从Word2Vec到FastText

文本处理大杀器

七月在线 加号

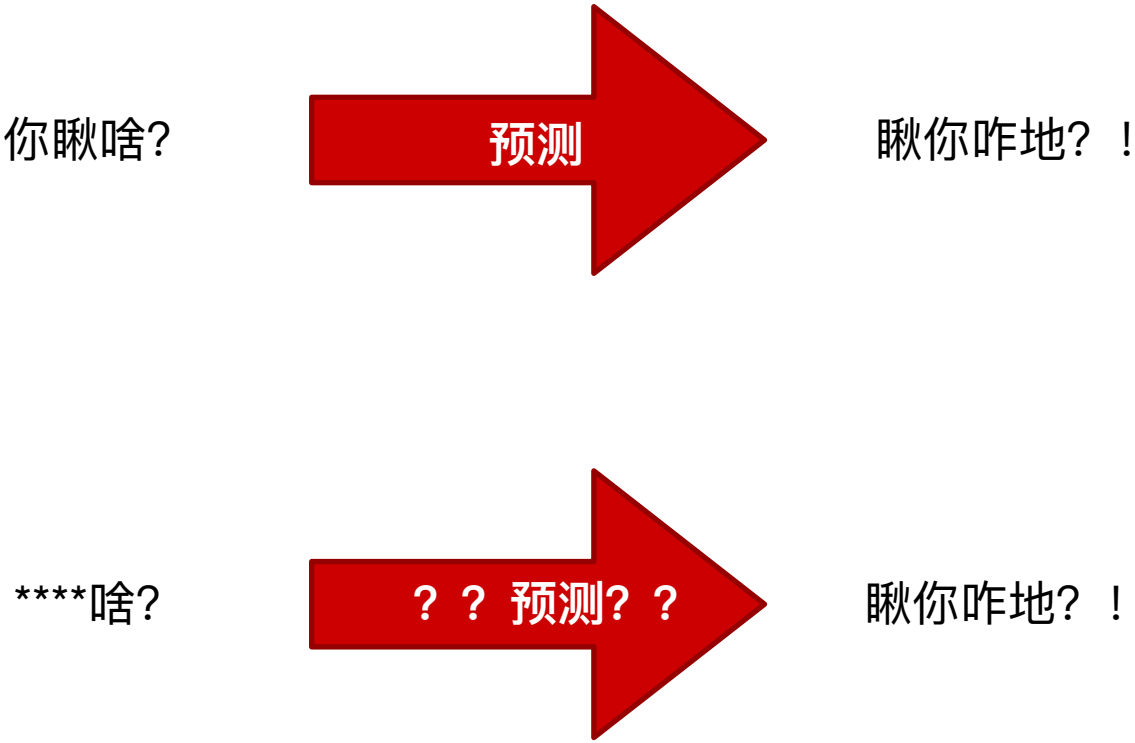
微博：@翻滚吧_加号

2016年11月20日

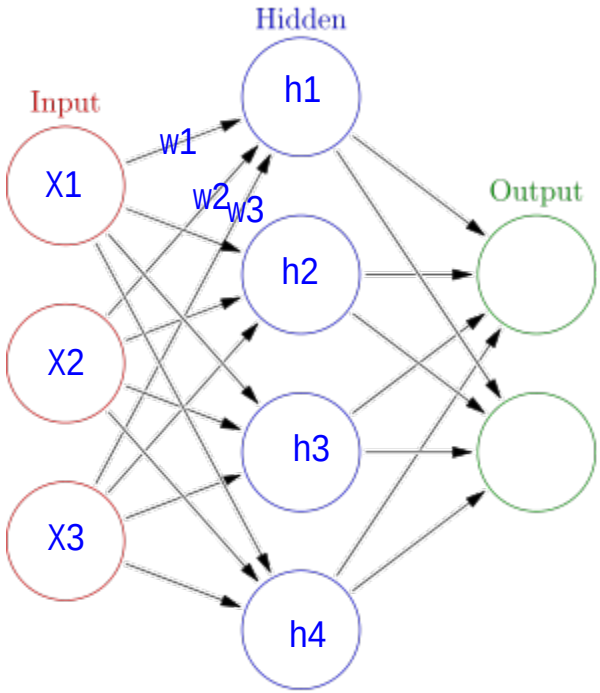
目录

- Word2Vec在深度学习中的应用
  - 文本生成（Word2Vec + RNN/LSTM）
  - 文本分类（Word2Vec + CNN）
- FastText
  - 原理
  - 应用

文本生成



普通神经网络



f函数可取:sigmoid,正切函数等

h1 = f(w1*x1+w2*x2+w3*x3)

带记忆神经网络

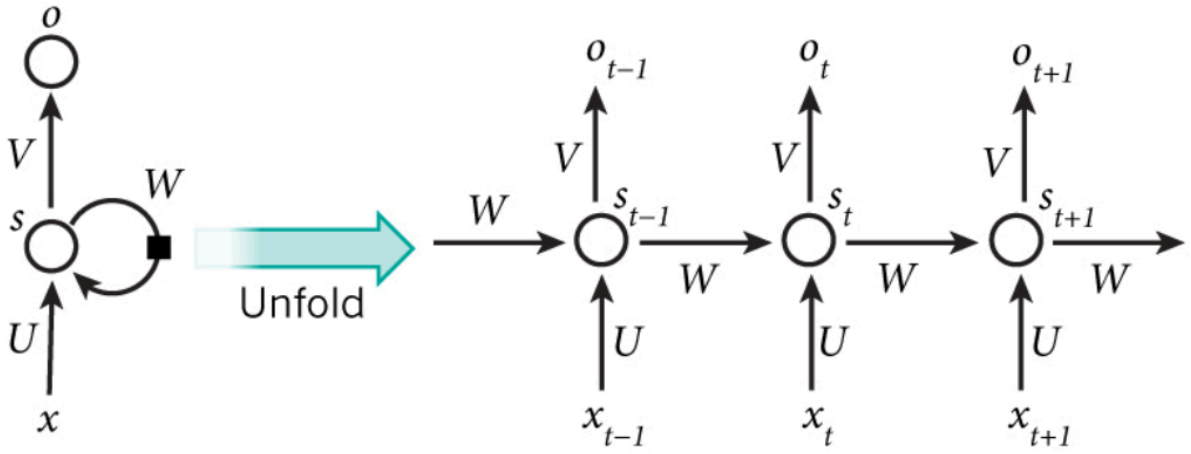
所以说，光直接feed不行，我们希望我们的分类器能够记得上下文前后文关系：

RNN的目的是让有sequential关系的信息得到考虑。

什么是sequential关系?
就是信息在时间上的前后关系。

相比于普通神经网络：

RNN



当前的记忆 = 权重U*输入 + 权重W*上一次的记忆

每个时间点中的S计算

$$s_t = f(Ux_t + Ws_{t-1})$$

这个神经元最终的输出，
基于最后一个S

$$o_t = \text{softmax}(Vs_t)$$

简单来说，对于t=5来说，其实就相当于把一个神经元拉伸成五个
换句话说，S就是我们所说的记忆（因为把t从1-5的信息都记录下来了）

记忆

由前文可见，RNN可以带上记忆。

假设，一个『生成下一个单词』的例子：

『这顿饭真好』——>『吃』

很明显，我们只要前5个字就能猜到下一个字是啥了

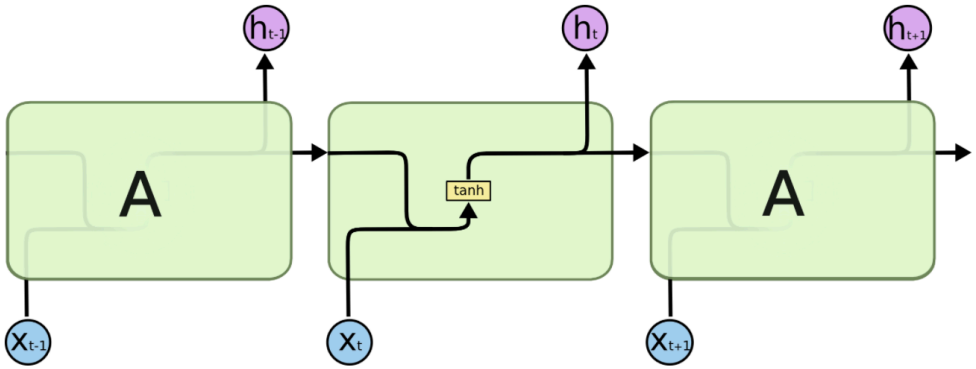
However，

如果我问你，『穿山甲说了什么？』

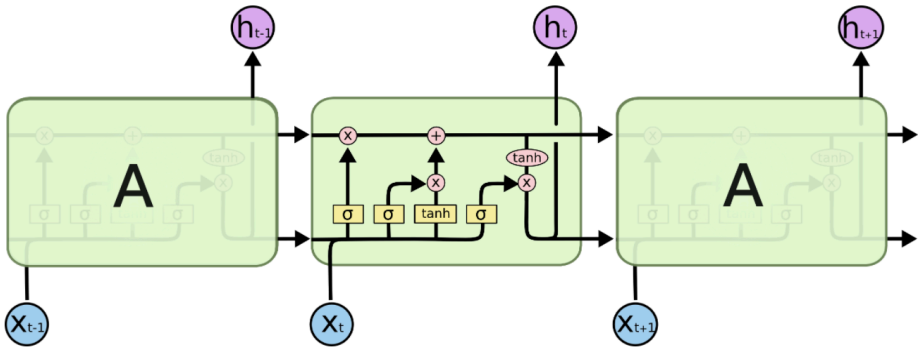
你能回答嘛?

(credit to 暴走漫画)

LSTM 长效短期记忆:RNN是什么都记住,LSTM是只记住该记住的

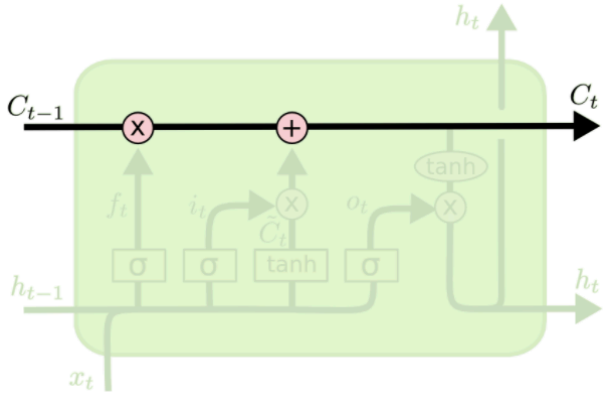


RNN

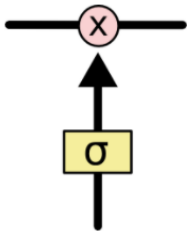


LSTM

LSTM



LSTM中最重要的就是这个Cell State，
它一路向下，贯穿这个时间线，
代表了记忆的纽带。
它会被XOR和AND运算符搞一搞，
来更新记忆



而控制信息的增加和减少的，
就是靠这些阀门：Gate

阀门嘛，就是输出一个1于0之间的值：
1 代表，把这一趟的信息都记着
0 代表，这一趟的信息可以忘记了

LSTM

下面我们来模拟一遍信息在LSTM里跑跑~

第一步：忘记门

来决定我们该忘记什么信息

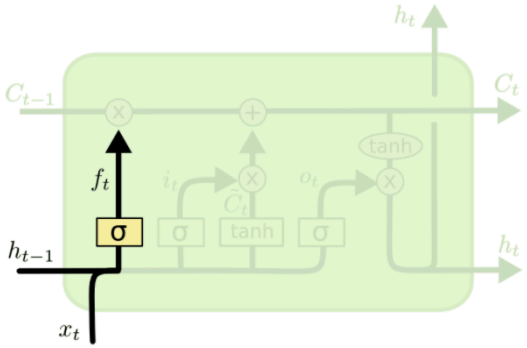
它把上一次的状态ht-1和这一次的输入xt相比较

通过gate输出一个0到1的值（就像是个activation function一样），

1 代表：给我记着！

ft = 0或1

0 代表：快快忘记！



Wf：权重　　bf：长度偏差值

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM

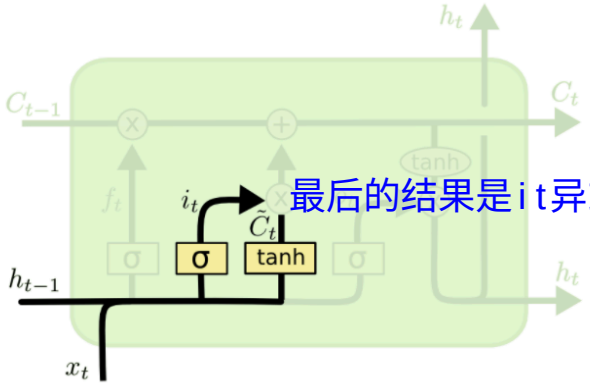
第二步：记忆门

哪些该记住

这个门比较复杂，分两步：

第一步，用sigmoid决定什么信息需要被我们更新（忘记旧的）

第二部，用Tanh造一个新的Cell State（更新后的cell state）如果不考虑之前的记忆，当前时刻的状态是怎么样的(应该会记住什么)



最后的结果是it异或Ct

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

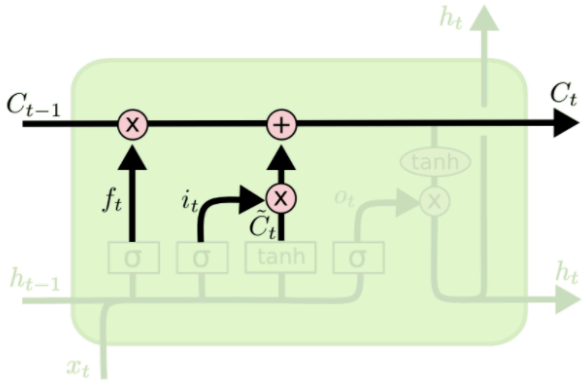
LSTM

第三步：更新门

把老cell state更新为新cell state

用XOR和AND这样的门来更新我们的cell state：

异或



C_t :时刻t的记忆该更新什么状态

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

该忘记的　　应该记住的

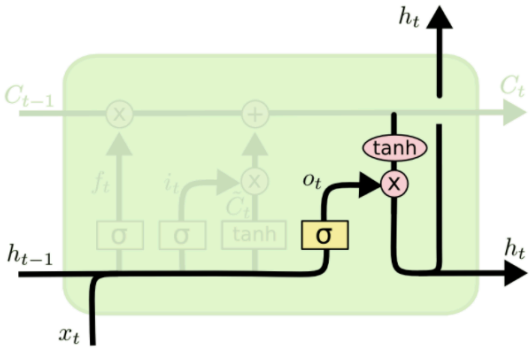
LSTM

第四步：输出门

由记忆来决定输出什么值

我们的Cell State已经被更新，
于是我们通过这个记忆纽带，来决定我们的输出：
（这里的Ot类似于我们刚刚RNN里直接一步跑出来的output）

ht：被记忆所处理过的h，会作为下一个LSTM的输入



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

哪些需要记住，哪些需要忘记

案例

题目原型：What’s Next?

可以用在不同的维度上：

维度1：下一个字母是什么?

案例

维度2：下一个单词是什么？

案例

维度3：下一个句子是什么?

问答机器人

案例

维度N：下一个图片/音符/....是什么？

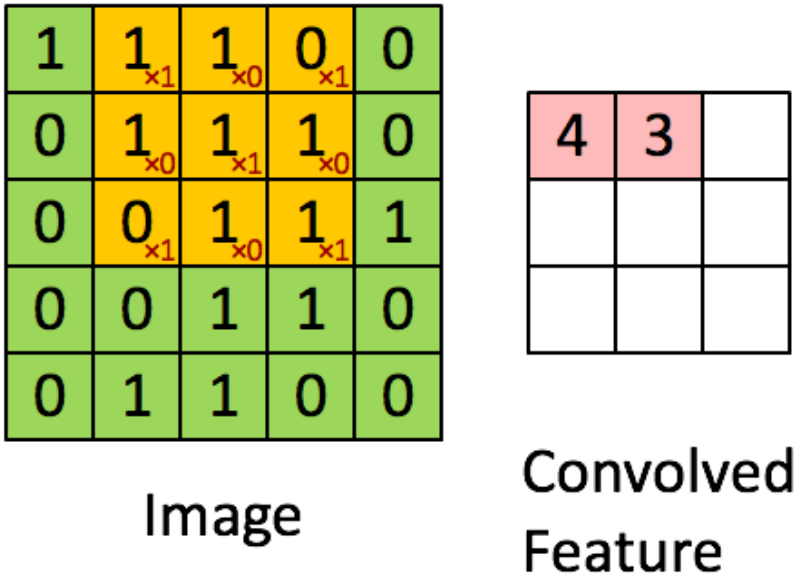
代码

【详见随堂 iPython Notebook】

文本分类

Baseline	BoW + SVM (Fan et al. 2008)
Deep Learning	CNN for Text (Kim, 2014)

CNN4Text



CNN4Text

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

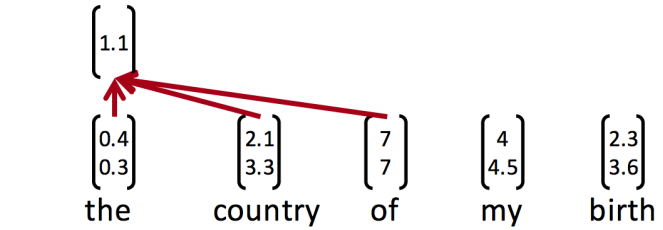
	0	1	0	
	1	-4	1	
	0	1	0	

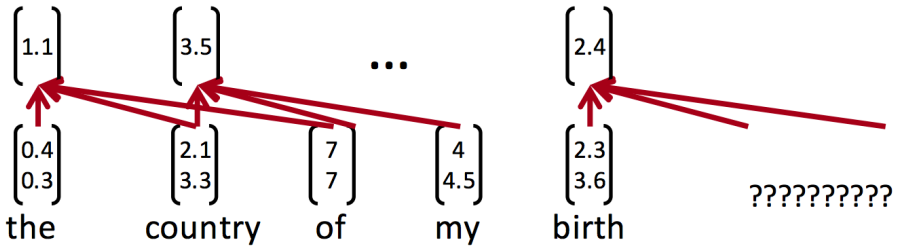
CNN4Text

如何迁移到文字处理?

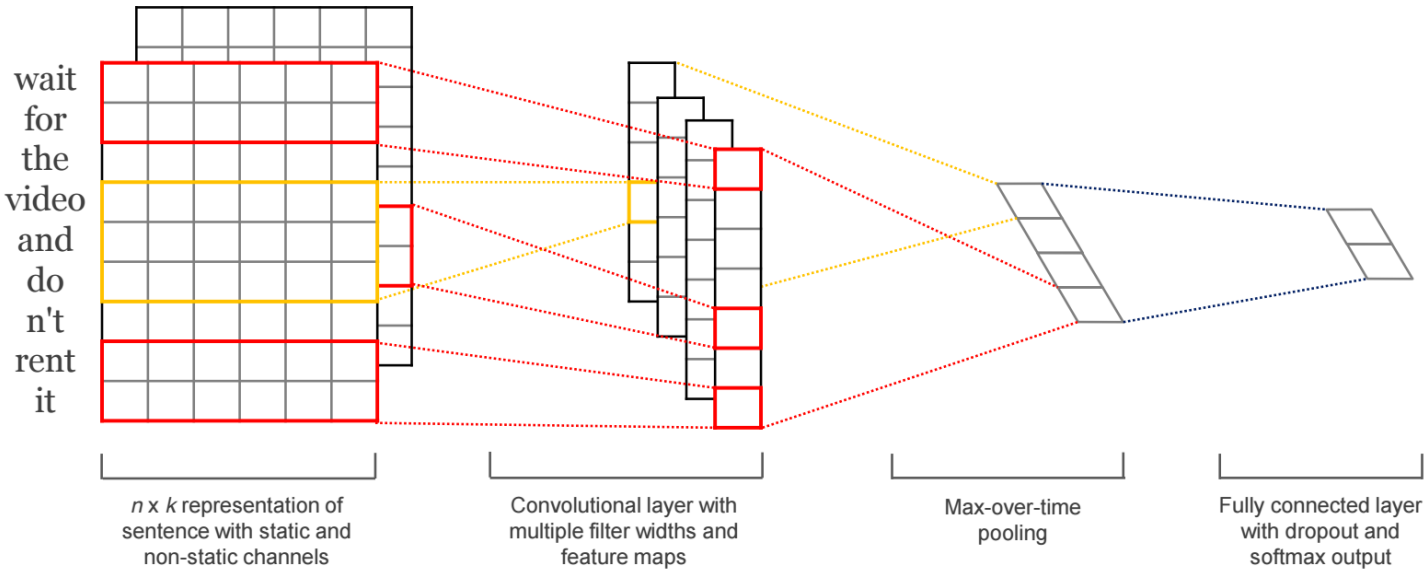
1. 把文字表示成图片

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$



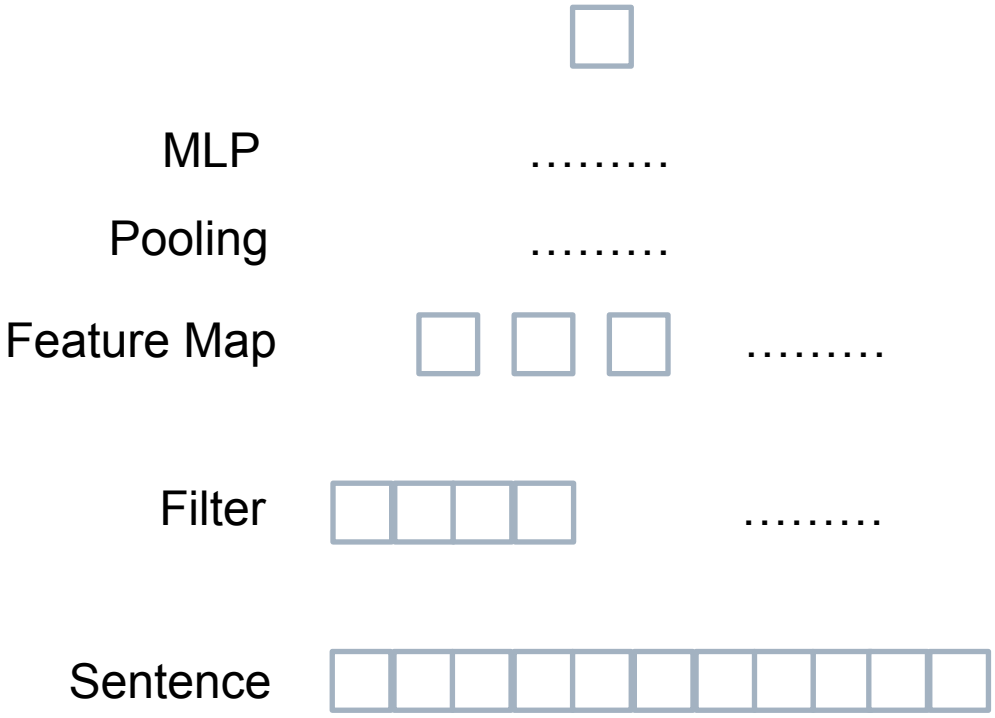


CNN4Text


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

CNN4Text

2. 把CNN做成1D



CNN4Text

RNN hypothesis

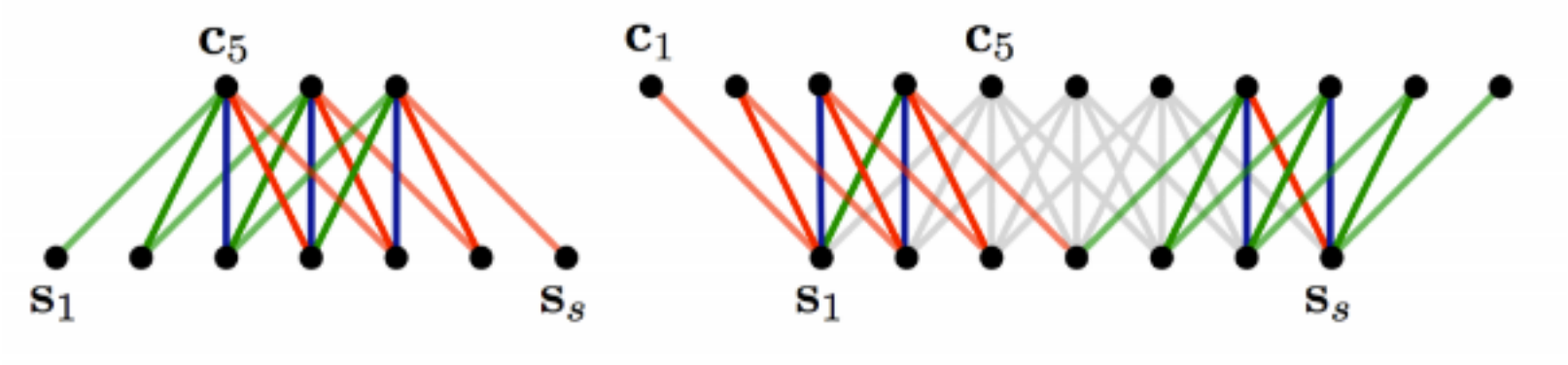
CNN4Text

CNN hypothesis

你们到看这话句，
并不会到感违和。

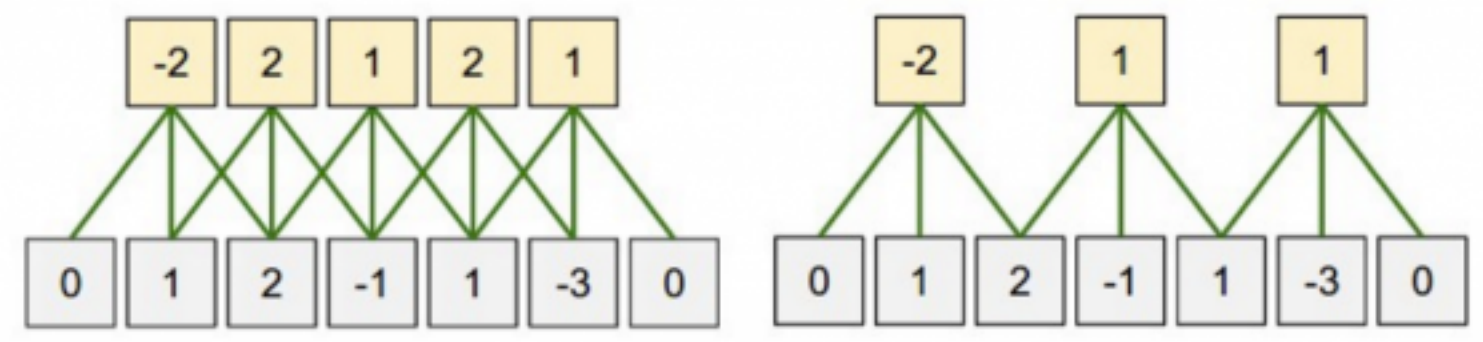
CNN4Text

边界处理:
Narrow vs Wide



CNN4Text

Stride size:
步伐大小



代码

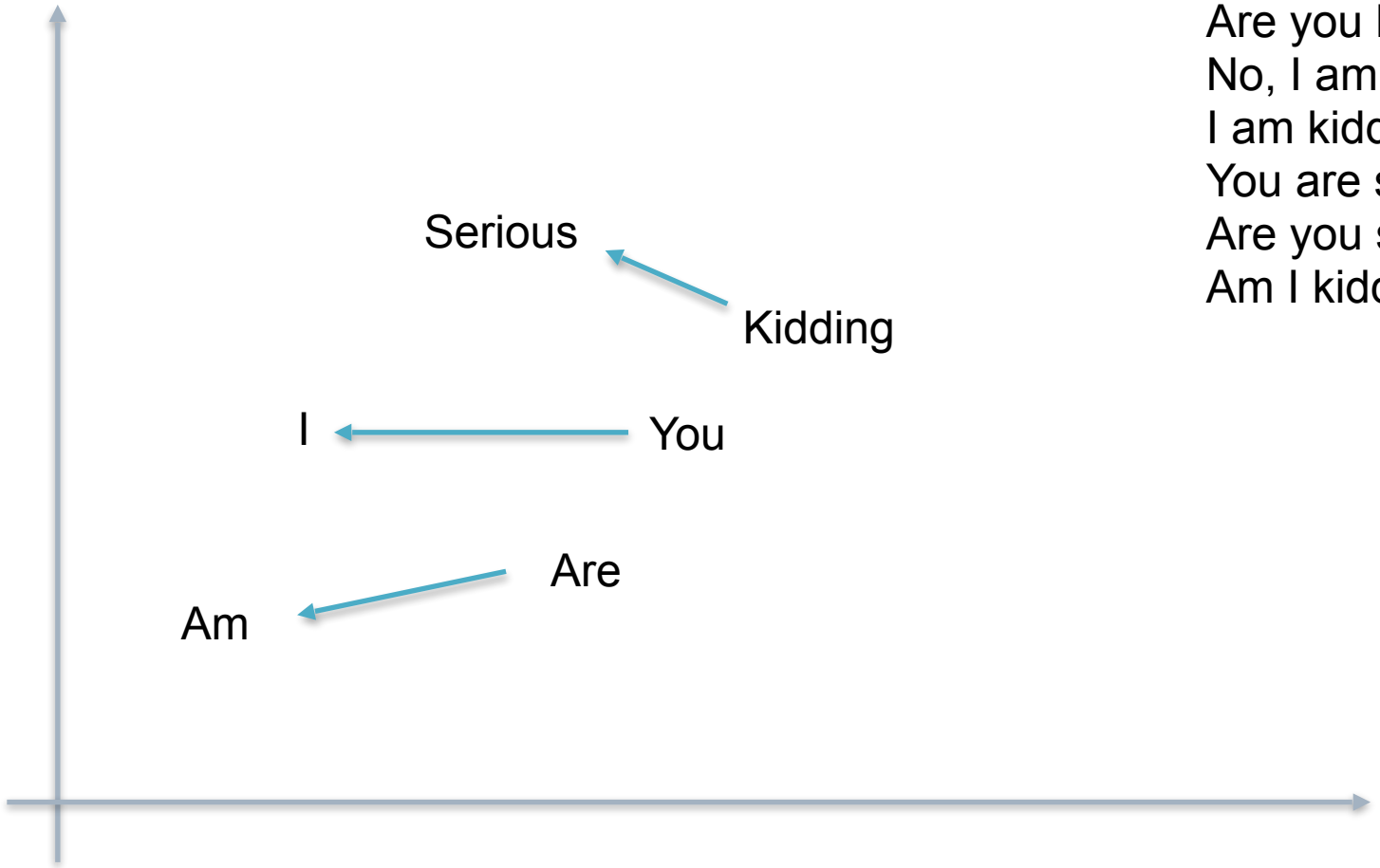
【详见随堂 iPython Notebook】

FastText

Bag of Tricks for Efficient Text Classification

Armand Joulin **Edouard Grave** **Piotr Bojanowski** **Tomas Mikolov**
Facebook AI Research
{ajoulin,egrave,bojanowski,tmikolov}@fb.com

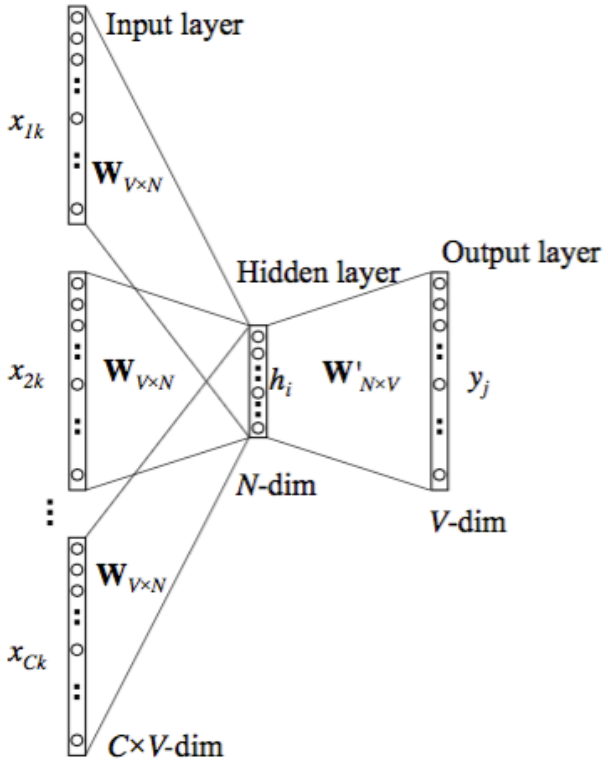
回顾Word2Vec



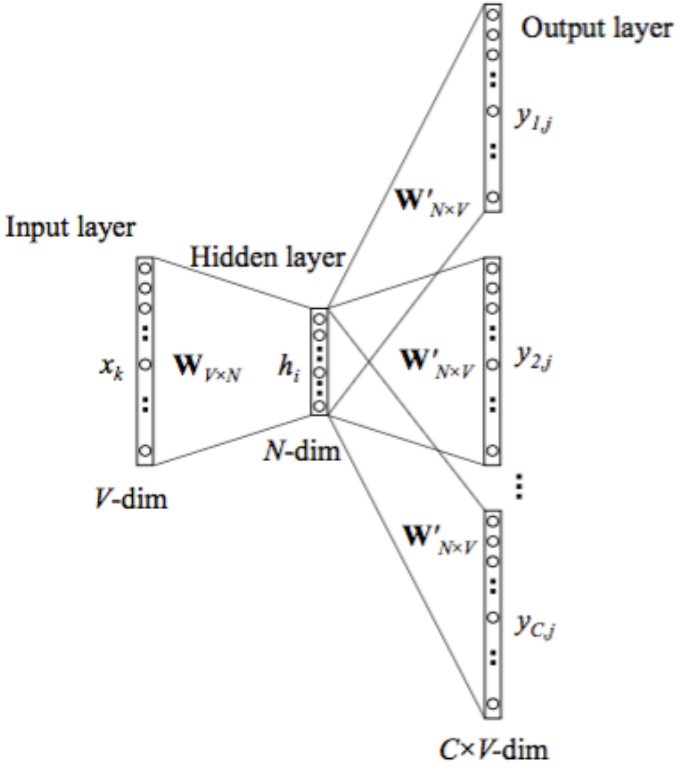
Are you kidding?
No, I am serious.
I am kidding.
You are serious.
Are you serious?
Am I kidding?

回顾Word2Vec

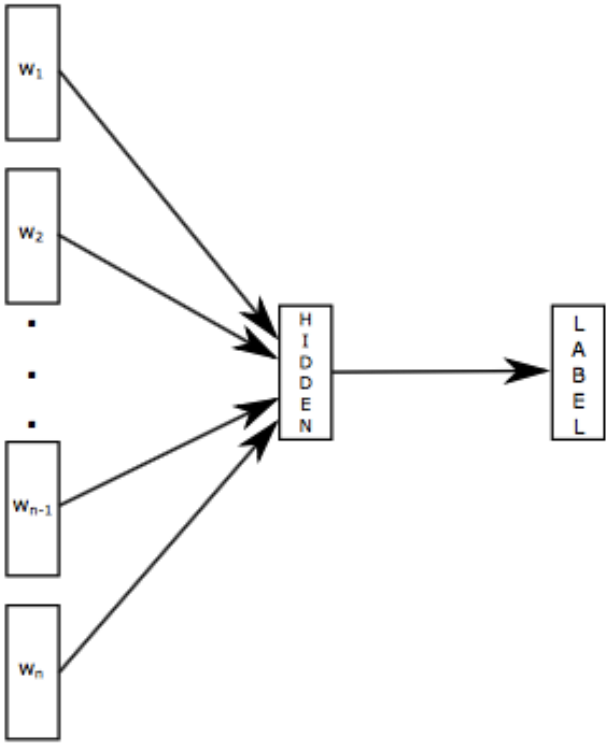
CBoW



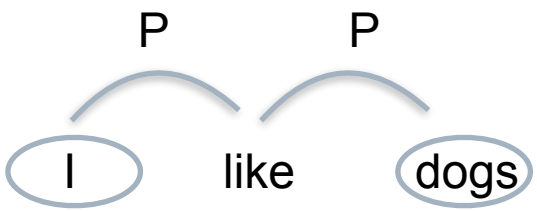
Skip-Gram

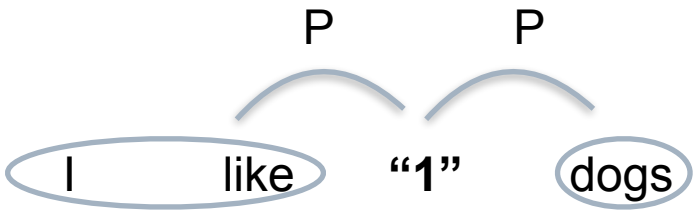


FastText模型构架



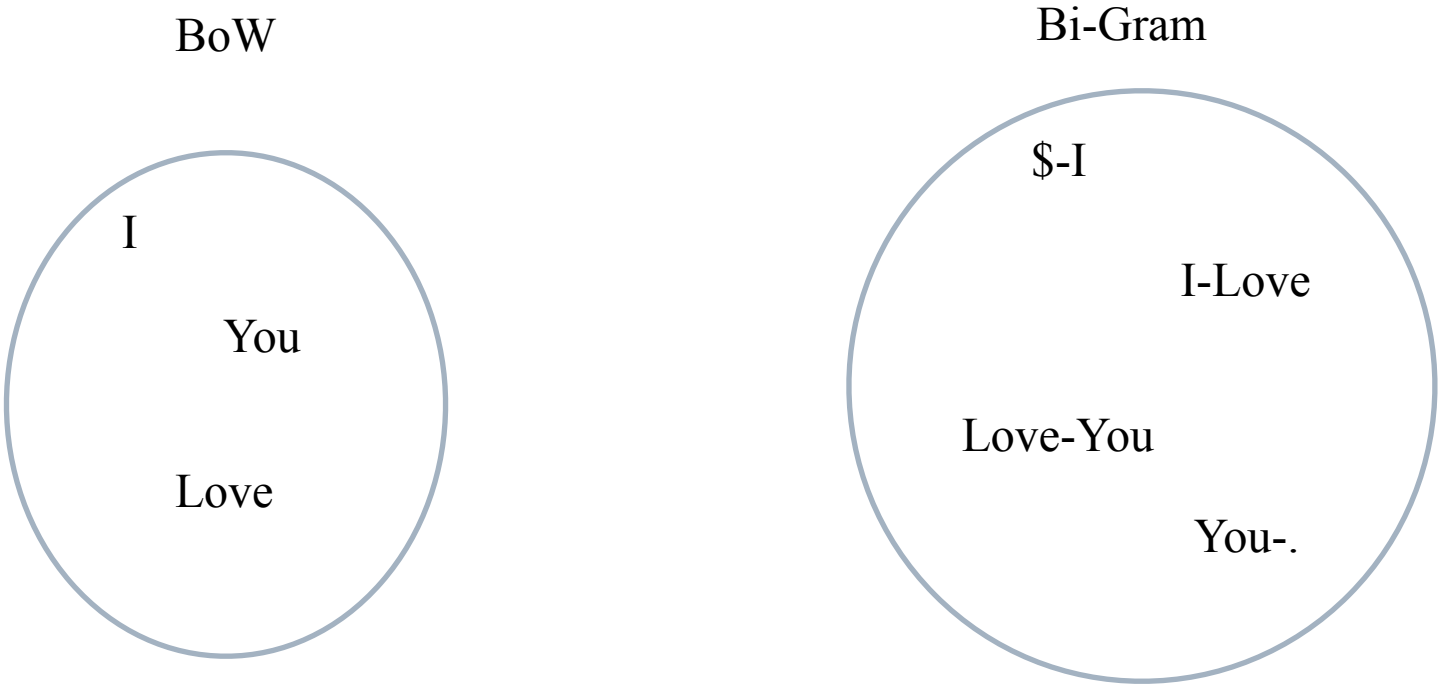
模型意义





模型改进

I Love You.



模型改进

普通玩法

1: hi, 2: hello, 3: what, 4:….

Kernel Trick

$$k(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle$$

Hashing Trick

“What dose the fox say?”

h(what) mod 5 = 0
h(does) mod 5 = 1
h(the) mod 5 = 1
h(fox) mod 5 = 1
h(say) mod 5 = 3

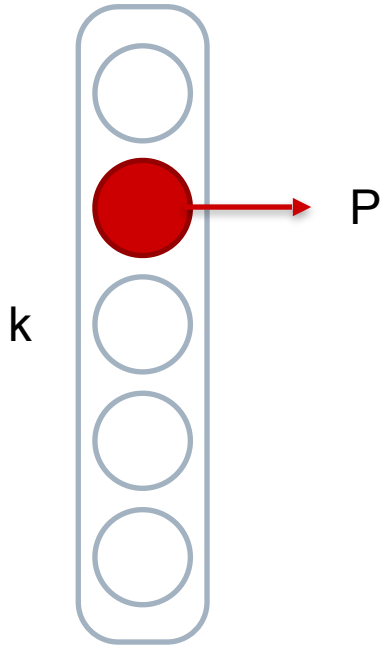
(1, 3, 0, 1, 0)

$h_2(x)$ = +1 / -1

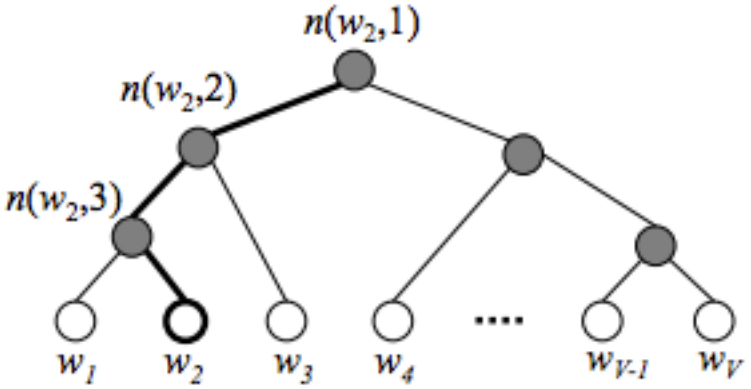
模型改进

Softmax

$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$$

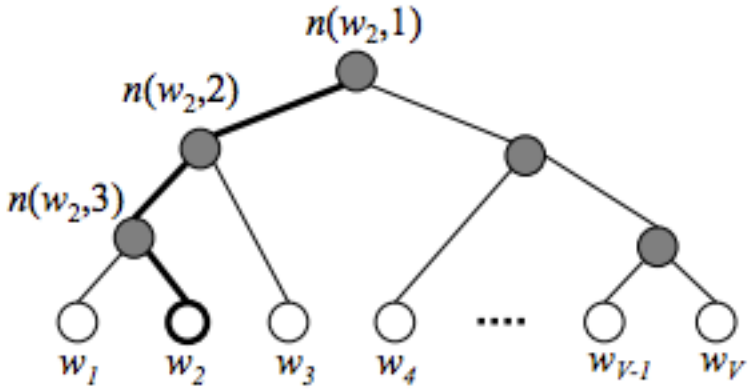


Hierarchy Softmax



模型改进

深度优先搜索



霍夫曼编码

K	N	G	I	H	E
4	2	5	1	3	7

常用的往Root放，不常用的往leaf放

总结

模型用途+空间时间提速

W2V → FastText

应用

更加快速的在线学习与反馈学习: 社交网络广告推荐，信息流推荐

更加细分+可拓展化的分类任务：个性化新闻，网页，。。。。

部署轻量级设备：手机端（Mobile DL），IoT，etc

效果

Tag Prediction

YFCC100M (Thomee et al., 2016)

Model	prec@1	Running time	
		Train	Test
Freq. baseline	2.2	-	-
Tagspace, $h = 50$	30.1	3h8	6h
Tagspace, $h = 200$	35.6	5h32	15h
`fastText`, $h = 50$	31.2	6m40	48s
`fastText`, $h = 50$, bigram	36.7	7m47	50s
`fastText`, $h = 200$	41.1	10m34	1m29
`fastText`, $h = 200$, bigram	46.1	13m38	1m37

效果

Sentiment Analysis

	Zhang and LeCun (2015)		Conneau et al. (2016)			fastText
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

代码

【详见随堂 iPython Notebook】

感谢大家！

恳请大家批评指正！