NLP paper review <Efficient Estimation of Word Representations in Vector Space>

2023年7月31日 6:22

Day1:word2vec研究成果意义

- 语言模型：计算一个句子是句子的概率的模型
  - 通用语言模型：语言学家基于语法给出的模型，因为语言的变化更新和不同语言不同规则而不通用
  - 统计语言模型：用语料中的频率代替概率

$$P(s)=P(w_1)P(w_2|w_1)\cdots P(w_n|w_1\cdots w_n)\rightarrow\begin{cases}P(w_i)=\dfrac{count(w_i)}{N} & N:\text{语料大小}\\ P(w_i|w_{i-1})=\dfrac{P(w_i,w_{i-1})}{P(w_{i-1})}=\dfrac{\frac{count(w_i,w_{i-1})}{N}}{\frac{count(w_{i-1})}{N}}=\dfrac{count(w_i,w_{i-1})}{count(w_{i-1})}\end{cases}$$

问题&解决方法：

- 平滑操作：给那些没有出现过的词或者词组一个比较小的概率（Laplace smooth/加1平滑：每个词在原来出现的基础上+1）
  - i. 有些词在语料中很少见
  - ii. 短语太长，在语料中出现频率很低

$$\begin{cases}P(w_i)=\dfrac{count(w_i)+1}{N+V}, V:\text{词\&词组个数}\\ P(w_i|w_{i-1})=\dfrac{count(w_{i'}w_{i-1})+2}{count(w_{i-1})+1}\end{cases}$$

- Markov假设：下一个词的出现仅依赖于前面一个词或者两个词
  - i. 参数（$w_1,w_2,\cdots w_n$）空间过大：$V+V^2+V^3+\cdots+V^l$，$l:$句子长度
  - ii. 数据稀疏严重：浪费存储空间，浪费计算量

$$\begin{cases}unigram:P(s)=P(w_1)P(w_2)\cdots P(w_n) & \text{参数空间：}V\\ bigram:P(s)=P(w_1)P(w_2|w_1)\cdots P(w_n|w_{n-1}) & :V+V^2\\ trigram:P(s)=P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_n|w_{n-1}w_{n-2}) & :V+V^2+V^3\\ k-gram:P(s)=P(w_1)\cdots P(w_n|w_{n-k+1}\cdots w_{n-1}) & :V+V^2+V^3+\cdots+V^k\end{cases}$$

- 语言模型评价指标：Perplexity(困惑度)

$$P(s)=P(w_1)P(w_2|w_1)\cdots P(w_n|w_1\cdots w_n)$$

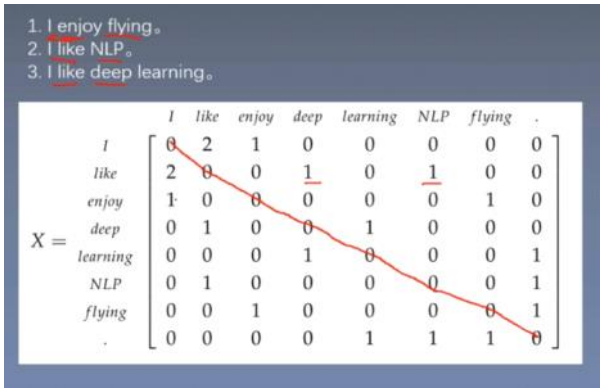
	Input	label	P
<start>	None	w_1	$P(w_1)$
	w_1	w_2	$P(w_2\|w_1)$
	w_1,w_2	w_3	$P(w_3\|w_1w_2)$
	.....	.....	.....
	$w_1,w_2,\cdots,w_{n-1}$	w_n	$P(w_n\|w_1\cdots w_n)$

$$\mathrm{PP}(s)=P(w_1,w_2,\cdots w_n)=\sqrt[n]{\frac{1}{P(w_1,w_2,\cdots w_n)}}$$

（句子概率越大，语言模型越好，困惑度越小，困惑越少越好）

Day1-word2vec-研究成果意义

- 词的表示方式（word representation）
  - One-hot Representation
    1. 表示简单【000010000000000...】
    2. 问题：词越多，维数越高（词表大小V）无法表示词和词之间的关系
       1) SVD(Window based Co-occurrence Matrix)
          i) 优点：可以一定程度上得到词与词之间的相似度
          ii) 缺点：矩阵越大，SVD矩阵分解效率低，学习得到的词向量可解释性差



σ_k代表特征值，特征值越大向量越重要

$X = U\Sigma V^T$

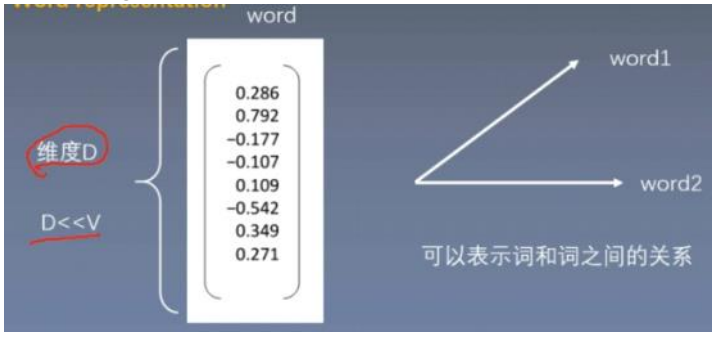
$$|V| \begin{bmatrix} & X & \end{bmatrix} = |V| \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} |V| \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} |V| \begin{bmatrix} - & v_1 & - \\ & v_2 & \\ & \vdots & \end{bmatrix}$$

$x_{ij} = \sum_{k=1}^{n} u_{ik}\sigma_k v_{jk}$ ⇓ 保留前k个

$$|V| \begin{bmatrix} & X & \end{bmatrix} = |V| \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} k \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} k \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix}$$

- Distributed Representation

  Word embedding: 可以表示词和词之间的关系



研究成果：

- 提出新的模型结构
- 提出优化训练的方法，使得训练速度加快
- 给出训练代码word2vec, 使得单机训练成为可能
- 成果：训练的词向量，又快又好，并且能够在大规模语料上进行词向量训练

研究意义：

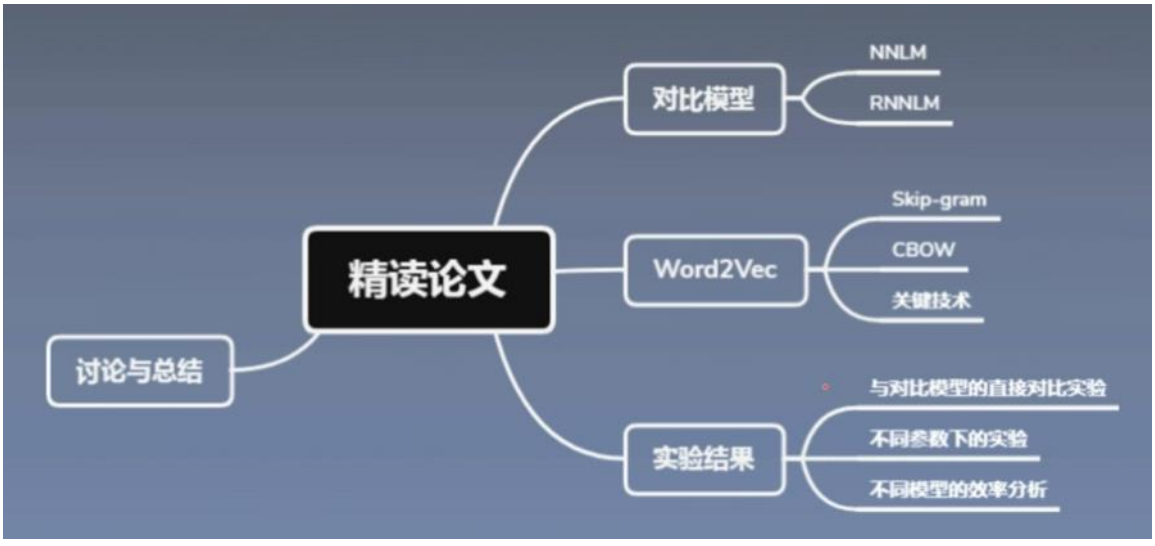
- 衡量词向量之间的相似程度 $sim(word1, word2) = \cos(wordvec1, wordvec2)$
- 词类比analogy: $\cos(word1 - word2 + word3, wordvec4)$， $v(France) - v(Paris) = v(Italy) - v(Rome)$
- 作为预训练模型提升NLP任务：
  - 外部任务，比如命名实体识别、文本分类。
  - 应用到其他NLP任务上，相当于半监督训练
- 双语单词嵌入

论文总览：

- Abstract: 提出2种新的高效计算词向量结构，并使用词相似度任务验证效果

- Introduction: 介绍词向量背景：本文目标，前人工作
- Model Architectures: LSA/LDA ； 前向神经网络；循环神经网络；并行网络计算
- Examples of the Learned Relationships: 例子：学习到词与词之间的关系
- Results: 评价任务描述；最大化正确率；模型结构比较；模型上大量数据的并行计算；微软研究院句子完成比赛
- New Log-Linear Models:介绍两种新模型结构：CBOW,skip-grams
- Conclusion:高质量词向量；高效率训练方式；作为预训练词向量作用与其他NLP任务能提升效果
- Follow-Up work: C++单机coding
- References

Day 2: 对比模型-03 （论文精讲）

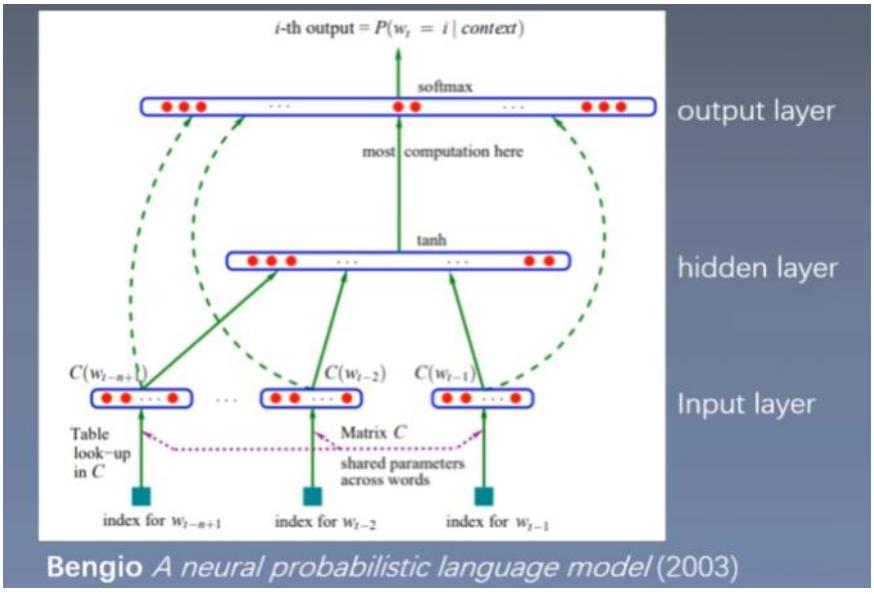


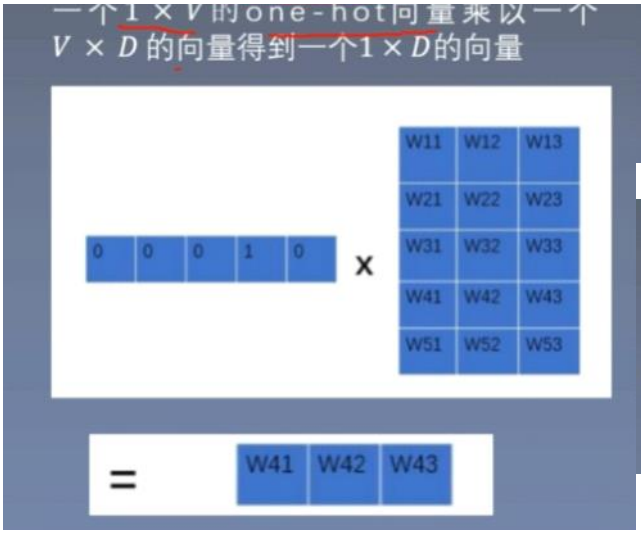
1. Feedforward Neural Net Language Model(NNLM):根据前n-1个词推测第n个词的概率（基于Markov假设）。优化模型就是使得输出正确单词的概率最大化。

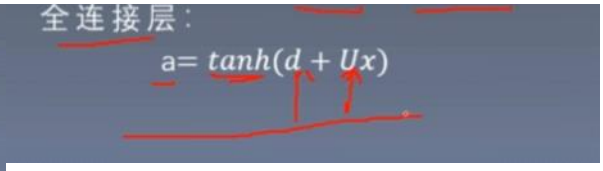
$$\left.\begin{matrix} word2id \\ id2word \end{matrix}\right\} dict$$

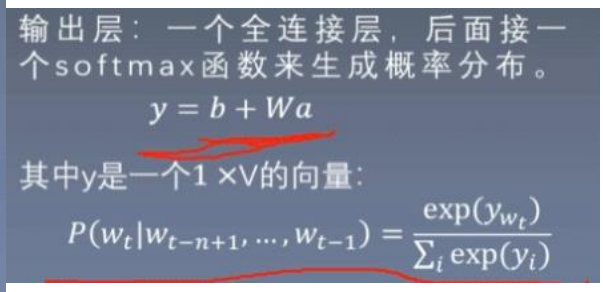
语言模型都是unsupervise模型

a. 过程：









b. 语言模型困惑度和Loss的关系：

$Loss: L = -\frac{1}{T}\sum_{i=1}^{T} \log p(w_i|w_{i-n+1}, \dots, w_{i-1})$ ⇒ T: 句子中词的个数

$$PP(s) = P(w_1, \dots, w_T)^{-\frac{1}{T}} = \sqrt[T]{\frac{1}{P(w_1, \dots, w_T)}}$$

$$\log(PP(s)) = -\frac{1}{T}\log(P(w_1)P(w_2|w_1)\dots P(w_T|w_{T-n+1}, \dots, w_{T-1})$$

$$\log(PP(s)) = -\frac{1}{T}\sum_{i=1}^{T} \log p(w_i|w_{i-n+1}, \dots, w_{i-1})$$

$$PP(s) = e^L$$

注意：由于batch训练，为了每次句子长度一样要补充pad，补充后算困惑度的时候记得考虑把pad影响去掉

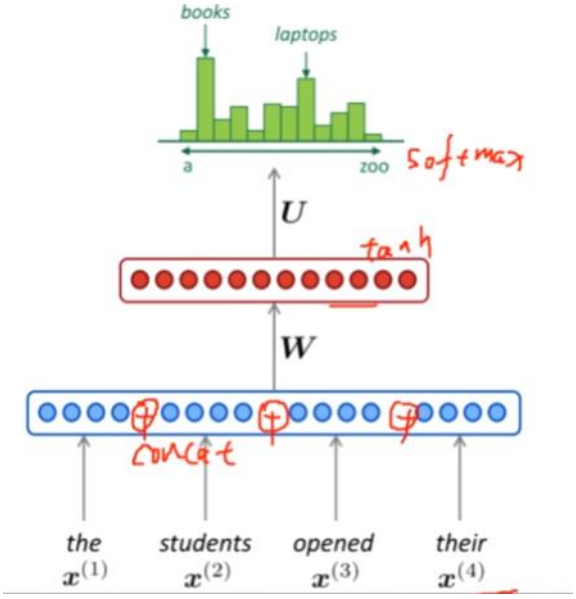
c. 改进：

i) 仅对一部分输出进行梯度传播，针对the,a,and这种出现频率高但是没什么用的词

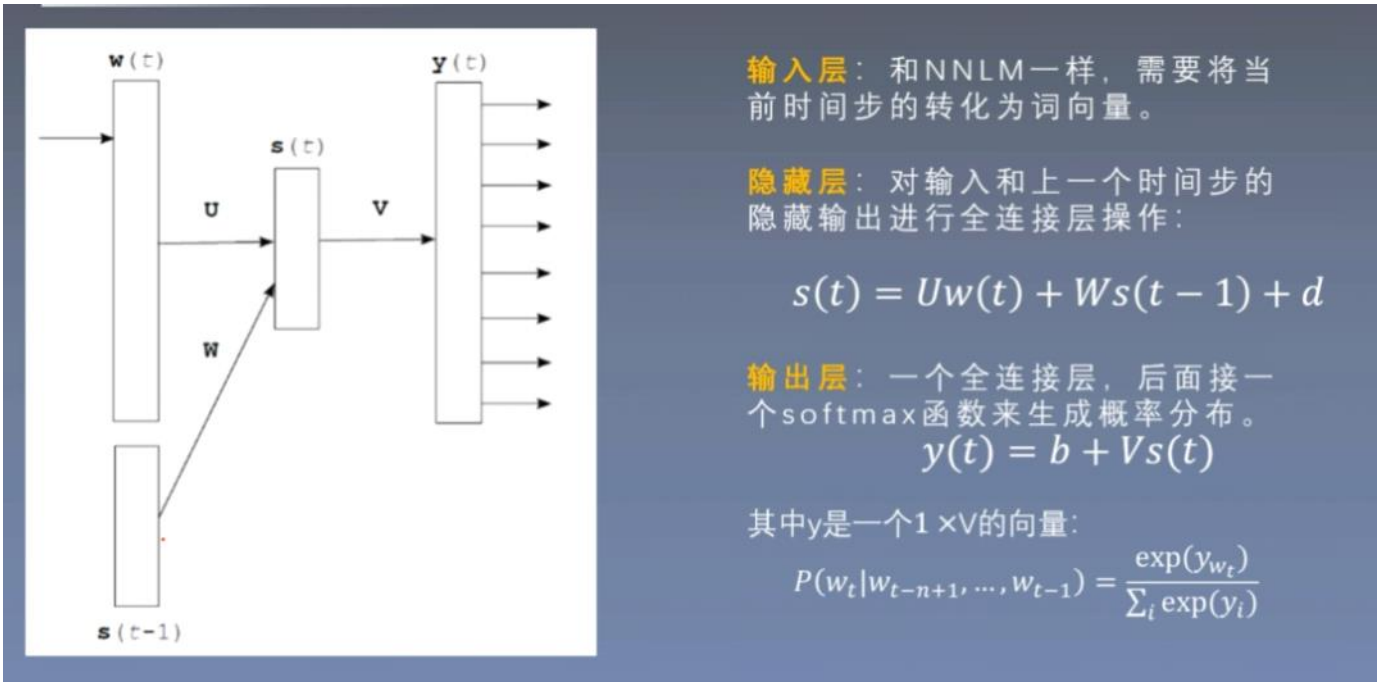
ii) 引入先验知识，如词性等。（但要考虑网络本身能否学习词性->可以 学习到的词性够不够用/准确->不太够用）

iii) 解决一词多义问题。通过上下文

iv) 加速softmax层:1)层次softmax 2)负采样

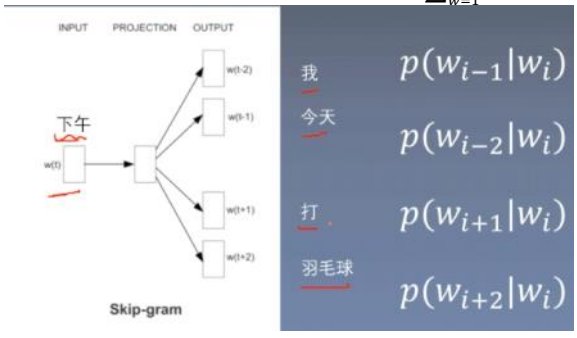


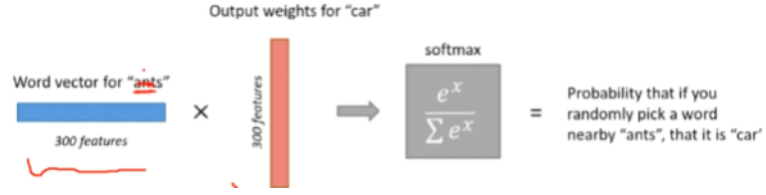
2. Recurrent Neural Net Language Model RNNLM:每个时间步预测一个词，在预测第n个词的时候用了前面n-1个词的信息。（不需要基于Markov）



Day2: word2vec-对比模型04

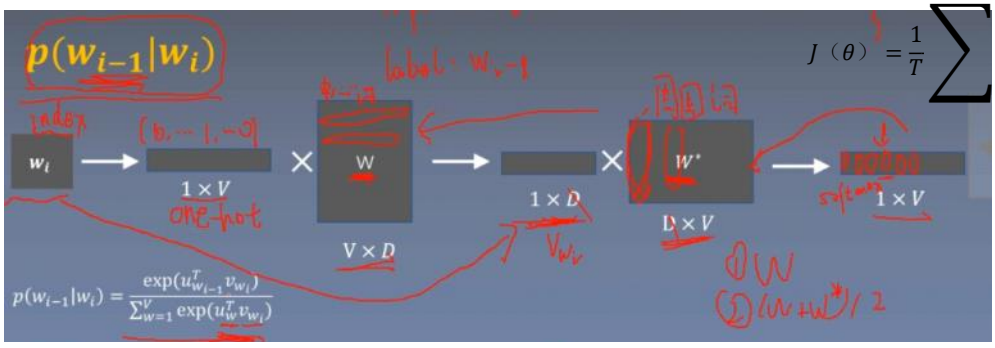
- Log-linear model:将语言模型的建立看成一个多分类问题，相当于在线性分类器加上softmax：$Y = softmax(wx + b)$
- 语言模型的基本思想是：句子中下一个词的出现和前面的词是有关系的，所以可以使用前面的词预测下一个词。
- Word2vec基本思想：句子中相近的词之间是有联系的，比如今天后面经常出现上午，下午和晚上。所以word2Vec的基本思想就是用词来预测词，skip-gram使用中心词预测周围词，cbow使用周围词来预测中心词。
  - Skip-gram（跳字模型）：$p(w_{i-1}|w_i) = \frac{\exp(u_{w_{i-1}}^T v_{w_i})}{\sum_{w=1}^{V} \exp(u_w^T v_{w_i})}$





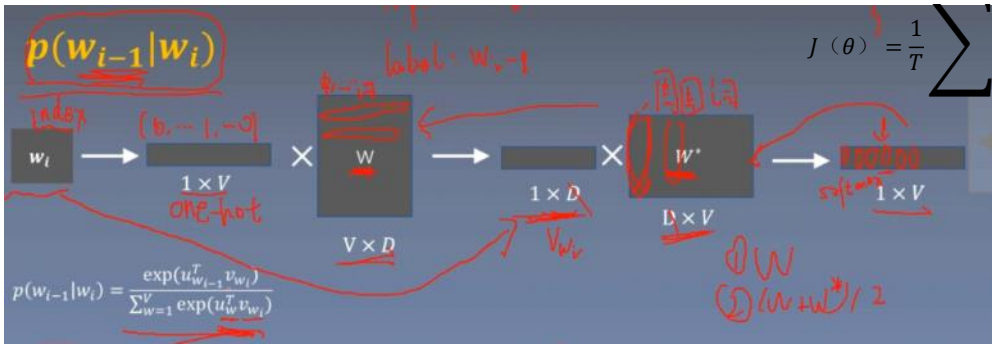
$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$，$v_c$是中心词向量

u_o，u_w是窗口内上下文词向量



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

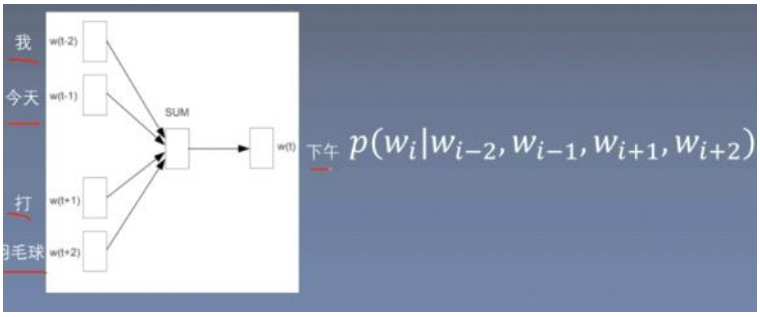
求最大值

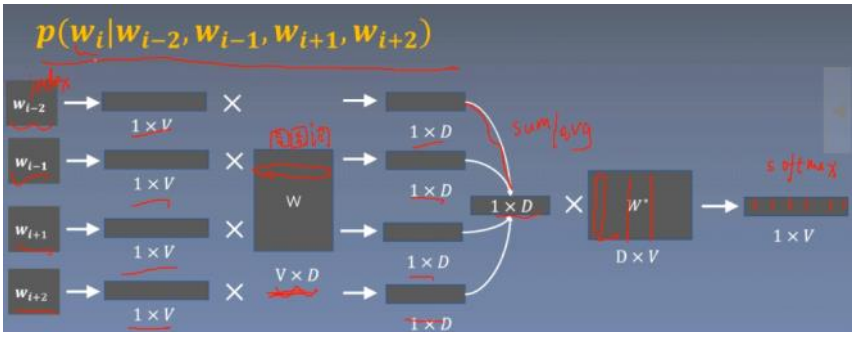


$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} \log p\left(w_{t+j}\middle|w_t\right)$$

求最大值

- CBOW:





$$J(\theta) = \frac{1}{T}\sum_{T}\sum \log P(c|o) = \frac{1}{T}\sum \frac{\exp\{u_o^T v_c\}}{\sum_{j=1}^{V}\exp\{u_o^T v_j\}}$$

e_1, e_2, e_3, e_4上下文词

$u_o = \text{sum}(e_1, e_2, e_3, e_4)$

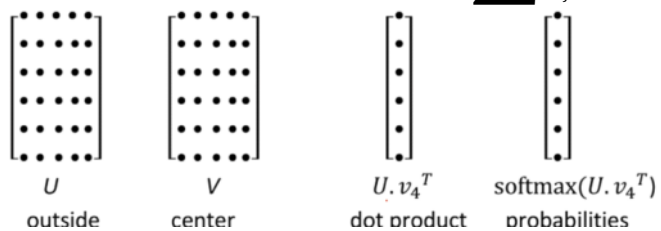
- $$P(c|o) = \sum \frac{\exp\{u_o^T v_c\}}{\sum_{j=1}^{V}\exp\{u_o^T v_j\}} \rightarrow softmax$$

u_o是窗口内上下文词向量和

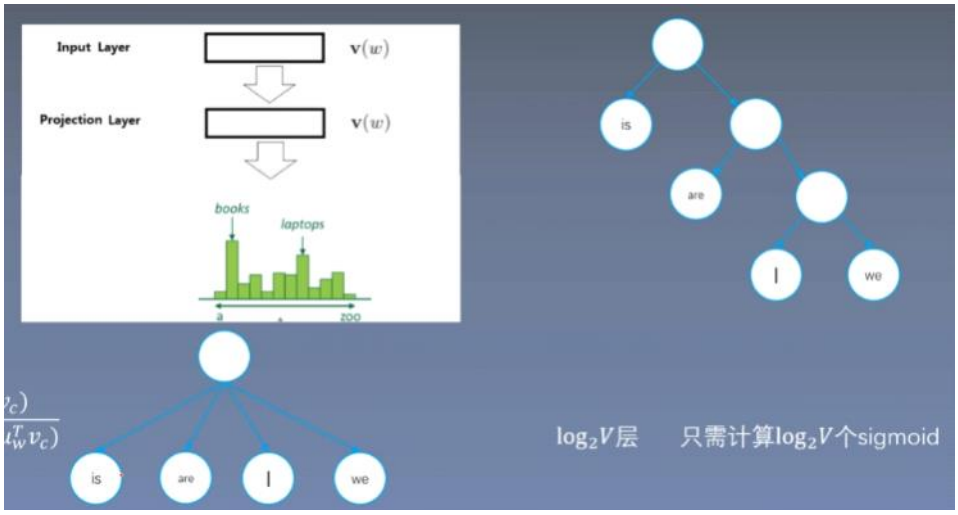
v_c,v_j 是中心词向量

Day2-word2vec 关键技术 05

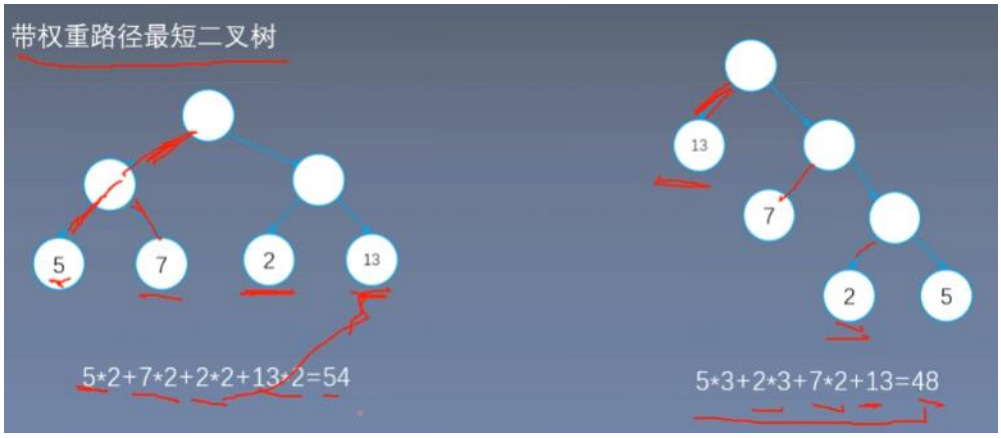
- 如何降低复杂度：softmax要输出V个概率，V特别大 $P(c|o) = \sum \frac{\exp\{u_o^T v_c\}}{\sum_{j=1}^{V}\exp\{u_w^T v_c\}}$ *V 个指数操作*



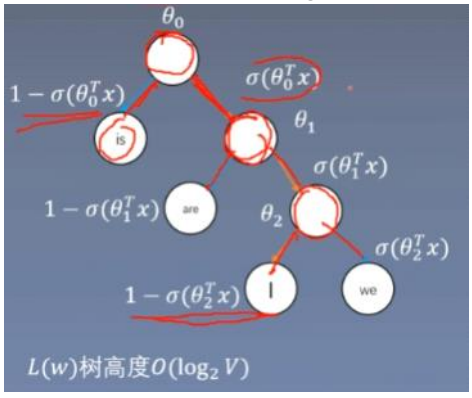
- Hierarchical softmax 层次softmax： softmax->sigmoid
  - 二叉树

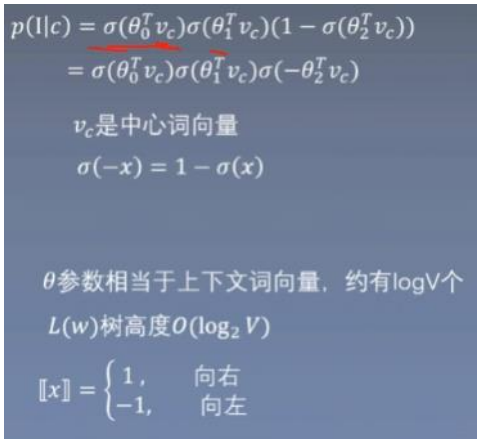


- Huffman树：带权重路径最短二叉树

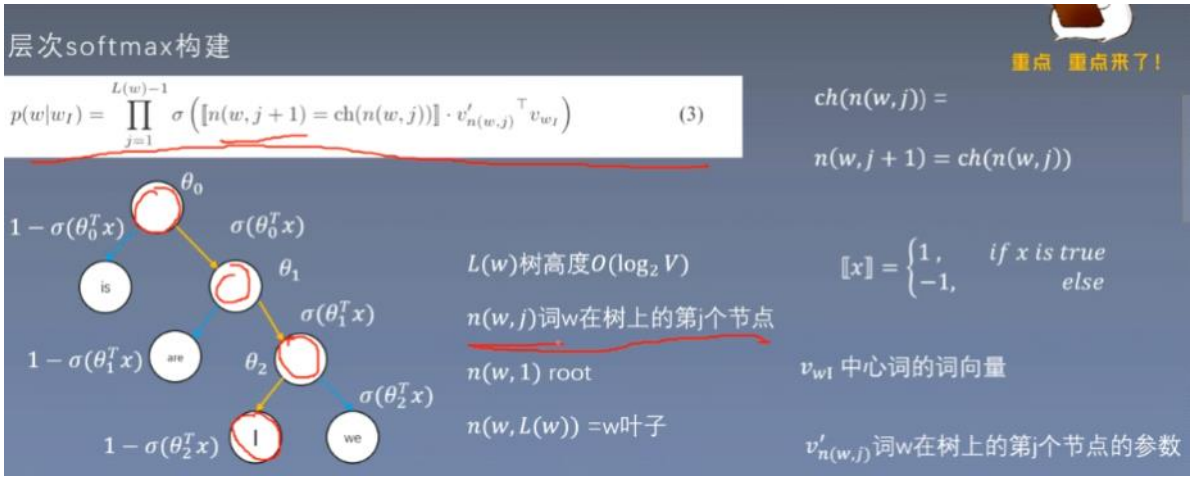


- Hierarchical softmax 构建：skip-gram举例

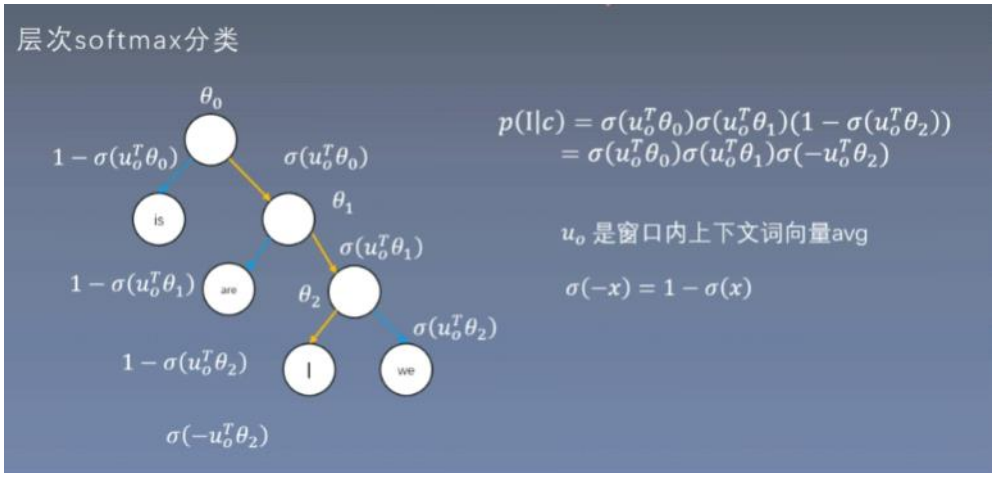




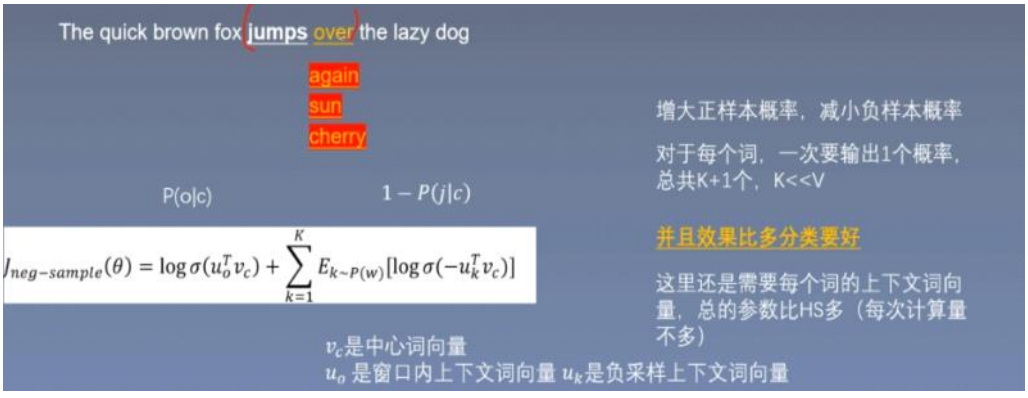
- 层次softmax构建

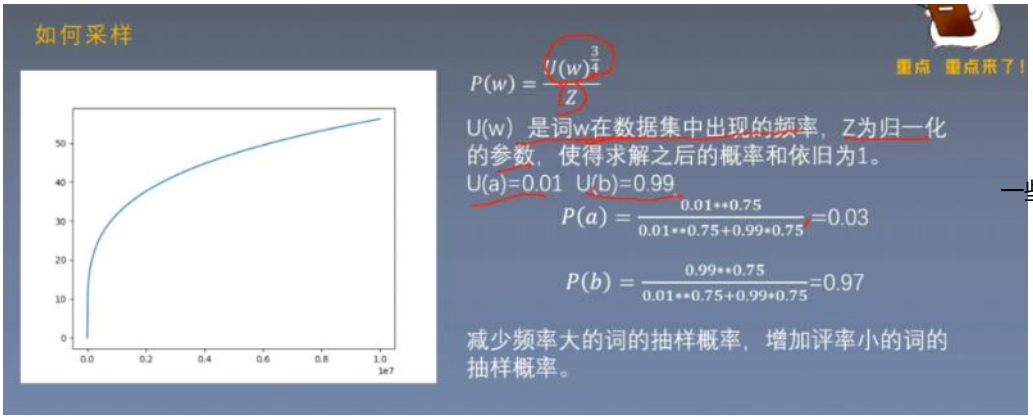


  - CBOW hierarchical softmax :

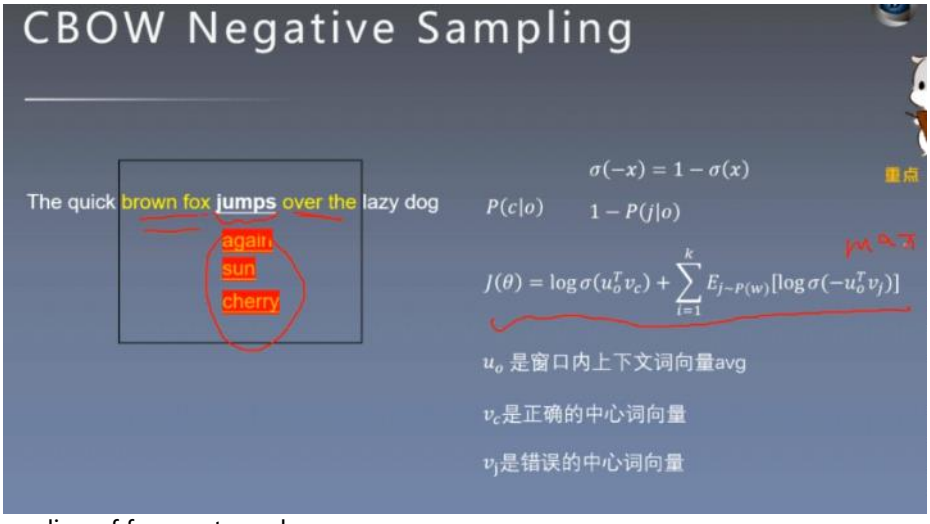


- Negative Sampling : 舍弃多分类，提升速度。增大正样本概率，减少负样本概率，对于每个词，一次要输出1个概率，总共K+1个，K<<V 一般情况下选3~10（K）个负样本，1个负样本会有偏差

  - 

  - 

    一些重要词出现频率反而很小

- 

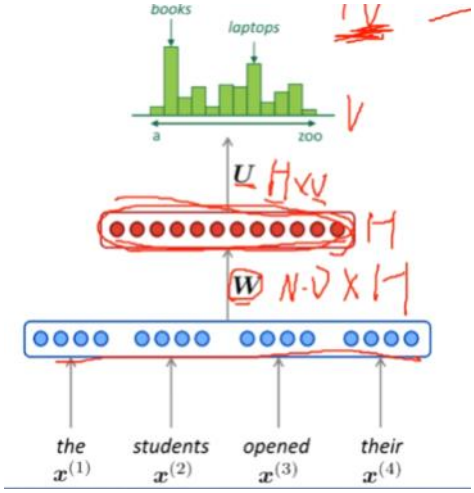
- Subsampling of frequent words:
  - 自然语言处理共识：文档或者数据集中出现频率高的词往往携带信息较少，比如the,is,a,and， 而出现频率低的词往往携带信息多
  - 重采样原因：
    - 想要更多地训练重要的词对，比如训练‘France’和‘Paris’之间的关系比训练'France'和 ‘the’之间的关系要有用。
    - 高频词很快就训练好了，而低频词需要更多的轮次。
      - 重采样方法：$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$ 其中$f(w_i)$为词w_i在数据集中出现的概率。文中t选取为10^{-5}，训练集中的词w_i会以的$P(w_i)$概率被删除。
      - 词频越大，$f(w_i)$越大，$P(w_i)$越大，那么词w_i就有更大的概率被删除，反之亦然。如果词w_i的词频小等于t，那么不会被删除。
      - 优点：加速训练，能够得到更好的词向量。

结论：

- word2vec介绍了两种模型
  - skip-gram：用中心词预测周围词
  - CBOW: 用周围词预测中心词：预测方法是使用周围词的平均来预测中心词
- 其中还有两种加速训练的方法：
  - Hierarchical softmax: 使用二叉树/Huffman树将softmax转成多此sigmoid（不再常用）
  - Negative sampling：将多分类问题转换成二分类问题，其中一个中心词和一个周围词就是一个正样本，一个中心词和一个随机采样的周围词就是一个负样本
  - Subsampling of frequent words:把高频词删除一些，低频词多做保留，这样就能训练的更快

Day2-word2vec-模型复杂度

- Model complexity: O=E*T*Q
  - O:training complexity
  - E:number of the training epochs 训练多少轮
  - T: number of words in the training set 每轮训练多少次
  - Q: model computational complexity 每次训练的时间（本文用参数数目代替。这里比较的就是Q）
- 模型复杂度：
  - Bengio A neural probabilistic language model(2003)

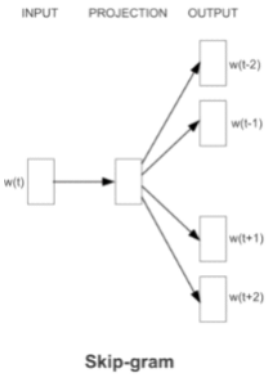
- 

  - $y = U\tanh(d + Wx)$（U维度：$V * H, W$维度：$N * D * H$）$\xrightarrow{softmax} p(x_i|x_{i-1}, \dots, x_{i-N}) = \frac{e^{y_i}}{\sum_{j=1}^{V} e^{y_i}}$
  - $y = [y_1, y_2, \dots, y_V]$　　N个上文词，D：词向量维度，V：词表大小
    $x = [x_1, x_2, x_3, x_4]$维度: $N * D$　　H：隐藏层大小

$$Q = V * H + N * D * H + N * D$$
$$\uparrow \text{Hierarchical softmax} \rightarrow \log_2 V * H$$

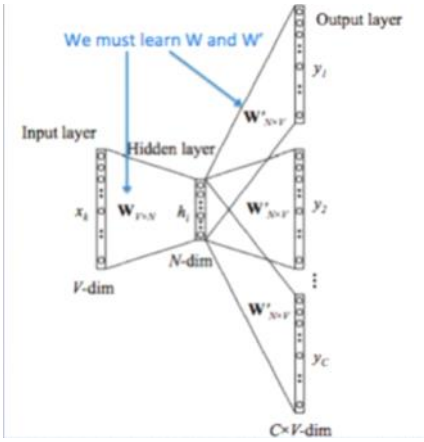
- RNNLM:
  $s(t) = Uw(t) + Ws(t-1)$
  $y(t) = Vs(t) \xrightarrow{softmax} p(x_i|x_{i-N}, \dots, x_{i-1}) = \frac{e^{y_i}}{\sum_{j=1}^{V} e^{y_i}}$
  - $y = [y_1, \dots, y_V]$
    $w: H, s: H$
    $W: H * H, U: H * H$　　$w(t)$: 第t个时刻的当前输入单词的词向量
    $V: H * V$　　$s(t-1)$: 隐藏层的前一次输出
    $Q = H * H + H * V$　　$y(t)$: 输出词的index

    $Q = 1 * D + D * H + H * H + H * V$
    $\approx 1 * H + H * H + H * H + H * V$
    $\approx H * (1 + 2H) + H * V$
    $\approx H * H + H * V$
- Skip-gram
  $$Q = C(D + D * V)$$
  - $\xrightarrow{Hierarchical} Q = C(D + D * \log_2 V)$　C: *周围词个数*



Skip-gram

  - $\xrightarrow{NegativeSampling} Q = C(D + D * (K + 1))$

- 

- CBOW:
  - $Q = N * D + D * V$
  - $\xrightarrow{Hierarchical} N * D + D * \log_2 V$
  - $\xrightarrow{NegativeSampling} N * D + D * (K + 1)$
- 前馈神经网络$Q = N * D + N * D * H + H * \log_2 V$

  循环神经网络$Q = H * H + H * \log_2 V$

  CBOW+HS: $Q = N * D + D * \log_2 V$

  Skip-Gram+HS: $Q = C(D + D * \log_2 V)$

  CBOW+NEG: $Q = N * D + D * (K + 1))$

  Skip-Gram+NEG: $Q = C(D + D * (K + 1))$

N:周围词的数目，左5右5->10

D：1000

H：1000

V：10000->$\log_2 V \approx 14$

K:5

Day2 实验结果

- 任务描述：5个语义类，9个语法类
  - 常见国家首都: capital-common-countries
    各国首都: capital-world
    货币: currency
    州-城市: city-in-state
    家庭关系: family
    形容词-副词: gram1-adjective-to-adverb
    反义词: gram2-opposite
    比较级: gram3-comparative
    最高级: gram4-superlative
    现在进行式: gram5-present-participle
    国家的形容词: gram6-nationality-adjective
    过去式: gram7-past-tense
    复数: gram8-plural
    第三人称单数: gram9-plural-verb
- 最大化正确率（优化参数）
  - 用小数据集调参，选择最好的参数
  - 维度，以及训练数据量，是2个需要寻找的参数
- 模型比较：
  - Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

    - RNNLM:单机用了8周， NNLM计算量更大
    - RNN：相对在语法上更好

- NNLM:效果更好
- CBOW:更好
- skip-gram：更平衡，在语义上效果好

- 与其他人开源的词向量比较

Table 4: *Comparison of publicly available word vectors on the Semantic-Syntactic Word Relationship test set, and word vectors from our models. Full vocabularies are used.*

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	**64.5**	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	**50.0**	55.9	**53.3**

Table 5: *Comparison of models trained for three epochs on the same data and models trained for one epoch. Accuracy is reported on the full Semantic-Syntactic data set.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

3 epoch>1 epoch
3 epoch 300 dim < 1 epoch 600 dim

- Large scale parallel training of models

Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

- Microsoft Research Sentence Completion Challenge

Table 7: *Comparison and combination of models on the Microsoft Sentence Completion Challenge.*

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

- Compare HS and Neg
  - 重采样加速训练，Neg比HS好

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55