FastText

Bag of Tricks for Efficient Text Classification
Armand Joulin, Edouard Grave, Piotr Bojanowski, Tomas Mikolov
TACL, 2016

Outline

- Word2vec
- FastText
- Reference

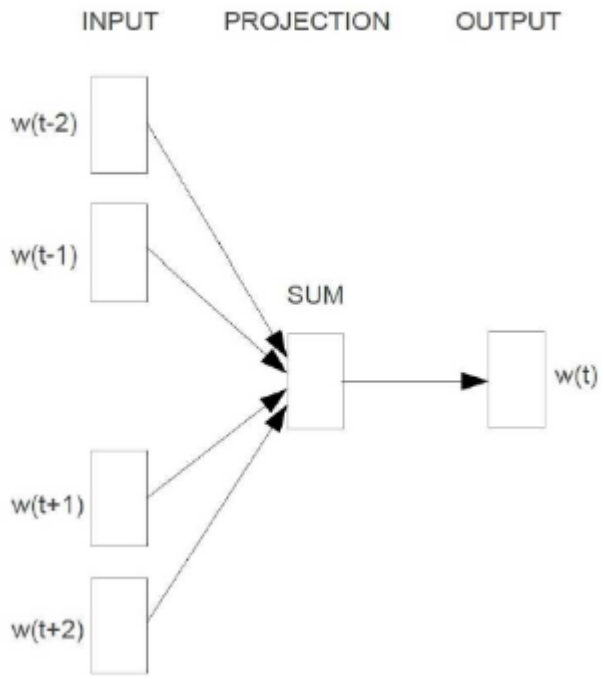
Word2vec-简介

- Word2vec是Google于2013年开源的word vector工具包，简单高效
- 利用语义分布假说：上下文相似的词，其语义也相似
- 三层浅层网络，包括CBOW模型和Skip-gram模型
- 训练技巧：分层Softmax、负采样（Negative Sampling）

Word2vec-模型

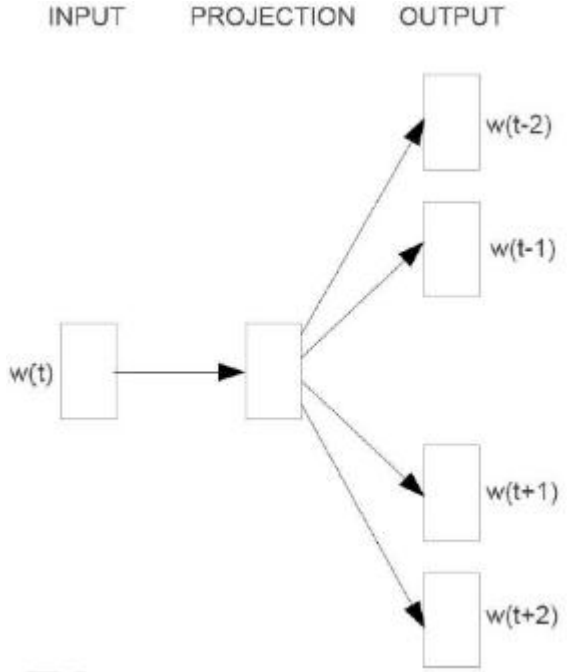
CBOW

- CBOW模型根据**中心词W(t)周围的词**来预测中心词



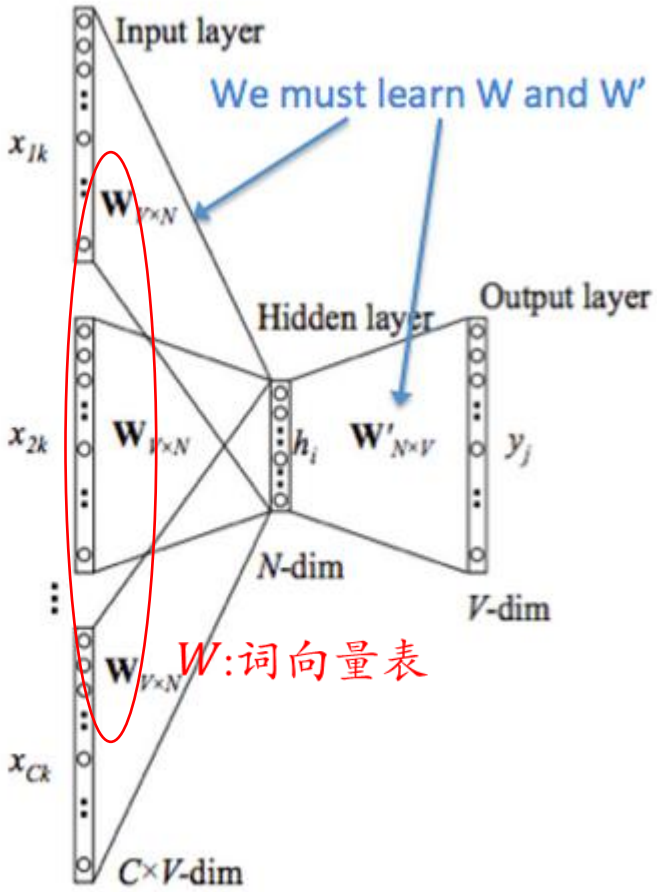
Skip-gram

- Skip-gram模型则根据**中心词W(t)**来预测周围词



Word2vec-模型

- 如何学习到词向量？以CBOW为例



输入层：

one − hot向量：X_I

隐藏层：

$\mathrm{H} = W^T * X$ （没有激活函数）

输出层：

$U = W^T * \mathrm{H}$

$$P(w_j|w_I) = y_i = \frac{\exp(u_j)}{\sum_{k\in V}\exp(u_k)}$$

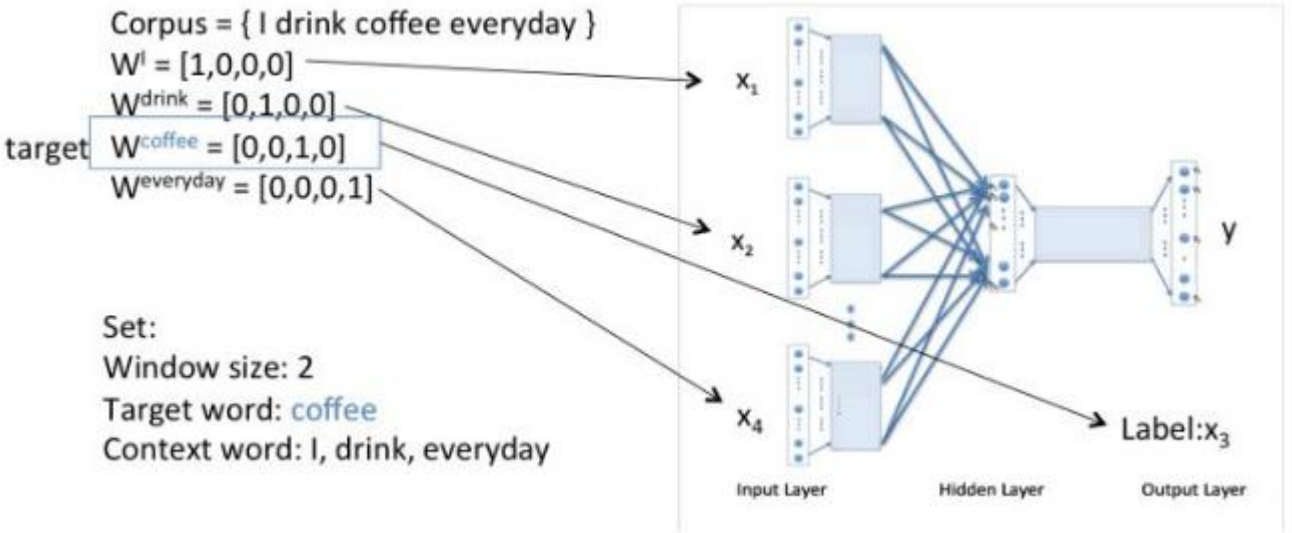
交叉熵损失函数

$$L = \sum_{k\in V} \log p(w|context(w))$$

$$= \sum_{k\in V} \log p(w_j|w_I)$$

Word2vec-举例

- Input : X



Word2vec-举例

- Hidden: $W^T X$

Corpus = { I drink coffee everyday }

Initialize:

$$W = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

Ex:

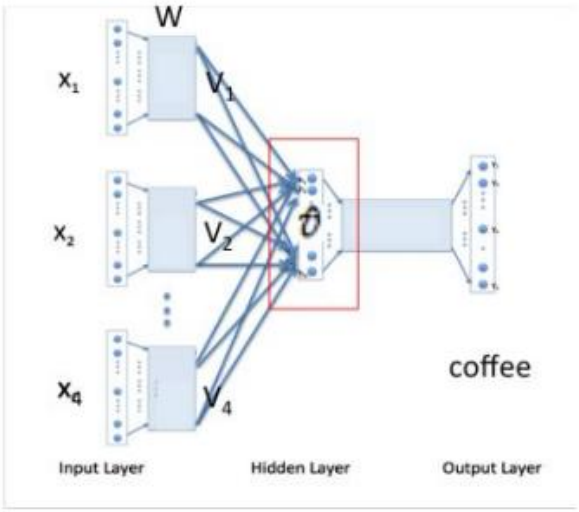
$W^{drink} = [0,1,0,0]$

$$Wx_2 = v_2$$

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 1 & 2 & 1 & 2 \\ -1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

$$\frac{v_1 + v_2 + v_4}{3} = \hat{v}$$

$$\frac{1}{3}\left(\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1.67 \\ 0.33 \end{bmatrix}$$



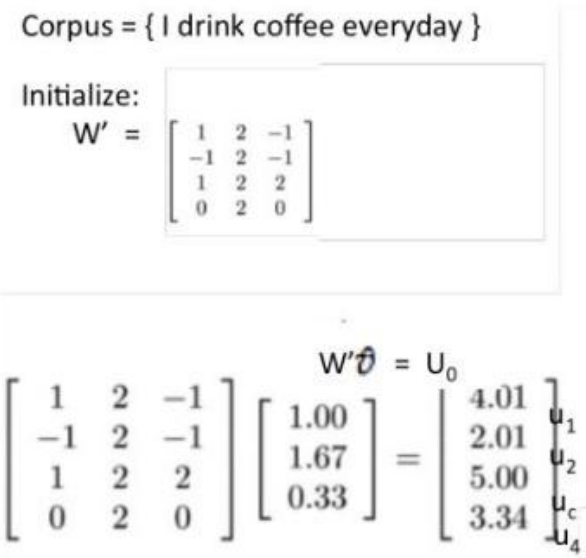
Word2vec-举例

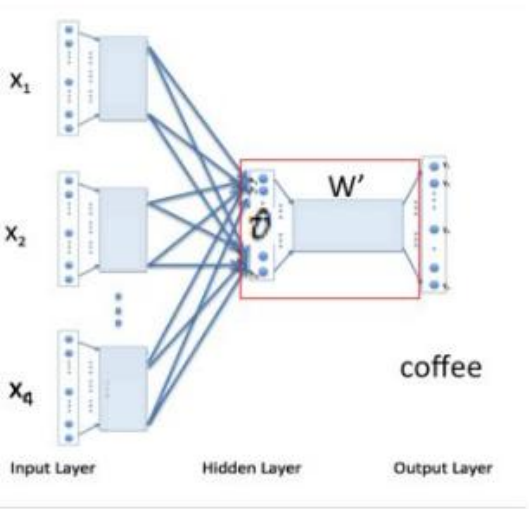
- Output

Corpus = { I drink coffee everyday }

Initialize:

$$W' = \begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 0 & 2 & 0 \end{bmatrix}$$





$$softmax(\mathbf{u}_o) = \mathbf{y}$$

$$softmax\left(\begin{bmatrix} 4.01 \\ 2.01 \\ 5.00 \\ 3.34 \end{bmatrix}\right) = \begin{bmatrix} 0.23 \\ 0.03 \\ 0.62 \\ 0.12 \end{bmatrix}$$

Probability of “coffee”

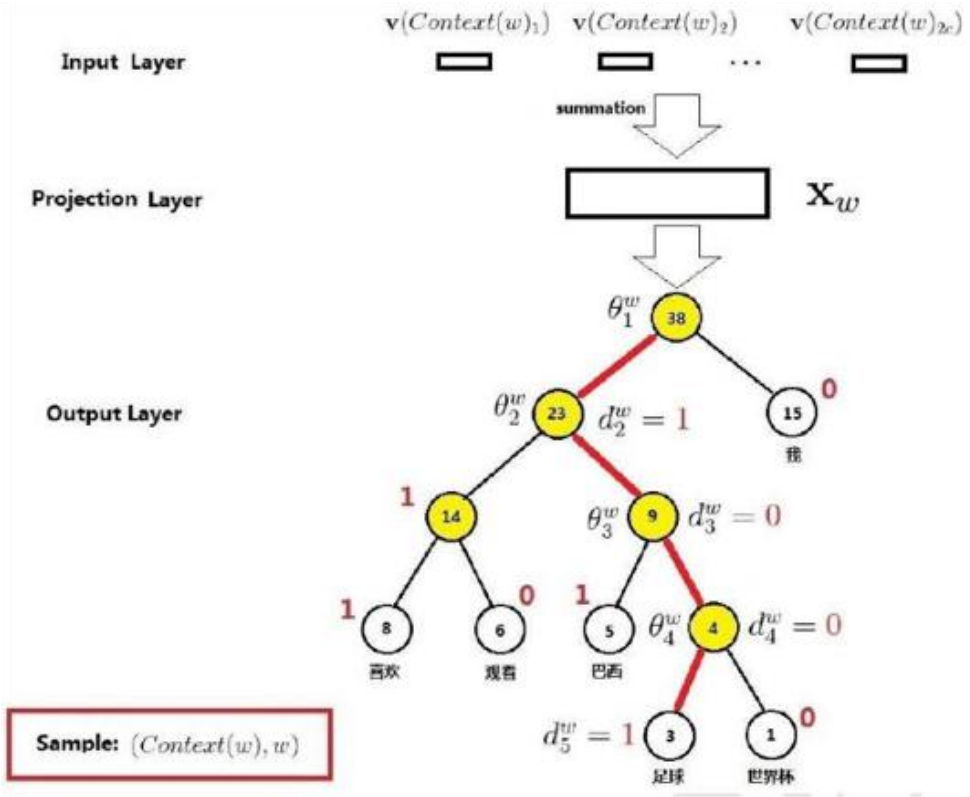
训练出来的look up table应该为矩阵W。即**任何一个单词的one-hot表示乘以这个矩阵W都将得到自己的word embedding**

Word2vec-softamx

- Softmax 函数
- $P(w_j|w_I) = y_i = \frac{\exp(u_j)}{\sum_{k \in V} \exp(u_k)}$，时间复杂度$O(|V|)$
- $|V|$是词表大小
- 计算一次y_i时间复杂度太高，需要计算所有的单词

Word2vec-Hierarchical Softmax

- Hierarchical Softmax
  - 根据单词出现频率，构造Huffman Tree
  - 对于词典中任意单词，必存在一条到达该单词结点的路径。路径上每个中间结点看做是一次二分类问题，每一次分类产生一个概率，将这些概率连乘即是p(w|context(w))
  - 是否满足softmax函数归一化的要求？满足



Word2vec-Hierarchical Softmax

- Hierarchical Softmax
- $O(|V|) \rightarrow O(\log|V|)$
- 对于一些生僻词（单词频率低），结点所在路径会很长，计算时间耗时
- 如何加快对生僻词的处理?

Word2vec-NEG

- Negative Sampling
  - 用来提高训练速度并改善词向量质量
  - 不再使用复杂的Huffman Tree，而是使用随机负采样，作为Hierarchical Softmax的一种替换

Word2vec-NEG

- Negative Sampling
  - 以CBOW模型为例，已知词w的上下文$context(w)$预测w，词w就是正样本，其他词就是负样本
  - 最大化$g(w)$即是增大正样本的概率同时降低负样本的概率，这里可视为一种二分类问题

假定现在已经选好了一个关于 w 的负样本子集 $NEG(w) \neq \emptyset$. 且对 $\forall\ \widetilde{w} \in \mathcal{D}$, 定义

$$L^w(\widetilde{w}) = \begin{cases} 1, & \widetilde{w} = w; \\ 0, & \widetilde{w} \neq w, \end{cases}$$

表示词 $\widetilde{w}$ 的标签, 即正样本的标签为 1, 负样本的标签为 0.

对于一个给定的正样本 $(Context(w), w)$, 我们希望最大化

$$g(w) = \prod_{u \in \{w\} \cup NEG(w)} p(u|Context(w)),$$

$$p(u|Context(w)) = \begin{cases} \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 1; \\ 1 - \sigma(\mathbf{x}_w^\top \theta^u), & L^w(u) = 0, \end{cases}$$

Word2vec-总结

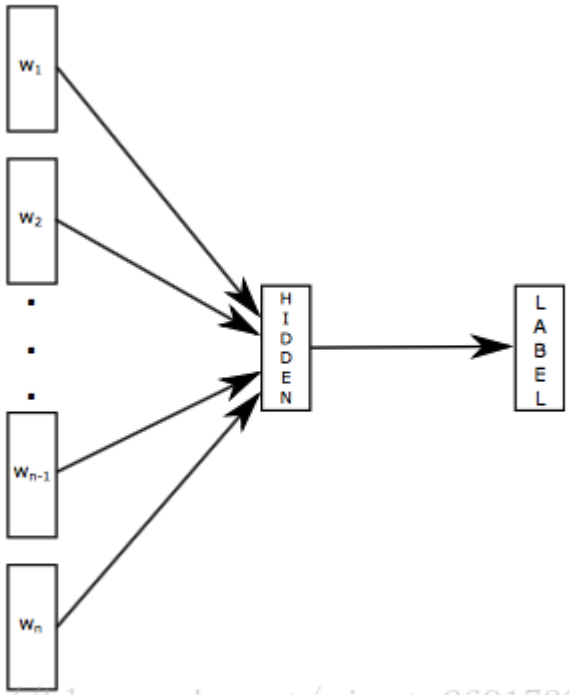
- 词向量生成工具
- 浅层神经网络
- 利用Hierarchical Softmax和Negative Sampling加快计算速度

FastText-简介

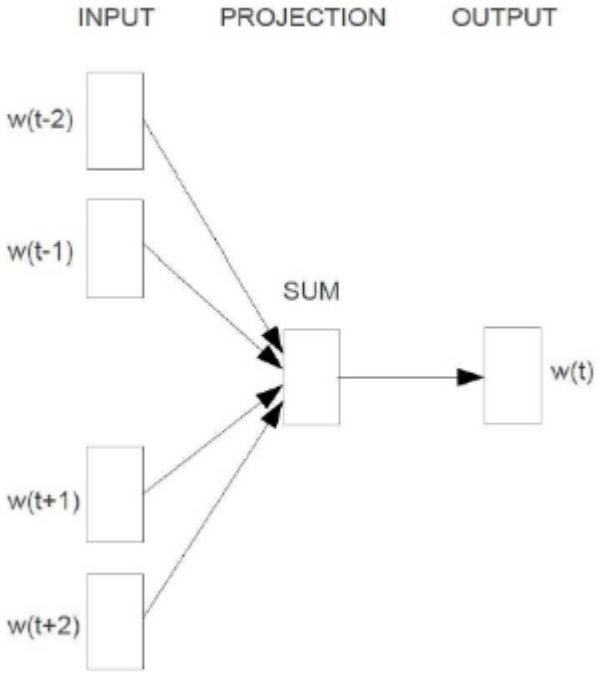
- 由Facebook开发的快速文本分类器，提供简单高效的文本分类和表征学习方法
- 与复杂的深度网络相比，实验效果相当，但时间快几个数量级

FastText-模型

- FastText：预测标签

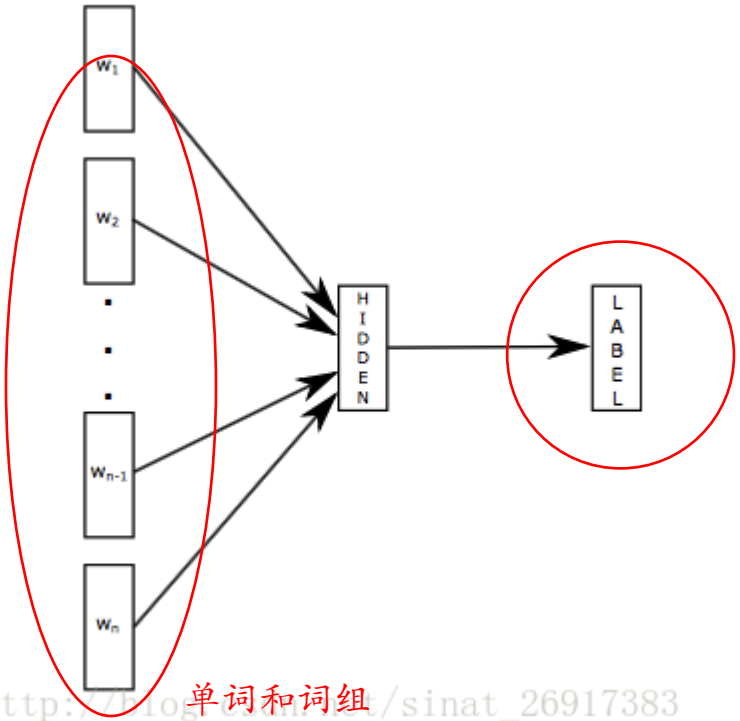


- CBOW：预测单词



FastText-模型

- FastText：预测标签



- 输入：文本的序列（单词和n-gram）
- 隐层：没有非线性激活函数，序列向量均值
- 输出：属于类别的概率

FastText-N-gram

- N-gram：将文本内容按照字节顺序进行大小为N的滑动窗口操作，最终形成长度为N的字节片段序列。
- n-gram中的gram根据粒度不同，有不同的含义，可以是字粒度，也可以是词粒度的。
- Bigram：

  单词“apple”，字符粒度下，n的取值为2

  它的bigram有：“<a”，“ap”，“pp” “pl”，“le”，“e>”
- Trigram：

  单词“apple”，字符粒度下，n的取值为3

  它的trigram有：“<ap”，“app”，“ppl”，“ple”，“le>”

N-gram

- word2vec把语料库中的每个单词当成原子的，它会为每个单词生成一个向量。这忽略了单词内部的形态特征，比如：“apple”和“apples”，两个单词都有较多公共字符，即它们的内部形态类似，但是在传统的word2vec中，这种单词内部形态信息因为它们被转换成不同的id丢失了。
- 为了克服这个问题，fastText使用了字符级别的n-grams来表示一个单词。
- **优点**
  - 对于低频词生成的词向量效果会更好。因为它们的n-gram可以和其它词共享。
  - 对于训练词库之外的单词，仍然可以构建它们的词向量。我们可以叠加它们的字符级n-gram向量。

Preformance

- Sentiment analysis

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
fastText, $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
fastText, $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

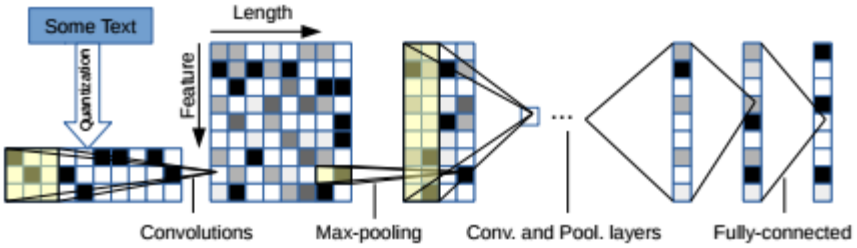
Deep net

	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

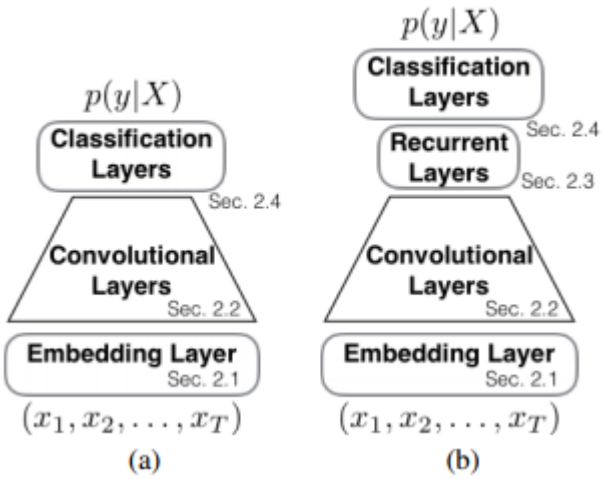
Char-CNN:字符级别的CNN网络
6层卷积+3层全连接

VDCNN: very deep CNN

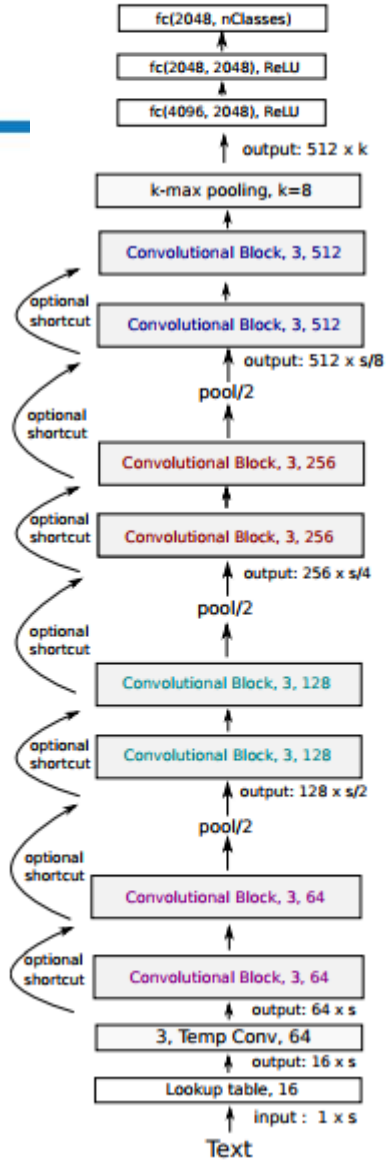
Baseline



char-CNN



char-CRNN

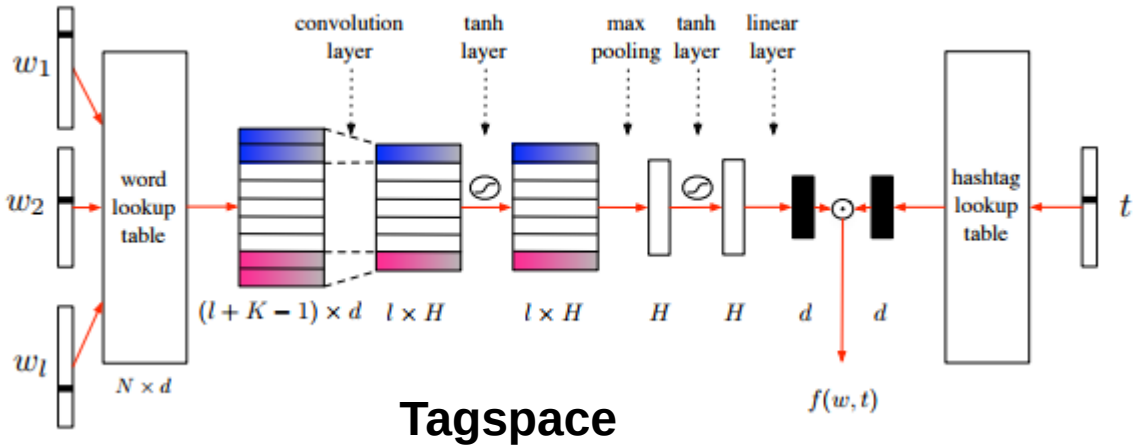


DVCNN

Preformance

- Tag prediction: tag数量多

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



Tagspace

与word2vec相比

- 输入层：
  - word2vec是 context window 内的term；
  - fasttext 对应的整个sentence的内容，包括term和n-gram的内容；
- 输出层：
  - Word2vec输出层对应是每一个term，计算某term的概率最大；
  - fasttext的输出层对应分类的label。
- 训练技巧：
  - word2vec使用Hierarchical-softmax和NEG
  - Fasttext在tag数量大时，会使用Hierarchical-softmax，但不使用NEG（精度损失）

与深度网络相比

- 原文：“DNN have in theory much higher representational power than shallow models, it is not clear if simple text classification problems are the right ones to evaluate them.”
- 结合n-gram信息和快速计算技巧（Hierarchical Softmax）
- 实验效果相当但简单高效

深度网络

- Deep Learning在各个领域不断渗透、颠覆，尤其是在图像、语音方面有着极为出色的表现。但在文本方面却一直没什么特别突出的成就。
- 语言、文本是经过很多人、很多年沉积下来的，是存在一定的规则性、关联性的，跟图像、语音有比较大的区别。
- 文本分类问题不足以展示deep learning强大的能力，在长文本分类问题中，使用简单TFIDF就可以得到良好的效果
- Simple is better

Reference

- Word2vec,2013
- Bag of Tricks for Efficient Text Classification,2016
- http://www.cnblogs.com/peghoty/p/3857839.html,word2vec的数学原理详解
- https://www.zhihu.com/question/44832436, word2vec是如何得到词向量的