0 大纲

1. 回顾词向量和word2vec
2. 优化基础
3. 基于统计的词向量
4. Glove模型和词向量

1 回顾词向量和word2vec

关于word2vec的更多细节

1 为什么每个词要对应两个词向量？

虽然一个词对应一个词向量也能够实现，但每个词对应两个词向量的平均优化时间更短。

2 两个模型

1. Skip-grams（SG）

   给出中心词（center word）预测上下文的词（outside）

2. Continuous Bag of Words（CBOW）

   从大量的上下文词汇中预测中心词。（朴素贝叶斯）

3 训练技巧

1. Negative sampling（负采样）
2. Naive softmax（simple, but expensive, training method）

2 优化基础

部分优化方法可以查看上节课笔记的第5部分：

https://blog.csdn.net/whatiscode/article/details/106944218

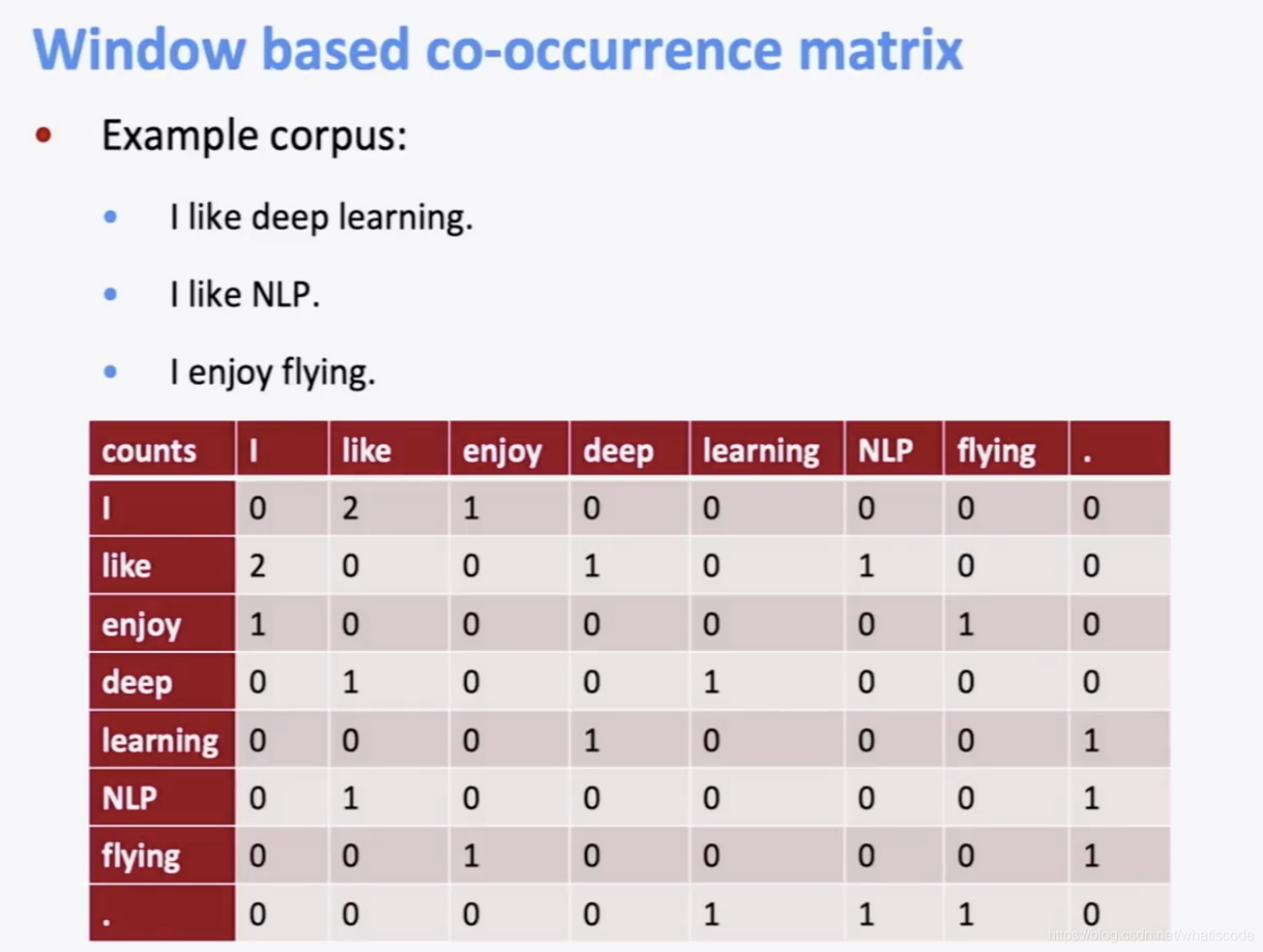
梯度下降（GD）、随机梯度下降（SGD）、小批量随机梯度下降法(Mini_Batch Gradient Densent)的区别可以查看这篇文章：

https://blog.csdn.net/weixin_40769885/article/details/85776177

3 基于统计的词向量

词向量目的：希望通过低维稠密向量来表示词的含义

3.1 基于窗口的co-occurrence矩阵



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

3.2 简单co-occurrence向量存在的问题

- 当词的数目是在不断增长，则词向量的维度也在不断增长
- 词汇很多时，矩阵很大，需要的存储空间也很大
- 矩阵很稀疏，即词向量很稀疏，会遇到稀疏计算的问题，从而导致模型不够鲁棒

解决办法：Low dimension vectors

3.3 Low dimension vectors

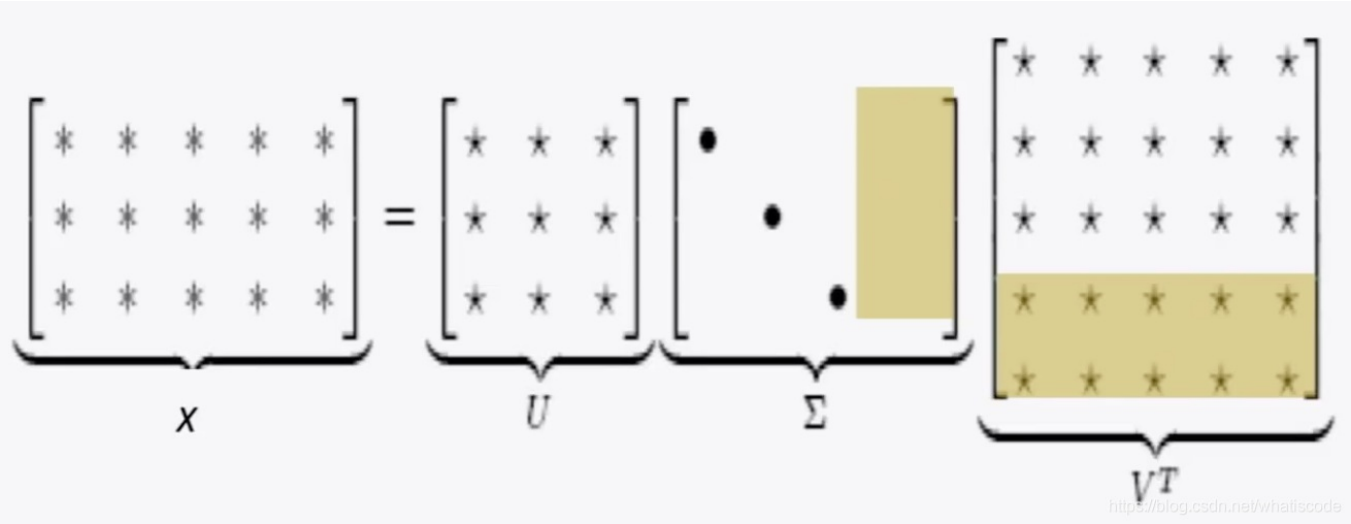
主要思想

将大部分重要信息用固定的、比较小的维度来存储：使用稠密向量。

向量的维度通常在25～1000之间，与word2vec类似。

那怎么样降低维度呢？

- Singular Value Decomposition（SVD，奇异值分解）

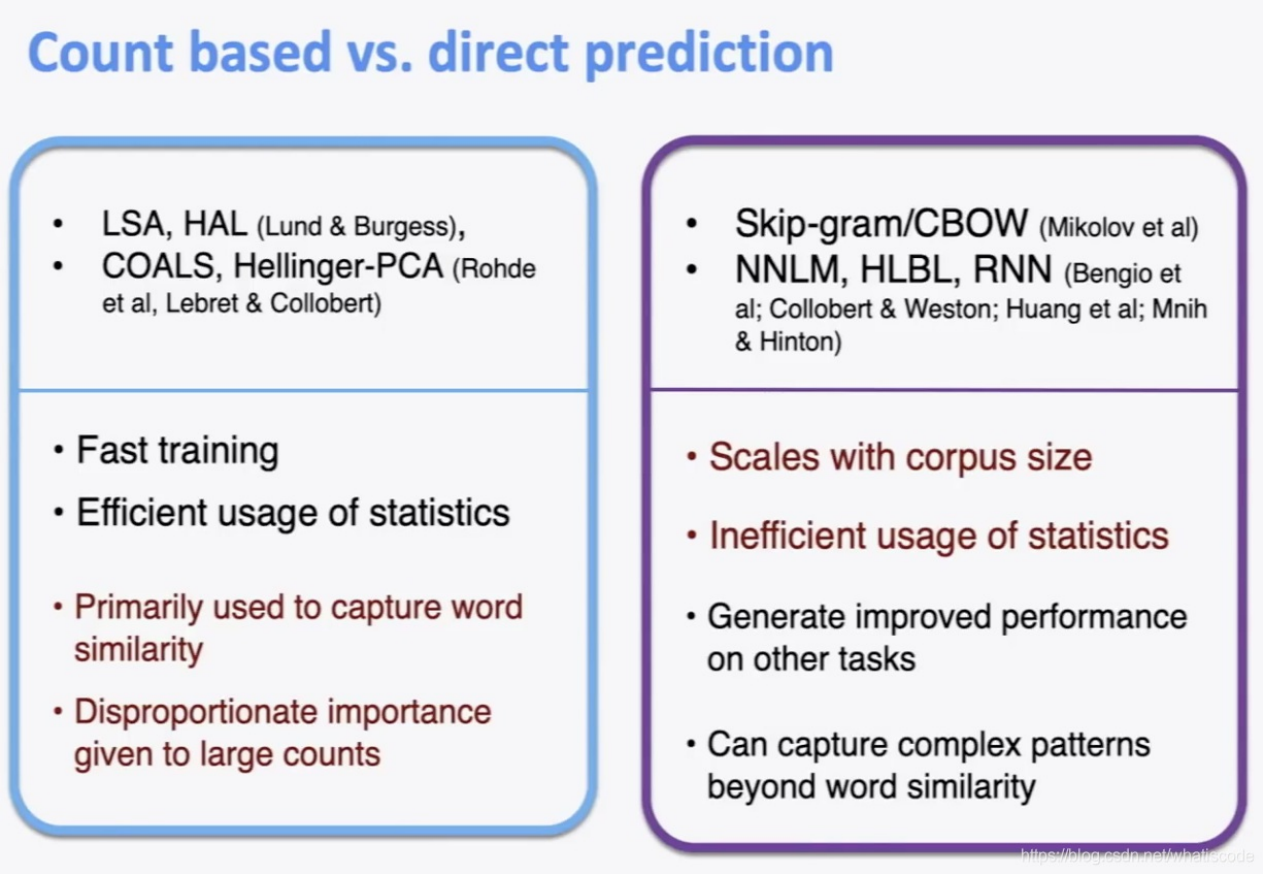


3.4 Hacks to X

https://pdfs.semanticscholar.org/73e6/351a8fb61afc810a8bb3feaa44c41e5c5d7b.pdf

上述链接中的文章对例子中简单的计数方法进行了改进，包括去掉停用词、使用倾斜窗口、使用皮尔逊相关系数等，提出了COALS模型，该模型得到的词向量效果也不错，具有句法特征和语义特征。

3.5 基于统计vs直接预测



4 Glove模型和词向量

模型目标：词进行向量化表示，使得向量之间尽可能多地蕴含语义和语法的信息。

流程：输入语料库–> 统计共现矩阵–> 训练词向量–>输出词向量。

4.1 构建统计共现矩阵X

X i j X_{i j} Xij代表单词

i表示上下文单词

j表示在特定大小的上下文窗口（context window）内共同出现的次数。这个次数的最小单位是1，但是GloVe不这么认为：它根据两个单词在上下文窗口的距离dd.

提出了一个衰减函数（decreasing weighting）：用于计算权重，也就是说距离越远的两个单词所占总计数（total count）的权重越小。

4.2 构建词向量和共现矩阵之间的关系

w i T w ~ ∗ j + b ∗ i + b ~ ∗ j = log ⁡ (X ∗ i j) w_{i}^{T} \tilde{w}*{j}+b*{i}+\tilde{b}*{j}=\log \left(X*{i j}\right) wiTw~∗j+b∗i+b~∗j=log(X∗ij)

其中， w i T w_{i}^{T} wiT 和 w ~ ∗ j \tilde{w}*{j} w~∗j是我们最终要求解的词向量； b ∗ i b*{i} b∗i和 b ~ ∗ j \tilde{b}*{j} b~∗j分别是两个词向量的bias term那它到底是怎么来的，为什么要使用这个公式？为什么要构造两个词向量 w ∗ i T w*{i}^{T} w∗iT 和 w ~ j \tilde{w}_{j} w~j?

有了上述公式之后，我们可以构建Loss function: J = ∑ i , j = 1 V f (X i j) (w i T w ~ ∗ j + b ∗ i + b ~ ∗ j − log ⁡ (X ∗ i j)) 2 J=\sum_{i, j=1}^{V} f\left(X_{i j}\right)\left(w_{i}^{T} \tilde{w}*{j}+b*{i}+\tilde{b}*{j}-\log \left(X*{i j}\right)\right)^{2} J = i,j=1∑V f(Xij)(wiTw~∗j+b∗i+b~∗j−log(X∗ij))2 loss function的基本形式就是最简单的mean square loss，只不过在此基础上加了一个权重函数 f (X i j) f\left(X_{i j}\right) f(Xij)，那么这个函数起了什么作用，为什么要添加这个函数呢？我们知道在一个语料库中，肯定存在很多单词他们在一起出现的次数是很多的（frequent co-occurrences），那么我们希望：

- 这些单词的权重要大于那些很少在一起出现的单词，因此这个函数要是非递减函数（non-decreasing）；
- 但这个权重也不能过大，当到达一定程度之后当不再增加；
- 如果两个单词没有在一起出现，也就是 X i j X_{i j} Xij,那么他们应该不参与到loss function的计算当中去，也就是f(x)要满足f(x)=0

为此，作者提出了以下权重函数：

$$f(x)=\begin{cases}(x/x_{\max})^{\alpha} & \text{if } x<x_{\max}\\ 1 & \text{otherwise}\end{cases}.$$