第8章 词向量表示

One-hot 表示

- One-hot 表示

把每个词表示为一个很长的向量。这个向量的维度是词表大小，其中绝大多数元素为 0，只有一个维度的值为 1，这个维度就代表了当前的词。

举个栗子，
“话筒”表示为 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 …]
“麦克”表示为 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 …]
每个词都是茫茫 0 海中的一个 1。

这种 One-hot Representation 如果采用稀疏方式存储，会是非常的简洁：也就是给每个词分配一个数字 ID。比如刚才的例子中，话筒记为 3，麦克记为 8（假设从 0 开始记）

One-hot 表示

- 缺陷
  - 容易受维数灾难的困扰
  - 词汇鸿沟，不能很好地刻画词与词之间的相似性
  - 强稀疏性

分布式表示

- 最早是Hinton于1986年提出的，可以克服One-Hot Representation的上述缺点。其基本想法是：通过训练将某种语言中的每一个词 映射成一个固定长度的短向量

分布式相似度的表示

- Distributional similarity based representations

通过一个词语的上下文可以学到这个词语的很多知识

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

The dog is walking

The cat is walking

基于上下文的词向量表示-共现矩阵

- 如何使用上下文来表示单词
  - 共现矩阵
  - 基于窗口的共现矩阵：一个简单例子

窗口长度是1（一般是5-10）
对称（左右内容无关）
语料样例
I like deep learning.
I like NLP.
I enjoy flying

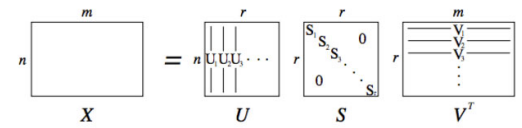
counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

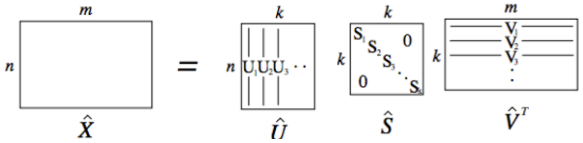
共现矩阵

- idea: 将最重要的信息存储在固定的，低维度的向量里：密集向量（dense vector）
- 维数通常是25-1000
- 对词共现矩阵采用SVD进行降维

SVD（奇异值分解）

将一个比较复杂的矩阵用更小更简单的3个子矩阵的相乘来表示，这3个小矩阵描述了大矩阵重要的特性。





$\hat{X}$ is the best rank k approximation to X, in terms of least squares.

- 对X做矩阵分解（如奇异值分解）得到矩阵**正交矩阵U**，对U进行**归一化**得到矩阵，即视为所有词的词向量：

I	0.24	0.21	0.10	0.38	-0.18	-0.18	-0.42	-0.06
like	0.20	0.82	-0.17	0.31	0.18	-0.23	0.13	0.14
enjoy	0.37	0.64	0.16	0.00	-0.58	0.64	0.00	-0.31
deep	0.36	0.38	0.35	-0.07	0.45	0.08	0.55	-0.47
learning	0.40	0.52	-0.50	-0.43	0.35	0.16	-0.47	-0.40
NLP	0.35	0.35	-0.22	-0.19	0.13	0.49	0.21	0.66
flying	0.41	0.42	-0.40	-0.38	-0.51	-0.43	0.42	-0.12
.	0.38	0.58	0.59	-0.62	-0.03	-0.23	-0.26	0.24

Python中简单的词向量SVD分解

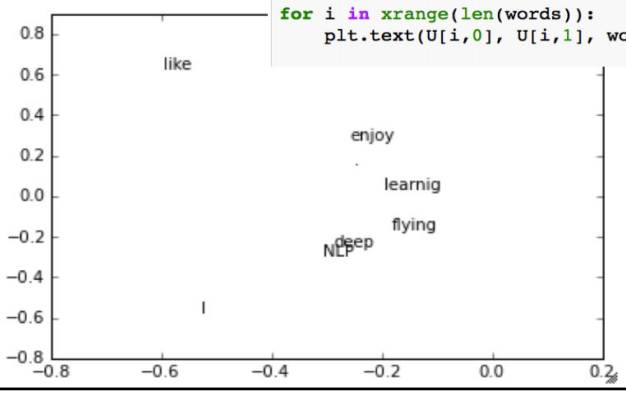
- 语料：I like deep learning. I like NLP. I enjoy flying

```
import numpy as np
la = np.linalg
words = ["I", "like", "enjoy",
         "deep","learnig","NLP","flying","."]
X = np.array([[0,2,1,0,0,0,0,0],
              [2,0,0,1,0,1,0,0],
              [1,0,0,0,0,0,1,0],
              [0,1,0,0,1,0,0,0],
              [0,0,0,1,0,0,0,1],
              [0,1,0,0,0,0,0,1],
              [0,0,1,0,0,0,0,1],
              [0,0,0,0,1,1,1,0]])

U, s, Vh = la.svd(X, full_matrices=False)
```

- 打印U矩阵的前两列这也对应了最大的两个奇异值

```
for i in xrange(len(words)):
    plt.text(U[i,0], U[i,1], words[i])
```



Hacks to X

- 功能词(the, he, has)过于频繁，对语法有很大影响，解决办法是降低使用或完全忽略功能词
- 延展窗口增加对临近词的计数
- 用皮尔逊相关系数代替计数，并置负数为0

http://tedlab.mit.edu/~dr/Papers/RohdeGonnermanPlaut-COALS.pdf

- 使用SVD存在的问题
  - 对于n*m矩阵来说计算的时间复杂度是o(mn^2) 当单词或者文档数以百万计时很糟糕
  - 对于新词或者新的文档很难及时更新
  - 相对于其他的DL模型，有着不同的学习框架

解决方案：直接学习低维度的词向量

神经语言模型

- 神经网络语言模型（Neural Network Language Model，NNLM）是一种用神经网络建模语言的方法。NNLM 通过学习文本序列中的词汇之间的概率关系，能够捕捉到语言的结构和语境，从而能够生成自然语言文本或进行其他与语言相关的任务。
- 最经典的论文就是Bengio于2003年发表的《A Neural Probabilistic Language Model》：正式提出神经网络语言模型nnlm，这应该算是第一个用来学习词向量表示的神经网络模型
- Word2Vec，CNN，LSTM

神经语言模型的发展

2018年
3月：ELMo，11月：Bert（google）

2013，2014年
- Hinton组把RNN用在语音识别上取得巨大突破
- Yoon Kim 提出来Textcnn用于短文本分类

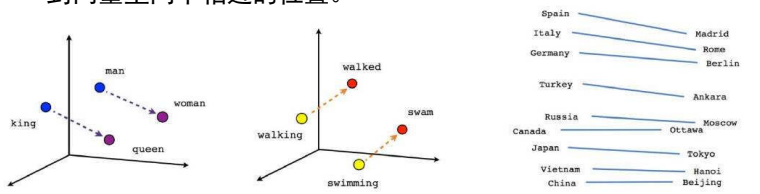
2014，2015年
- 提出一个多对多模型Seq2Seq
- Attention机制

2013年
Google团队发表了word2vec工具

2003年
Bengio 等人提出了神经网络语言模型nnlm；第一个用来学习词向量表示的神经网络模型

Word2Vec

- Word2Vec可以将One-Hot Encoder转化为低维度的连续值，也就是稠密向量，并且其中意思相近的词将被映射到向量空间中相近的位置。



Male-Female　　Verb tense　　Country-Capital

vector('Paris') - vector('France') +vector('Italy') ≈vector('Rome')

vector('king') - vector('man') + vector('woman') ≈vector('queen')

Word2Vec

- word2vec模型背后的基本思想是对出现在上下文环境里的词进行预测。
- 对于每一条输入文本，我们选取一个上下文窗口和一个中心词，并基于这个中心词去预测窗口里其他词出现的概率。因此，word2vec模型可以方便地从新增语料中学习到新增词的向量表达，是一种高效的在线学习算法（online learning）。

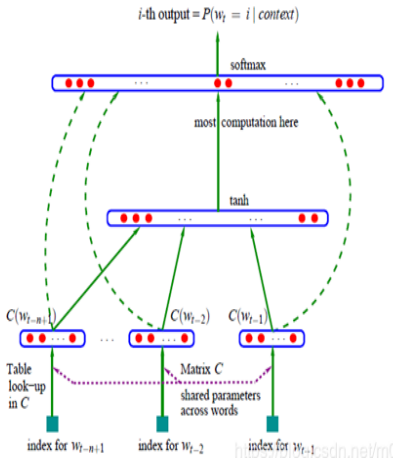
神经网络语言模型（NNLM）

- **输入：**

（1）首先是对整个词汇表建立一个索引，每个单词对应一个索引号，其实就是one-hot编码

（2）one-hot编码建立索引矩阵D，维度为 (n−1)×|V|，即每一行代表一个单词的one hot。

（3）而矩阵C每一行为一个单词的词向量，这样D·C就抽取出对应单词的向量了，这就是图中的table look-up in c



神经网络语言模型（NNLM）

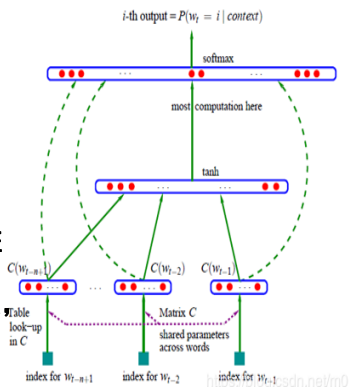
- **输入：**

（4）找出对应的词向量后，将这些词向量拼接在一起，形成一个(n−1)m维的列向量x

（5）经过隐含层tanh函数的激活，再经过softmax输出层的输出，这就得到了函数g的输出向量。

- **输出：**

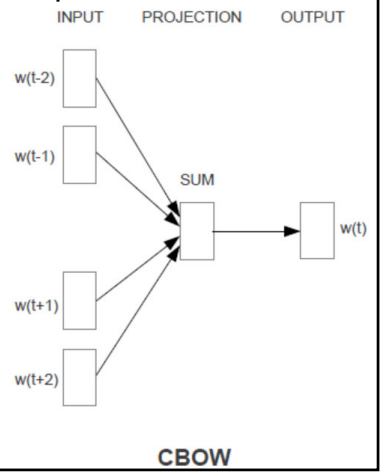
函数g把输入的上下文单词的特征向量($C(w_{t-n+1}),...,C(w_{t-1})$)映射为下一个单词$w_t$的条件概率分布函数当然，这个单词wt在字典V中。



Word2Vec

- Word2Vec模型中，主要有Skip-Gram和CBOW两种模型。

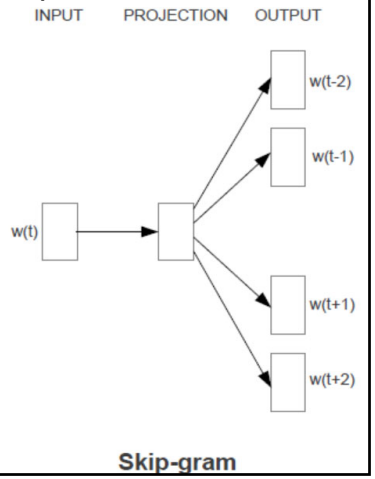
CBOW是给定上下文，来预测input word



Word2Vec

- Word2Vec模型中，主要有Skip-Gram和CBOW两种模型。

Skip-Gram是给定input word来预测上下文。



Skip-gram

Word2Vec

- 例：”Hangzhou is a nice city”

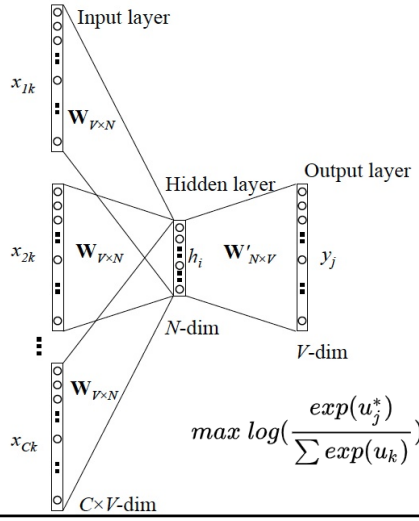
  这里假设滑窗尺寸为1

  CBOW可以制造的映射关系为：[Hangzhou,a]—>is，[is,nice]—>a，[a,city]—>nice

  Skip-Gram可以制造的映射关系为:(is,Hangzhou)，(is,a)，(a,is)， (a,nice)，(nice,a)，(nice,city)

CBOW对小型数据库比较合适，而Skip-Gram在大型语料中表现更好。

CBOW(Continuous Bag-of-Words)

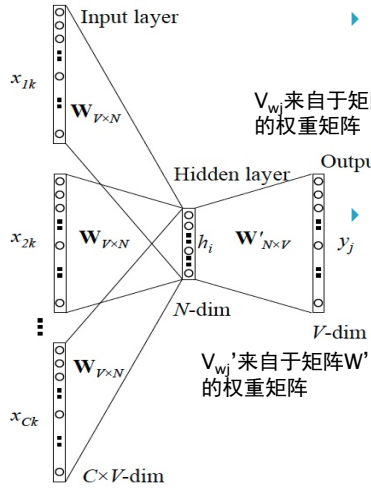


目标函数：

$$L = max\ log\ p(w|Context(w))$$
$$= max \log\ (y_j^*)$$
$$= max\ log(\frac{exp(u_j^*)}{\sum exp(u_k)})$$

$$max\ log(\frac{exp(u_j^*)}{\sum exp(u_k)}) = max\ u_j^* - log\sum_{k=1}^{V} exp(u_k)$$

CBOW(Continuous Bag-of-Words)



- 隐藏层：

$$= \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I},$$

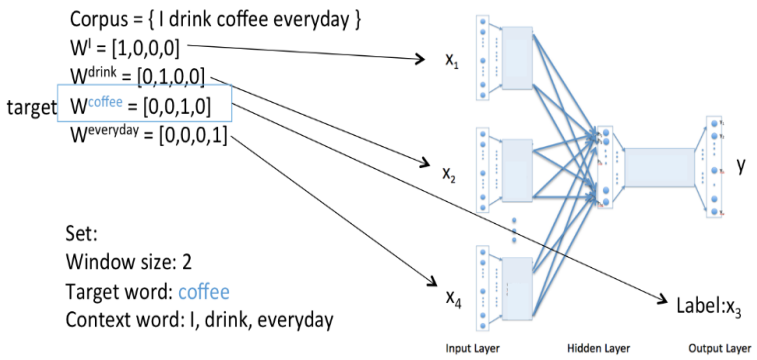
V_{wj}来自于矩阵W的行向量，也就是输入 -> 隐藏层的权重矩阵

- 隐藏层和输出层之间

$$u_j = \mathbf{v}'^{T}_{w_j}\mathbf{h},$$

V_{wj}’来自于矩阵W’ 的列向量，也就是隐藏层 -> 输出层的权重矩阵

An example of CBOW Model



窗口大小是2，表示选取coffee前面两个单词和后面两个单词，作为input词。

CBOW

- 2.然后将 one-hot 表征结果[1,0,0,0]、[0,1,0,0]、[0,0,0,1]，分别乘以:3x4的输入层到隐藏层的权重矩阵W「这个矩阵也叫嵌入矩阵，可以随机初始化生成」，比如可以是

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

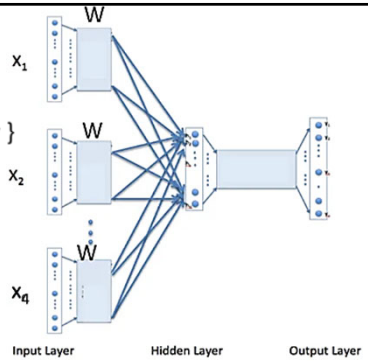
CBOW

Corpus = { I drink coffee everyday }

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

Ex:

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



Continuous bag-of-words (Mikolov et al., 2013)

$$Wx_2 = v_2$$

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

CBOW

- 3.将得到的结果向量求平均作为隐藏层向量:[1,1.67,0.33]

Corpus = { I drink coffee everyday }

Initialize:

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

$$\frac{V_1 + V_2 + V_4}{3} = \hat{v}$$

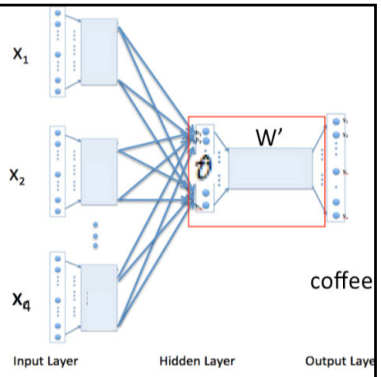
$$\frac{1}{3}\left(\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1.67 \\ 0.33 \end{bmatrix}$$

4.然后将隐藏层[1,1.67,0.33]向量乘以:4x3的隐藏层到输出层的权重矩阵W'(这个矩阵也是嵌入矩阵，也可以初始化得到)，得到输出向量:
[4.01,2.01,5.00,3.34]

Corpus = { I drink coffee everyday }

Initialize:

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



$$W'\hat{v} = U_0$$

$$\begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1.00 \\ 1.67 \\ 0.33 \end{bmatrix} = \begin{bmatrix} 4.01 \\ 2.01 \\ 5.00 \\ 3.34 \end{bmatrix} \begin{matrix} u_1 \\ u_2 \\ u_c \\ u_4 \end{matrix}$$

CBOW

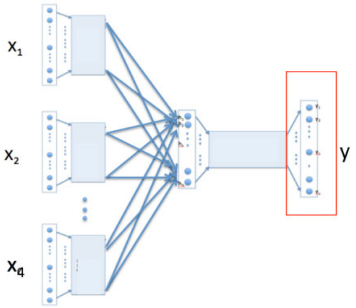
- 最后对输出向量[4.01, 2.01, 5.00, 3.34] 做 softmax 激活处理得到实际输出[0.23, 0.03, 0.62, 0.12]，

Output: Probability distribution

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

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

Probability of "coffee"

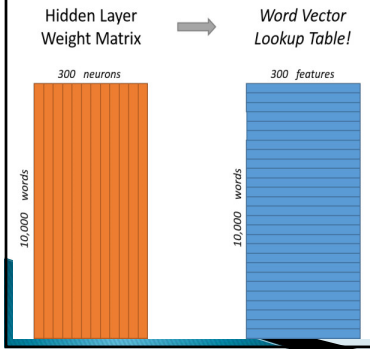


We desire probability generated to match the true probability(label) x_3 [0,0,1,0]

Use gradient descent to update W and W'

隐层

Google在最新发布的基于Google news数据集训练的模型中使用的就是300个特征的词向量。词向量的维度是一个可以调节的超参数（在Python的gensim包中封装的Word2Vec接口默认的词向量大小为100， window_size为5）。

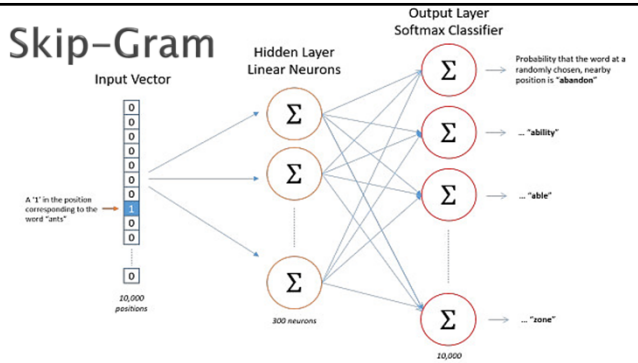


如果你从行的角度看，这个隐层的W实际上就是我们要的word vector。

$$[0\ 0\ 0\ 1\ 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10\ 12\ 19]$$

学习隐藏层权重

Skip-Gram

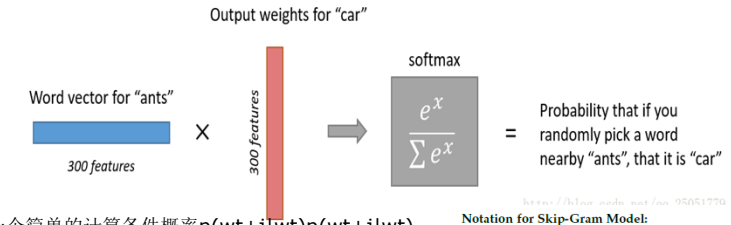


对于一个长度为T的语料库，假设我们为每一个词选取的上下文窗口的大小是m（指的是上下文各m个词），则我们的目标函数是最大化训练语料的对数似然概率：

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log p(w_{t+j}|w_t)$$

输出层

当”ants”1×300的词向量被送到输出层的时候，实际遇到的是一个**Softmax回归分类器**。 它的要点是每个输出神经元将产生0和1之间的输出（代表概率），并且所有这些输出值的总和相加等于1。



一个简单的计算条件概率p(wt+j|wt)p(wt+j|wt)的公式如下：

$$p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{W} exp(u_w^T v_c)}$$

Notation for Skip-Gram Model:

- w_i: Word i from vocabulary V
- $\mathcal{V} \in \mathbb{R}^{n \times |V|}$: Input word matrix
- v_i: i-th column of $\mathcal{V}$, the input vector representation of word w_i
- $\mathcal{U} \in \mathbb{R}^{n \times |V|}$: Output word matrix
- u_i: i-th row of $\mathcal{U}$, the output vector representation of word w_i

Word2Vec

- Word2Vec的作者在它的第二篇论文中强调了这些问题，下面是作者在第二篇论文中的三个创新：
  - 将常见的单词组合（word pairs）或者词组作为单个“words”来处理。
  - 对高频次单词进行抽样来减少训练样本的个数。
  - 对优化目标采用“negative sampling”方法，这样每个训练样本的训练只会更新一小部分的模型权重，从而降低计算负担。
- 事实证明，对常用词抽样并且对优化目标采用“negative sampling”不仅降低了训练过程中的计算负担，还提高了训练的词向量的质量。

Word2Vec的应用

- 如果两个不同的单词有着非常相似的“上下文”（也就是窗口单词很相似，比如“Kitty climbed the tree”和“Cat climbed the tree”），那么通过我们的模型训练，这两个单词的嵌入向量将非常相似。
- 这种方法实际上也可以帮助你进行词干化（stemming），例如，神经网络对”ant“和”ants”两个单词会习得相似的词向量。

Word2Vec的应用-查询扩展

- 在进行搜索的时候，文档中的同一个词往往有多种表达方法，这种现象叫做同义词。比如用户输入“充电宝”，而文档里有“移动电源”，这时如果搜索系统没有做查询优化的话用户是不能找到相关文档的。由此我们引出查询扩展的概念
- 查询扩展技术是利用计算机语言学，信息学的技术，在原用户查询的基础上，通过一定的方法和策略，把与原查询词相关的词组添加到查询序列中，组成新的，更准确表达用户查询意图的查询序列。
- 弥补用户查询信息不足的缺陷，改善检索中的查全率和查准率低下的问题。

查询扩展

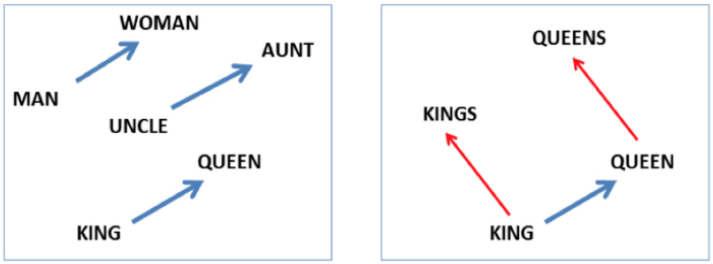
- 20世纪六十年代，搜索引擎用户的一个基本工具是在线叙词表（thesaurus）。
- 叙词表又叫做受控词表（controlled vocabulary），描述了文档集合索引的所有词汇及同义词相关词或短语的信息。
- 通过叙词表，用户可以决定在查询中使用到哪些词语或短语，并且能够用同义词和相关词来扩展最初的查询。
- 最著名的叙词表是是美国国立医学图书馆编制的权威性主题词表《医学主题词表》(Medical Subject Headings，简称MeSh)。
  - 《MeSh》汇集约18,000多个医学主题词。
  - 在进行检索时，用户输入一个主题词后，系统会自动显示该主题词所能组配的副主题词。

查询扩展

- 利用Word2Vec产生包含同义词和相关词的扩展词列表

相关或者相似的词，在距离上更接近

自动实现：1）单词语义相似性的度量；2）词汇的语义的类比



Word2Vec的应用-文本分类

- TextCNN

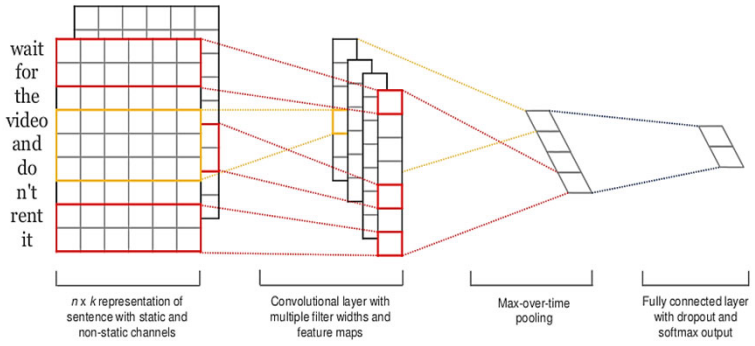
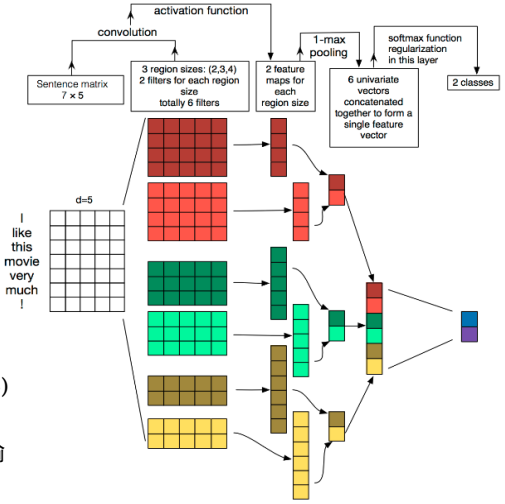


Figure 1: Model architecture with two channels for an example sentence.

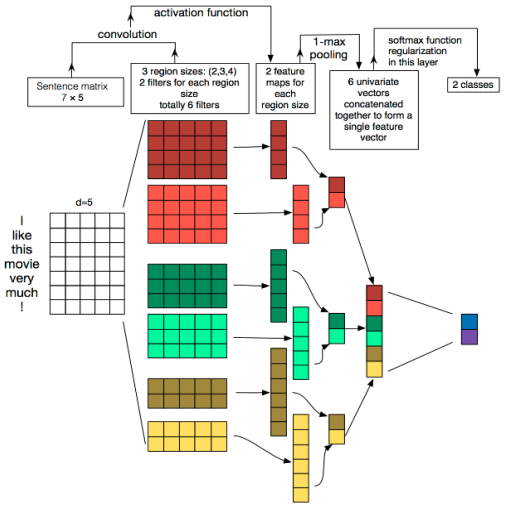
文本分类

- **Embedding**：第一层是图中最左边的7乘5的句子矩阵，每行是词向量，维度=5，这个可以类比为图像中的原始像素点。
- **Convolution**：然后经过kernel_sizes=(2,3,4)的一维卷积层，每个kernel_size 有两个输出 channel。



文本分类

- **MaxPolling**：第三层是一个1-max pooling层，这样不同长度句子经过pooling层之后都能变成定长的表示。
- **FullConnection and Softmax**：最后接一层全连接的softmax 层，输出每个类别的概率。



Word2Vec的应用-文本分类

- 通道（Channels）：
  - 图像中可以利用 (R, G, B) 作为不同channel；
  - 文本的输入的channel通常是不同方式的embedding方式（比如 word2vec或Glove），实践中也有利用静态词向量和fine-tunning词向量作为不同channel的做法。

Channel 1　　Channel 2

Word2Vec的应用-文本分类

- 在词向量构造方面可以有以下不同的方式：
  - CNN-rand：随机初始化每个单词的词向量通过后续的训练去调整。
  - CNN-static：使用预先训练好的词向量，如word2vec训练出来的词向量，在训练过程中不再调整该词向量。
  - CNN-non-static：使用预先训练好的词向量，并在训练过程进一步进行调整。CNN-multichannel：将static与non-static作为两通道的词向量。

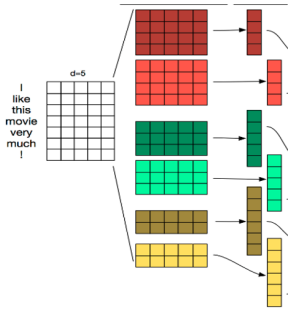
Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4

Word2Vec的应用-文本分类

- 一维卷积（conv-1d）：

图像是二维数据；

文本是一维数据，因此在TextCNN卷积用的是一维卷积（在**word-level**上是一维卷积；虽然文本经过词向量表达后是二维数据，但是在embedding-level上的二维卷积没有意义）。一维卷积带来的问题是需要**通过设计不同 kernel_size 的 filter 获取不同宽度的视野**。



Word2Vec的应用-文本分类

- Pooling层：
  利用CNN解决文本分类问题的文章还是很多的，比如这篇 A Convolutional Neural Network for Modelling Sentences 最有意思的输入是在 pooling 改成 **(dynamic) k-max pooling** ，pooling阶段保留 k 个最大的信息，保留了全局的序列信息。
  比如在情感分析场景，举个例子：
  “我觉得这个地方景色还不错，但是人也实在太多了”
  虽然前半部分体现情感是正向的，全局文本表达的是偏负面的情感，利用 k-max pooling能够很好捕捉这类信息。

其他模型

- GloVe和word2vec的思路相似：
  - GloVe: Global Vectors for Word Representation
- Glove是斯坦福大学Jeffrey Pennington等人提出的，他们认为虽然skip-gram模型在计算近义词方面比较出色，但它们只是在局部上下文窗口训练模型，并且它很少使用语料中的一些统计信息，因此Jeffrey Pennington等人又提出了一个新型模型GloVe。
- 思想：和哪个上下文单词在一起的多，那么这个单词与这个上下文单词在一起要比与其他词在一起意义要大。
- 例如i=ice, j=steam，假设有共现词k，但是k与ice的联系要比与steam的联系强，也就是说单词k与ice出现的概率比与 steam出现的概率大，比如说k=solid，那么我们认**Pik/Pjk会很大。**

其他模型

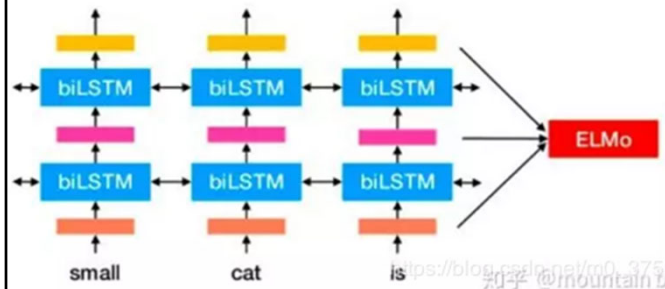
- **ELMo-动态词向量**
  - 艾伦研究所开发，并于6月初在NAACL 2018年发布ELMo（深度语境化的单词表示）。ELMo(Embeddings from Language Models)，被称为时下最好的通用词和句子嵌入方法，来自于语言模型的词向量表示，也是利用了深度上下文单词表征
- 该模型的优势：
  能够处理单词用法中的复杂特性（比如句法和语义）
- 这些用法在不同的语言上下文中如何变化（比如为词的多义性建模）

其他模型

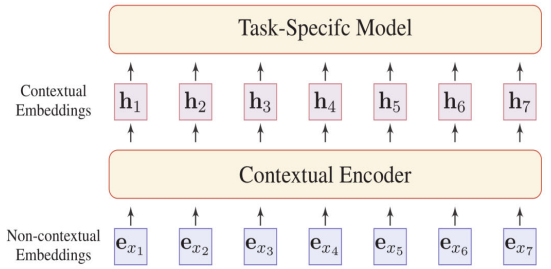
- **ELMo与word2vec、glove最大的不同：**即词向量不是一成不变的，而是根据上下文而随时变化，这与word2vec或者glove具有很大的区别。

举个例子：针对某一词多义的词汇 w="苹果"
文本序列1=“我 买了 六斤 苹果。”
文本序列2=“我 买了一个 苹果 7。”
上面两个文本序列中都出现了“苹果”这个词汇，但是在不同的句子中，它们的含义显示是不同的，一个属于水果领域，一个属于电子产品呢领域，如果针对“苹果”这个词汇同时训练两个词向量来分别刻画不同领域的信息呢？答案就是使用ELMo。

- ELMo是双向语言模型bilm的多层表示的组合，基于大量文本，ELMo模型是从深层的双向语言模型（deep bidirectional language model）中的内部状态(internal state)学习而来的，而这些词向量很容易加入到QA、文本对齐、文本分类等模型中。

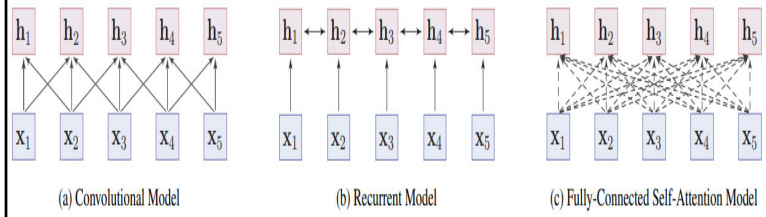


Generic Neural Architecture for NLP



模型驱动
+
数据驱动

Contextual Encoder



- NLP应用：情感分析
  - 传统的方法：精选的情感词典+词袋模型（忽略词序）+人工设计的特征（很难覆盖所有的信息）
  - 深度学习：和上述词素，句法和语义相似的深度学习模型->RNN
  - Demo: http://nlp.stanford.edu/sentiment/
- http://cs224d.stanford.edu/

- 开源工具：
  gensim
  tensorflow

小测试

- Word2Vec常用于词向量转换，在进行词向量转换时，常使用的模型有两种，分别是___和____

A. CBOW 和Skip-gram B.SVD和Skip-gram

- 下面哪句是正确的(　　)
  A. TextCNN可以沿着行和列进行卷积运算
  B. CNN可以有多个卷积核，大小不相等
  C. word2vector可以产生稠密的词向量

- 下列哪种词嵌入支持上下文建模（Context Modeling）？
- A. Word2Vec B. GloVe C. Elmo D. 以上所有

Thank You !