2021年度暑期强化课程

词表示模型：神经网络语言模型

授课人：曹亚男

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

2.词向量表示和神经语言模型

2.1	词的表示方法
2.2	统计语言模型
2.3	神经网络语言模型
2.4	词向量表示应用和对比
2.5	更多研究方向

2.词向量表示和神经语言模型

如何表示一个词的含义?

- 通常的回答：使用WordNet、HowNet等语义词典，可表示一个词的上位信息和同义词集

Wordnet: synonym sets (enjoy)

```
>>> for synset in wn.synsets('enjoy'):
...     print synset.lemma_names()
...

[u'enjoy', u'bask', u'relish', u'savor',
 u'savour']
[u'enjoy']
[u'love', u'enjoy']
[u'enjoy']
[u'delight', u'enjoy', u'revel']
```

Wordnet: antonym sets (like)

```
>>> for synset in wn.synsets('like'):
...     for lemma in synset.lemmas():
...         print lemma.antonyms()
...
[]
[Lemma('dislike.v.01.dislike')]
[]
[Lemma('unlike.a.01.unlike')]
[]
[Lemma('unlike.a.02.unlike')]
[]
[Lemma('unalike.a.01.unalike')]
```

- 不能体现词间的细微差别；词典不能及时更新；主观性强，耗时耗力

One-hot表示

- 大部分基于规则和基于统计的自然语言处理任务把词看作原子符号

enjoy, like, learn

- 采用向量空间模型表示，每个词都是茫茫0海中的一个1

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

问题

- 向量维度较高：20K(口语)- 50K(PTB)- 500K(大词表)-13M(谷歌1T)
- 采用one-hot表示，任意两个词之间都是孤立的（词义鸿沟）

enjoy [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND

like [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] =0

基于分布相似度的表示

- 通过一个词的上下文来学习该词的表示

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

(J. R. Firth 1957: 11)

- 统计自然语言处理最成功的idea之一

If you enjoy something, you find pleasure and satisfaction in doing it or experiencing it. I thought that I knew everything about Jemma: her likes and dislikes, her political viewpoints.

These words will represent *enjoy/like*

- 如何表示？两个选择
  - 利用全部上下文词：基于word-document共现矩阵（LSA）
  - 利用一定窗口长度内的上下文词捕获语法和语义信息

基于共现矩阵的分布表示

样例语料

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

统计窗口为1

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	0
flying	0	0	1	0	0	0	0	0
.	0	0	0	0	1	1	1	0

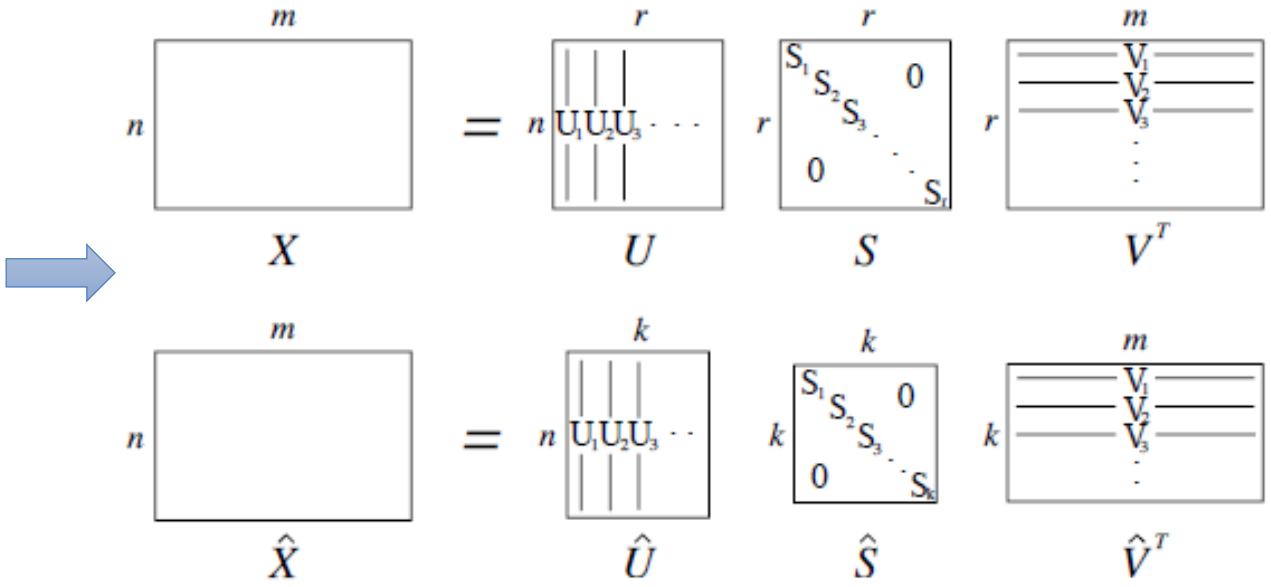
维度高

数据稀疏

降维！

降维方法：SVD分解

```
[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]
```



$O(|V|^3)$
复杂度

$$enjoy = \begin{bmatrix} -0.256 \\ 0.274 \\ 0.159 \end{bmatrix} \qquad like = \begin{bmatrix} -.0594 \\ 0.630 \\ -0.170 \end{bmatrix}$$

能否直接学习低维词向量?

- 不直接计算词之间的共现频度，直接基于词的上下文词来预测当前词（或基于当前词预测上下文词）

- 更快的学习速度：将$O(|V|^3)$降到$O(|V|)$，$O(log|V|)$?

- 更好的灵活性：加入新句子、新文档，词表里加入新词

2.词向量表示和神经语言模型

统计语言模型

- 如何判断一句话是人（正常人）说的？

一个例子

(1) The cat is small. 

(2) Small the is cat. 

(3) The cat small is. 

统计语言模型

- 统计语言模型的作用是为一个长度为m的字符串确定一个概率分布$P(w_1, w_2, \ldots, w_m)$，表示其存在的可能性。其中w_1到w_m依次表示这段文本中的各个词。

$$P(w_1, w_2, ..., w_m) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)$$
$$...P(w_i \mid w_1, w_2, \ldots, w_{i-1})...P(w_m \mid w_1, w_2, \ldots, w_{m-1})$$
$$= \prod_{i=1}^{m} P(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

$w_1, w_2, \ldots, w_{i-1}$ → *Context(w_i)*

- 其中，$P(w_i \mid w_1, w_2, \ldots, w_{i-1})$是语言模型的参数。如果句子长度为$M$，语料库对应的词典大小为$N$，那么生成一个长度为$M$的任意句子，理论上有$N^M$种可能。当句子较长时，$P(w_i \mid w_1, w_2, \ldots, w_{i-1})$很难估算。

n-gram模型

- *n*-gram模型假设一个词的出现概率只与它前面的*n*-1个词相关，距离大于等于*n*的上文词会被忽略：

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

- 一元（unigram）模型：$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i)$ → 无词序信息
- 二元（bigram）模型：$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-1})$ → 保留局部词序
- 三元（trigram）模型：$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-2}, w_{i-1})$ → 保留局部词序

- *n*-gram模型中，*n*越大，能够保留的词序信息越多，语言模型越有效

*百万词级别语料中，常使用三元模型

Gregory W Lehser, Effects of ngram order and training text size on word prediction

n-gram模型

- 在*n-gram*模型中，传统的方法一般采用频率计数的比例来估算*n*元条件概率

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_i, w_{i-(n-1)}, \ldots, w_{i-1})}{count(w_{i-(n-1)}, \ldots, w_{i-1})}$$

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

$$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

Bi-gram的例子

corpus

- I like deep learning.
- I like NLP.
- I enjoy flying.
- He enjoy NLP.

频度矩阵

I	he	like	enjoy	deep	learning	NLP	flying	.
3	1	2	2	1	1	2	1	3

Bi-gram 矩阵

cnt	I	he	like	enjoy	deep	learning	NLP	flying	.
I	0	0	2	1	0	0	0	0	0
he	0	0	0	1	0	0	0	0	0
like	0	0	0	0	1	0	1	0	0
enjoy	0	0	0	0	0	0	1	1	0
deep	0	0	0	0	0	1	0	0	0
learning	0	0	0	0	0	0	0	0	1
NLP	0	0	0	0	0	0	0	0	2
flying	0	0	0	0	0	0	0	0	1

$$p(\text{I enjoy NLP.})$$
$$= p(I)\,p(enjoy \mid I)\,p(NLP \mid enjoy)\,p(. \mid NLP)$$
$$= \frac{3}{16} * \frac{1}{3} * \frac{1}{2} * 1 = \frac{1}{32}$$

$$p(\text{I NLP enjoy.}) = 0$$

Bi-gram 概率矩阵

p(y|x)	I	he	like	enjoy	deep	learning	NLP	flying	.
I	0	0	2/3	1/3	0	0	0	0	0
he	0	0	0	1	0	0	0	0	0
like	0	0	0	0	1/2	0	1/2	0	0
enjoy	0	0	0	0	0	0	1/2	1/2	0
deep	0	0	0	0	0	1	0	0	0
learning	0	0	0	0	0	0	0	0	1
NLP	0	0	0	0	0	0	0	0	1
flying	0	0	0	0	0	0	0	0	1

n-gram模型

- 在*n-gram*模型中，传统的方法一般采用频率计数的比例来估算*n*元条件概率

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_i, w_{i-(n-1)}, \ldots, w_{i-1})}{count(w_{i-(n-1)}, \ldots, w_{i-1})}$$

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

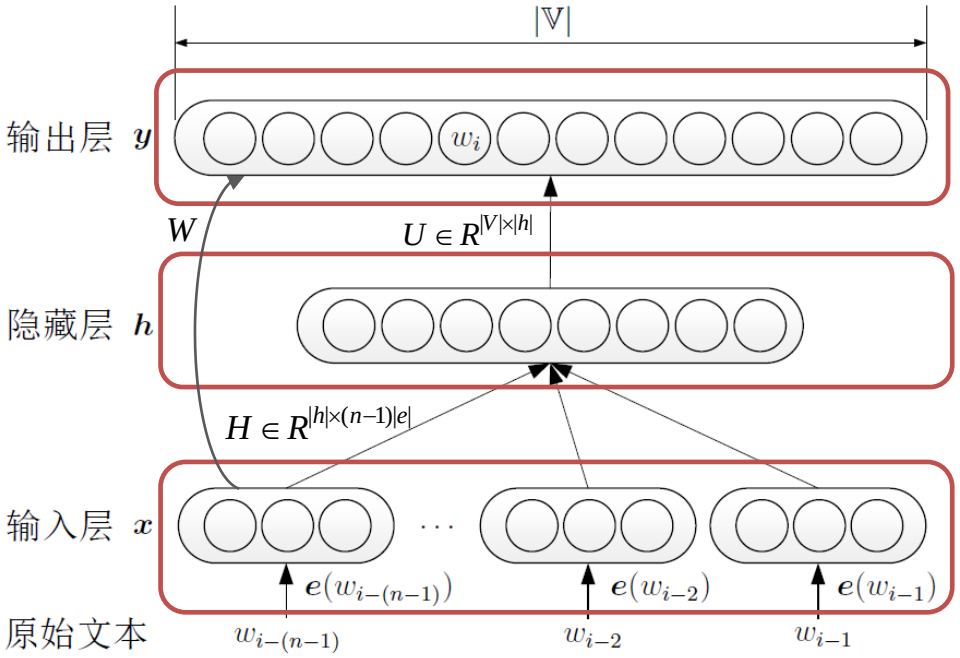
$$P(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

- 当*n*较大时，长度为*n*的序列出现的次数非常少，因此在估算*n*元条件概率时，会遇到数据稀疏问题，导致估算结果不准确

2.词向量表示和神经语言模型

神经网络语言模型（NNLM）

- 采用神经网络结构对n-gram模型进行建模，估算$P(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$的值。输入为条件部分的整个词序列，输出为目标词$w_i$的分布



输出层：有|V|个元素，表示下一个词为V中某个词的可能性

$$y = b^{(2)} + Wx + Uh$$

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\exp(y(w_i))}{\sum_{k=1}^{|V|} \exp(y(v_k))}$$

隐藏层：对x进行加权求和，权重矩阵为H，并用tanh函数激活

$$h = \tanh(b^{(1)} + Hx)$$

输入层：输入n-1个词$w_{i-(n-1)}, \ldots, w_{i-1}$的词向量拼接

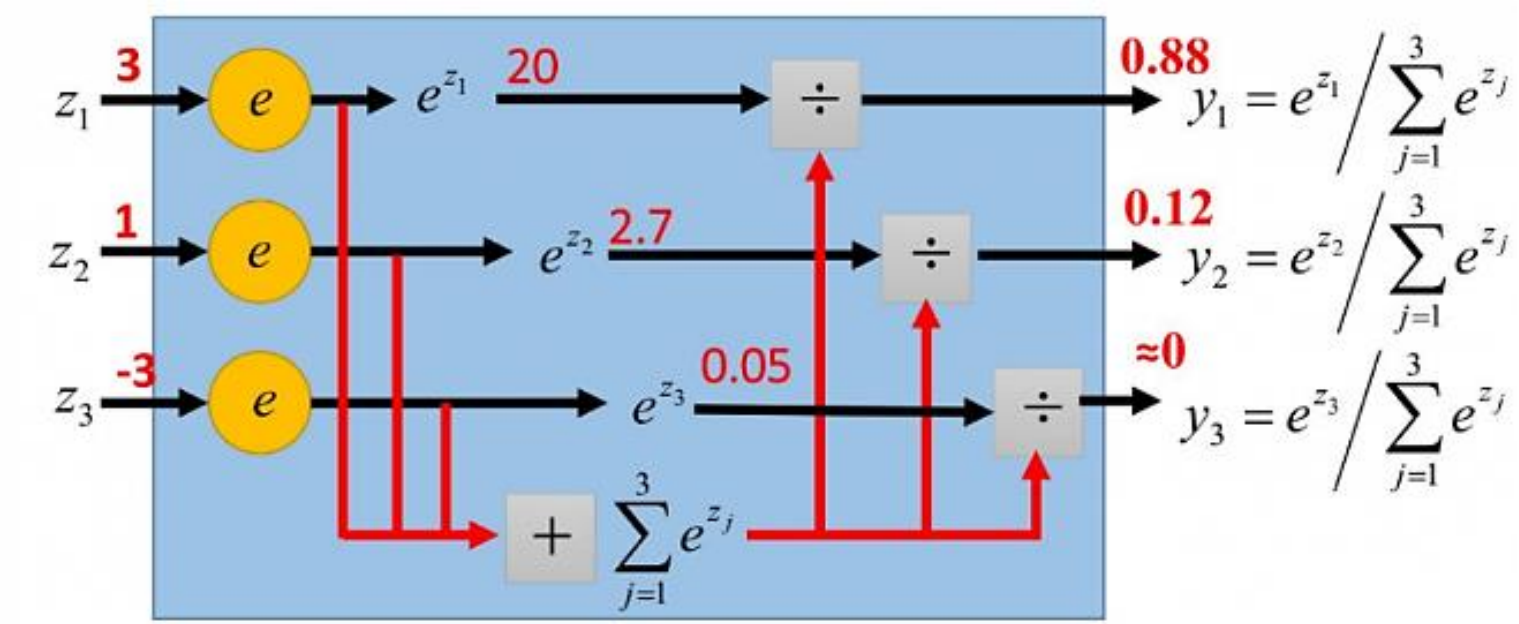
$$x = [e(w_{i-(n-1)}); \ldots; e(w_{i-1})]$$

神经网络语言模型（NNLM）结构图

Yoshua Bengio, et.al. A neural probabilistic language model（2001, 2003）

Softmax函数

- 函数形式 $y_i = e^{z_i} / \sum_{j=1} e^{z_j}$
- 因$0<y_i<1$，且$\sum_i y_i = 1$，softmax可以看作输出概率；



神经网络语言模型（NNLM）

- 输出层的分量$y(w_i)$描述的是在上文为$w_{i-(n-1)}, \ldots, w_{i-1}$的条件下，下一个词为$w_i$的可能性，体现了上文序列与目标词之间的关系

$$y(w_i) = b^{(2)} + U(\tanh(b^{(1)} + H[e(w_{i-(n-1)}); \ldots; e(w_{i-1})])$$

→ O(|V|*|h|)

$$P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{\exp(y(w_i))}{\sum_{k=1}^{|V|} \exp(y(v_k))}$$

→ O(|V|)

- 对于整个语料而言，语言模型需要最大化：

$$\sum_{w_{i-(n-1)}: i \in D} \log P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

采用log损失函数
$L(Y, P(Y|X)) = -\log P(Y|X)$

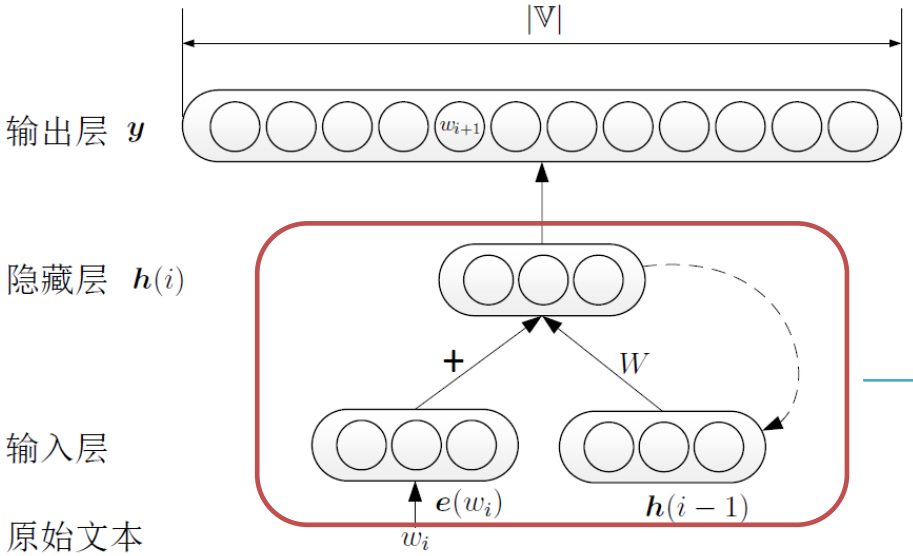
- 使用随机梯度下降法来优化训练目标：每次迭代，从语料D中随机选取一段文本$w_{i-(n-1)}, \ldots, w_{i-1}$作为训练样本，进行一次梯度迭代

$$\theta \leftarrow \theta + \alpha \frac{\partial \log P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})}{\partial \theta}$$

$e(w_i)$、H、U、$b^{(1)}$、$b^{(2)}$　　学习率

循环神经网络语言模型（RNNLM）

- 不同于NNLM是对n-gram建模，RNNLM直接对$P(w_i|w_1,w_2,\ldots,w_{i-1})$进行建模。RNNLM可以利用$w_i$所有上文信息，预测下一个词



循环神经网络语言模型（RNNLM）结构图

独特的隐藏层算法：

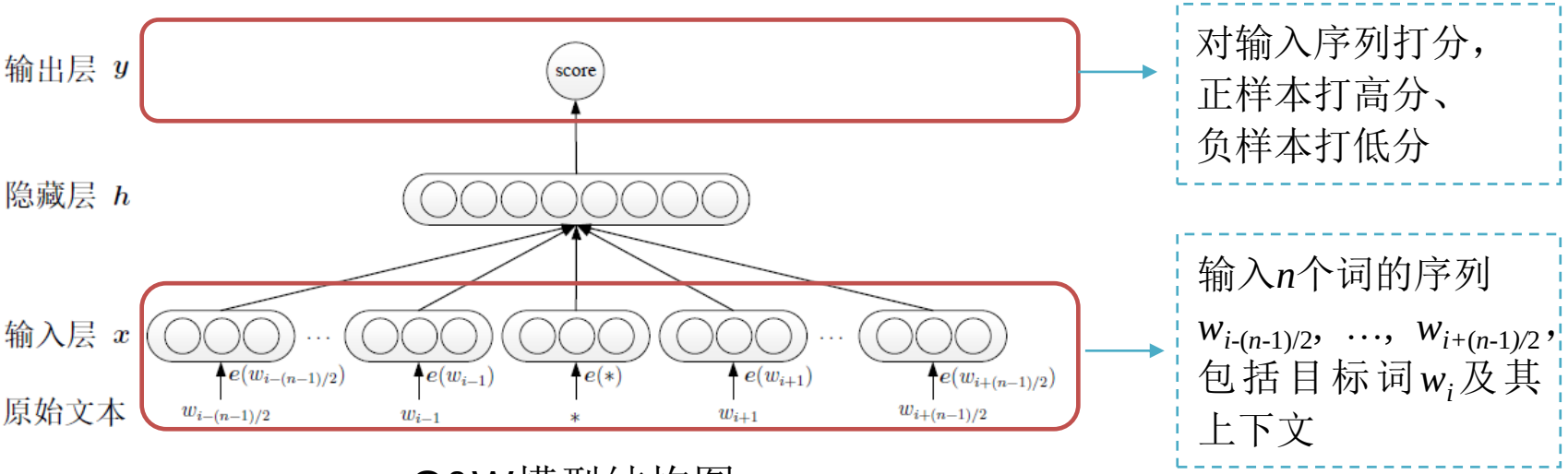
$$\boldsymbol{h}(i) = \phi(\boldsymbol{e}(w_i) + W\boldsymbol{h}(i-1))$$

- $h(i)$表示文本中第i个词对应的隐藏层，由当前词的词向量及上一个词对应的隐藏层结合得到
- 隐藏层初始状态为$h(0)$，模型逐个读入语料中的词w_1, w_2，…，隐藏层不断更新为$h(1), h(2)$，…

1. Tomas Mikolov, et.al. Statistical language models based on neural networks. 2012
2. Tomas Mikolov, et.al. Recurrent neural network based language model. 2010

C&W模型

- C&W是第一个直接以生成词向量为目标的模型，希望能够更快速地生成词向量。采用对n元短语打分的方式替代求解条件概率

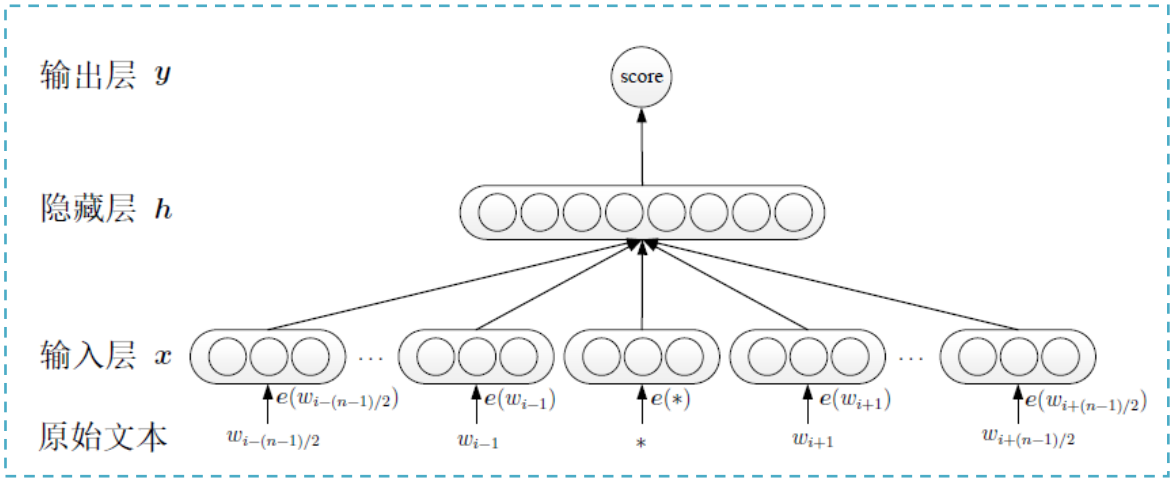


C&W模型结构图

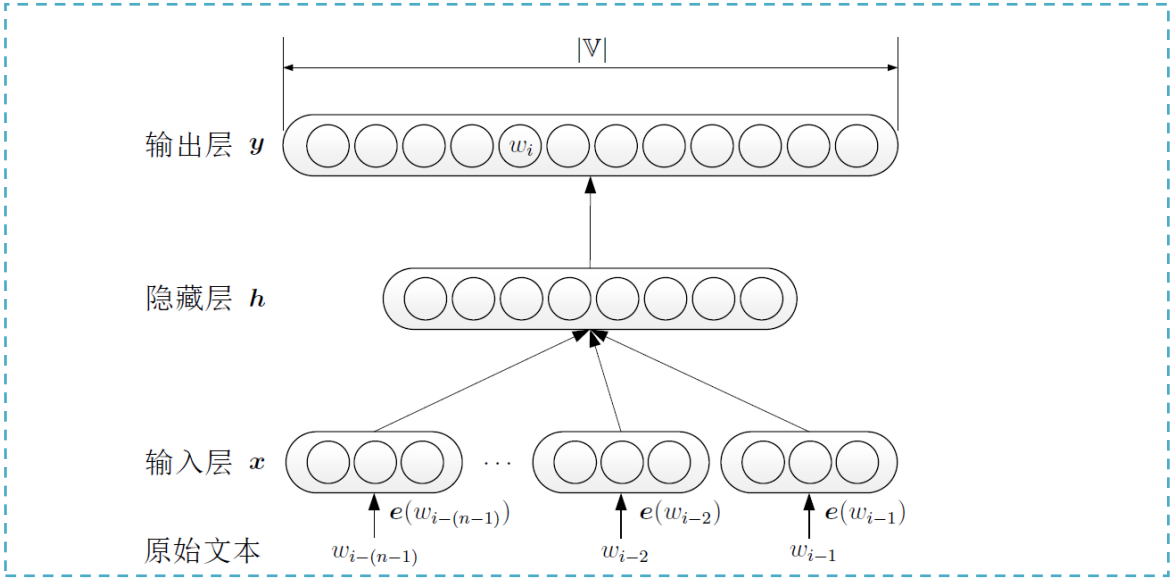
- C&W的优化目标是最小化：$\sum_{(w,c)\in D}\sum_{w'\in V}\max(0,1-\text{score}(w,c)+\text{score}(w',c))$
  - (w,c)为从语料中选出的一个n元短语；w为序列中的中间词（目标词）；c为w的上下文；w'为字典中的某个词

Ronan Collobert：A unified architecture for natural language processing: Deep neural networks with multitask learning, 2008

C&W VS. NNLM



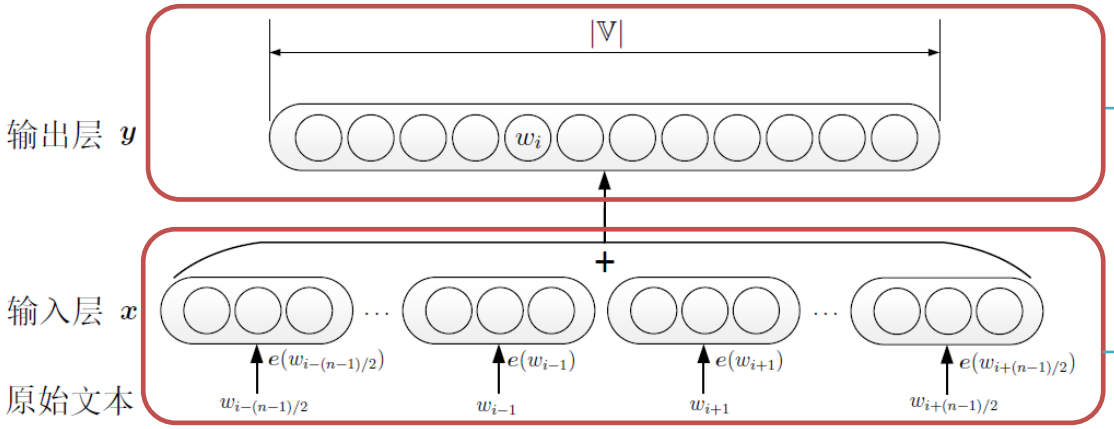
- 目标词在输入层
- 输出层只有1个节点
- 最后一层只需|h|次运算



- 目标词在输出层
- 输出层有|V|个节点
- 最后一层需|V|*|h|次运算，且需要进行softmax运算

CBOW模型

- CBOW借鉴了C&W使用上下文作为输入的思想，并在NNLM基础上进行了简化



CBOW模型结构图

根据上下文的表示，直接对目标词进行预测

$$P(w|c)=\frac{\exp(e'(w)^T x)}{\sum_{w'\in V}\exp(e'(w)^T x)}$$

对于一段长度为n的训练样本，输入为

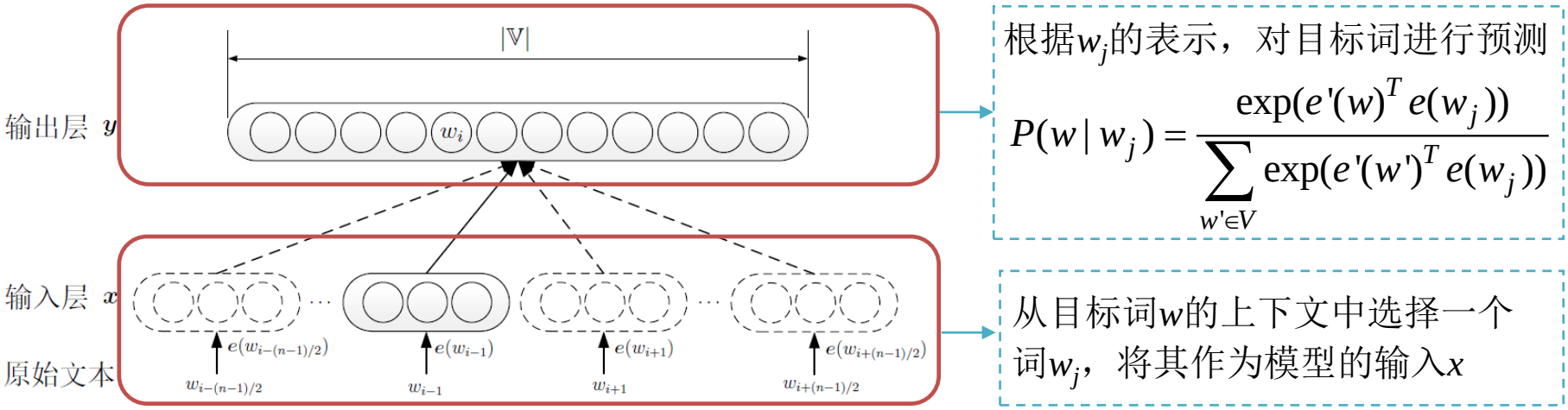
$$x=\frac{1}{n-1}\sum_{w_j\in c}e(w_j)$$

- CBOW的优化目标为最大化：$\sum_{(w,c)\in D}\log P(w|c)$
- 与NNLM的区别：1. 去除上下文的词序信息；2. 无隐藏层，NN转化为log线性结构

Tomas Mikolov：Efficient estimation of word representations in vector space，2013

Skip-gram模型

- Skip-gram与CBOW模型类似，模型中没有隐藏层；但模型的输入和优化目标不同



Skip-gram模型结构图

- Skip-gram的优化目标是最大化：$\sum_{(w,c) \in D} \sum_{w_j \in c} \log P(w \mid w_j)$
- 更多优化和提升：负采样、二次采样、层次softmax……

Tomas Mikolov：Efficient estimation of word representations in vector space，2013

神经网络语言模型小结

指标	分析
模型复杂度	NNLM>C&W>CBOW>Skip-gram
参数个数	NNLM>(CBOW=Skip-gram)>C&W
时间复杂度	NNLM>(CBOW=Skip-gram)>C&W
上下文表示	NNLM、C&W：n-gram的词向量拼接 CBOW ：n-gram中各词词向量的平均值 Skip-gram：上下文中的某个词的词向量
目标词与上下文词之间的关系	C&W：上下文和目标词都在输入层，优化组合关系 NNLM、CBOW、Skip-gram：上下文在输入层、目标词在输出层，优化预测关系

回答三个问题

- 不直接计算词之间的共现频度，直接基于词的上下文词来预测当前词（或基于当前词预测上下文词）

- 更快的学习速度：将$O(|V|^3)$降到$O(|V|)$，$O(log|V|)$？
  - CBOW和Skip-gram复杂度为$O(|e||V|)$
  - 层级softmax优化为$O(log|V|)$

- 更好的灵活性：加入新句子、新文档，词表里加入新词

2.词向量表示和神经语言模型

词向量模型应用

- 利用词向量的语言学特性完成任务
  - 分布假说：语义相似的词，其词向量空间距离更相近
  - 语义相关性、同义词检测、单词类比
- 将词向量作为特征，提高自然语言处理任务的性能
  - 使用静态词向量，在模型训练过程中，只调整模型参数，不调整输入词向量
  - 基于平均词向量的文本分类、命名实体识别等
- 将词向量作为神经网络的初始值（动态词向量），提升神经网络模型的优化效果
  - 使用动态词向量，模型训练过程中会调整词向量的初值（预训练→微调）
  - 基于卷积神经网络的文本分类、词性标注

同义词检测

Glove results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

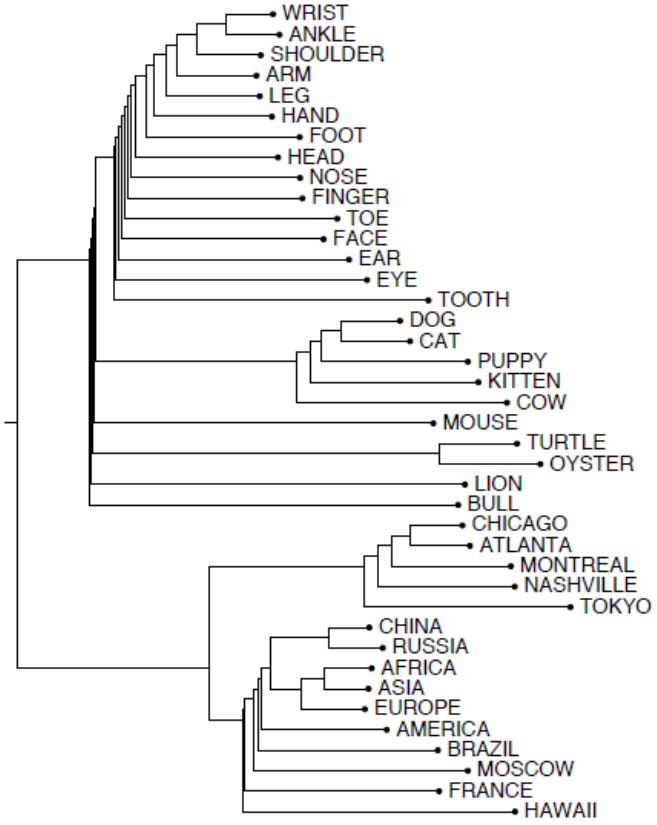
leptodactylidae

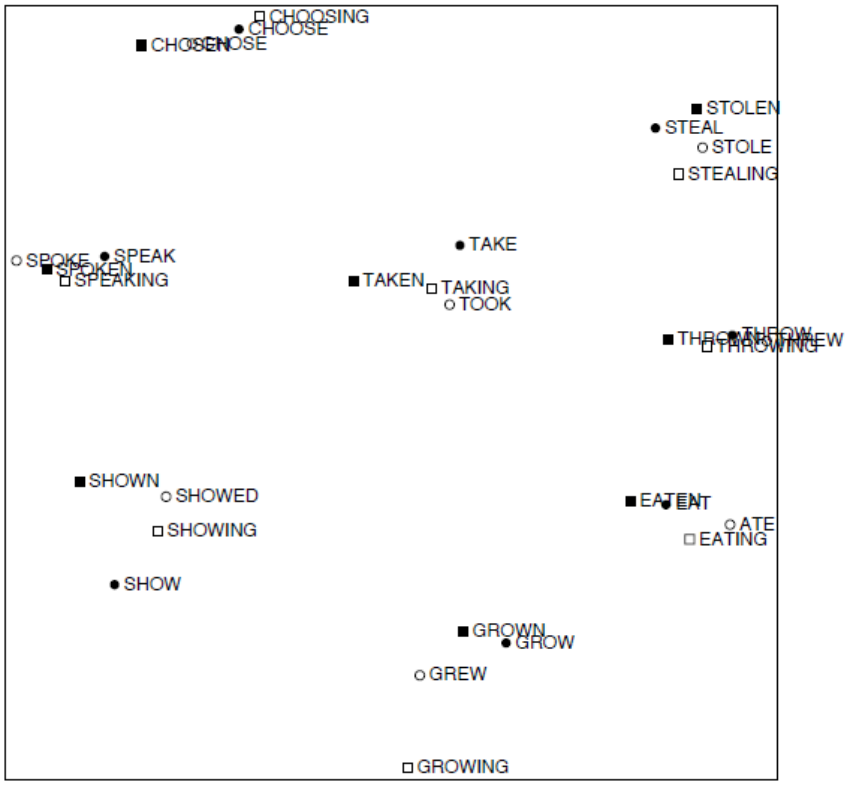
rana

eleutherodactylus

Pennington et al. Glove: Global Vectors for Word Representation, 2014

语义相似度度量





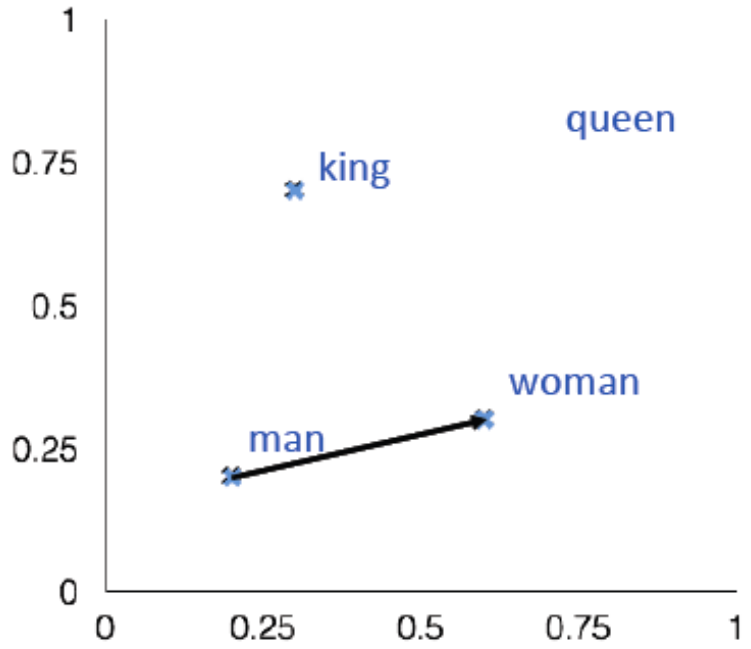
Rohde et al. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence, 2005

单词类比

- 相似关系词对的词向量之差也相似，直接使用词向量的加减法
  - 句法层：$x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
  - 语义层：$x_{king} - x_{man} \approx x_{queen} - x_{woman}$

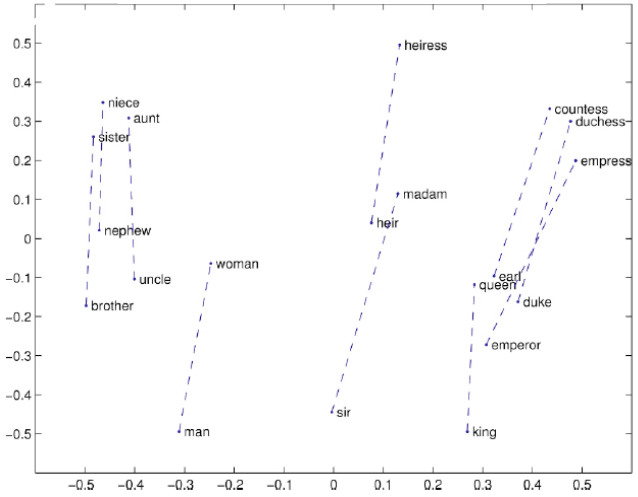
man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

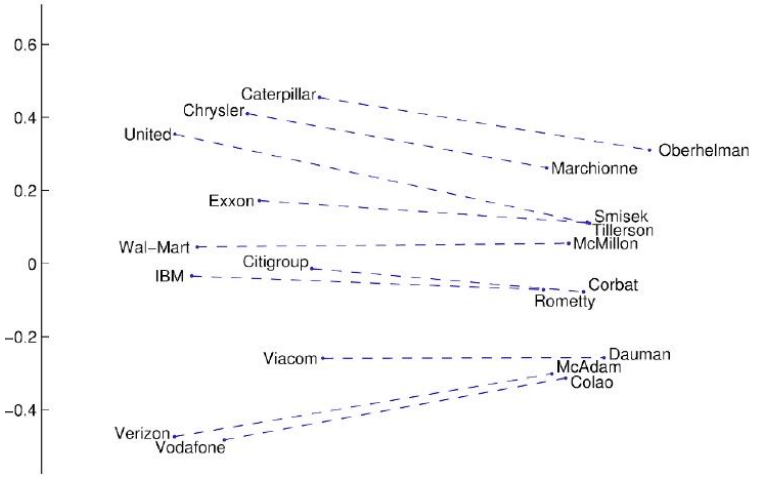


单词类比

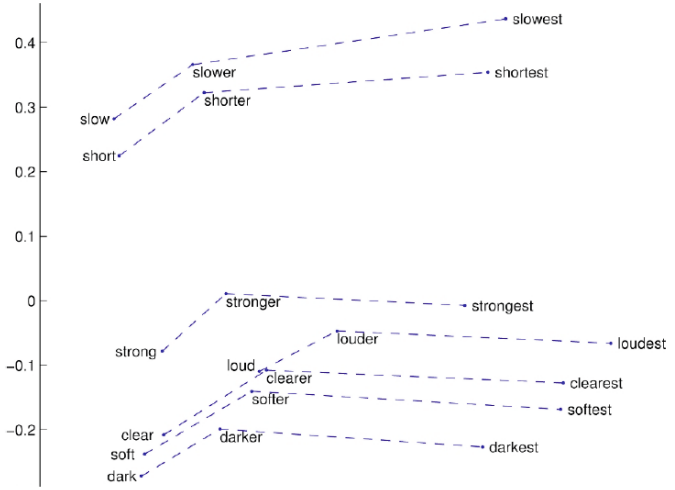
Glove Visualizations



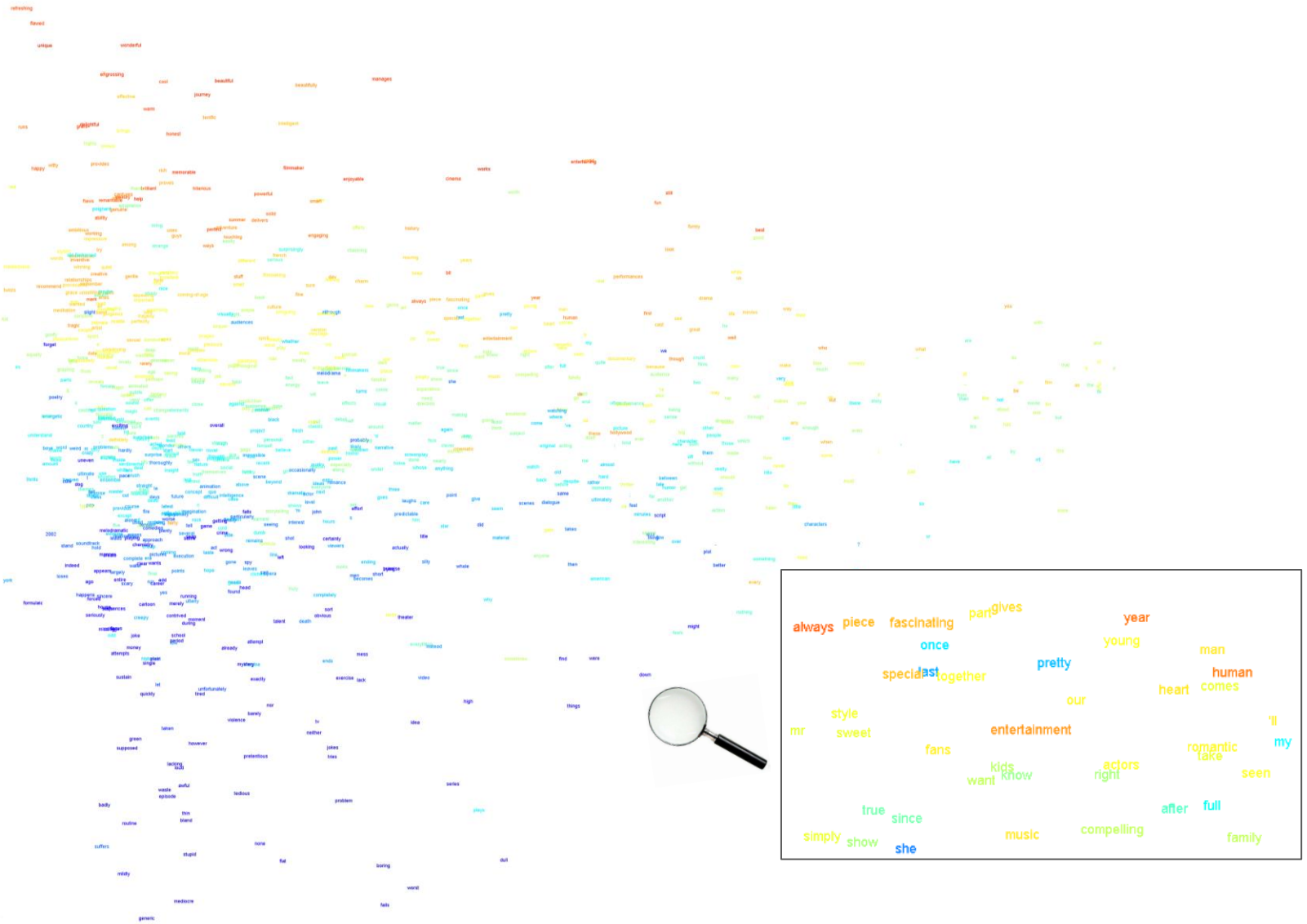
Company - CEO



Superlatives



情感分类


always
piece
fascinating
part
gives
year
young
once
man
pretty
special
last
together
human
heart
comes
our
style
mr
sweet
entertainment
'll
my
fans
romantic
take
kids
want
know
right
actors
seen
true
since
after
full
simply
show
she
music
compelling
family

应用词向量的几个问题

- 问题一：使用哪个模型效果更好？
  - 简单的模型在大多数情况下已经足够好。语料较大时，可以选择复杂的模型。预测目标词的模型比目标词与上下文呈组合关系的模型要好
- 问题二：训练语料的大小及领域对词向量有什么影响？
  - 使用领域内的语料，对同领域的任务有显著的提升。在此前提下，语料规模越大越好
- 问题三：如何选择迭代次数，以获得足够好的词向量，同时避免过拟合？
  - 迭代优化的终止条件需要根据具体任务的验证集来判断，或者近似地选取其它类似任务作为指标，不应选用训练词向量时的损失函数
- 问题四：多少维的词向量效果最理想？
  - 一般选择50维及以上。维度越高，其对词义的刻画粒度越细，效果越好（训练语料及训练时间允许的情况下）

2.词向量表示和神经语言模型

结合*n*-gram特征的嵌入表示

- 训练词向量时考虑单词*n*-gram特征（局部词序），如english-born和british-born共享后缀
  - 目标函数：$\sum_{t}^{T}\left[\sum_{c\in\mathcal{C}_t}\log\left(1+e^{-s(w_t,w_c)}\right)+\sum_{n\in\mathcal{N}_{t,c}}\log(1+e^{s(w_t,n)})\right]$

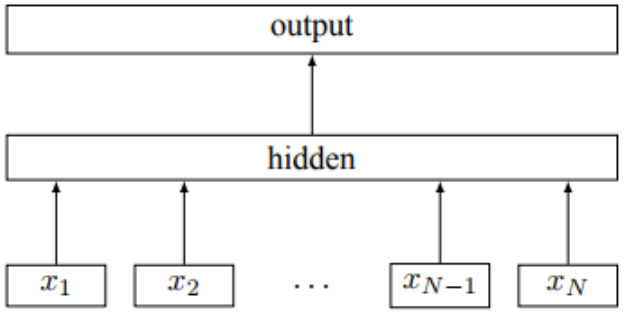


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

- 评分函数：$s(w,c)=\sum_{g\in\mathcal{G}_w} z_g^T v_c$
  - (w, c) 是一个词语对，w为中间词，c为上下文词
  - n 是词典中除去 w 和 c 外的其他词
  - z_g是*n*-gram 的向量表示
  - v_c是上下文词的向量表示
  - 用*n*-grams (n<N) 的所有向量的和表示中间词

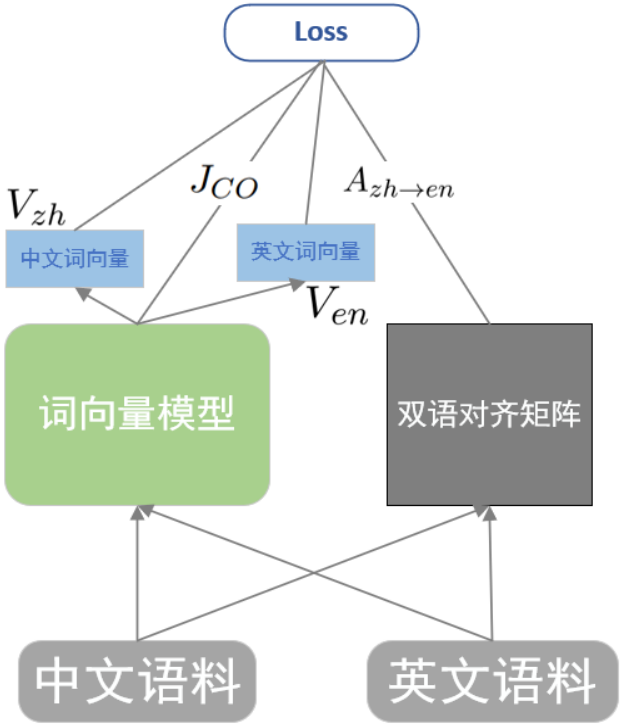
Piotr Bojanowski , et.al. Enriching Word Vectors with Subword Information, 2017

Charagram: Embedding Words and Sentences via Character n-grams

Bag of Tricks for Efficient Text Classification.

多语言词嵌入

- 双语词嵌入对齐：把不同语言单词嵌入到一个共享的空间

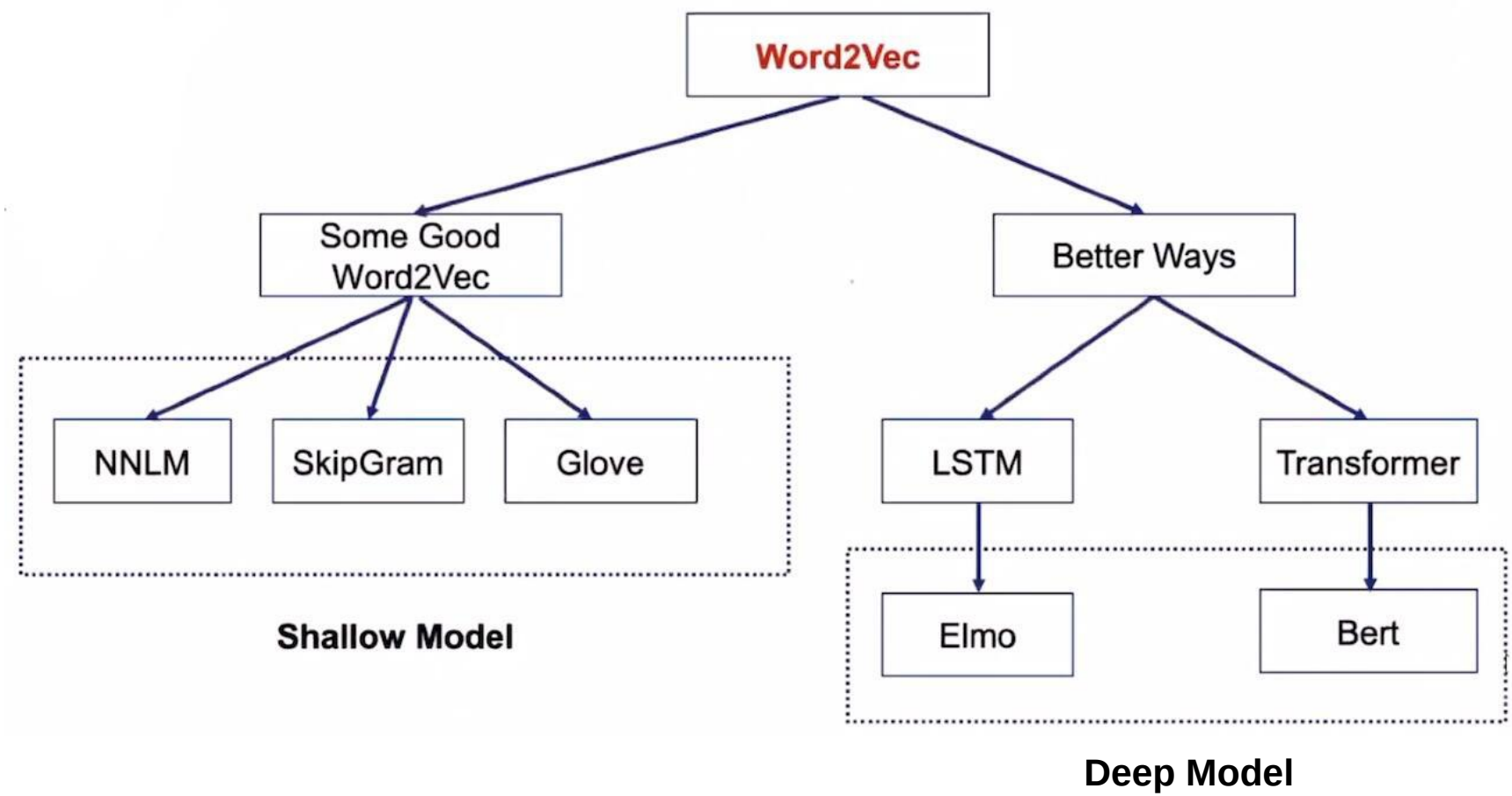


- 通过平行语料构造双语对齐矩阵
- 分别训练出中英文词向量并获得相应的训练误差
- 利用对齐矩阵，训练误差和词向量的乘积来计算最终误差：

$$J_{CO\text{-}zh} + \lambda J_{TEO\text{-}en\to zh}$$

$$J_{TEO\text{-}en\to zh} = \left\| V_{zh} - A_{en\to zh} V_{en} \right\|^2$$

Word2Vec—Shallow Model



参考文献

- Lectures
  - CS224d-Lecture2(Simple Word Embedding)
  - CS224d-Lecture3 (Word Embedding)
- Papers
  - 博士论文：基于神经网络的词和文档语义向量表示方法研究
  - Gregory W Lehser, Effects of ngram order and training text size on word prediction
  - Yoshua Bengio, et.al. A neural probabilistic language model（2001, 2003）
  - Ronan Collobert：A unified architecture for natural language processing: Deep neural networks with multitask learning, 2008
  - Tomas Mikolov：Efficient estimation of word representations in vector space，2013
  - Tomas Mikolov：Efficient estimation of word representations in vector space，2013
  - Pennington et al. Glove: Global Vectors for Word Representation, 2014
  - Rohde et al. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence, 2005
  - Piotr Bojanowski , et.al. Enriching Word Vectors with Subword Information, 2017

欢迎加入DL4NLP！

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS