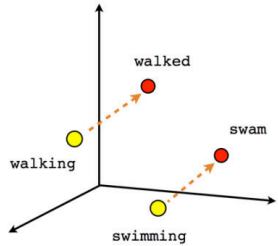
- Word2Vec


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)

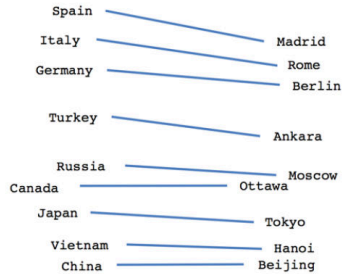
- Word2Vec 效果



Male-Female



Verb tense



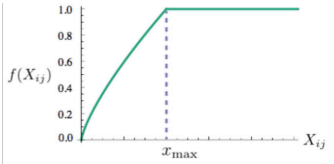
Country-Capital

- Word2Vec 小结

1. CBOW 和 Skip-Gram 基本就是 Softmax 网络
2. 目标是负的 Log 损失 (交互熵)
3. 中间是线性单元，而不是 Tanh
4. 输入是上下文（周围词），而不是上文
5. 计算速度快，容易重现
6. 近似计算 Softmax
    - Hierarchical Softmax （同 Hierarchical NNLM）
    - Noise Constractive Estimation
    - Negative Sampling (简化自 NCE)

- Glove（Global Vectors for word representation)：Jeffrey 和 Socher

Probability and Ratio	k = *solid*	k = *gas*	k = *water*	k = *fashion*
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

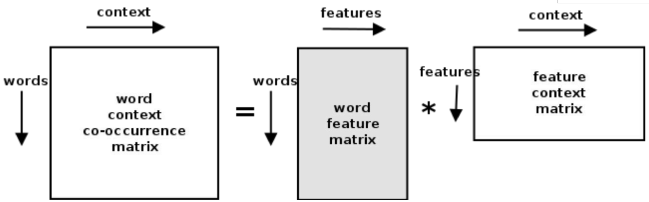


共现权重

$$\sum_{i,j=1}^{V} f(X_{i,j})(u_i^T v_j + b_i + c_j - \log X_{i,j})^2$$

加权最小二乘目标

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$



- Glove 目标定义

加权最小二乘目标：

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ik} \right)^2$$

定义条件概率（i, j 的词频）：

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$$

条件概率比值（给定k后，i, j 的条件概率比）：

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

假设存在词向量**加性**关系：

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

假设存在词向量**乘性**关系：

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

并且**统一函数**关系：

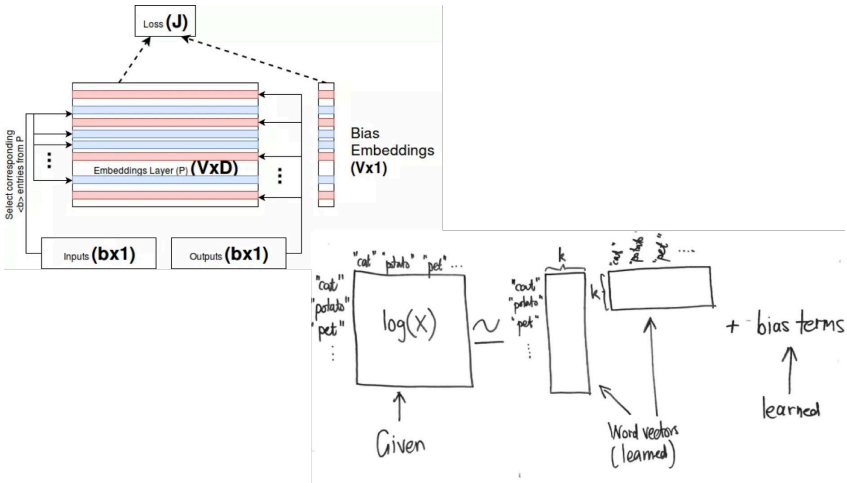
$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

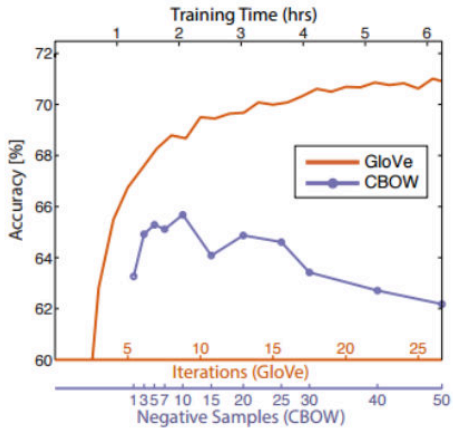
统一函数为指数函数：

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}$$

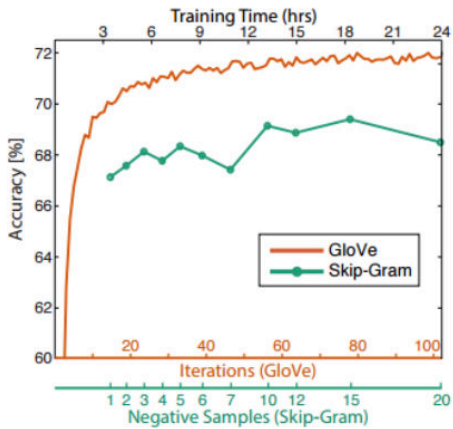
- Glove


Loss (J)
Select corresponding <b> entries from P
Embeddings Layer (P) (VxD)
Bias Embeddings (Vx1)
Inputs (bx1)
Outputs (bx1)
"cat" "potato" "pet" ...
"cat"
"potato"
"pet"
log(X)
k
+ bias terms
learned
Given
Word vectors (learned)

- Glove vs Word2Vec

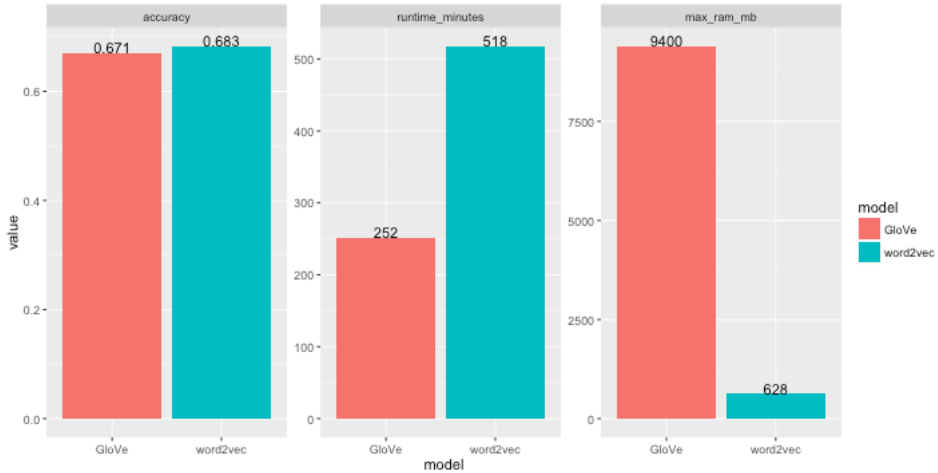


(a) GloVe vs CBOW



(b) GloVe vs Skip-Gram

- Glove vs Word2Vec

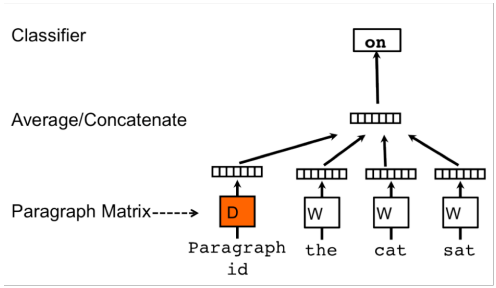


- Glove 小结

1. 是 LSI 和 Log Bi-Linear 的回归的融合
2. 吸收了 Huang's Model 全局思想，分为 Globall Matrix Factorization 和 Local Context Window
3. 引入共现权重
4. 训练速度比 Word2Vec 要快，效果不比 Word2Vec 差，就是内存占用高于 Word2Vec
5. 在 word analogies，word similarity，NER 任务中取得不错的效果

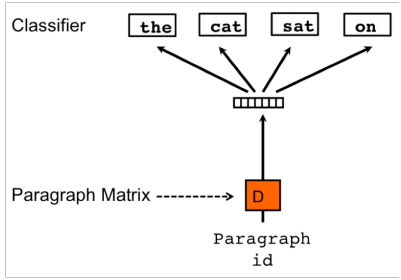
- Paragraph2Vec

Paragraph Vector



PV-DM

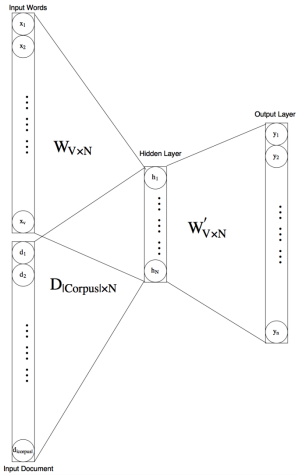
Distributed Memory



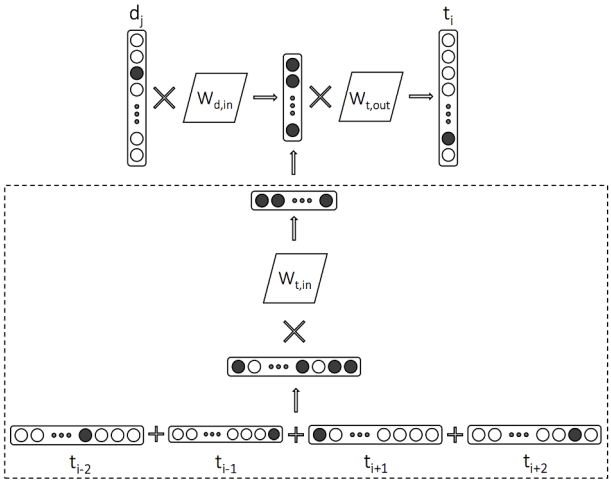
PV-DBOW

Distributed Bag of Words

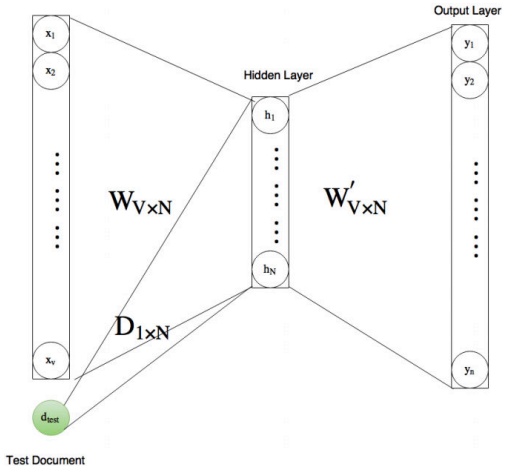
- Paragraph2Vec 训练



- Paragraph2Vec 窗口操作



- Paragraph2Vec 推理



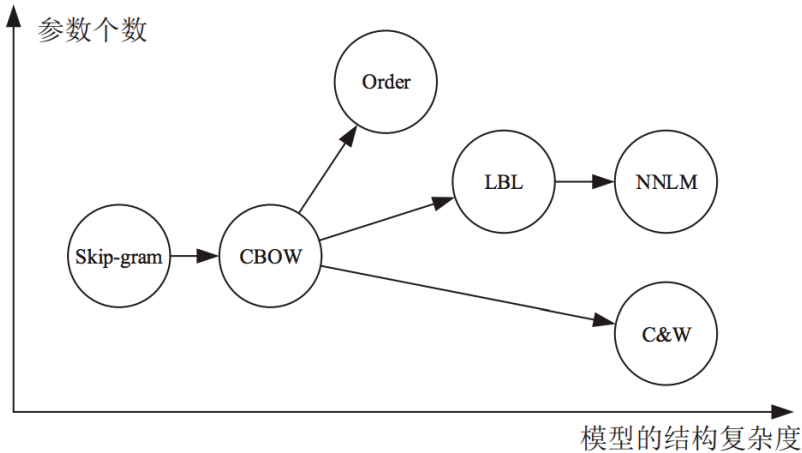
- Paragraph2Vec 对比

Model	Error rate
Vector Averaging	10.25%
Bag-of-words	8.10 %
Bag-of-bigrams	7.28 %
Weighted Bag-of-bigrams	5.67%
Paragraph Vector	**3.82%**

- Paragraph2Vec 小结

1. Paragraph2Vec 保留了部分词的顺序和语义空间
   - 固定窗口输入
   - 滑动窗口到整个文本
   - Paragraph 共享一个 ID，类似 Bagging
2. 拼接处理效果优于求平均
3. PV-DM 的效果要优于 PV-DBOW
4. 效果略微优于 n-gram 的 BoW 模型

- 各种模型的比较



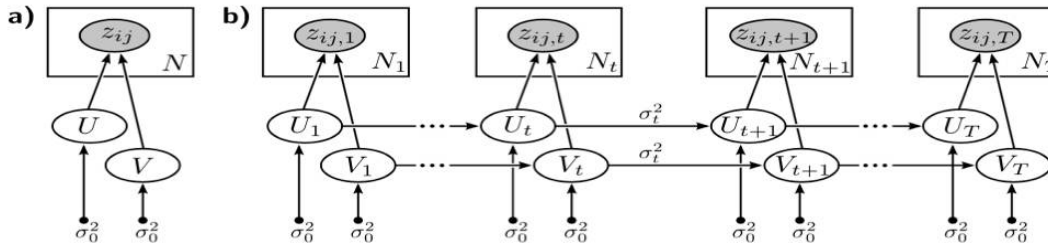
- 各种模型的比较

名称	上下文	上下文与目标词之间的建模（技术手段）
LSA/LSI [56]	文档	矩阵
HAL [70]	词	
GloVe [90]	词	
Jones & Mewhort [45]	n-gram	
Brown Clustering [12]	词	聚类
Skip-gram [73]	词	神经网络
CBOW [73]	n-gram（加权）	
Order （2.2.7 小节）	n-gram（线性组合）	
LBL [79]	n-gram（线性组合）	
NNLM [7]	n-gram（非线性组合）	
C&W [17]	n-gram（非线性组合）	

- Word Embedding VS Topic Model

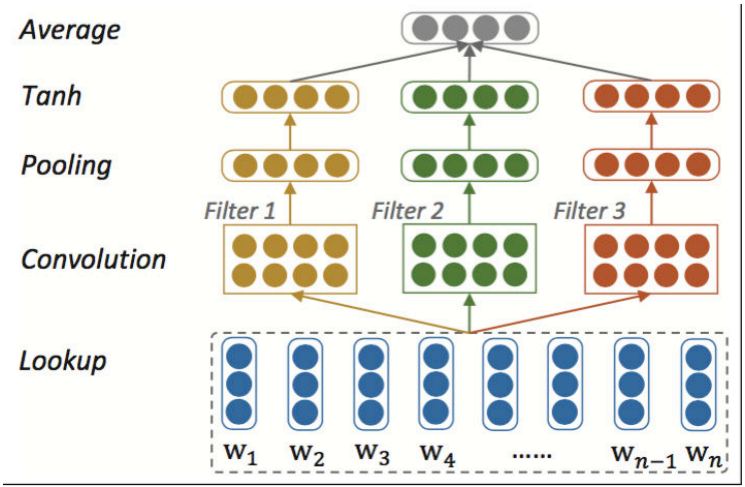
1. 词嵌入在相似度方面和词类比（word analogy）现象由于主题模型
2. 主题模型丢失了词的顺序信息
3. 主题模型的可解释性高一些，词嵌入难以解释，也没有统一的表示 (例如，概率分布)
4. 词嵌入是浅层神经网络，而主题模型是浅层概率图模型
5. 词嵌入训练太慢

- 词嵌入和概率图模型的融合



Bayesian Skip-Gram

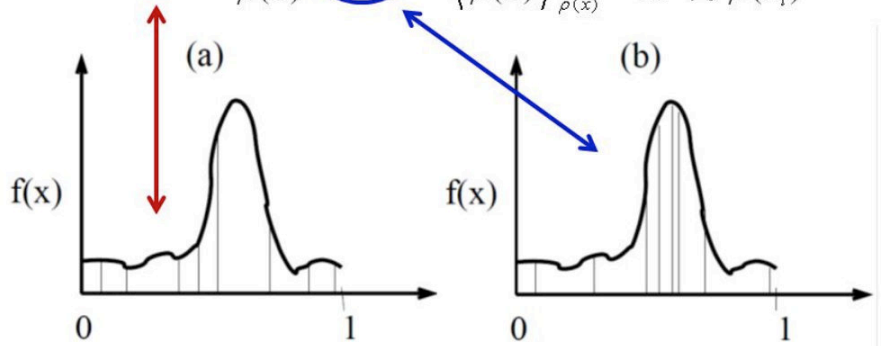
- CNN 直接词特征提取


Average
Tanh
Pooling
Filter 1
Filter 2
Filter 3
Convolution
Lookup
w_1 w_2 w_3 w_4 ..... w_{n-1} w_n

3.3 Softmax 近似计算

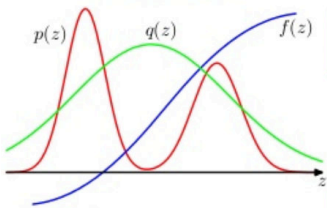
- Importance Sampling：未知分布

$$A = \int_a^b f(x)dx = \int_a^b \frac{f(x)}{\rho(x)}\rho(x)dx \equiv \left\langle \frac{f(x)}{\rho(x)} \right\rangle_{\rho(x)} \cong \frac{1}{M}\sum_{i=1}^{M} \frac{f(x_i)}{\rho(x_i)}$$



- Importance Sampling：未知分布上的期望

Basic idea:
Transform the integral into an expectation over a simple, known distribution



Conditions:
q(z) > 0 when f(z)p(z) ≠ 0
Easy to sample from q(z)

Integral
$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Proposal
$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\frac{q(\mathbf{z})}{q(\mathbf{z})}d\mathbf{z}$$

Importance weight
$$\mathbb{E}[f] = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})d\mathbf{z}$$

correct the bias introduced by sampling from a wrong distribution

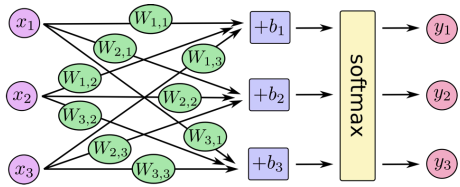
Normalized

$$w^{(s)} \propto \frac{p(\boldsymbol{z}^{(s)})}{q(\boldsymbol{z}^{(s)})} \qquad \boldsymbol{z}^{(s)} \sim q(\boldsymbol{z})$$

Monte Carlo
$$\mathbb{E}[f] = \frac{1}{S}\sum_{s} w^{(s)} f(\mathbf{z}^{(s)})$$

- All the generated samples are retained

- Softmax 求解看成是求期望



$$J_\theta = -\log \frac{\exp(h^\top v'_w)}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})}$$

$$J_\theta = -h^\top v'_w + \log \sum_{w_i \in V} \exp(h^\top v'_{w_i})$$

$$h^\top v'_w = -\mathcal{E}(w)$$

$$J_\theta = \mathcal{E}(w) + \log \sum_{w_i \in V} \exp(-\mathcal{E}(w_i))$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \nabla_\theta \log \sum_{w_i \in V} \exp(-\mathcal{E}(w_i))$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \frac{1}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \sum_{w_i \in V} \exp(-\mathcal{E}(w_i)) \nabla_\theta(-\mathcal{E}(w_i))$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \sum_{w_i \in V} \frac{\exp(-\mathcal{E}(w_i))}{\sum_{w_i \in V} \exp(-\mathcal{E}(w_i))} \nabla_\theta(-\mathcal{E}(w_i))$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) + \sum_{w_i \in V} P(w_i) \nabla_\theta(-\mathcal{E}(w_i))$$

$$\nabla_\theta J_\theta = \nabla_\theta \mathcal{E}(w) - \sum_{w_i \in V} P(w_i) \nabla_\theta \mathcal{E}(w_i)$$

$$\sum_{w_i \in V} P(w_i) \nabla_\theta \mathcal{E}(w_i) = \mathbb{E}_{w_i \sim P}[\nabla_\theta \mathcal{E}(w_i)]$$

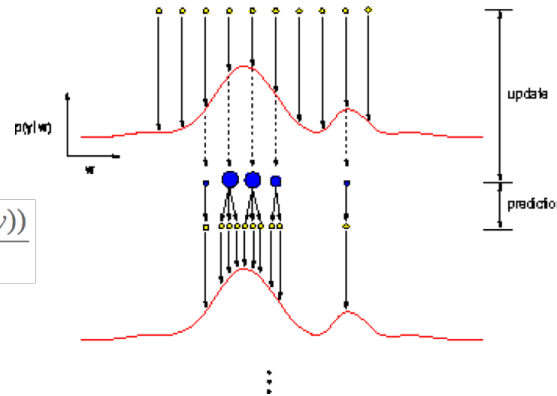
- Softmax 的 Importance Sampling 求解

$$\mathbb{E}_{w_i \sim P}[\nabla_\theta \mathcal{E}(w_i)] \approx \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \mathcal{E}(w_i)$$

$$\mathbb{E}_{w_i \sim P}[\nabla_\theta \mathcal{E}(w_i)] \approx \frac{1}{R} \sum_{i=1}^{m} r(w_i) \nabla_\theta \mathcal{E}(w_i)$$

$$r(w) = \frac{\exp(-\mathcal{E}(w))}{Q(w)}$$

$$R = \sum_{j=1}^{m} r(w_j)$$



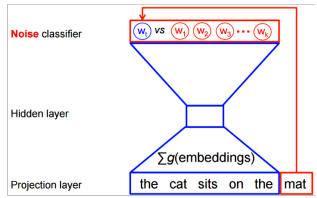
- Noise Contrastive Estimation： Gutmann and Hyvärinen 2010

noise distribution Q

$$J_\theta = -\sum_{w_i \in V}[\log P(y=1 \mid w_i, c_i) + k\, \mathbb{E}_{\tilde{w}_{ik} \sim Q}[\log P(y=0 \mid \tilde{w}_{ij}, c_i)]]$$

$$J_\theta = -\sum_{w_i \in V}[\log P(y=1 \mid w_i, c_i) + k \sum_{j=1}^{k} \frac{1}{k} \log P(y=0 \mid \tilde{w}_{ij}, c_i)]$$

$$J_\theta = -\sum_{w_i \in V}[\log P(y=1 \mid w_i, c_i) + \sum_{j=1}^{k} \log P(y=0 \mid \tilde{w}_{ij}, c_i)]$$



$$P(y, w \mid c) = \frac{1}{k+1} P_{\text{train}}(w \mid c) + \frac{k}{k+1} Q(w) \qquad P(y=1 \mid w, c) = \frac{\frac{1}{k+1} P_{\text{train}}(w \mid c)}{\frac{1}{k+1} P_{\text{train}}(w \mid c) + \frac{k}{k+1} Q(w)}$$

$$P(y=1 \mid w, c) = \frac{P_{\text{train}}(w \mid c)}{P_{\text{train}}(w \mid c) + k\, Q(w)}$$

$$P(y=1 \mid w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + k\, Q(w)}$$

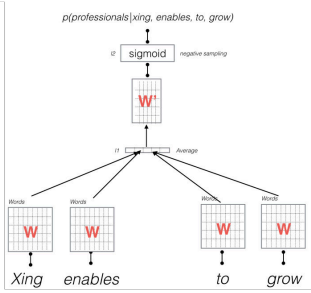
$$P(w \mid c) = \frac{\exp(h^\top v'_w)}{\sum_{w_i \in V} \exp(h^\top v'_{w_i})} \qquad P(w \mid c) = \frac{\exp(h^\top v'_w)}{Z(c)} \qquad Z(c) = 1$$

$$P(w \mid c) = \exp(h^\top v'_w).$$

$$J_\theta = -\sum_{w_i \in V}[\log \frac{\exp(h^\top v'_{w_i})}{\exp(h^\top v'_{w_i}) + k\, Q(w_i)} + \sum_{j=1}^{k} \log (1 - \frac{\exp(h^\top v'_{\tilde{w}_{ij}})}{\exp(h^\top v'_{\tilde{w}_{ij}}) + k\, Q(\tilde{w}_{ij})})]$$

- Negative Sampling：Mikolov 2013

$$J_\theta = -\sum_{w_i \in V}[\log \frac{\exp(h^\top v'_{w_i})}{\exp(h^\top v'_{w_i}) + k\, Q(w_i)} + \sum_{j=1}^{k} \log\,(1 - \frac{\exp(h^\top v'_{\tilde{w}_{ij}})}{\exp(h^\top v'_{\tilde{w}_{ij}}) + k\, Q(\tilde{w}_{ij})})]$$



$$P(y=1 \mid w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + k\, Q(w)} \qquad k\,Q(w) = 1$$

$$P(y=1 \mid w, c) = \frac{\exp(h^\top v'_w)}{\exp(h^\top v'_w) + 1} \qquad k = |V|$$

$$P(y=1 \mid w, c) = \frac{1}{1 + \exp(-h^\top v'_w)}$$

$$J_\theta = -\sum_{w_i \in V}[\log \frac{1}{1 + \exp(-h^\top v'_{w_i})} + \sum_{j=1}^{k} \log\,(1 - \frac{1}{1 + \exp(-h^\top v'_{\tilde{w}_{ij}})}]$$

$$J_\theta = -\sum_{w_i \in V}[\log \frac{1}{1 + \exp(-h^\top v'_{w_i})} + \sum_{j=1}^{k} \log\,(\frac{1}{1 + \exp(h^\top v'_{\tilde{w}_{ij}})}]$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

$$J_\theta = -\sum_{w_i \in V}[\log \sigma(h^\top v'_{w_i}) + \sum_{j=1}^{k} \log \sigma(-h^\top v'_{\tilde{w}_{ij}})]$$

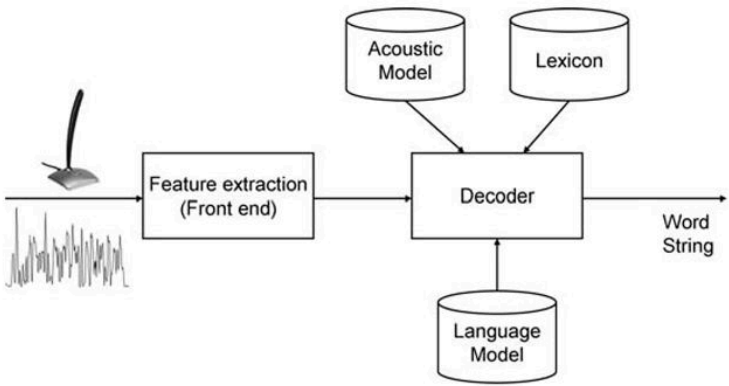
- Softmax 近似小结

1. NCE 要比 IS 稳定：NCE 利用了 2 分类结果
2. NS 是 NCE 一种近似

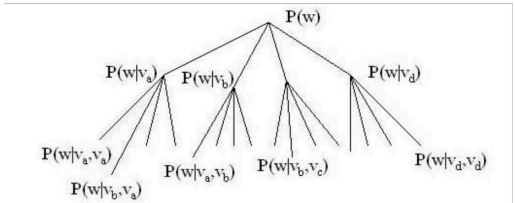
4 语言模型

4.1 从概率模型到神经网络模型

- 语言模型



- N-Gram 概率模型



Word-level unigrams

Text	Token Sequence	Token Value
One Two Three Four	1	One
One Two Three Four	2	Two
One Two Three Four	3	Three
One Two Three Four	4	Four

Word-level bigrams

Text	Token Sequence	Token Value
One Two Three Four	1	One Two
One Two Three Four	2	Two Three
One Two Three Four	3	Three Four

Word-level trigrams

Text	Token Sequence	Token Value
One Two Three Four	1	One Two Three
One Two Three Four	2	Two Three Four

$$\hat{p}(w_a) = \frac{c(w_a)}{N}$$

$$\hat{p}(w_b|w_a) = \frac{c(w_a, w_b)}{\Sigma_{w_b} c(w_a, w_b)} \approx \frac{c(w_a, w_b)}{c(w_a)}$$

- N-Gram 概率模型

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

$P(w_{i+1}=\text{of} \mid w_i=\text{tired}) = 1$
$P(w_{i+1}=\text{of} \mid w_i=\text{use}) = 1$
$P(w_{i+1}=\text{sister} \mid w_i=\text{her}) = 1$
$P(w_{i+1}=\text{beginning} \mid w_i=\text{was}) = 1/2$
$P(w_{i+1}=\text{reading} \mid w_i=\text{was}) = 1/2$

$P(w_{i+1}=\text{bank} \mid w_i=\text{the}) = 1/3$
$P(w_{i+1}=\text{book} \mid w_i=\text{the}) = 1/3$
$P(w_{i+1}=\text{use} \mid w_i=\text{the}) = 1/3$

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

- 语言模型的评价: 困惑度 Perplexity

熵(Entropy)的计算：

$$H = -\lim_{m\to\infty}\frac{1}{m}\sum_{w_1,w_2,\ldots,w_m}(P(w_1,w_2,\ldots,w_m)\ \log_2 P(w_1,w_2,\ldots,w_m))$$

所有情况（全概率公式）：

$$H = -\lim_{m\to\infty}\frac{1}{m}\log_2 P(w_1,w_2,\ldots,w_m)$$

有限近似：

$$\hat{H} = -\frac{1}{m}\log_2 P(w_1,w_2,\ldots,w_m)$$

困惑度(Perplexity)的计算：

$$PP = \hat{P}(w_1,w_2,\ldots,w_m)^{-\frac{1}{m}}$$

困惑度和熵的关系：

$$PP = 2^{\hat{H}}$$

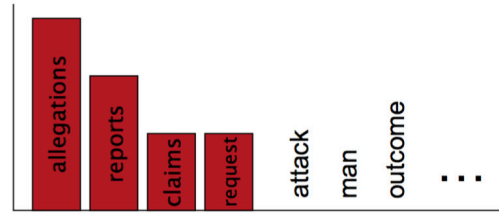
- N-Gram 模型问题

1. 概率为 0 的问题
    - Laplace (Add-One) Smoothing
    - Good-Turing
    - Interpolation
    - Backoff
    - Kneser-Ney
2. 马尔可夫假设不满足问题
    - N-Gram 内在问题
    - Long-Term Dependency 问题

- 概率为 0 的问题

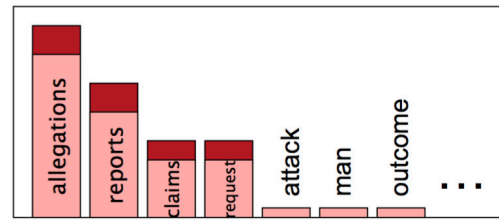
  - When we have sparse statistics:

    P(w | denied the)
    3 allegations
    2 reports
    1 claims
    1 request
    7 total



  - Steal probability mass to generalize better

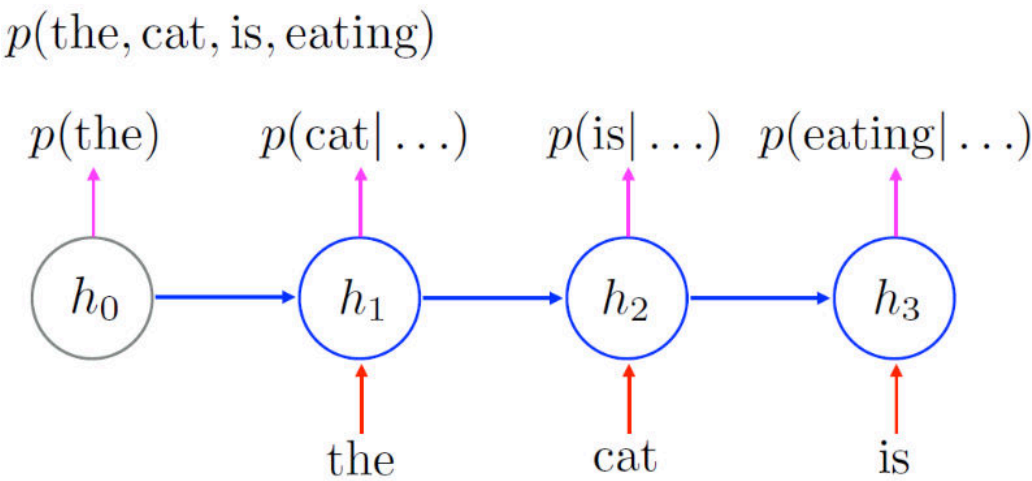
    P(w | denied the)
    2.5 allegations
    1.5 reports
    0.5 claims
    0.5 request
    2 other
    7 total



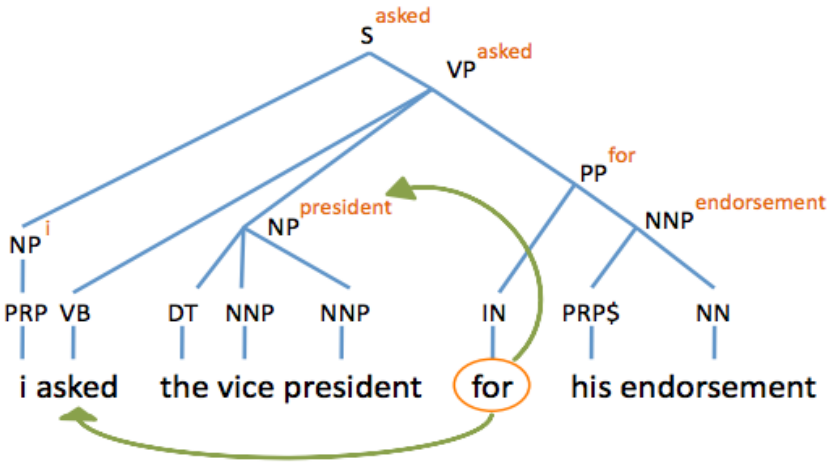
- Smoothing: 概率为 0 的问题

1. Lapalace (add-one, add-α) [unseen words same probability]
2. Deleted (held-out) estimation [things-we-saw-once]
3. Good Turing [things new, assuming leave-one-out]
4. Back-off (Katz) smoothing [non-linear zero-count]
5. Interpolation (Jelinek-Mercer) smoothing [linear zero-count]
6. Witten-Bell [interpolation, diversity of predicted word]
7. Absolute discounting [approximating Good-Turing]
8. Kneser-Ney [back-off, diversity of history]
9. Modified Kneser-Ney [interpolated back-off]

- 马尔可夫假设：N-Gram 模型



- 长距离依赖问题 Long-Term Dependency



- Log-Linear 语言模型

$$\hat{p}(y \mid x) \stackrel{\text{def}}{=} \frac{\text{count}(x,y)}{\text{count}(x)}$$

$$u(x,y) \stackrel{\text{def}}{=} \exp \sum_{k=1}^{K} (\theta_k \cdot f_k(x,y)) = \exp\left(\vec{\theta} \cdot \vec{f}(x,y)\right) > 0$$

$$Z(x) \stackrel{\text{def}}{=} \sum_{y} u(x,y)$$

$$\hat{p}(y \mid x) \stackrel{\text{def}}{=} \frac{u(x,y)}{Z(x)}$$

$$p(y \mid x) \stackrel{\text{def}}{=} \frac{1}{Z(x)} \exp \sum_{k=1}^{K} \theta_k \cdot f_k(x,y)$$

$$\sum_y p(y \mid x) = 1$$

$$p(y \mid x) \stackrel{\text{def}}{=} \frac{1}{Z(x)} \prod_{\substack{k \text{ such that} \\ f_k(x,y)=1}} (\exp \theta_k)$$

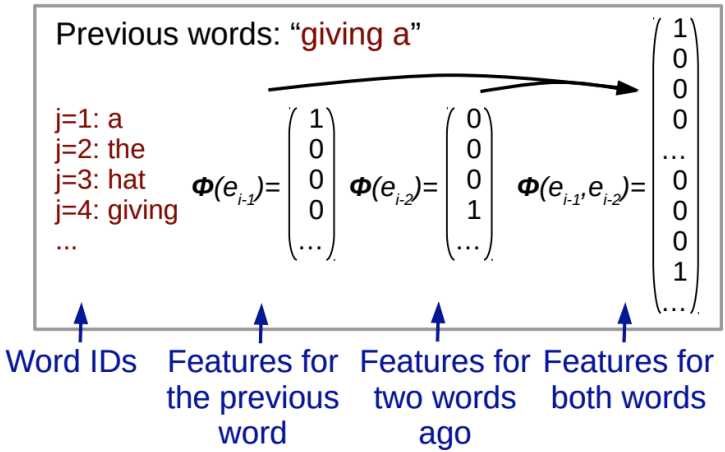
$$p(\text{solid circle}) \stackrel{\text{def}}{=} \frac{\exp(\theta_{\text{circle}} + \theta_{\text{solid}})}{\exp(\theta_{\text{circle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{circle}} + \theta_{\text{striped}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{striped}})}$$

$$p(\text{striped circle}) \stackrel{\text{def}}{=} \frac{\exp(\theta_{\text{circle}} + \theta_{\text{striped}})}{\exp(\theta_{\text{circle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{circle}} + \theta_{\text{striped}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{striped}})}$$

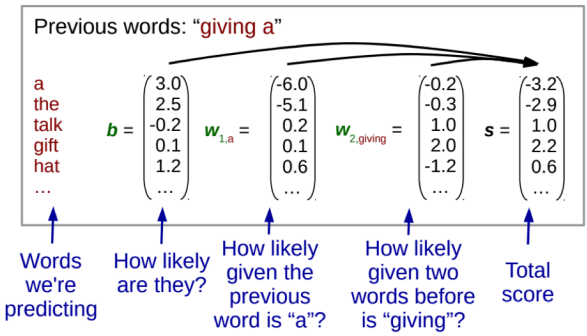
$$p(\text{solid triangle}) \stackrel{\text{def}}{=} \frac{\exp(\theta_{\text{triangle}} + \theta_{\text{solid}})}{\exp(\theta_{\text{circle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{circle}} + \theta_{\text{striped}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{striped}})}$$

$$p(\text{striped triangle}) \stackrel{\text{def}}{=} \frac{\exp(\theta_{\text{triangle}} + \theta_{\text{striped}})}{\exp(\theta_{\text{circle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{circle}} + \theta_{\text{striped}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{solid}}) + \exp(\theta_{\text{triangle}} + \theta_{\text{striped}})}$$

- Log-Linear 特征向量



- Log-Linear 到 Softmax 激活



$$\boldsymbol{s} = W\boldsymbol{x} + \boldsymbol{b}.$$

$$\boldsymbol{s} = \sum_{\{j:x_j!=0\}} W_{\cdot,j}x_j + \boldsymbol{b}$$

$$\boldsymbol{p} = \text{softmax}(\boldsymbol{s})$$

$$p_j = \frac{\exp(s_j)}{\sum_{\tilde{j}} \exp(s_{\tilde{j}})}$$

- Log-Linear 优化更新

$$
\begin{aligned}
\boldsymbol{x} &= \boldsymbol{\phi}(e_{t-m+1}^{t-1}) \\
\boldsymbol{s} &= \sum_{\{j:x_j!=0\}} W_{\cdot,j} x_j + \boldsymbol{b} \\
\boldsymbol{p} &= \text{softmax}(\boldsymbol{s}) \\
\ell &= -\log \boldsymbol{p}_{e_t}.
\end{aligned}
$$

$$
\begin{aligned}
\frac{d\ell(e_{t-n+1}^{t}, W, \boldsymbol{b})}{d\boldsymbol{b}} &= \frac{d\ell}{d\boldsymbol{p}} \frac{d\boldsymbol{p}}{d\boldsymbol{s}} \frac{d\boldsymbol{s}}{d\boldsymbol{b}} \\
\frac{d\ell(e_{t-n+1}^{t}, W, \boldsymbol{b})}{dW_{\cdot,j}} &= \frac{d\ell}{d\boldsymbol{p}} \frac{d\boldsymbol{p}}{d\boldsymbol{s}} \frac{d\boldsymbol{s}}{dW_{\cdot,j}}
\end{aligned}
$$

$$
\begin{aligned}
\frac{d\ell(e_{t-n+1}^{t}, W, \boldsymbol{b})}{d\boldsymbol{b}} &= \boldsymbol{p} - \text{onehot}(e_t) \\
\frac{d\ell(e_{t-n+1}^{t}, W, \boldsymbol{b})}{dW_{\cdot,j}} &= x_j(\boldsymbol{p} - \text{onehot}(e_t))
\end{aligned}
$$

- Log-Linear 优化更新推导

$$P(y \mid x, \mathbf{W}) = \frac{e^{\sum_k \mathbf{W}_k \phi_k(x,y)}}{\sum_{y' \in \mathcal{Y}} e^{\sum_k \mathbf{W}_k \phi_k(x,y')}}$$

$$\log P(y \mid x, \mathbf{W}) = \underbrace{\mathbf{W} \cdot \phi(x,y)}_{\text{Linear term}} - \underbrace{\log \sum_{y' \in \mathcal{Y}} e^{\mathbf{W} \cdot \phi(x,y')}}_{\text{Normalization term}}$$

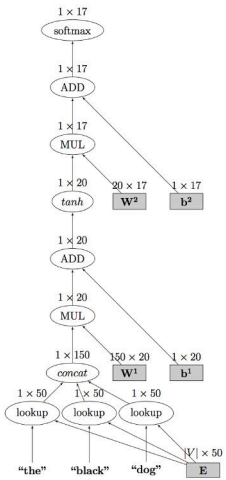
$$\begin{aligned} L(\mathbf{W}) &= \sum_{i=1}^{n} \log P(y_i \mid x_i) \\ &= \sum_{i=1}^{n} \mathbf{W} \cdot \phi(x_i, y_i) - \sum_{i=1}^{n} \log \sum_{y' \in \mathcal{Y}} e^{\mathbf{W} \cdot \phi(x_i, y')} \end{aligned}$$

$$\begin{aligned} \left.\frac{dL}{d\mathbf{W}}\right|_{\mathbf{W}} &= \sum_{i=1}^{n} \phi(x_i, y_i) - \sum_{i=1}^{n} \frac{\sum_{y' \in \mathcal{Y}} \phi(x_i, y') e^{\mathbf{W} \cdot \phi(x_i, y')}}{\sum_{z' \in \mathcal{Y}} e^{\mathbf{W} \cdot \phi(x_i, z')}} \\ &= \sum_{i=1}^{n} \phi(x_i, y_i) - \sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \phi(x_i, y') \frac{e^{\mathbf{W} \cdot \phi(x_i, y')}}{\sum_{z' \in \mathcal{Y}} e^{\mathbf{W} \cdot \phi(x_i, z')}} \\ &= \underbrace{\sum_{i=1}^{n} \phi(x_i, y_i)}_{\text{Empirical counts}} - \underbrace{\sum_{i=1}^{n} \sum_{y' \in \mathcal{Y}} \phi(x_i, y') P(y' \mid x_i, \mathbf{W})}_{\text{Expected counts}} \end{aligned}$$

$$\Delta = \left.\frac{dL}{d\mathbf{W}}\right|_{\mathbf{W}}$$

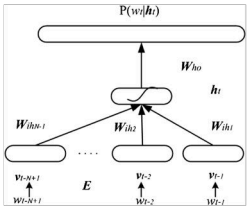
$$\mathbf{W} \leftarrow \mathbf{W} + \beta_* \Delta$$

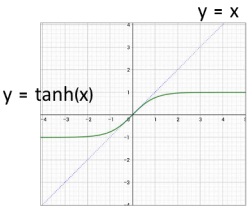
- NNLM: 加 tanh(x) 截断的 Log-Linear



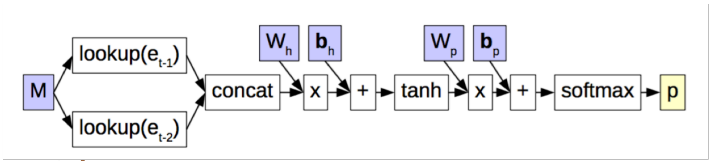
$$
\begin{aligned}
\boldsymbol{h}' &= W_{xh}\boldsymbol{x} + \boldsymbol{b}_h \\
\boldsymbol{h} &= \tanh(\boldsymbol{h}') \\
y &= \boldsymbol{w}_{hy}\boldsymbol{h} + b_y
\end{aligned}
$$

$$
\boldsymbol{p} = \mathrm{softmax}(y)
$$





- NNLM (trigram)：从加和变成拼接



tri-gram

NNLM

Log-Linear

$$\boldsymbol{s} = W^{(1)}_{\cdot,e_{t-1}} + W^{(2)}_{\cdot,e_{t-2}} + \boldsymbol{b}$$
$$\boldsymbol{p} = \text{softmax}(\boldsymbol{s}),$$

$$\boldsymbol{m} = \text{concat}(M_{\cdot,e_{t-2}}, M_{\cdot,e_{t-1}})$$
$$\boldsymbol{h} = \tanh(W_{mh}\boldsymbol{m} + \boldsymbol{b}_h)$$
$$\boldsymbol{s} = W_{hs}\boldsymbol{h} + \boldsymbol{b}_s$$
$$\boldsymbol{p} = \text{softmax}(\boldsymbol{s})$$

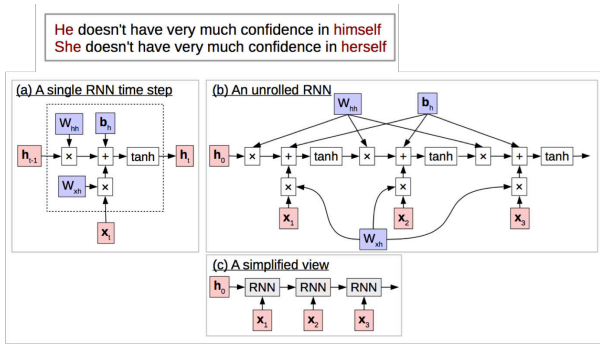
- NNLM 小结

1. 吸收了 Log-Linear 的优点，可以跳跃周边词汇
   - 让上下文表达变得灵活
   - 更强的泛化能力
2. 加和变成拼接，减少信息丢失
   - 灵活的组合方式
3. 依然对 N 有限制
   - 无法彻底解决长距离依赖问题

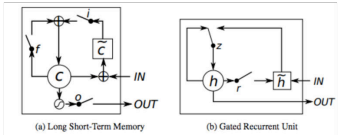
4.2 从神经网络模型到 RNN 模型

- RNN-LM 模型

RNN 顺序信息压缩



记忆学习



- RNN-LM 模型

上下文宽带限制

$$
\begin{aligned}
\boldsymbol{m} &= \text{concat}(M_{\cdot,e_{t-2}}, M_{\cdot,e_{t-1}}) \\
\boldsymbol{h} &= \tanh(W_{mh}\boldsymbol{m} + \boldsymbol{b}_h) \\
\boldsymbol{s} &= W_{hs}\boldsymbol{h} + \boldsymbol{b}_s \\
\boldsymbol{p} &= \text{softmax}(\boldsymbol{s})
\end{aligned}
$$

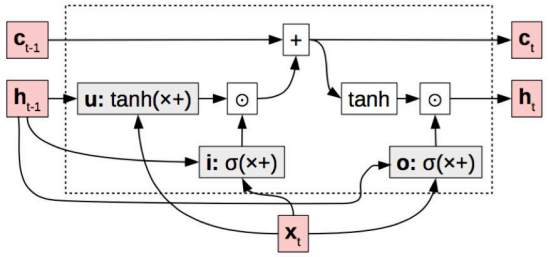
顺序压缩

$$
\begin{aligned}
\boldsymbol{m}_t &= M_{\cdot,e_{t-1}} \\
\boldsymbol{h}_t &= \begin{cases} \tanh(W_{mh}\boldsymbol{m}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) & t \geq 1, \\ \boldsymbol{0} & \text{otherwise.} \end{cases} \\
\boldsymbol{p}_t &= \text{softmax}(W_{hs}\boldsymbol{h}_t + b_s).
\end{aligned}
$$

带记忆顺序压缩

$$
\begin{aligned}
\boldsymbol{m}_t &= M_{\cdot,e_{t-1}} \\
\boldsymbol{h}_t &= \text{RNN}(\boldsymbol{m}_t, \boldsymbol{h}_{t-1}) \\
\boldsymbol{p}_t &= \text{softmax}(W_{hs}\boldsymbol{h}_t + b_s).
\end{aligned}
$$

- RNN-LM 模型：LSTM 和 GRU



$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

$$\boldsymbol{z} = \boldsymbol{x} \odot \boldsymbol{y}$$
$$z_i = x_i * y_i$$

LSTM

$$\begin{aligned}
\boldsymbol{u}_t &= \tanh(W_{xu}\boldsymbol{x}_t + W_{hu}h_{t-1} + \boldsymbol{b}_u)\\
\boldsymbol{i}_t &= \sigma(W_{xi}\boldsymbol{x}_t + W_{hi}h_{t-1} + \boldsymbol{b}_i)\\
\boldsymbol{f}_t &= \sigma(W_{xf}\boldsymbol{x}_t + W_{hf}h_{t-1} + \boldsymbol{b}_f)\\
\boldsymbol{o}_t &= \sigma(W_{xo}\boldsymbol{x}_t + W_{ho}h_{t-1} + \boldsymbol{b}_o)\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t).
\end{aligned}$$

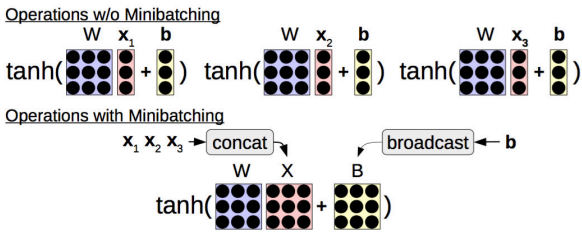
GRU

$$\begin{aligned}
\boldsymbol{r}_t &= \sigma(W_{xr}\boldsymbol{x}_t + W_{hr}h_{t-1} + \boldsymbol{b}_r)\\
\boldsymbol{z}_t &= \sigma(W_{xz}\boldsymbol{x}_t + W_{hz}h_{t-1} + \boldsymbol{b}_z)\\
\tilde{\boldsymbol{h}}_t &= \tanh(W_{xh}\boldsymbol{x}_t + W_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h)\\
\boldsymbol{h}_t &= (1-\boldsymbol{z}_t)\boldsymbol{h}_{t-1} + \boldsymbol{z}_t\tilde{\boldsymbol{h}}_t.
\end{aligned}$$

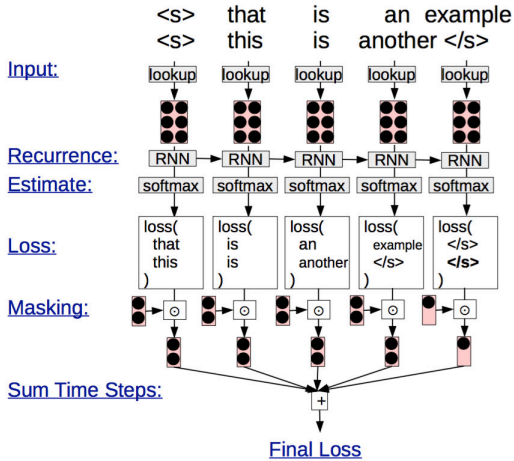
- RNN-LM 模型：批处理

Algorithm 2 A batch learning algorithm

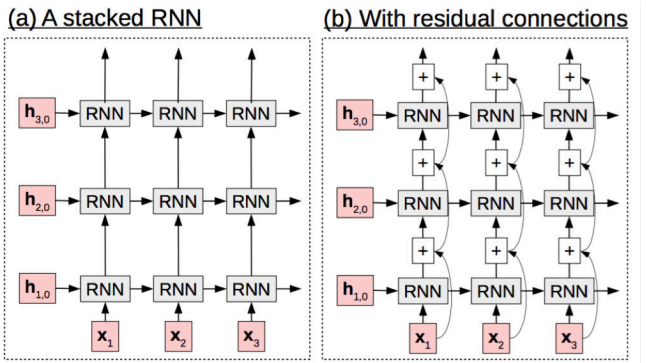
```
procedure BATCH
    for several epochs of training do
        for each training example in the data do
            Calculate and accumulate gradients of the loss
        end for
        Update the parameters according to the accumulated gradient
    end for
end procedure
```



- RNN-LM 模型：批处理


<s> that is an example
<s> this is another </s>
Input:
lookup
Recurrence:
RNN
Estimate:
softmax
Loss:
loss(that this)
loss(is is)
loss(an another)
loss(example </s>)
loss(</s> </s>)
Masking:
Sum Time Steps:
Final Loss

- 扩展 RNN-LM 模型：堆栈和残差



$$\begin{aligned}\boldsymbol{h}_{1,t} &= \mathrm{RNN}_1(\boldsymbol{x}_t, \boldsymbol{h}_{1,t-1}) \\ \boldsymbol{h}_{2,t} &= \mathrm{RNN}_2(\boldsymbol{h}_{1,t}, \boldsymbol{h}_{2,t-1}) \\ \boldsymbol{h}_{3,t} &= \mathrm{RNN}_3(\boldsymbol{h}_{2,t}, \boldsymbol{h}_{3,t-1}),\end{aligned}$$

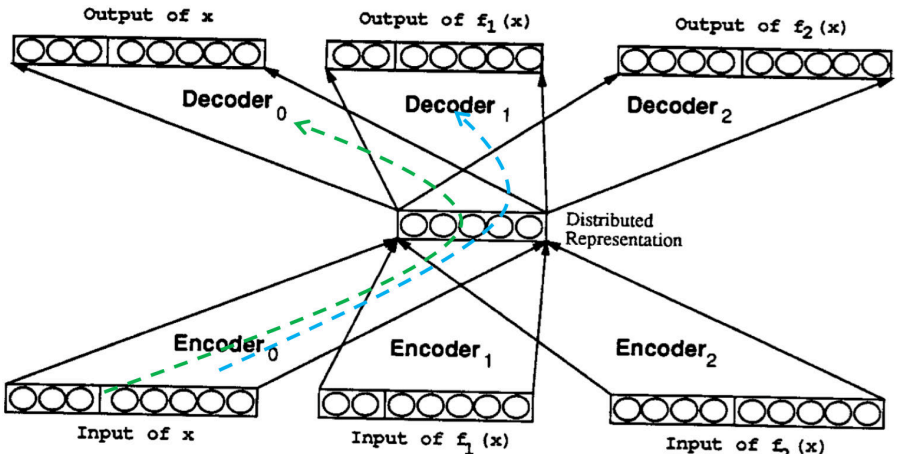
$$\begin{aligned}\boldsymbol{h}_{1,t} &= \mathrm{RNN}_1(\boldsymbol{x}_t, \boldsymbol{h}_{1,t-1}) + \boldsymbol{x}_t \\ \boldsymbol{h}_{2,t} &= \mathrm{RNN}_2(\boldsymbol{h}_{1,t}, \boldsymbol{h}_{2,t-1}) + \boldsymbol{h}_{1,t} \\ \boldsymbol{h}_{3,t} &= \mathrm{RNN}_3(\boldsymbol{h}_{2,t}, \boldsymbol{h}_{3,t-1}) + \boldsymbol{h}_{2,t}.\end{aligned}$$

- RNN-LM 小结

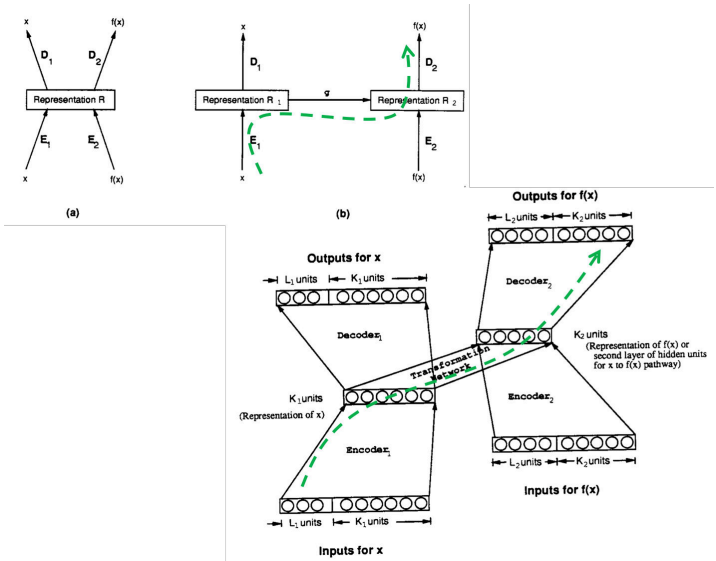
1. RNN 解决长距离依赖问题
    - 保留了 NNLM 的优点：上下文、拼接等
2. RNN 变种的各种优点
    - 记忆能力
    - 多种顺序信息压缩
3. 问题：增大了计算量

4.3 从 RNN 模型到 Encoder-Decoder 模型

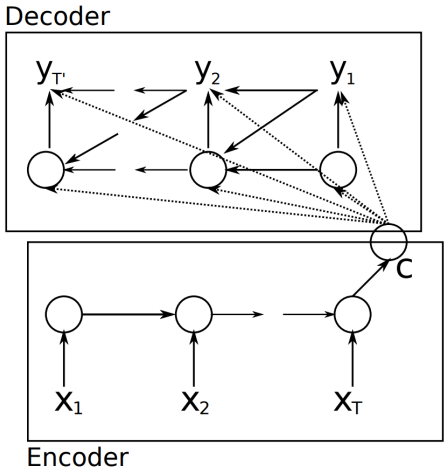
- 端到端翻译：双口 Encoder-Decoder 模型



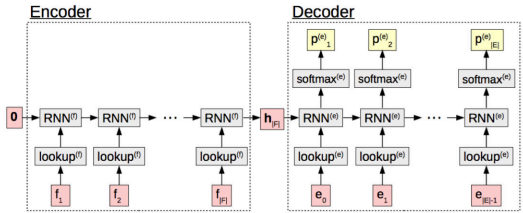
- 端到端翻译：带转换的双口 Encoder-Decoder 模型



- 端到端翻译：Encoder-Decoder 模型 + RNN 语言模型

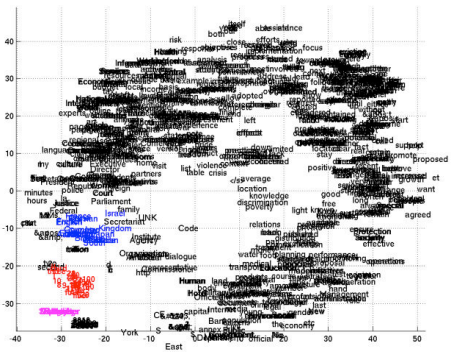


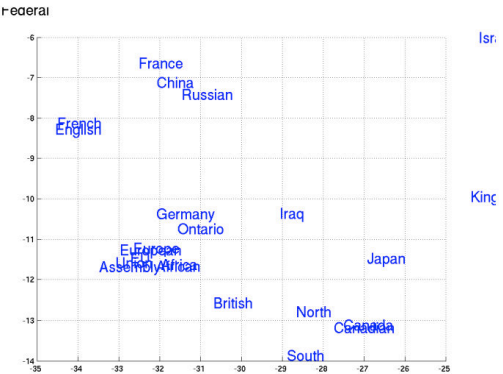
- RNN Encoder-Decoder 模型



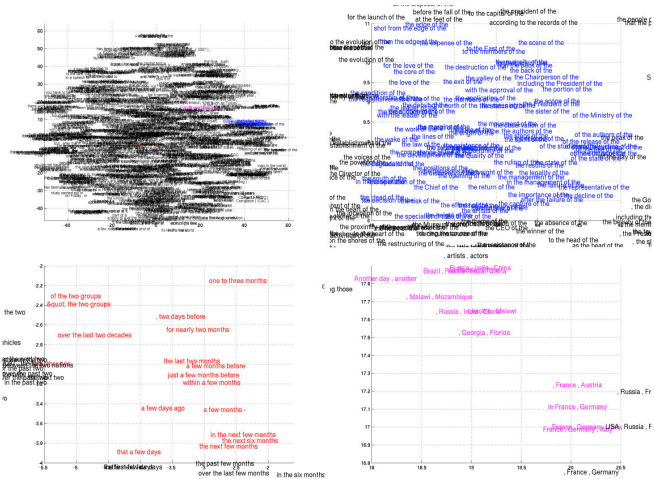
$$
\begin{aligned}
\boldsymbol{m}_t^{(f)} &= M_{\cdot,f_t}^{(f)} \\
\boldsymbol{h}_t^{(f)} &= \begin{cases} \mathrm{RNN}^{(f)}(\boldsymbol{m}_t^{(f)}, \boldsymbol{h}_{t-1}^{(f)}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases} \\
\boldsymbol{m}_t^{(e)} &= M_{\cdot,e_{t-1}}^{(e)} \\
\boldsymbol{h}_t^{(e)} &= \begin{cases} \mathrm{RNN}^{(e)}(\boldsymbol{m}_t^{(e)}, \boldsymbol{h}_{t-1}^{(e)}) & t \geq 1, \\ \boldsymbol{h}_{|F|}^{(f)} & \text{otherwise.} \end{cases} \\
\boldsymbol{p}_t^{(e)} &= \mathrm{softmax}(W_{hs}\boldsymbol{h}_t^{(e)} + b_s)
\end{aligned}
$$

- RNN Encoder-Decoder 自带词嵌入效果

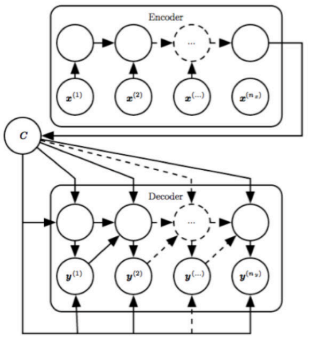


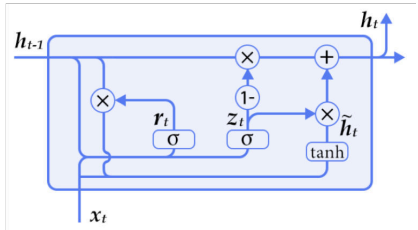


- RNN Encoder-Decoder 自带词嵌入效果



- GRU Encoder–Decoder





$$\mathbf{c} = \tanh\left(\mathbf{V}\mathbf{h}^{\langle N\rangle}\right)$$

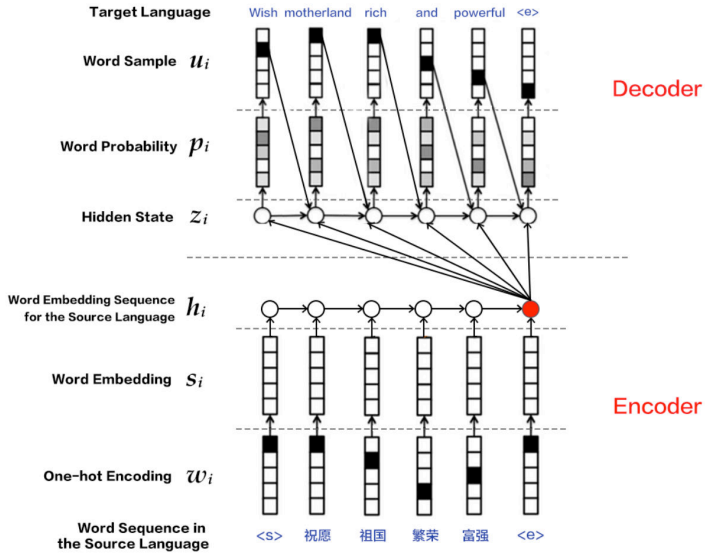
$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1 - z_j)\tilde{h}_j^{\langle t\rangle}$$

$$\begin{aligned}
\tilde{h}_j^{\langle t\rangle} &= \tanh\left(\left[\mathbf{W}e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right), \\
z_j &= \sigma\left(\left[\mathbf{W}_z e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right), \\
r_j &= \sigma\left(\left[\mathbf{W}_r e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right).
\end{aligned}$$

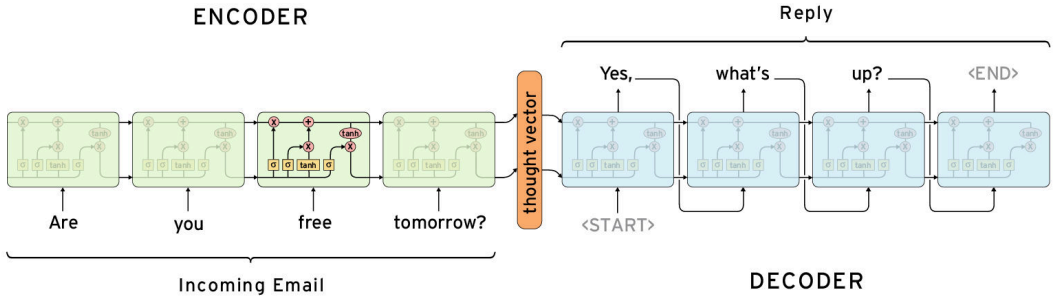
$$h'^{\langle t\rangle}_j = z'_j h'^{\langle t-1\rangle}_j + (1 - z'_j)\tilde{h}'^{\langle t\rangle}_j$$

$$\begin{aligned}
\tilde{h}'^{\langle t\rangle}_j &= \tanh\left(\left[\mathbf{W}'e(\mathbf{y}_{t-1})\right]_j + r'_j\left[\mathbf{U}'\mathbf{h}'_{\langle t-1\rangle} + \mathbf{C}\mathbf{c}\right]\right) \\
z'_j &= \sigma\left(\left[\mathbf{W}'_z e(\mathbf{y}_{t-1})\right]_j + \left[\mathbf{U}'_z\mathbf{h}'_{\langle t-1\rangle}\right]_j + \left[\mathbf{C}_z\mathbf{c}\right]_j\right) \\
r'_j &= \sigma\left(\left[\mathbf{W}'_r e(\mathbf{y}_{t-1})\right]_j + \left[\mathbf{U}'_r\mathbf{h}'_{\langle t-1\rangle}\right]_j + \left[\mathbf{C}_r\mathbf{c}\right]_j\right)
\end{aligned}$$

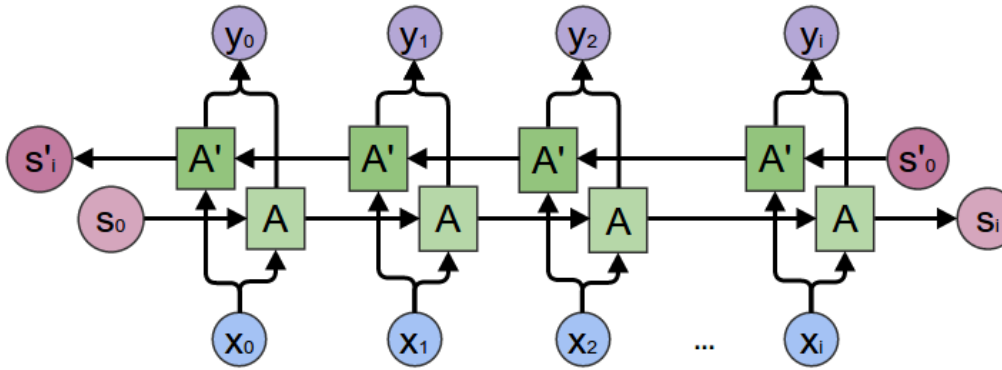
- RNN Encoder–Decoder：端到端翻译



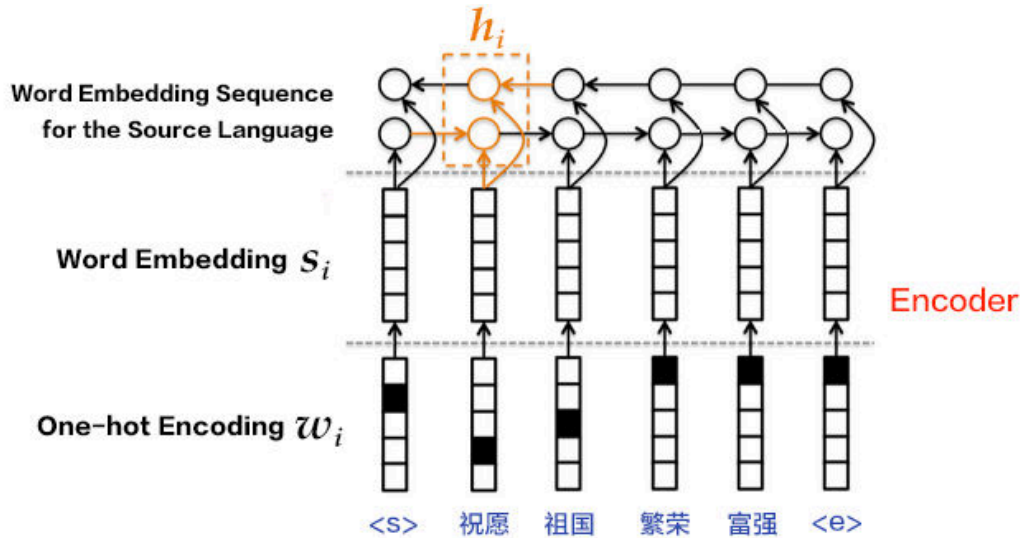
- LSTM Encoder–Decoder：端到端问答！



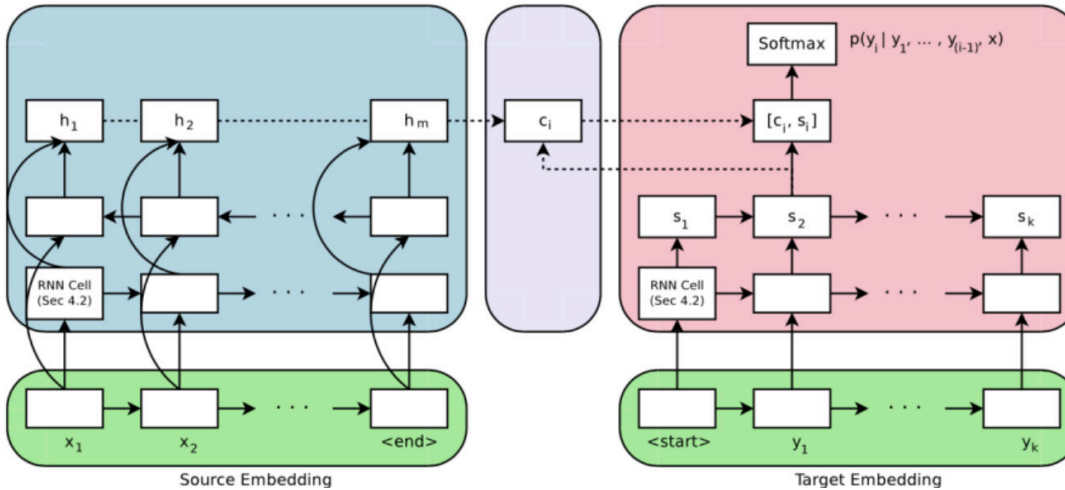
- 双向 RNN



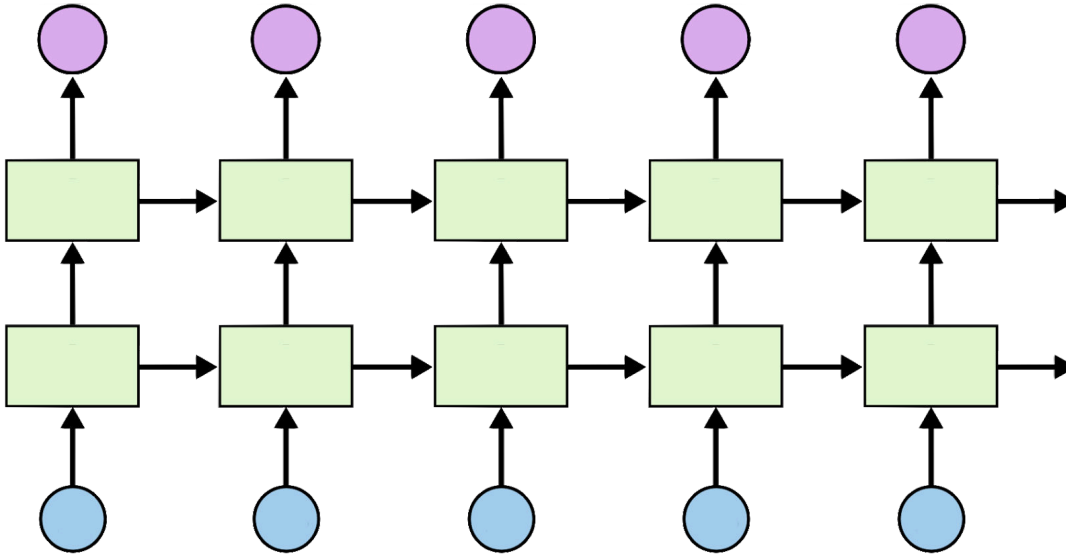
- 双向 RNN 词嵌入


h_i
Word Embedding Sequence
for the Source Language
Word Embedding s_i
One-hot Encoding w_i
Encoder
<s>
祝愿
祖国
繁荣
富强
<e>

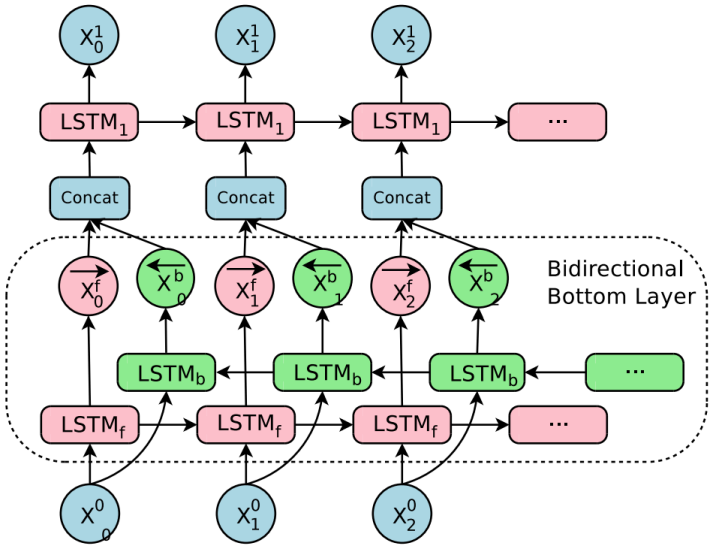
- 双向 RNN Encoder–Decoder



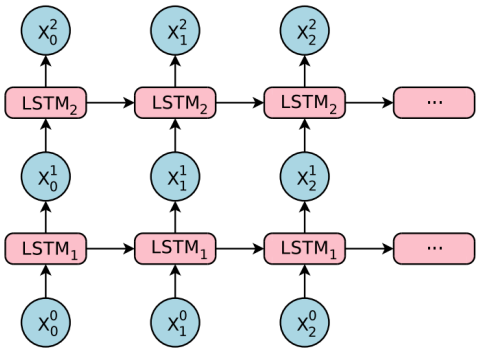
- 堆栈 RNN

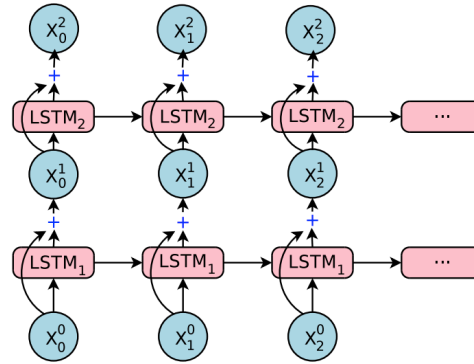


- 双向堆栈 RNN Encoder

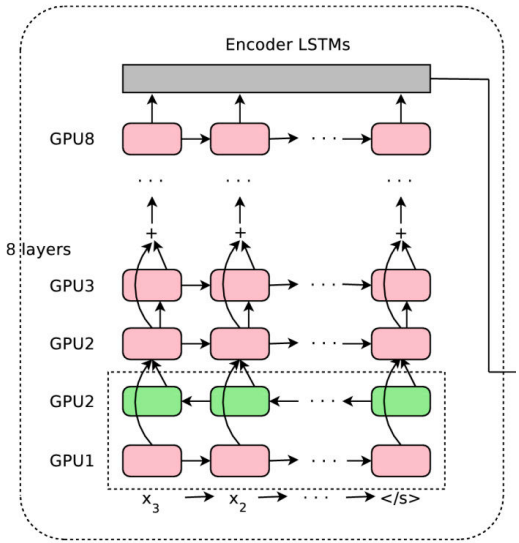


- 带残差的堆栈 RNN


X_0^2
X_1^2
X_2^2
LSTM$_2$
LSTM$_2$
LSTM$_2$
...
X_0^1
X_1^1
X_2^1
LSTM$_1$
LSTM$_1$
LSTM$_1$
...
X_0^0
X_1^0
X_2^0


X_0^2
X_1^2
X_2^2
+
LSTM$_2$
LSTM$_2$
LSTM$_2$
...
X_0^1
X_1^1
X_2^1
+
LSTM$_1$
LSTM$_1$
LSTM$_1$
...
X_0^0
X_1^0
X_2^0

- GNMT 谷歌神经翻译的 Encoder

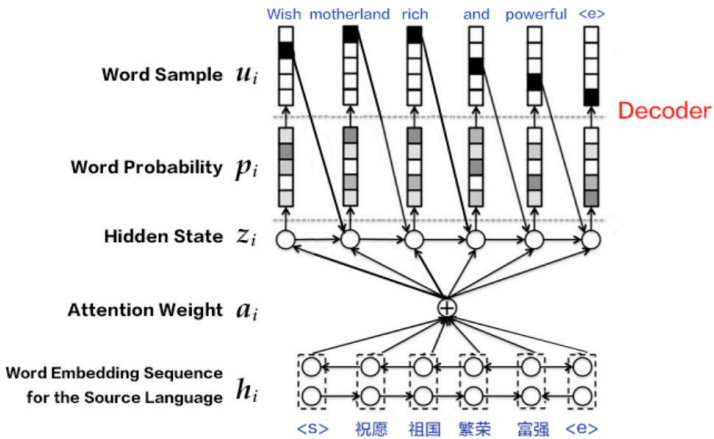


- RNN Encoder–Decoder 小结

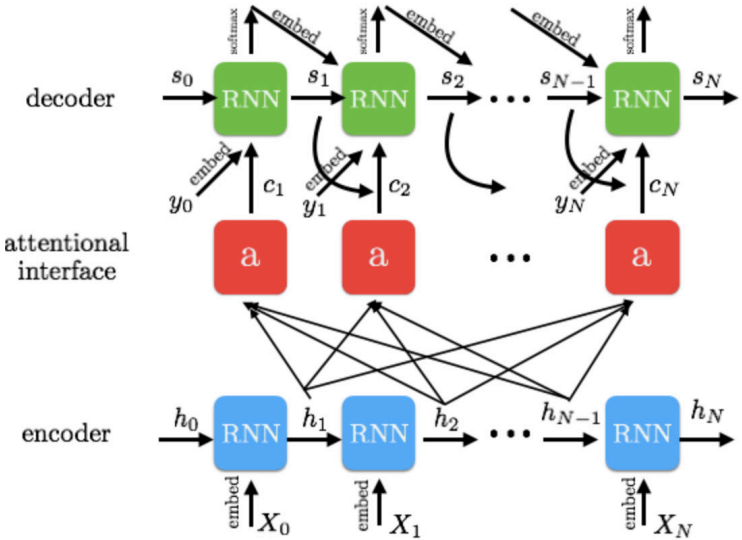
1. 使用 RNN 作为编解码网络
   - 双向 RNN
   - 双向 RNN + 堆栈 + 残差
   - 双向 RNN + 堆栈 + 残差
2. 计算量增大，GPU 加速
3. 原始的 Sequence to Sequence 模型

4.4 Attention 模型

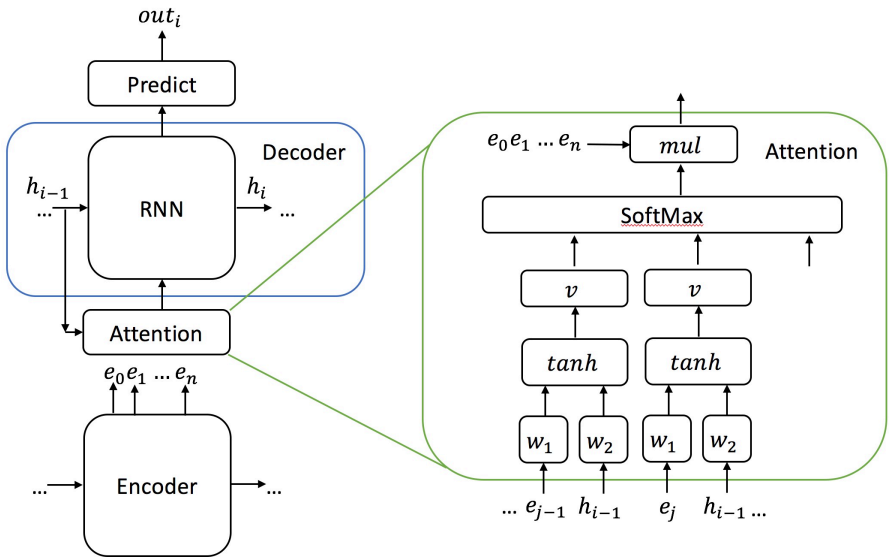
- 注意力机制的端到端翻译



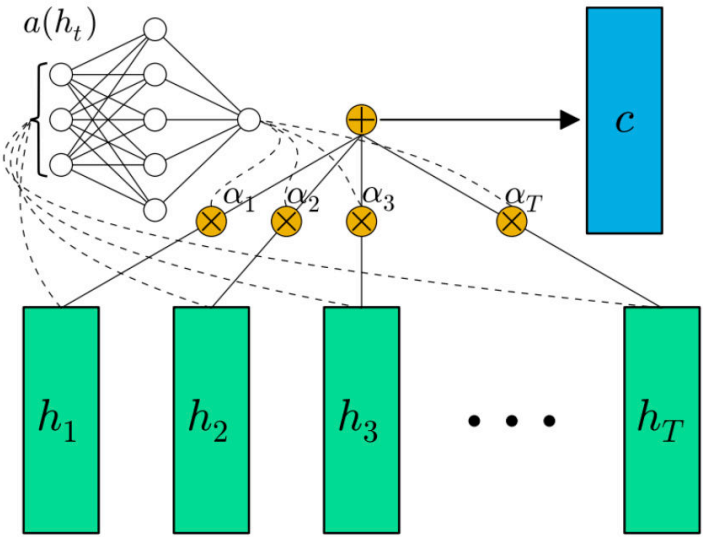
- 注意力机制的 Encoder-Decoder



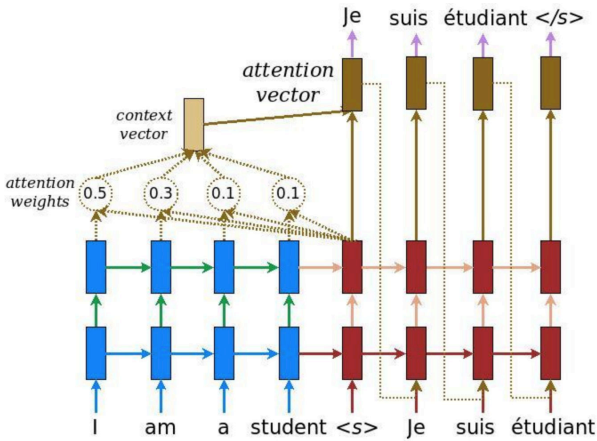
- 注意力网络


out_i
Predict
Decoder
h_{i-1}
...
RNN
h_i
...
Attention
$e_0 e_1 \dots e_n$
...
Encoder
...
$e_0 e_1 \dots e_n$
mul
Attention
SoftMax
v
v
$tanh$
$tanh$
w_1
w_2
w_1
w_2
... e_{j-1} h_{i-1} e_j h_{i-1} ...

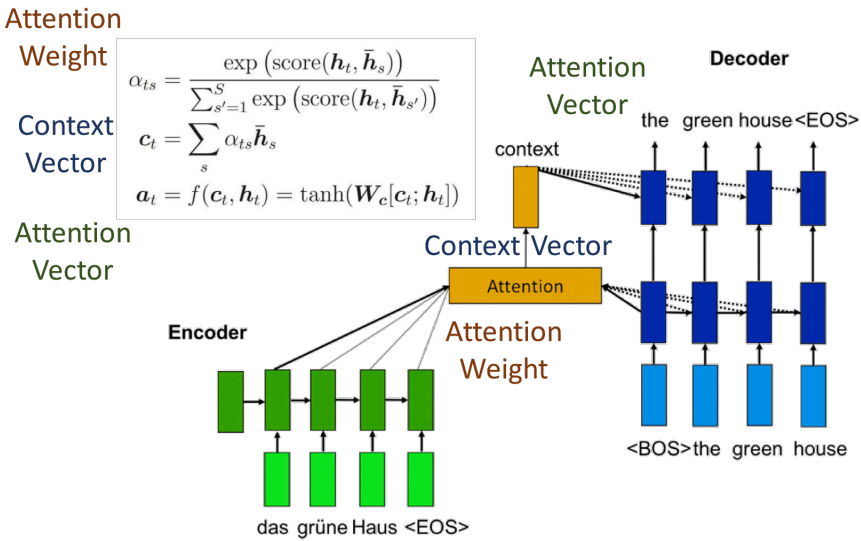
- 注意力权重



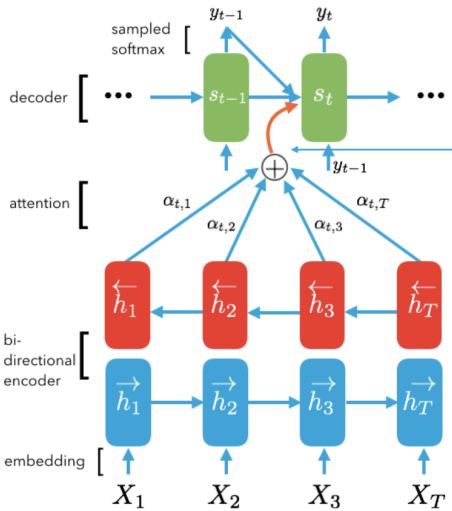
- 注意力机制：自动调整映射范围



- 注意力机制构成


Attention Weight
Context Vector
Attention Vector
$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$
$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s$
$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t])$
Attention Vector
Decoder
the green house <EOS>
context
Context Vector
Attention
Encoder
Attention Weight
<BOS> the green house
das grüne Haus <EOS>

- 注意力机制



$$p(y_i|y_1,...,y_{i-1},X) = g(y_{i-1},s_i,c_i)$$

$$s_i = f(s_{i-1},y_{i-1},c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

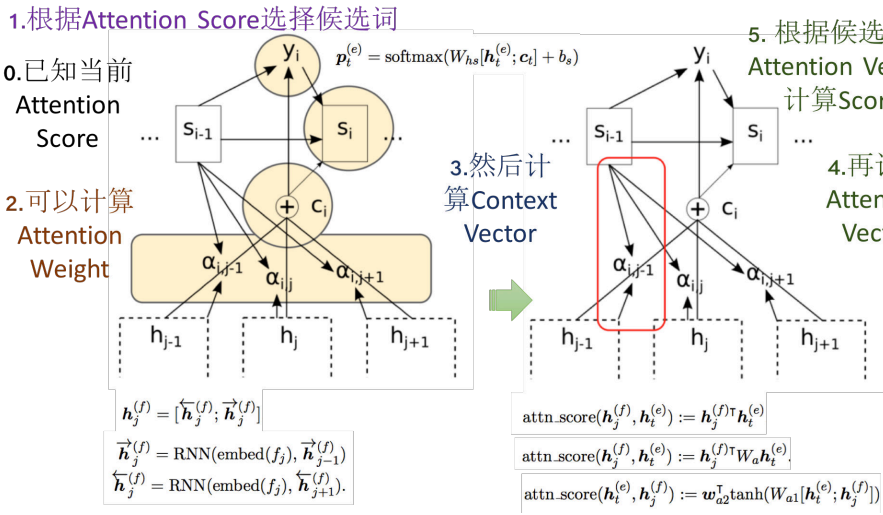
$$\alpha_{i,j} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1},h_j)$$

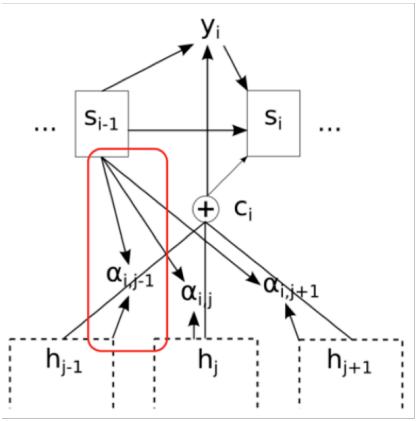
where a is a feed forward neural network.

$$h_j = [\rightarrow h_j; \leftarrow h_j]_{concat}$$

- 注意力机制计算流程


1.根据Attention Score选择候选词
0.已知当前 Attention Score
2.可以计算 Attention Weight
3.然后计算Context Vector
5. 根据候选词、Attention Vector 计算Score
4.再计算 Attention Vector
$p_t^{(e)} = \text{softmax}(W_{hs}[h_t^{(e)}; c_t] + b_s)$
$h_j^{(f)} = [\overleftarrow{h}_j^{(f)}; \overrightarrow{h}_j^{(f)}]$
$\overrightarrow{h}_j^{(f)} = \text{RNN}(\text{embed}(f_j), \overrightarrow{h}_{j-1}^{(f)})$
$\overleftarrow{h}_j^{(f)} = \text{RNN}(\text{embed}(f_j), \overleftarrow{h}_{j+1}^{(f)}).$
$\text{attn\_score}(h_j^{(f)}, h_t^{(e)}) := h_j^{(f)\top} h_t^{(e)}$
$\text{attn\_score}(h_j^{(f)}, h_t^{(e)}) := h_j^{(f)\top} W_a h_t^{(e)}$
$\text{attn\_score}(h_t^{(e)}, h_j^{(f)}) := w_{a2}^{\top} \tanh(W_{a1}[h_t^{(e)}; h_j^{(f)}])$

- 注意力得分 3 种常见计算方法



1. 简单的点积计算（类似cosine距离）

$$\text{attn\_score}(\boldsymbol{h}_j^{(f)}, \boldsymbol{h}_t^{(e)}) := \boldsymbol{h}_j^{(f)\intercal} \boldsymbol{h}_t^{(e)}$$

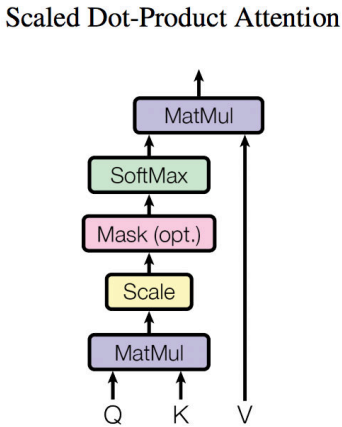
2. 通过矩阵映射计算（加权的cosine距离）

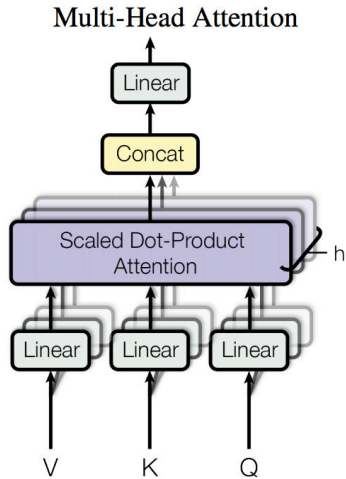
$$\text{attn\_score}(\boldsymbol{h}_j^{(f)}, \boldsymbol{h}_t^{(e)}) := \boldsymbol{h}_j^{(f)\intercal} W_a \boldsymbol{h}_t^{(e)}$$

3. 通过神经网络计算

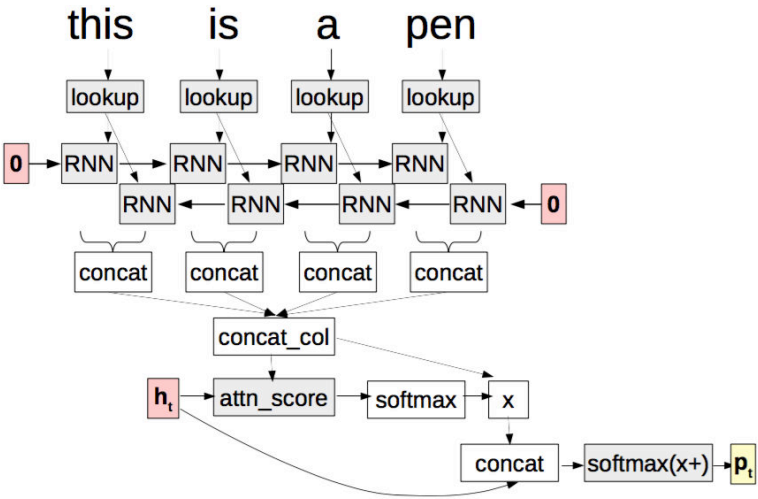
$$\text{attn\_score}(\boldsymbol{h}_t^{(e)}, \boldsymbol{h}_j^{(f)}) := \boldsymbol{w}_{a2}^{\intercal} \tanh(W_{a1}[\boldsymbol{h}_t^{(e)}; \boldsymbol{h}_j^{(f)}]$$

- 注意力得分网络

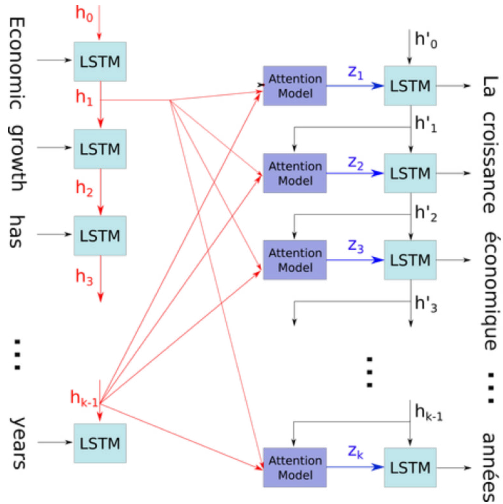

Scaled Dot-Product Attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

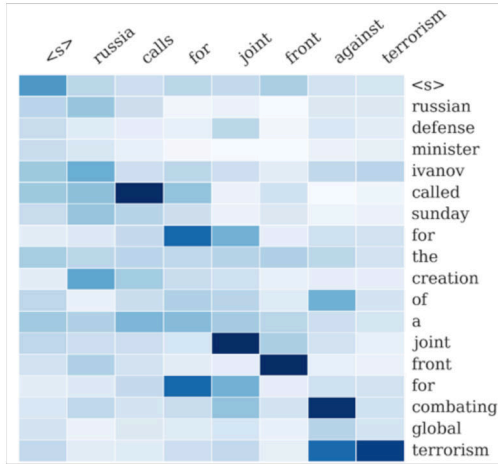

Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

- 注意力计算图

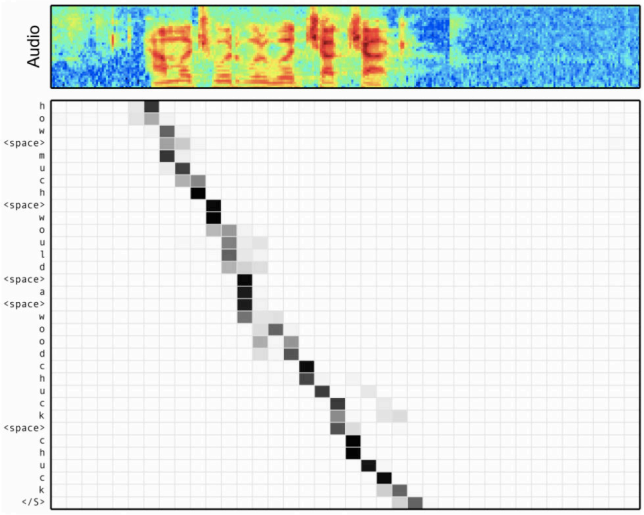

this
is
a
pen
lookup
lookup
lookup
lookup
0
RNN
RNN
RNN
RNN
RNN
RNN
RNN
RNN
0
concat
concat
concat
concat
concat_col
h_t
attn_score
softmax
x
concat
softmax(x+)
p_t

- 注意力得分和对齐

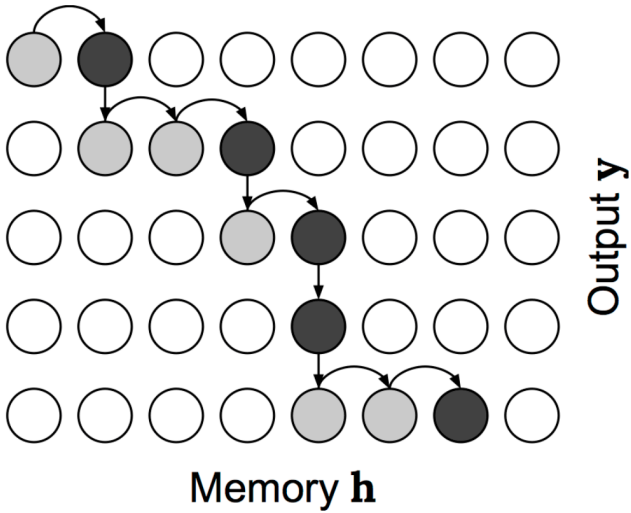

Economic growth has ... years
LSTM
Attention Model
La croissance économique ... années


<s>
russia
calls
for
joint
front
against
terrorism
russian
defense
minister
ivanov
called
sunday
for
the
creation
of
a
joint
front
for
combating
global
terrorism

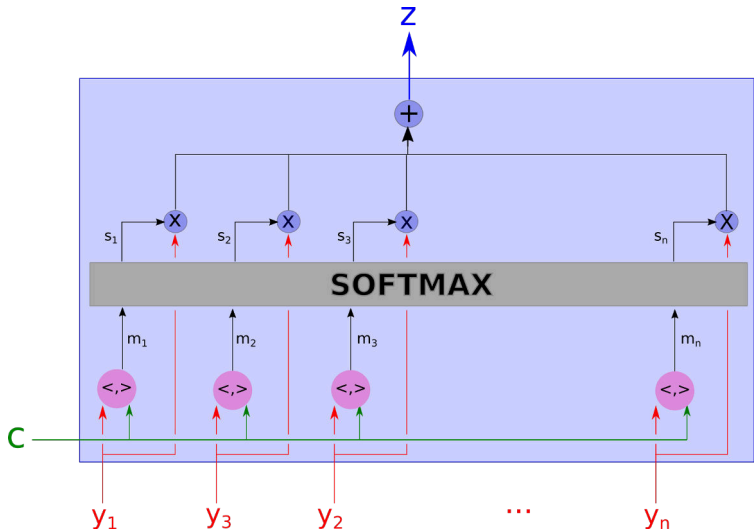
- 注意力机制对齐：语音和文字信号



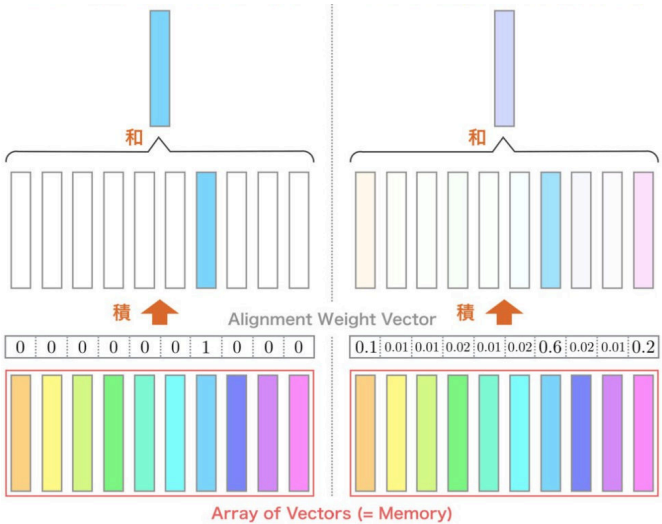
- 对齐机制



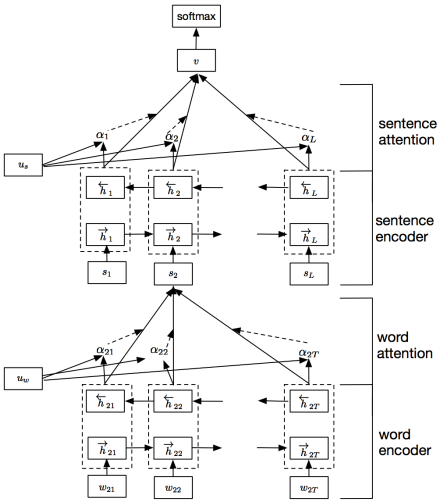
- 软注意力机制



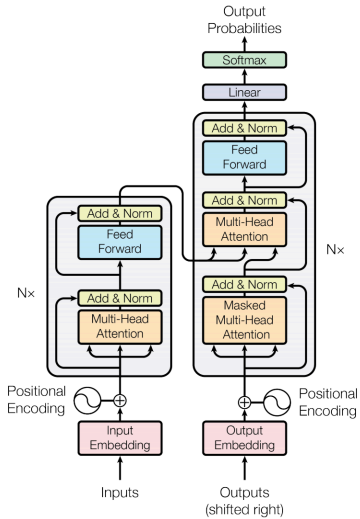
- 硬和软的注意力机制对比: 0/1 vs probability



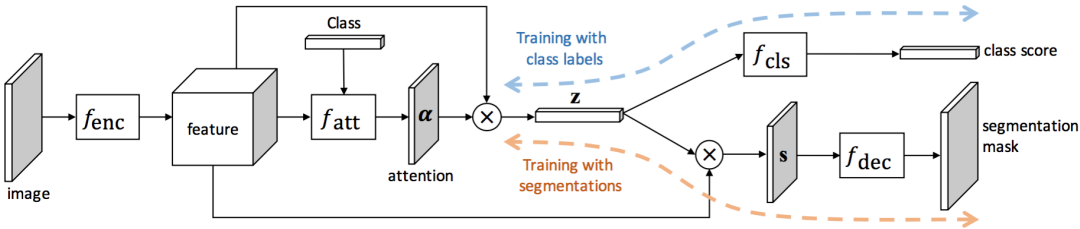
- 分层注意力机制


softmax
v
sentence attention
sentence encoder
word attention
word encoder

- Attention Is All You Need: Transformer Model



- 图像注意力机制

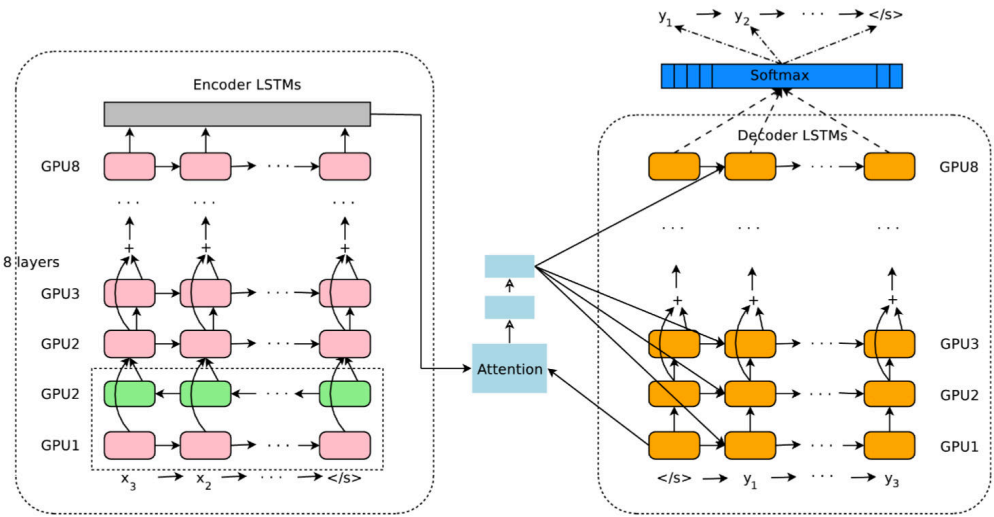

Class
Training with
class labels
f_{cls}
class score
image
f_{enc}
feature
f_{att}
α
attention
z
s
f_{dec}
segmentation
mask
Training with
segmentations

- Attention 机制小结

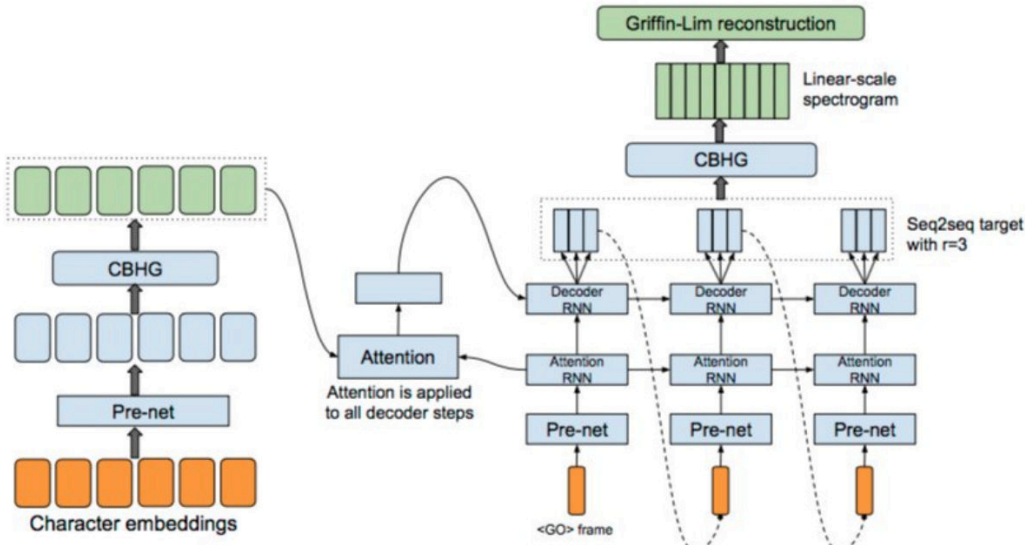
1. Attention 是上下文环境的自适应强化
2. Attention 可以有软硬的区别
3. Attention 和传统的对齐有很强的关联
4. Attention 计算分为 5 个步骤
    - 根据得分选定候选词
    - 计算权重
    - 计算上下文向量
    - 计算注意力向量
    - 重新计算得分
5. Attention 得分主要有 3 种方法：点积、矩阵积、神经网络
6. Attention 扩展：分层模型、转换器模型

4.5 Seq2Seq 模型：端到端

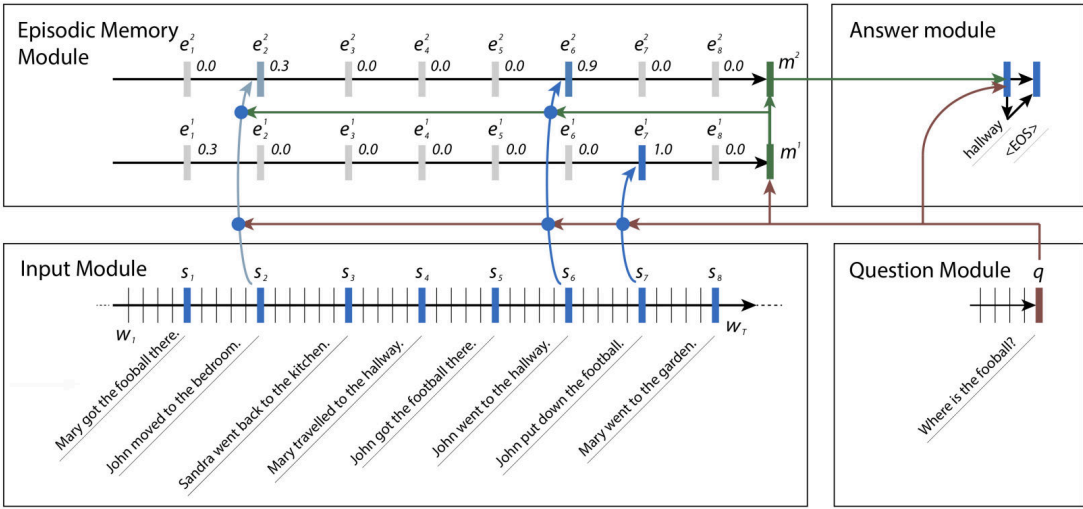
- 谷歌神经翻译 GNMT：Seq2Seq 模型



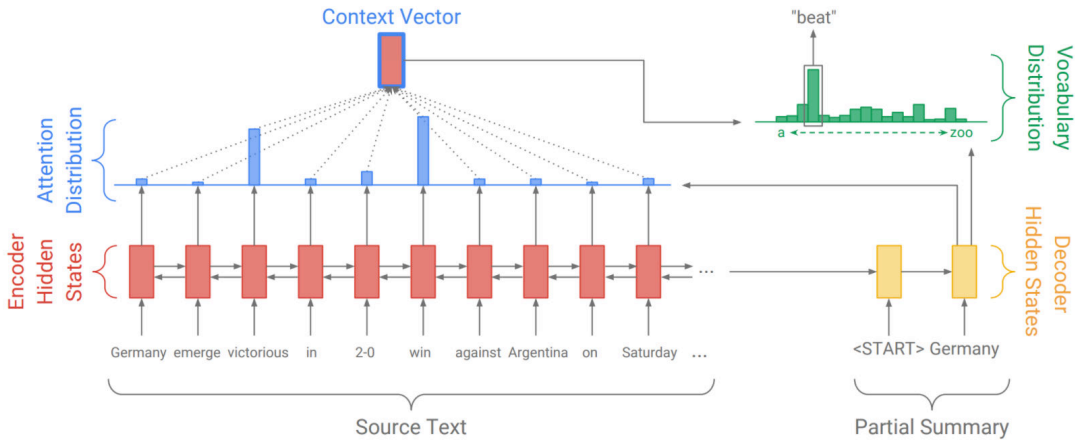
- Google 的 TTS 系统: Tacotron



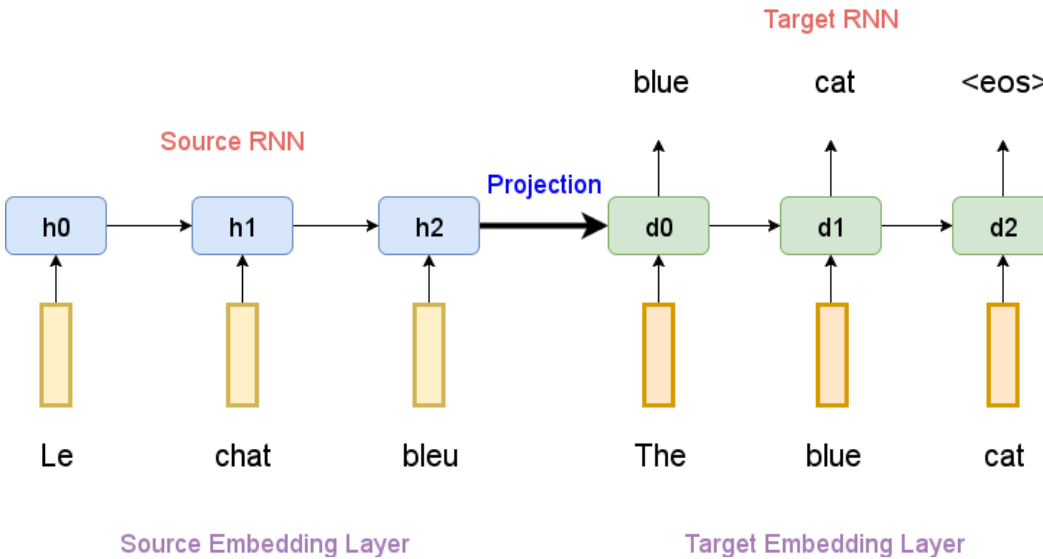
- 问答系统：


Episodic Memory Module
Answer module
Input Module
Question Module
hallway
<EOS>
Where is the fooball?
Mary got the fooball there.
John moved to the bedroom.
Sandra went back to the kitchen.
Mary travelled to the hallway.
John got the football there.
John went to the hallway.
John put down the football.
Mary went to the garden.

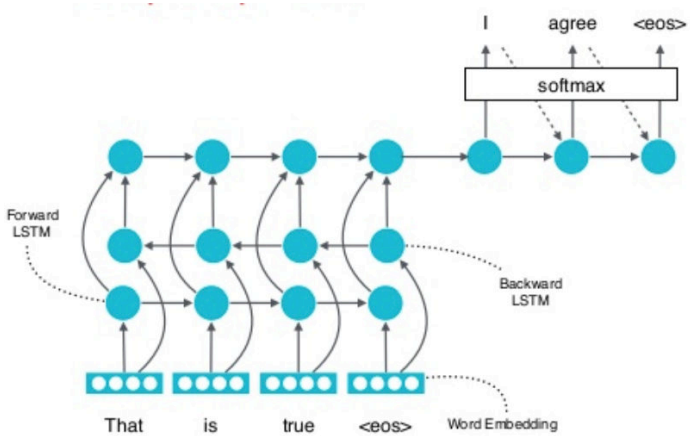
- 概要提取


Context Vector
"beat"
Vocabulary Distribution
a
zoo
Attention Distribution
Encoder Hidden States
Decoder Hidden States
Germany emerge victorious in 2-0 win against Argentina on Saturday ...
<START> Germany
Source Text
Partial Summary

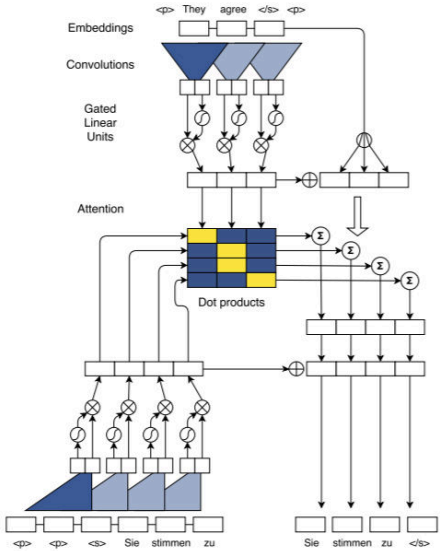
- 最简 Seq2Seq


Target RNN
blue
cat
<eos>
Source RNN
Projection
h0
h1
h2
d0
d1
d2
Le
chat
bleu
The
blue
cat
Source Embedding Layer
Target Embedding Layer

- Seq2Seq：RNN 强化编解码



- Seq2Seq：Attention 强化对齐

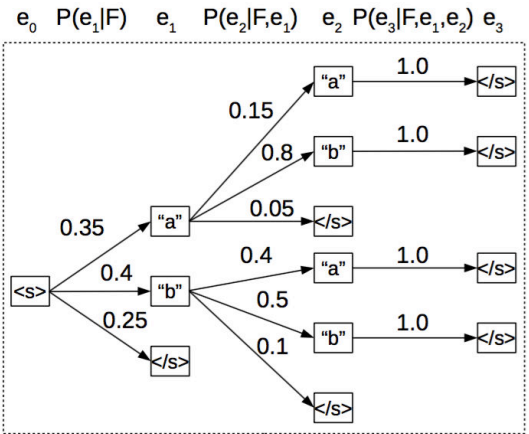


- Seq2Seq 模型小结

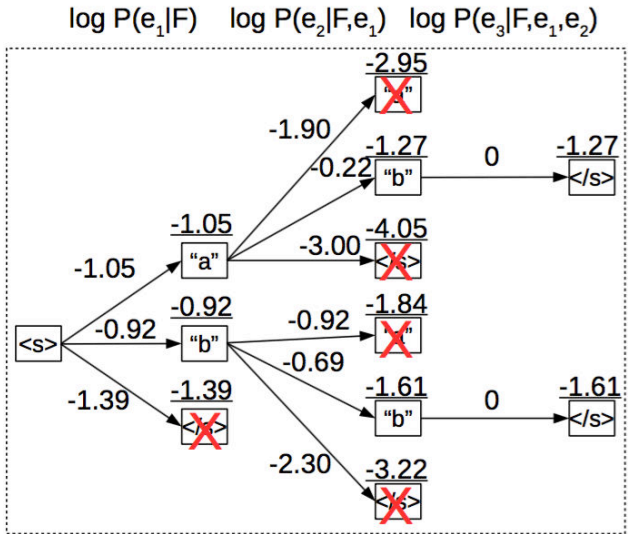
1. Seq2Seq 模型适用范围很广
    - 翻译、问答、概要、语音、图片
2. Seq2Seq 增强
    - RNN 变型
    - Attention 机制
3. Seq2Seq 计算量很大

4.6 翻译输出

- 贪心查找 Greedy Search 全局最优

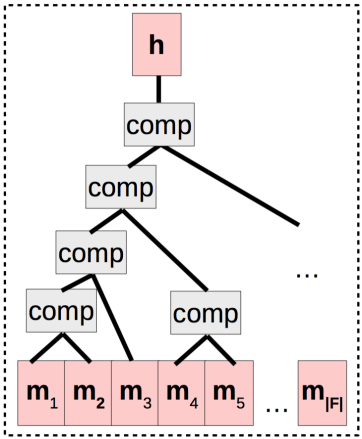


- 束查找 Beam Search 全局最优

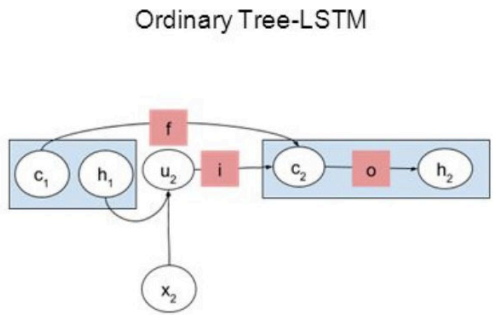


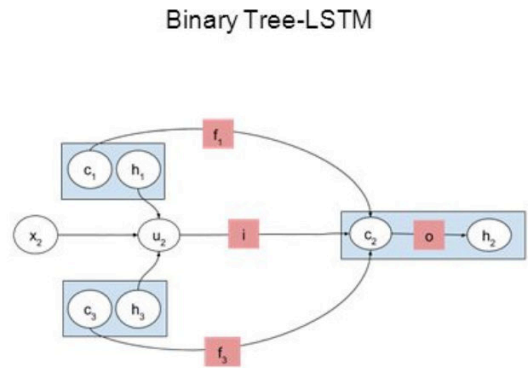
4.7 替代和增强 RNN 模型

- 树结构做特征提取

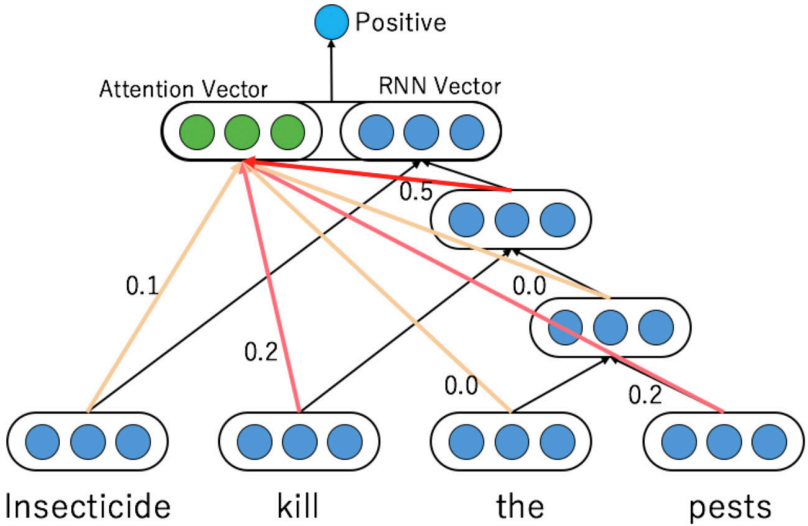


- Tree RNN

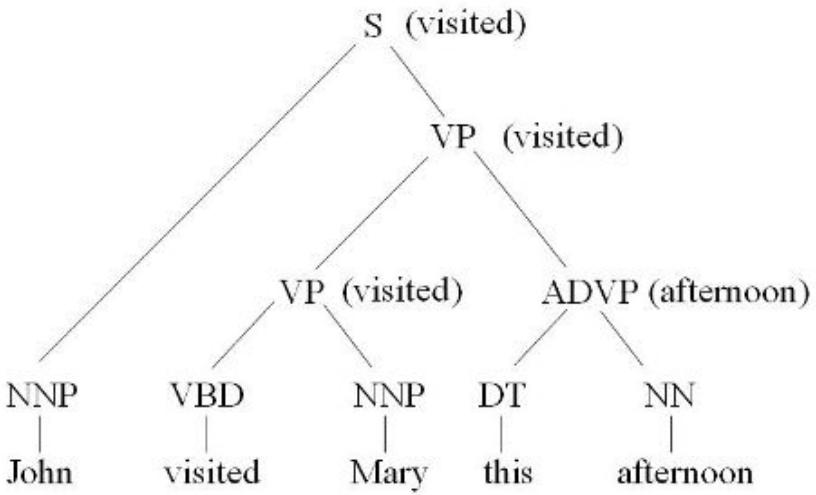

Ordinary Tree-LSTM
f
c1
h1
u2
i
c2
o
h2
x2


Binary Tree-LSTM
f1
c1
h1
x2
u2
i
c2
o
h2
c3
h3
f3

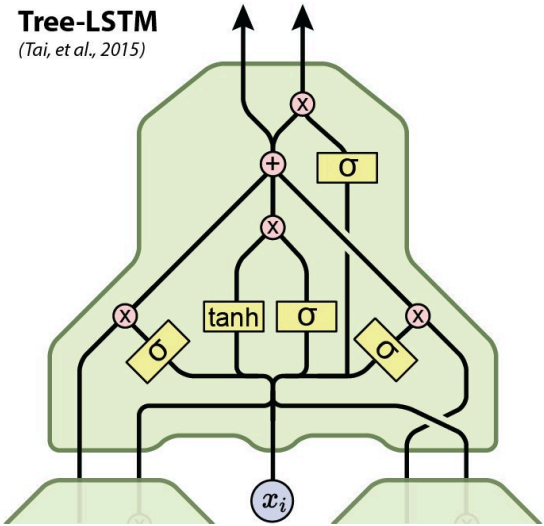
- 二叉树 RNN


Positive
Attention Vector
RNN Vector
0.5
0.1
0.0
0.2
0.0
0.2
Insecticide
kill
the
pests

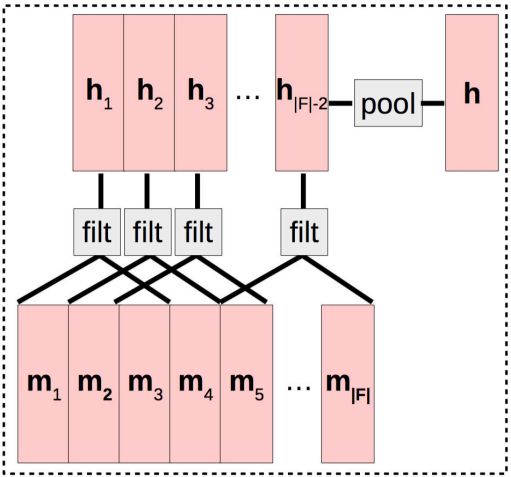
- 语法树


S (visited)
VP (visited)
VP (visited)
ADVP (afternoon)
NNP
VBD
NNP
DT
NN
John
visited
Mary
this
afternoon

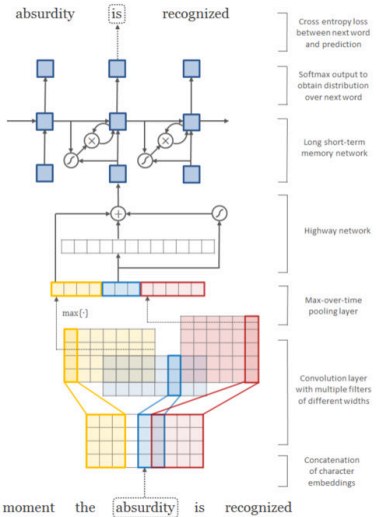
- 树结构化的 LSTM



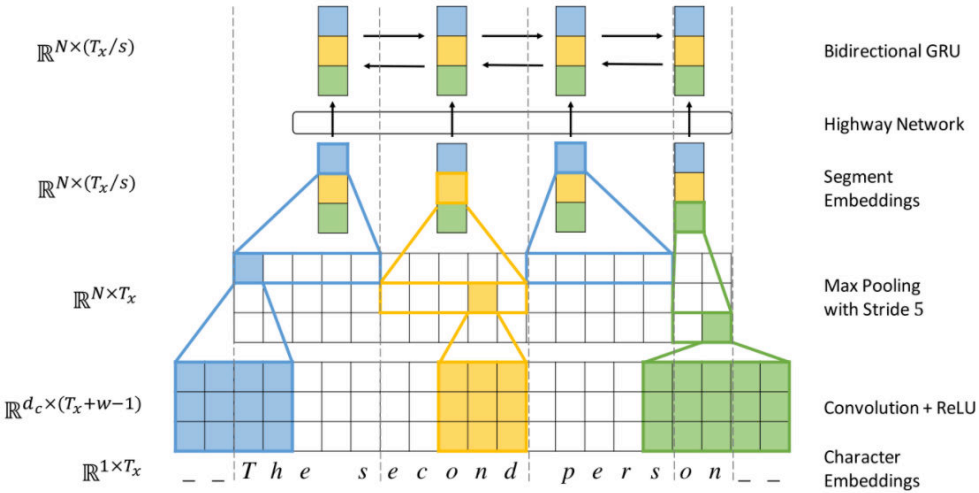
- CNN 做特征提取



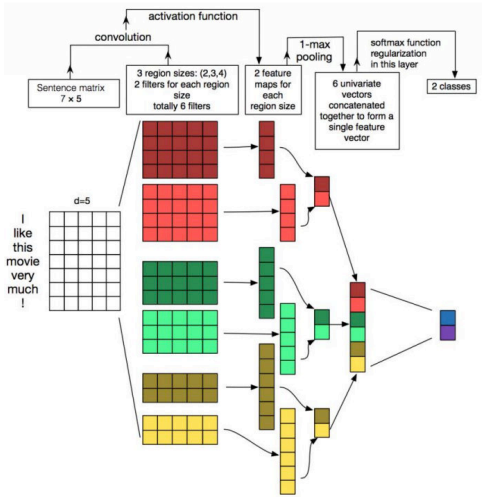
- CNN-RNN 模型：char-cnn-rnn 文本



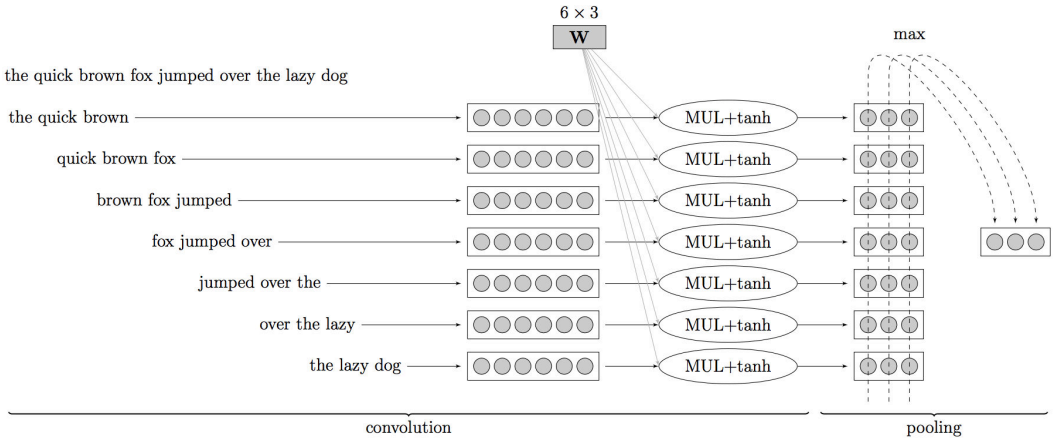
- Char-CNN-RNN 模型



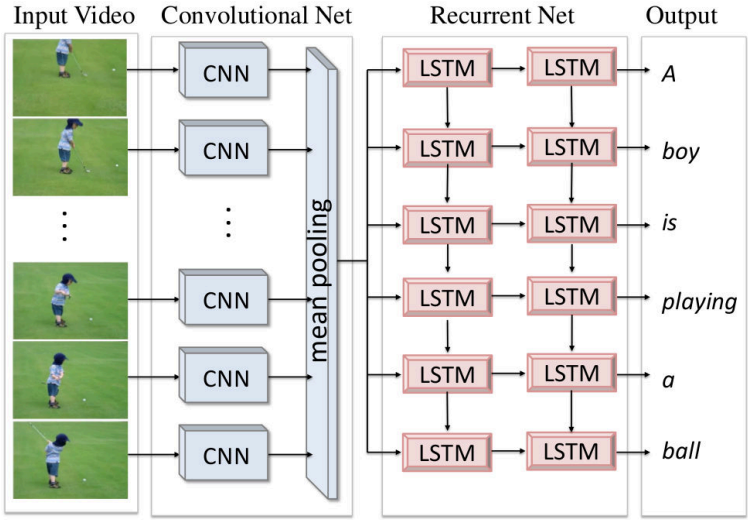
- CNN 文本处理


activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

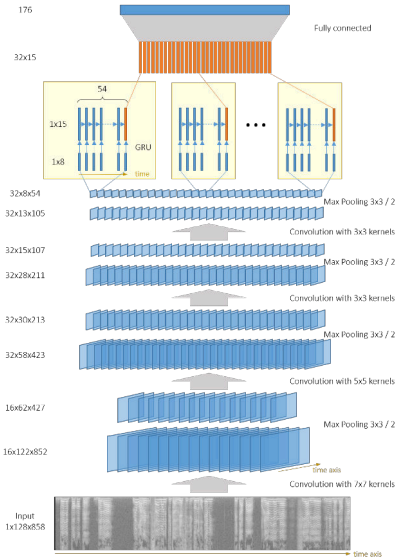
- CNN 文本处理：窗口滑动



- CNN-RNN 模型：图像处理



- CNN-RNN 模型：语音处理

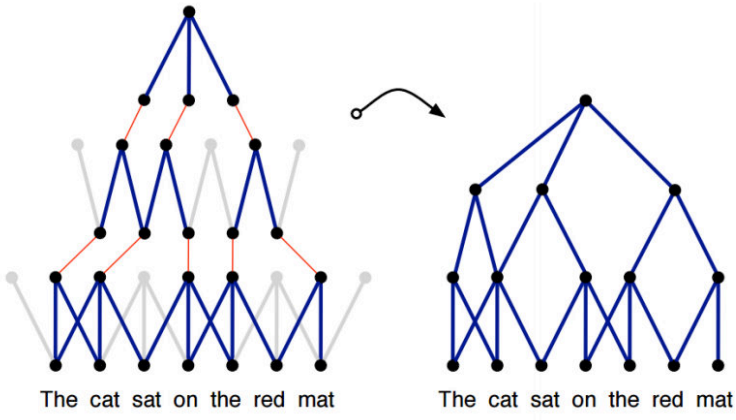


- Tree-RNN 和 CNN-RNN 小结

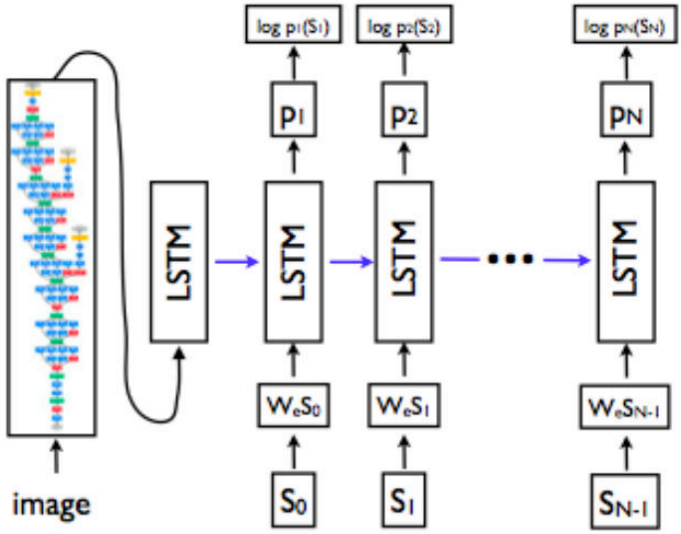
1. 通过结构化 RNN 引入结构先验
2. 通过 CNN 特征提取简化计算
3. 通过 Tree-RNN 和 CNN-RNN 强化稳定性

4.8 其他应用

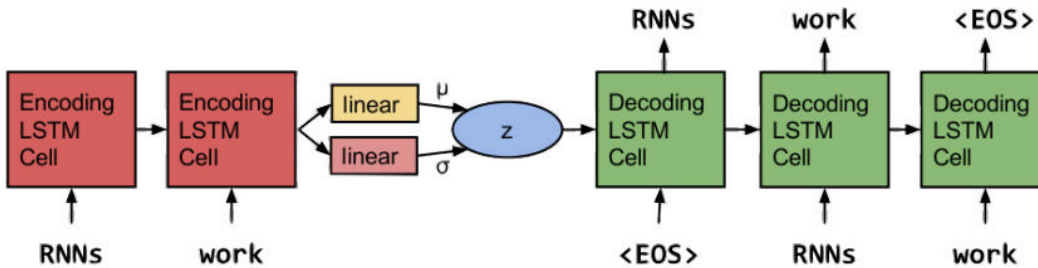
- 句子语言建模：DCNN (dynamic convolutional neural network)



- 图像标题：CNN-LSTM Decoder



- 句子生成：RNN-based VAE



感谢 Stanford, CMU, MIT 等网上公开课程和资料！

AI2ML