法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象
  - 新浪微博：ChinaHadoop

语言模型
Language Model

主讲人： 史兴

6/30/2017

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

什么是语言模型

XX 模型 ： Score(xx)

语言模型 (language Model): Score(语言)

Score(什么 是 语言 模型) --> 0.05
Score(什么 有 语言 模型) --> 0.01

什么是语言模型

XX 模型 ： Score(xx)

话题模型 (Topic Model): Score(话题|语言)

Score(NLP | “什么 是 语言 模型？”) --> 0.8
Score(三国 | “什么 是 语言 模型？”) --> 0.05

什么是语言模型

XX 模型 ： Score(xx)

狗叫模型 (Bark Model): Score(狗叫)

Score(“汪 汪 汪”) -> 0.5
Score(“汪 汪 喵”) -> 0.01

球星模型： Score(库里); Score(詹姆斯);
电影模型： Score(湄公河行动);

概率语言模型

概率语言模型(Statistical Language Model)

$$P(sentence) = P(w_1, w_2, ..., w_n)$$

$$\sum_{sentence \in L} P(sentence) = 1$$

语言模型的用处

- 输入法： P(隔壁老王) > P(隔壁老张)
- 机器翻译：
  I have a dream
  P(我有个梦想) > P(我有只梦想)
- 语音识别：
  P(我向你汇报) > P(我象你汇报)

核心： 通过分数来告诉机器怎么说人话

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

N-gram

计算 $P(w_1, w_2, ..., w_n)$

利用链式法则：

$$P(A, B, C) = P(A)P(B|A)P(C|A, B)$$

$$P(w_1, w_2, .., w_n) = P(w_1)P(w_2|w_1)...P(w_n|w_1, ..., w_{n-1})$$

N-gram

计算 $P(w_1, w_2, ..., w_n)$

马尔可夫(Markov)假设：
“无记忆性”：未来的事件，只取决于有限的历史

某厂

N-gram

计算 $P(w_1, w_2, ..., w_n)$

马尔可夫(Markov)假设：

“无记忆性”：未来的事件，只取决于有限的历史

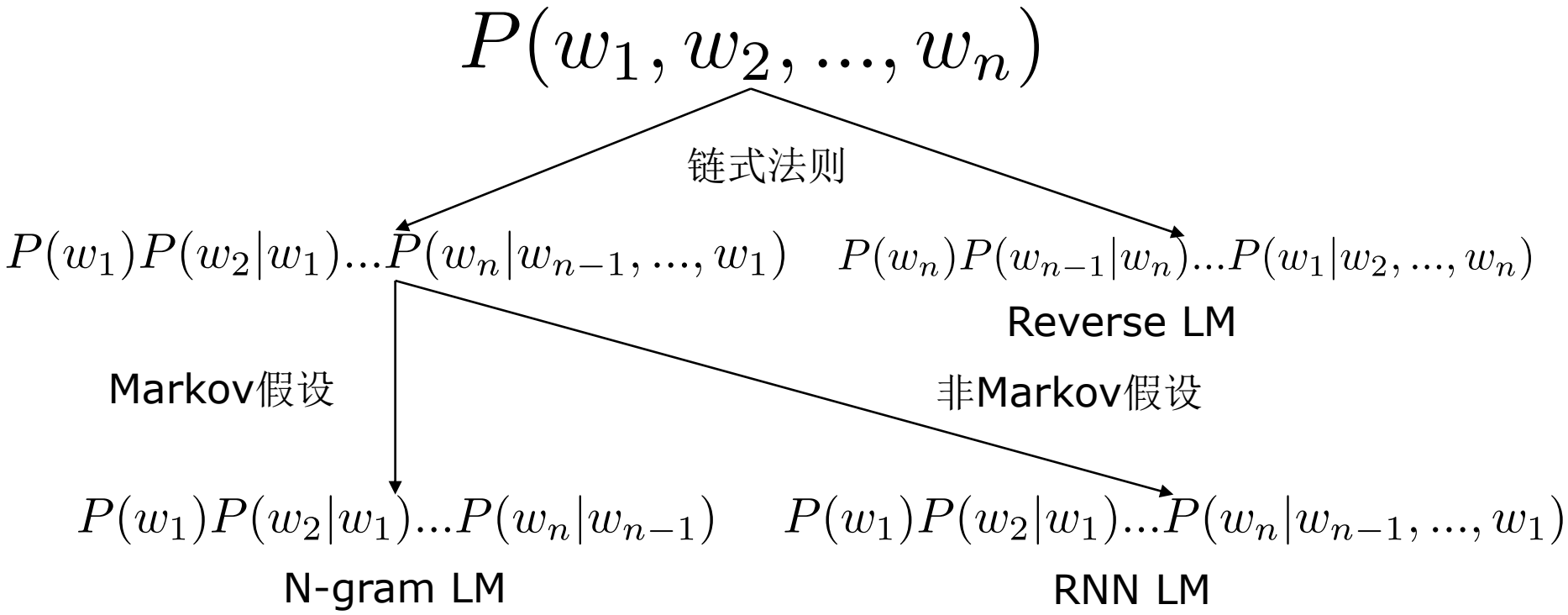
$$P(w_5|w_4, w_3, w_2, w_1)$$

$P(w_5)$ unigram　　$P(w_5|w_4)$ bigram　　$P(w_5|w_4, w_3)$ trigram

$$P(w_1, w_2, ..., w_n) = P(w_1)P(w_2|w_1)...P(w_n|w_{n-1})$$

$$P(w_1, w_2, ..., w_n) = P(w_1|START)P(w_2|w_1)...P(w_n|w_{n-1})P(EOS|w_n)$$

语言模型总结


$P(w_1, w_2, ..., w_n)$
链式法则
$P(w_1)P(w_2|w_1)...P(w_n|w_{n-1}, ..., w_1)$
$P(w_n)P(w_{n-1}|w_n)...P(w_1|w_2, ..., w_n)$
Reverse LM
Markov假设
非Markov假设
$P(w_1)P(w_2|w_1)...P(w_n|w_{n-1})$
$P(w_1)P(w_2|w_1)...P(w_n|w_{n-1}, ..., w_1)$
N-gram LM
RNN LM

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

语言模型的评价

	外在评价 Extrinsic Evaluation	内在评价 Intrinsic Evaluation
猫	能不能抓到老鼠	颜色；速度；力量
语言模型	语音识别的准确率	预测测试集的能力
优点	实践是掌握真理的唯一标准	快，简单
缺点	慢，复杂	与真正的目标有偏差

语言模型的评价

	外在评价 Extrinsic Evaluation	内在评价 Intrinsic Evaluation
猫	能不能抓到老鼠	颜色；速度；力量
语言模型	语音识别的准确率	**预测测试集的能力**

问鉴宝专家这是真品的概率？

鉴宝专家A ： 30%

鉴宝专家B： 70%

大家从<u>未见过</u>的<u>真</u>古董

语言模型的评价

预测测试集的能力

语言模型 ⬆ -> P(test set) ⬆ -> Perplexity(test set) ⬇

公式

$$W_{test} = \{w_1, w_2, ..., w_n; w_i \in V\}$$

$$Perplexity(W_{test}) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 q(w_i)}$$

实例

V	a	b	c	d
P 真实概率	0.4	0.2	0.2	0.2
Q LM概率	0.5	0.1	0.2	0.2

$$W_{test} = \{a, d, c, a, b\}$$

$$PPX(Q) = -\frac{1}{5} * (\log_2(0.5) + \log_2(0.2) + \log_2(0.2) + \log_2(0.5) + \log_2(0.1)) = 3.98$$

$$PPX(P) = 3.79$$

语言模型的评价

如何理解Perplexity

$$Perplexity(W_{test}) = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 q(w_i)}$$

$$= 2^{-\sum_{i=1}^{|V|}\frac{count(V_i)}{N}\log_2 q(v_i)}$$

$$= 2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$$

合同相同的词汇

$$\hat{p}(v_i) = \frac{count(v_i)}{N}$$

$-\log_2 p(v_i)$ 如果用概率分布q来编码v_i, 需要多少比特（bits）

$-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)$ v_i的分布服从p， 用q来编码v_i, 需要的比特数的期望。

$2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$ W_{test}的等效状态的数目

语言模型的评价

如何理解Perplexity

$2^{-\sum_{i=1}^{|V|}\hat{p}(v_i)\log_2 q(v_i)}$ W_{test}的等效状态的数目

PPX(投掷硬币)= 2
PPX(投掷骰子)= 6

N-gram	Unigram	Bigram	Trigram
PPX	-bug-	219	174

ptb dataset by KenLM

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

OOV

Trigram Model

$$P(w_i|w_{i-1}, w_{i-2}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})}$$

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

Testing Set

我 喜欢 上网
我 喜欢 编程
我 喜欢 王者荣耀

P(王者荣耀|我 喜欢) = 0 (Training中来没有出现的词)
→ OOV (Out of Vocabulary)

P(编程|我 喜欢) = 0 (Training中没有出现的trigram)
→ Smoothing

OOV

Trigram Model

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

$$P(w_i|w_{i-1},w_{i-2}) = \frac{count(w_{i-2},w_{i-1},w_i)}{count(w_{i-2},w_{i-1})}$$

OOV

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

Maximum Likelihood 方法

$$\max \log(L(D_{Train}))$$
$$= \log(\prod_i p(w_i|w_{i-1}, w_{i-2}))$$
$$= \sum_i \log(p(w_i|w_{i-1}, w_{i-2}))$$

= 3*log(p(我|-，-)) + log(p(你|-, -))
+ 3*log(p(喜欢|-, 我)) + log(p(喜欢|-,你))
+ log(p(开车|我 喜欢)) + log(p(上网|我 喜欢))
+ log(p(篮球|我 喜欢)) + log(p(编程|你 喜欢))

限制： p(我|-，-) + p(你|-，-) = 1

拉格朗日法：
L = 3*log(p(我|-，-)) + log(p(你|-, -))
+ lambda * (p(我|-，-) + p(你|-，-) - 1)

OOV

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

拉格朗日法：
L = 3*log(p(我|-，-)) + log(p(你|-, -))
+ lambda * (p(我|-，-) + p(你|-，-) - 1)

L对参数的导数等于零：
dL / d p(我|-，-) = 0
dL / d p(你|-，-) = 0
dL / d lambda = 0

3 / p(我|-,-) + lambda = 0
1 / p(你|-,-) + lambda = 0
p(我|-，-) + p(你|-，-) – 1 = 0

P(我|-,-) = 3 / (3 + 1)
= count(-,-,我) / (count(-,-,我) + count(-,-,你))

OOV

Trigram Model

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

$$P_{ML}(w_i|w_{i-1},w_{i-2}) = \frac{count(w_{i-2},w_{i-1},w_i)}{count(w_{i-2},w_{i-1})}$$

OOV

Trigram Model

$$P(w_i|w_{i-1}, w_{i-2}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})}$$

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 篮球
你 喜欢 编程

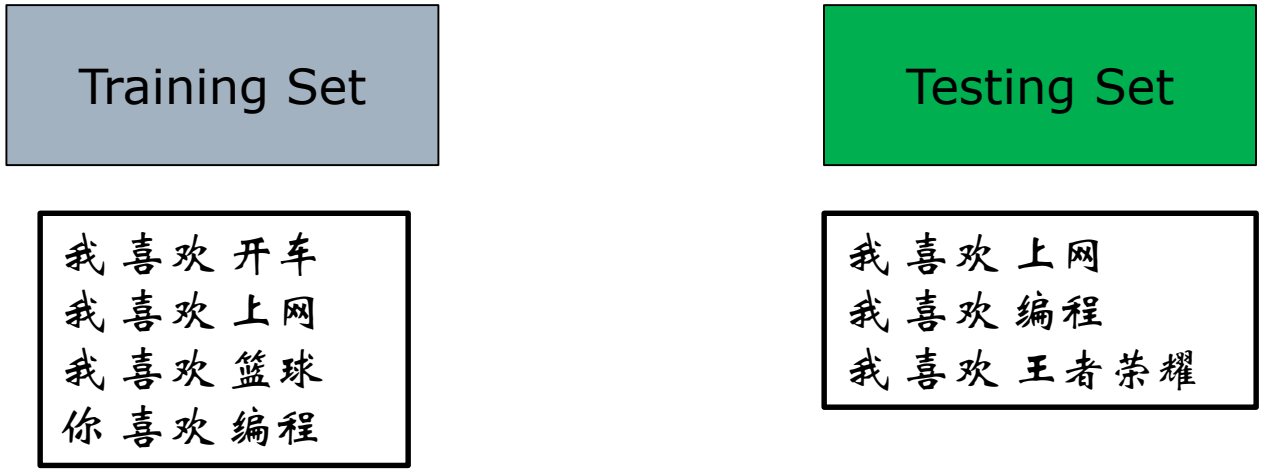
Testing Set

我 喜欢 上网
我 喜欢 编程
我 喜欢 王者荣耀

P(王者荣耀|我 喜欢) = 0 (Training中来没有出现的词)
→ OOV (Out of Vocabulary)

P(编程|我 喜欢) = 0 (Training中没有出现的trigram)
→ Smoothing

OOV



P(王者荣耀|我 喜欢) = 0 (Training中来没有出现的词)

解决方法：

1. 假设Training Set中出现了|V'|个不同的词汇，那么我们根据词频选择词频最高的|V|个词汇作为我们的词汇集V。
2. 在Training和Testing中，将不属于V的词汇都替换成特殊词汇UNK.

OOV

Training Set

我 喜欢 开车
我 喜欢 上网
我 喜欢 UNK
你 喜欢 编程

Testing Set

我 喜欢 上网
我 喜欢 编程
我 喜欢 UNK

V′ = {我 喜欢 开车 上网 篮球 编程}
V = {我 喜欢 开车 上网 编程 }

P(王者荣耀|我 喜欢)
= P(UNK|我 喜欢)
= count(我 喜欢 UNK) / count(我 喜欢)
= 1 / 3 = 0.333

提纲

- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

N-gram 平滑

P(编程|我 喜欢) = 0　　(Training中没有出现的trigram)

本质上，是一个“**贫富分化**”的问题

A. 政府给大家每人都发一笔钱
B.找父母要
C.劫富济贫

N-gram 平滑

P(编程|我 喜欢) = 0　　(Training中没有出现的trigram)

A. 政府给大家每人都发一笔钱
B. 找父母要
C. 劫富济贫

- +1 平滑　(A)
- Back-off 回退法 (B)
- Interpolate 插值法 (B)
- Absolute Discount (C)
- Kneser-Ney Smoothing (C)
- Modified Kneser-Ney Smoothing (最优的方法) (C)

N-gram 平滑

- +1 平滑　1920年提出

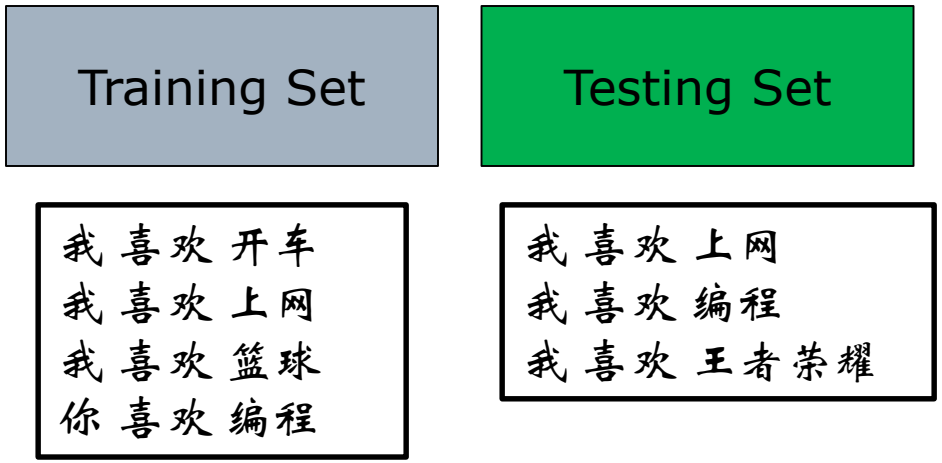
$$P_{add}(w_i|w_{i-1},w_{i-2}) = \frac{\delta+count(w_{i-2},w_{i-1},w_i)}{\delta|V|+\sum_{w_i} count(w_{i-2},w_{i-1},w_i)}$$

在别的分类问题中可能有用，
但是在语言模型中表现一般

政府给大家每人发一点钱

N-gram 平滑

- Back-off 回退法 1987年提出



Count(我 喜欢 编程) = 0
但是 count(喜欢 编程) > 0

思路：
使用 Trigram 如果 count(trigram) 满足一定的条件
否则使用Bigram;
否则使用Unigram;　　具体参考 “Katz smoothing”

自己有钱自己出；自己没钱，爸爸出；爸爸没钱爷爷出

N-gram 平滑

- Interpolate 插值法 1980 提出完善

将Trigram, Bigram, Unigram线性组合起来:

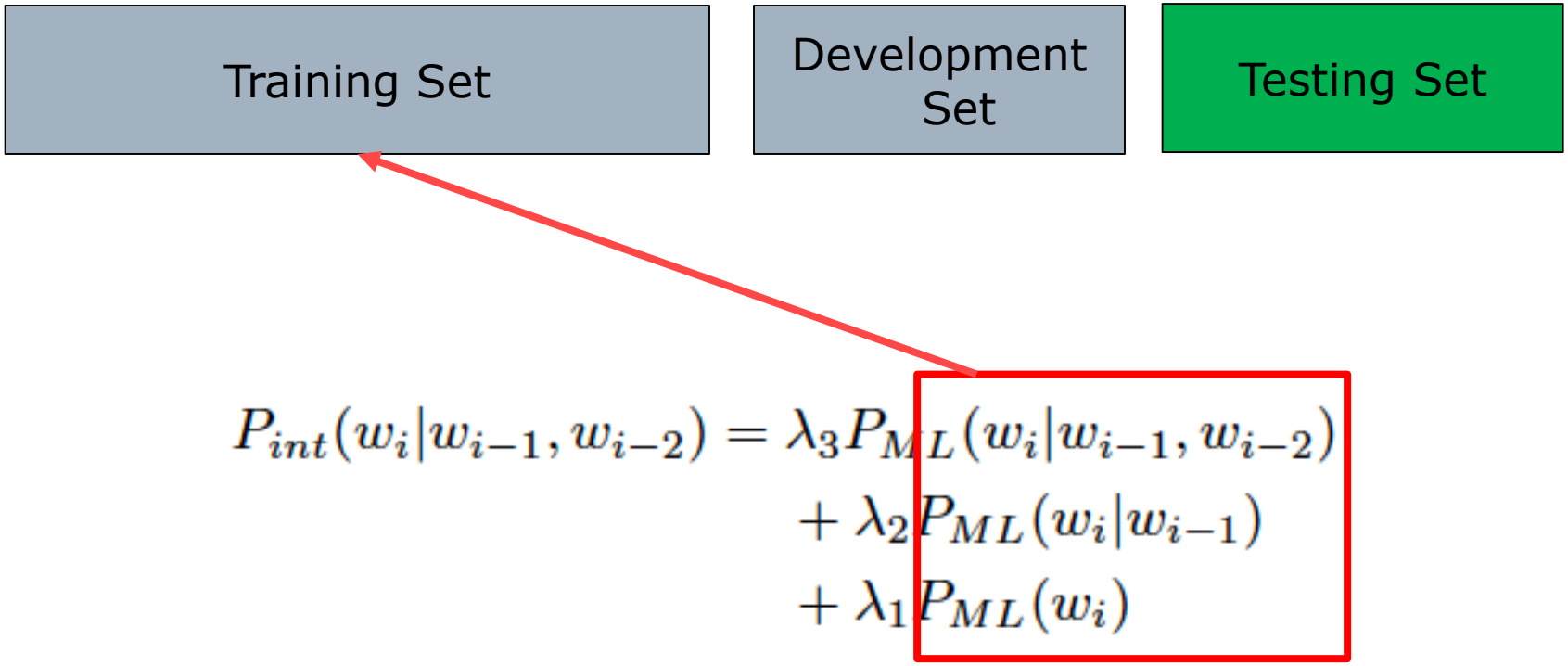
$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$\lambda_3 + \lambda_2 + \lambda_1 = 1$$

自己， 爸爸， 爷爷各自出一点钱

N-gram 平滑

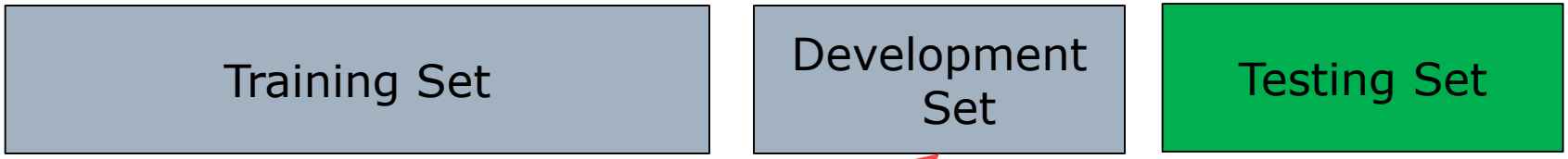
- Interpolate 插值法



$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

N-gram 平滑

- Interpolate 插值法

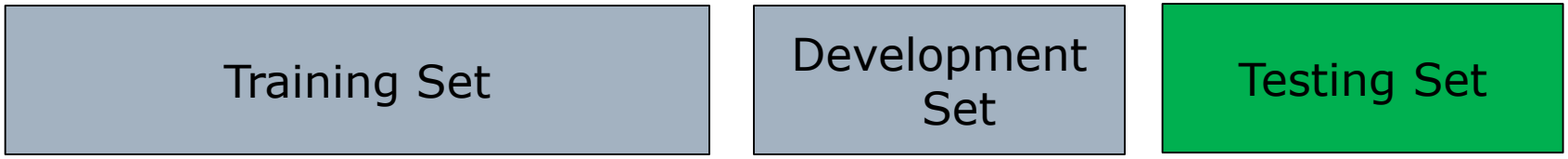


$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$L = \sum_h count(h) \log(\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))$$

N-gram 平滑

- Interpolate 插值法



$$\begin{aligned}&\max \log(L(D_{Train}))\\ &= \log(\prod_i p(w_i|w_{i-1}, w_{i-2}))\\ &= \sum_i \log(p(w_i|w_{i-1}, w_{i-2}))\end{aligned}$$

log里面只有一个参数
求导之后各个参数独立出来

$$L = \sum_h count(h) \log(\sum_{i=1}^{K} \lambda_i P_{ML,i}(h))$$

log里面只有几个参数的和
求导之后，各个参数耦合在一起

EM 算法 来解决

N-gram 平滑

- Interpolate 插值法 (更进一步)

$$P_{int}(w_i|w_{i-1},w_{i-2}) = \lambda_3 P_{ML}(w_i|w_{i-1},w_{i-2}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_1 P_{ML}(w_i)$$

$$\lambda_k \rightarrow \lambda_{k,w_{i-2}^{i}}$$

根据不同的上下文，选择不同的参数

N-gram 平滑

- Absolute Discounting “绝对折扣”

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1})$$

递归停止： Unigram 或者 zero-gram

有钱的，每个人交固定的税D, 建立一个基金；
没钱的，根据自己爸爸有多少钱来分了这个基金；

N-gram 平滑

- Absolute Discounting “绝对折扣”

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1})$$

$$\lambda_{w_{i-n+1}^{i}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet)$$

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i|c(w_{i-n+1}^{i}) > 0\}|$$

N-gram 平滑

Absolute Discounting “绝对折扣”

估算 D: leave-one-out

初始的训练集: a b c d c a d c

各个字母出现的次数r
a：2次
b：2次
c：3次
d：1次

p(a) = p_2
p(b) = p_2
p(c) = p_3
p(d) = p_1

出现r次的字母的个数 n_r
n_1 = 1
n_2 = 2
n_3 = 1

$$L = 2\log(p(a)) + 2\log(p(b)) + 3\log(p(c)) + \log(p(d))$$
$$= 2\log(p_2) + 2\log(p_2) + 3\log(p_3) + \log(p_1)$$
$$= \sum_{r=1}^{3} r n_r \log(p_r)$$

N-gram 平滑

Absolute Discounting “绝对折扣”

估算 D: leave-one-out

原来的训练集: a b c d c a d c

Train	Dev(one)
~~a~~ b c d c a d c	a
a ~~b~~ c d c a d c	b
a b ~~c~~ d c a d c	c
a b c ~~d~~ c a d c	d
a b c d ~~c~~ a d c	c
a b c d c ~~a~~ d c	a
a b c d c a ~~d~~ c	d
a b c d c a d ~~c~~	c

各个字幕在新的**Train**中出现的次数r`

a：1次
b：1次
c：2次
d：0次

p(a) = p_1
p(b) = p_1
p(c) = p_2
p(d) = p_0

N-gram 平滑

Absolute Discounting “绝对折扣”

估算 D: leave-one-out

原来的训练集: a b c d c a d c

各个字幕在新的**Train**中出现的次数r`

a：1次
b：1次
c：2次
d：0次

p(a) = p_1
p(b) = p_1
p(c) = p_2
p(d) = p_0

$$L = 2\log(p(a)) + 2\log(p(b)) + 3\log(p(c)) + \log(p(d))$$
$$= 2\log(p_1) + 2\log(p_1) + 3\log(p_2) + \log(p_0)$$
$$= \sum_{r=1}^{3} r n_r \log(p_{r-1})$$

N-gram 平滑

- Absolute Discounting “绝对折扣”

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1})$$

$$\lambda_{w_{i-n+1}^{i}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet)$$

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i|c(w_{i-n+1}^{i}) > 0\}|$$

$$D = \frac{n_1}{n_1 + 2n_2}$$

N-gram 平滑

Kneser-Ney Smoothing

回顾 Absolute Discounting:

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1})$$

递归停止： Unigram $P(w_i)$

填空： 我 想 买 太阳 _____

A 像章 count(太阳, 像章) = 0 count(像章) = 100
毛主席像章，金正恩像章，...

B 老铁 count(太阳，老铁) = 0 count(老铁) = 1000
扎心了老铁

需要选一个更加“适配”的词

N-gram 平滑

Kneser-Ney Smoothing

回顾 Absolute Discounting:

$$P_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{abs}(w_i|w_{i-n+2}^{i-1})$$

递归停止： Unigram $P(w_i)$

(刘强西 奶茶) * 40
(玉思聪 ZB) * 10, (玉思聪 LY) * 10, (玉思聪 LPD) * 10

(__, 网红SLD) ?

需要选一个更加“交际广泛”的人

N-gram 平滑

- Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{KN}(w_i|w_{i-n+2}^{i-1})$$

递归停止： Unigram $P(w_i)$

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}$$

有钱的，每个人固定的税 D, 建立一个基金；
没钱的，根据自己爸爸“交际广泛”的程度来分了这个基金；

N-gram 平滑

Modified Kneser-Ney Smoothing

回顾 KN Smoothing:

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i}) - D, 0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \lambda_{w_{i-n+1}^{i}} P_{KN}(w_i|w_{i-n+2}^{i-1})$$

设置三个D:
D_1 如果 c = 1
D_2 如果 c = 2
D_{3+} 如果 c >= 3

有钱的，每个人根据自己的收入交不同的税D, 建立一个基金；
没钱的，根据自己爸爸“交际广泛”的程度来分了这个基金；

N-gram 语言模型

总结

模型	简单理解	只需要记住
+1 平滑	政府印钱	没用
Backoff	用爸爸的钱	
Interpolate	自己和爸爸都出点	Development Set; EM
Absolute Discouting	有钱人缴固定税， 按爸爸的资产分配	Leave-one-out;
Kneser-Ney	有钱人缴固定税， 按爸爸人脉分配	词的适配度
Modified KN	有钱人缴阶梯税， 按爸爸人脉分配	阶梯税率 最好的方法！

N-gram 语言模型

- 工具: KenLM
  - https://kheafield.com/code/kenlm/
  - Modified Kneser-Ney Smoothing
  - 演示

N-gram 语言模型

- 应用案例：完形填空
  - “they were suddenly 2 forward. At that moment, the air-hostess 3 . She looked very pale, but was quite 4 .”
    - 2.A.shifted B.thrown C.put D.moved
  - 假如使用3gram LM
    - p(“were suddenly shifted forward .”)
    - p(“were suddenly thrown forward .”)
    - p(“were suddenly put forward .”)
    - p(“were suddenly moved forward .”)

N-gram 语言模型

- 应用案例：完形填空
  - https://github.com/shixing/xing_nlp/tree/master/LM
  - 代码运行环境
    - Anaconda (python 2.7)
    - NLTK
      - pip install nltk
      - 需要下载在nltk里面下载punck
    - KenLM
      - https://github.com/kpu/kenlm
      - python module:
        - pip install https://github.com/kpu/kenlm/archive/master.zip

N-gram 语言模型

- 应用案例：完形填空
  - http://edu.sina.com.cn/en/2002-09-27/6063.html
  - “they were suddenly 2 forward. At that moment, the air-hostess 3 . She looked very pale, but was quite 4 .”
    - 2.A.shifted B.thrown C.put D.moved
    - 3.A.showed B.presented C.exposed D.appeared
    - 4.A.well B.still C.calm D.quiet

N-gram 语言模型

潜在的问题:

P(梨|水果 包含) P(苹果|水果 包含)

对“词”的理解有限
N-gram 上下文的长度有限

提纲

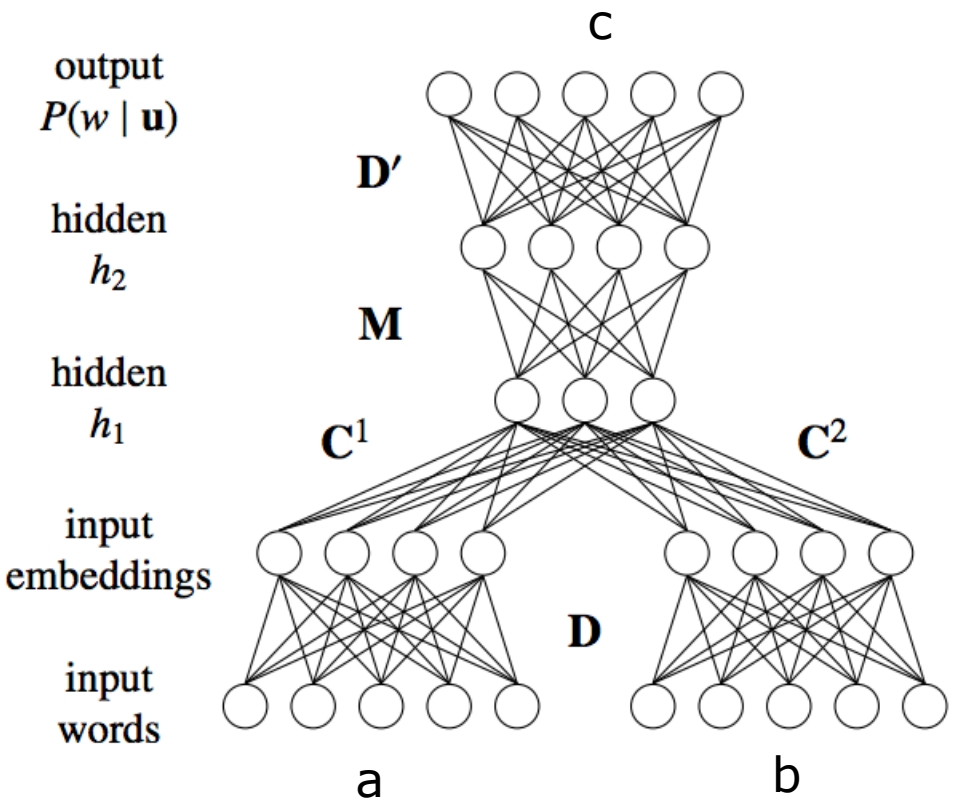
- 什么是语言模型
- N-gram 语言模型
- 语言模型的评价
- OOV
- 平滑方法
- NNLM / RNNLM
- 杂谈

神经网络语言模型

- Neural Network Language Model
  - ~~对“词”的理解有限：语义相似性~~
- Recurrent Neural Network Language Model
  - ~~N-gram 上下文的长度有限~~
- One more advanced model:
  - Recurrent Highway Network

Neural Network Language Model

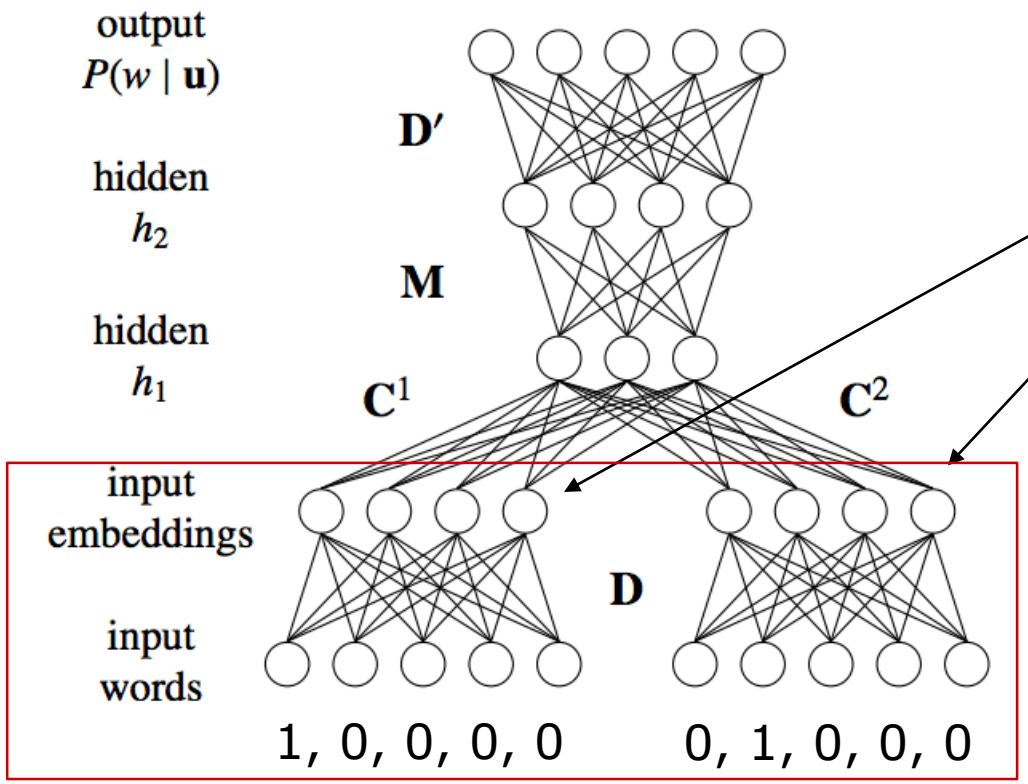
“a b c”



Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.

Neural Network Language Model

V = {a,b,c,d,e}　句子： a b c



Embedding Lookup

D = R^{5*4}

[0.2, -0.1, -0.5, 0.2],
[0.1, -0.2, 0.2, 0.1],
[-0.3, 0.2, 0.3, -0.2],
[-0.3, -0.2, -0.3, 0.2],
[0.1, 0.1, -0.3, -0.4]

embedding = dot(input,D)

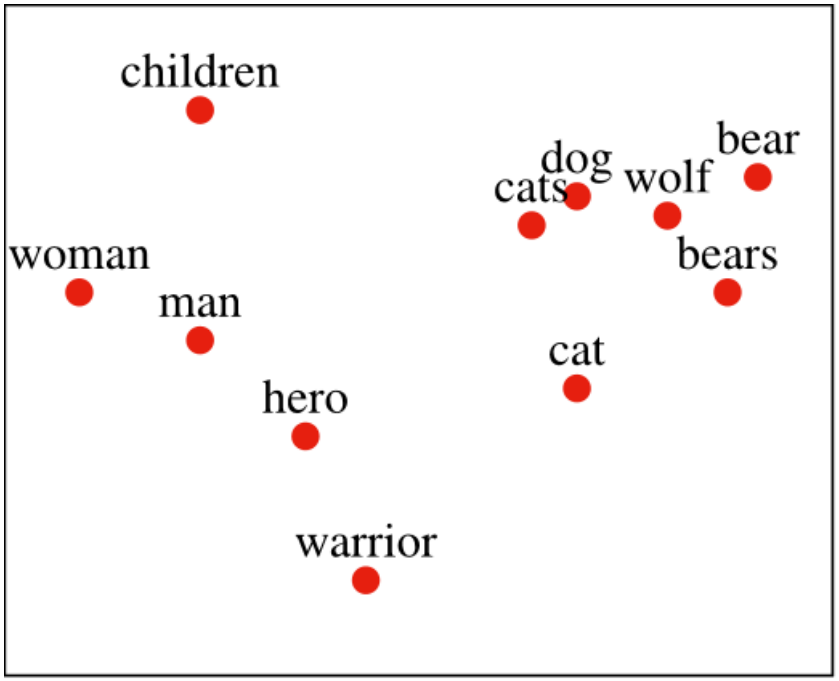
Neural Network Language Model

Embedding的作用：~~对“词”的理解有限~~

词法的相似性
good, better

语法的相似性
see, saw

语义相似性
dog, cat



Neural Network Language Model

为什么Embedding会有这样的作用？

Embedding的每一维相当于是机器学习出来的特征

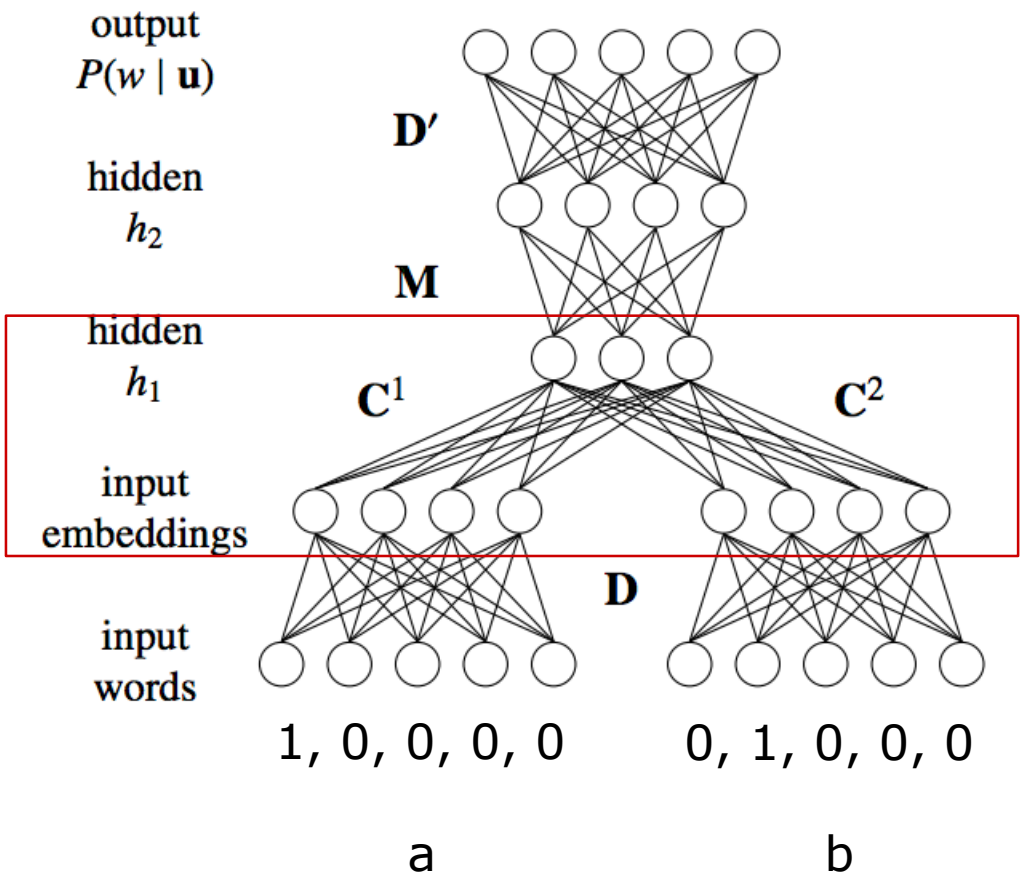
所谓的“特征学习/表示学习”
representation learning

ICLR
International Conference on Learning Representations

Deep learning的核心原因之一

Neural Network Language Model

V = {a,b,c,d,e}　句子： a b c



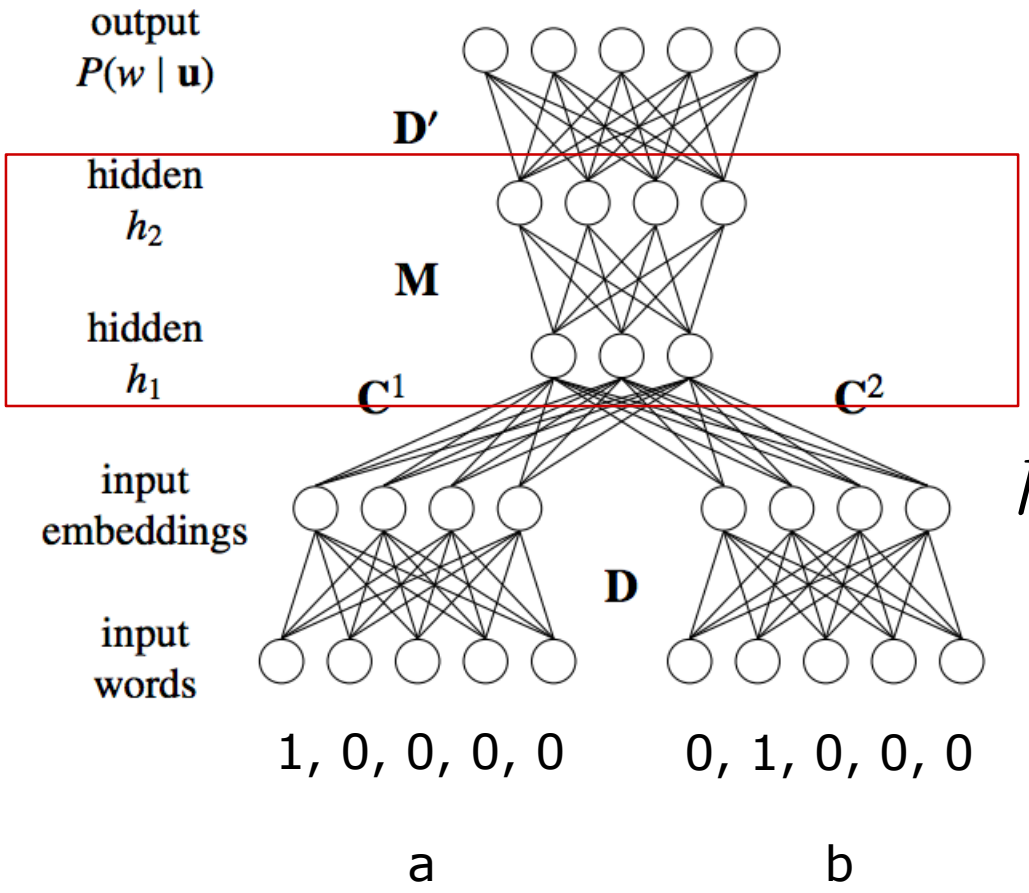
hidden h_1 的计算

$C\_1 = R^{4*3}$
$C\_2 = R^{4*3}$
$b\_1 = R^{1*3}$

$$h_1 = tanh(e_1 * C^1 + e_2 * C^2 + b_1)$$

Neural Network Language Model

V = {a,b,c,d,e}　句子： a b c



hidden h_2 的计算

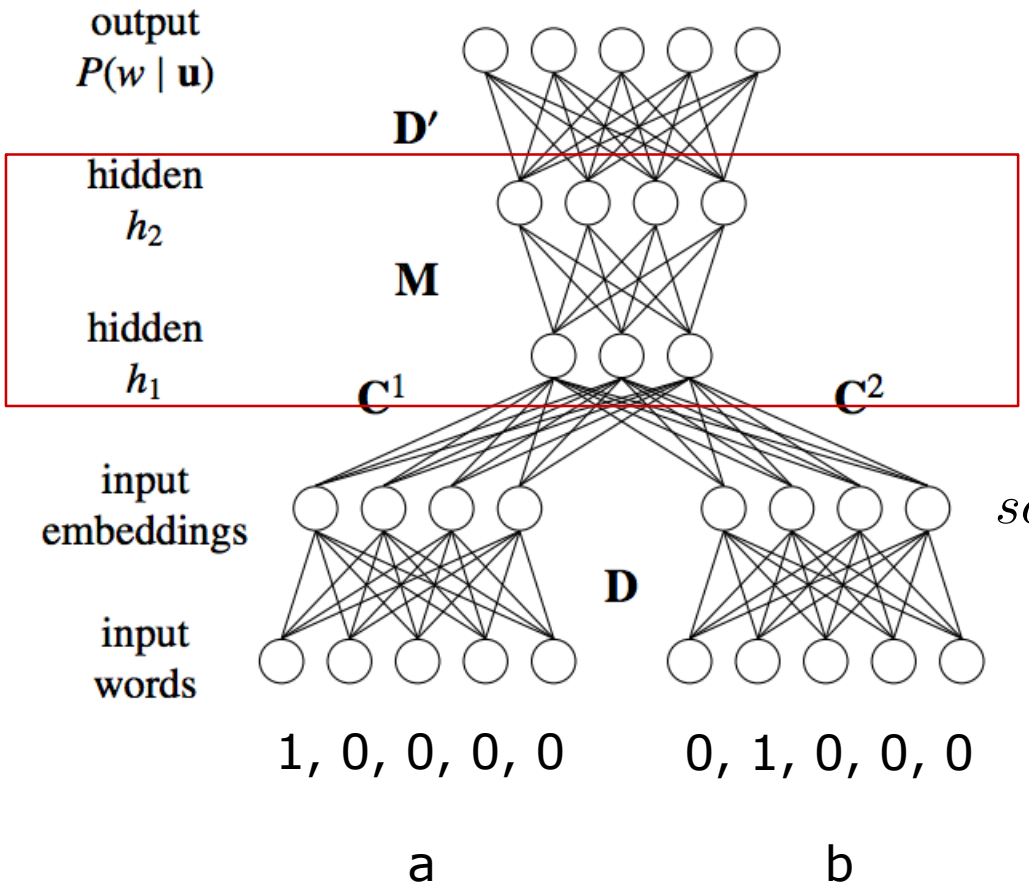
M = R^{3*4}

b_2 = R^{1*4}

$$h_2 = tanh(h_1 * M + b_2)$$

Neural Network Language Model

V = {a,b,c,d,e}　句子： a b c



softmax 的计算

D′ = R4*5

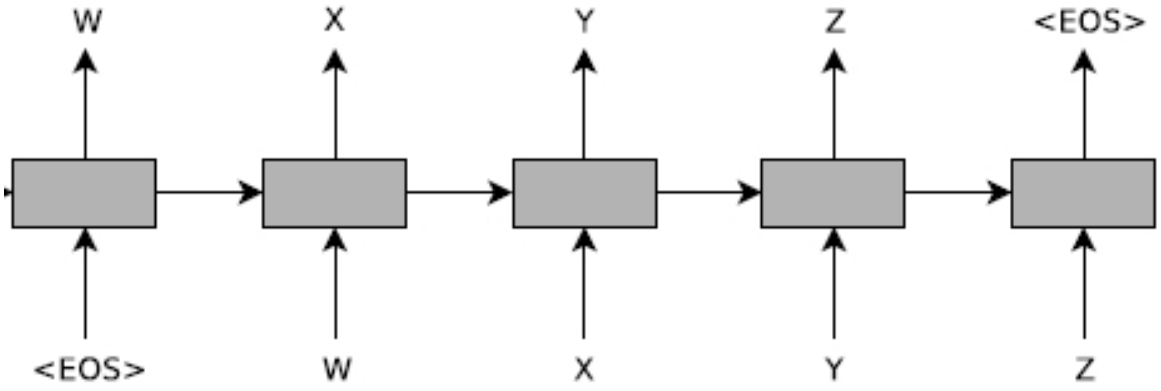
b_3 = R1*5

$$p(w|u) = softmax(h_2 * D' + b_3)$$

$$softmax(x[i]|x \in R^{1*n}) = \frac{e^{x[i]}}{\sum_j e^{x[j]}}$$

Recurrent Neural Network LM

句子："w x y z"



~~N-gram 上下文的长度有限~~

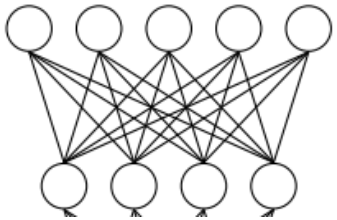
Recurrent Neural Network LM

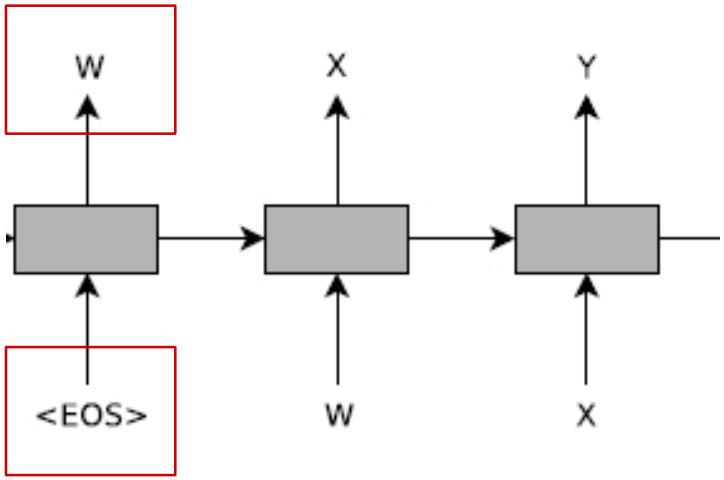
Softmax Layer

output
$P(w \mid \mathbf{u})$

$\mathbf{D}'$

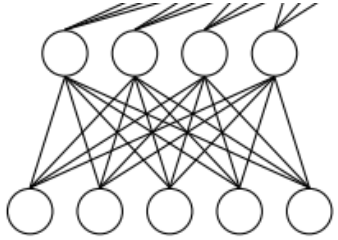
hidden
h_2





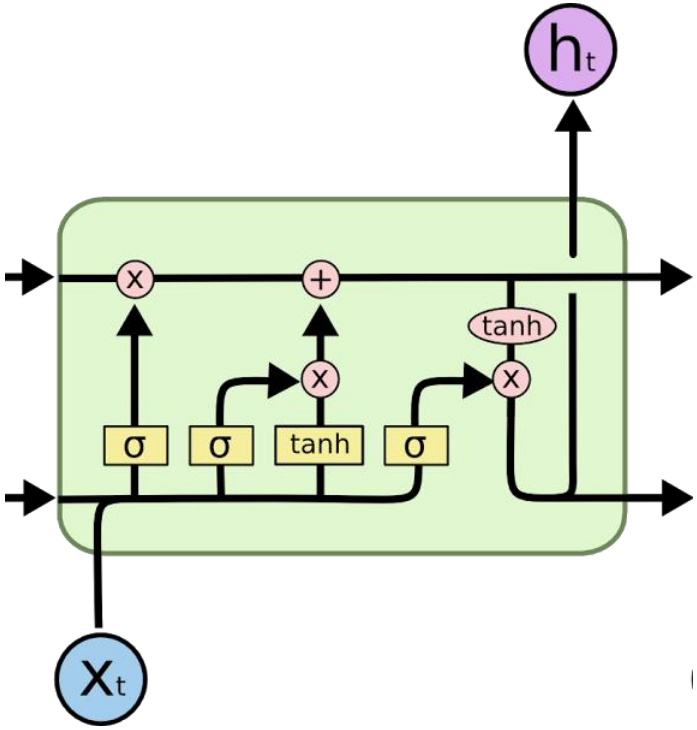
input
embeddings

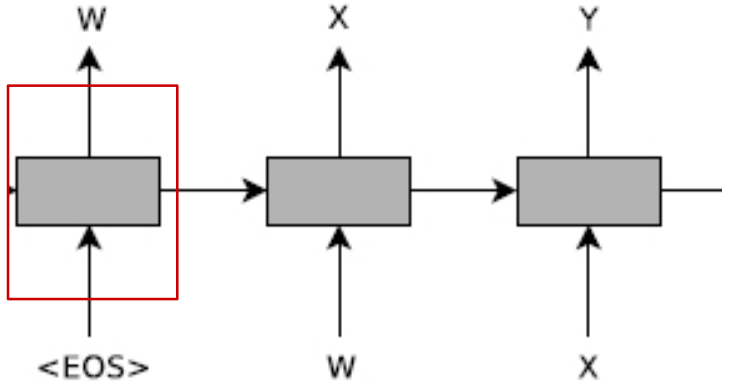
input
words



Embedding Lookup

Recurrent Neural Network LM

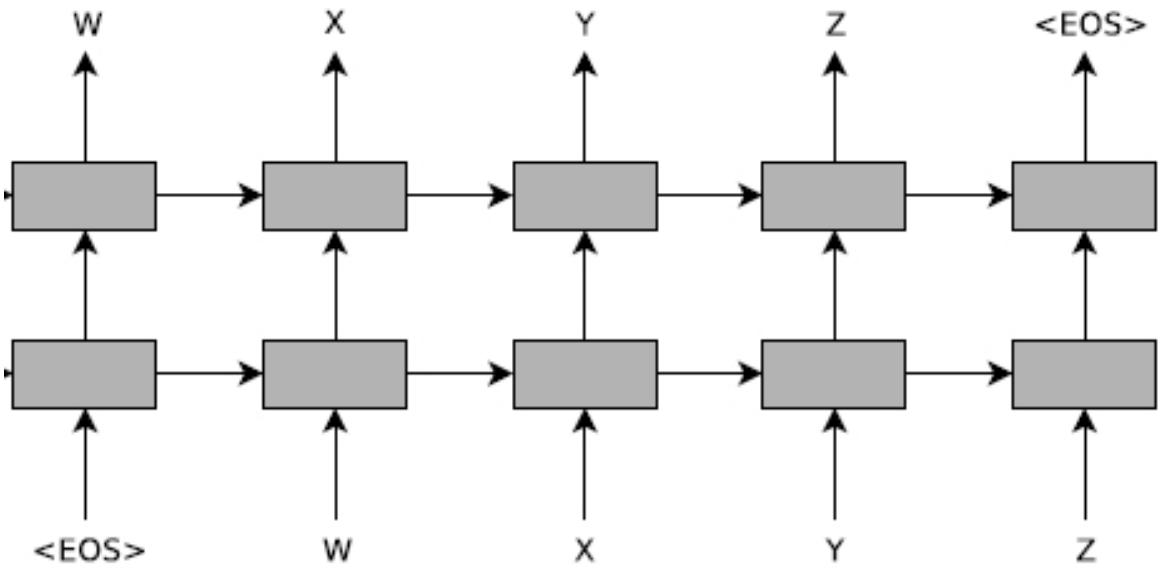




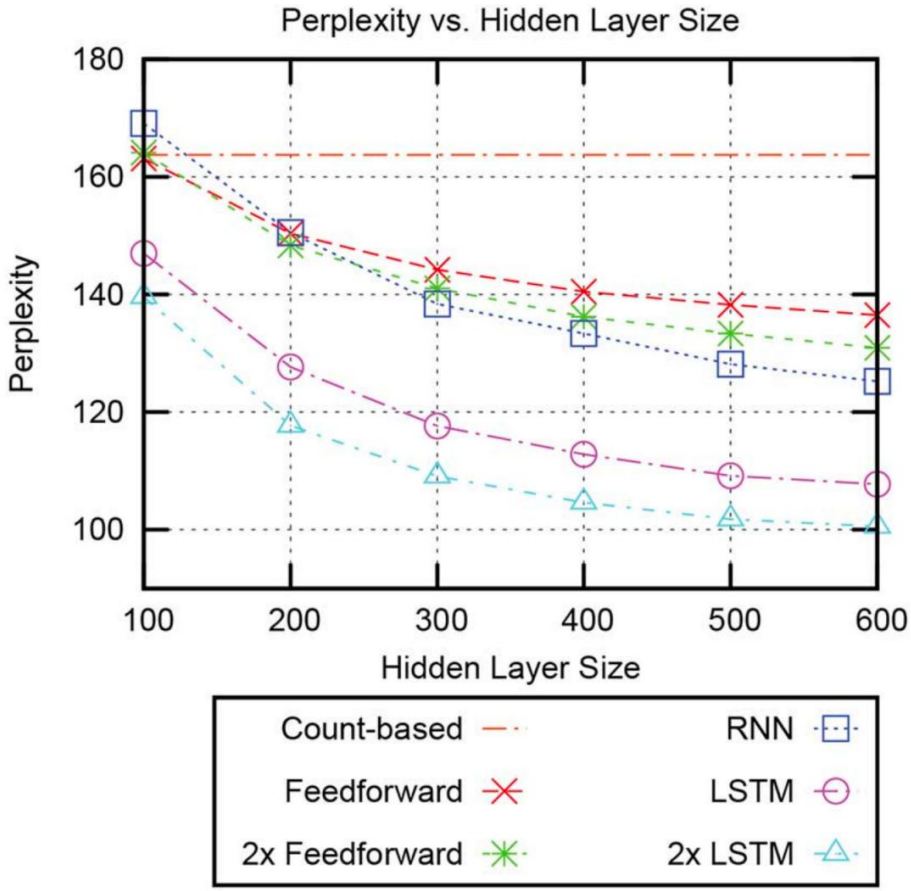
Long Short Term Memory Unit

Recurrent Neural Network LM

多层叠加

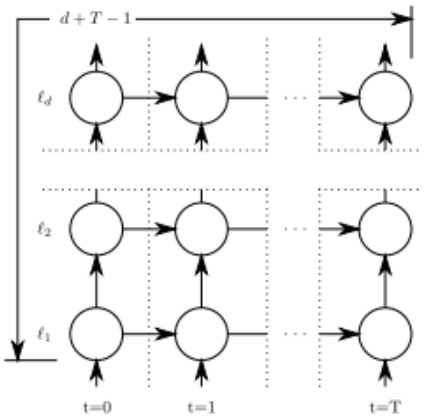


Recurrent Neural Network LM

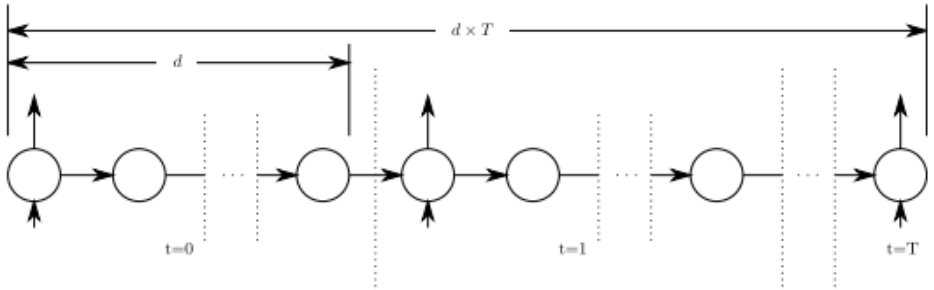


(from Sundermeyer, Ney, Schlüter, IEEE TASLP 2015)

Recurrent Highway Network



(a)



Zilly, Julian Georg, et al. "Recurrent highway networks." *arXiv preprint arXiv:1607.03474* (2016).

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：大数据分析挖掘
- 新浪微博：ChinaHadoop