

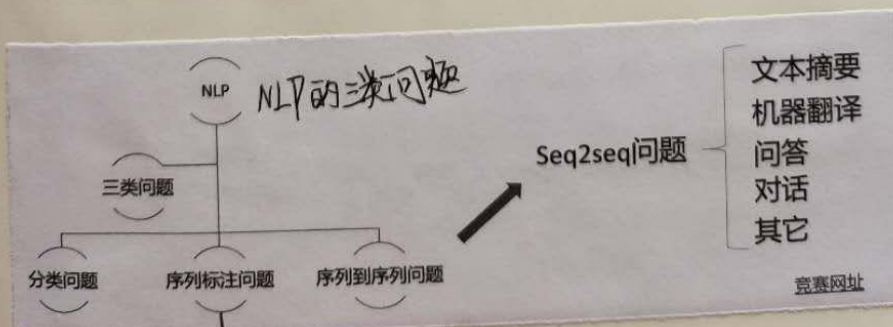
问答摘要与推理-项目简介

Outline

- 项目介绍和课程安排
- 初始词向量Word2vec
- Skip-gram、CBOW模型
- 初识深度学习框架

Outline

- 项目介绍和课程安排
- 初始词向量Word2vec
- Skip-gram、CBOW模型
- 初识深度学习框架



Outline

- 项目介绍和课程安排
- 初始词向量 Word2vec
- Skip-gram、CBOW模型
- 初识深度学习框架

语言模型:

今天欢迎同学们来到北京大学学习

某个位置 i 上的词的输出概率: $P(i) = P(w_1, w_2, \dots, w_n)$

$$= P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1, w_2, \dots, w_{n-1})$$

目标函数: $L = \sum_{w \in C} \log P(w | \text{context}(w))$

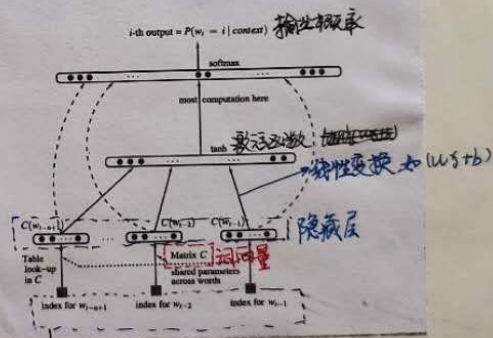
注:

- ① 其中 C 为词表
- ② w 为词表中的词

某位置

③ 为什么用 \log 表示: 避免法为加法, 方便运算; 便于求和, 避免溢出。

Neural Network Language Model 神经网络语言模型 (NNLM)



语言的表达形式

1. discrete symbols (离散模型)

缺点: 不能准确表达语义的相似性。

如不能很好的区别: 我爱北京和我喜欢北京。

2. by their context

我爱北京: 通过我和北京来计算爱 (的向量)

我喜欢北京: 通过我和北京来计算喜欢 (的向量)

Word2vec \rightarrow word embedding (词嵌入)

通过中心词来预测
输出词的频率

欢迎 来到 后厂 打工 学院

Word2vec 模型特点:

- ① 有一个很大的词表库
- ② 在词表中的每一个词都可以通过向量表征
- ③ 有一个中心词 c , 有一个输出词 o
- ④ 用词 c 和 o 的相似度来计算他们之间同时出现的概率
- ⑤ 调整这个词向量来获得最大的输出概率

新词的
词向量

0.286
0.792
-0.177
-0.107
0.109
-0.542
0.349
0.271

Outline

- 项目介绍和课程安排
- 初始词向量 Word2vec
- Skip-gram、CBOW模型
- 初识深度学习框架

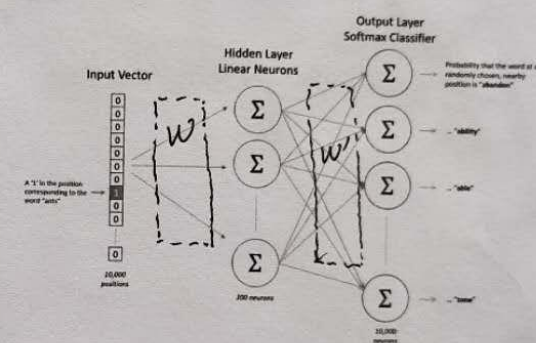
无监督学习

Skip-gram模型

两层神经网络模型
(隐藏层和输出层)
注: 输入层不算层

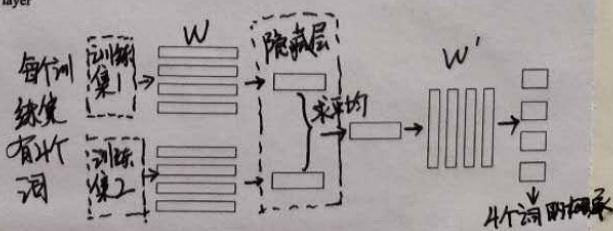
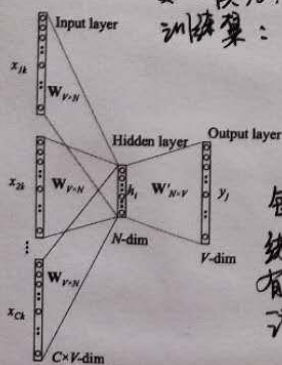


Skip-gram模型



CBOW模型

输入是两边的词, 输出是中间词
如词组为: [知识, 改变, 命运]
训练集: <pad> 改变 → 预测 知识
知识, 命运 → 预测 改变



附：

(1) 《神经网络与深度学习》，期附件中数学基础知识讲解。Github 链接如下：

<https://github.com/nndl/nndl.github.io>

(2) skip-gram 模型博文：<https://zhuanlan.zhihu.com/p/27234078>

(3) 优秀学员 github：<https://github.com/Light2077/>

(4) 推荐一个查阅资料的网站：<https://medium.com/> 和 cs224n

(5) 华为云 AI 平台：modelarts

(6) 百度 AI：黄埔学院

(7) 需要补充学些的知识点：交叉熵、反向求导

之前的笔记

HCT NLP week1 课程介绍及 word2vec

1. NLP 三个典型问题:

利用深度学习解决序列和
序列问题。—— Seq2seq 模型

CBOW 模型

↓
词袋模型

袋

问题:

★ 计算机算法与
数据结构

↓

2. 词向量 word2vec

Leetcode.com

word2vec - objective function
目标函数。

Padde padde

课后补充：

- ① softmax
- ② 反向梯度求导

$$L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m} P(w_{t+j} | w_t, \theta)$$

word2vec - objective function 目标函数

① $L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m} P(w_{t+j} | w_t, \theta)$

θ : 建立神经网络的所有参数, 即可得到所有词向量

t : 位置, 输入词

通过 t 位置输入, 预测 t 位置外一个位置 ($t+j$ 位置) 词的概率

所有位置词, 即 词向量: $[-m, m]$

对每个位置都要进行计算

对于 $[-m, m]$ 之间, 对于给定位置 t 的词, 其他位置词出现的概率

即: 对于给定位置 t 的词, 要计算所有位置上词出现的概率

优化以上目标函数得如下目标函数:

目标函数

② $J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t, \theta)$

T 个样本的均值

词向量问题即变为

优化函数 $J(\theta)$ 问题

最优时对应的 θ 即为

所要求的词向量

此时词向量目标称之为 Fake Task, 即

T 表示取了 T 个训练样本

计算 t 位置在 m 个词, 在 m 个词出现的概率 (不计其位置权重)

此求最优解问题最终并不是想求得最优解 P ，而是想要得到二权重值（权重值代表每个词在空间中意义）
得到 w_1, w_2

总结：

- 词向量问题，并不是为了求得最优解 (P) ，而是为了通过最优解，得到最优解的过程中计算并得到权重矩阵 W_1 和 W_2 。

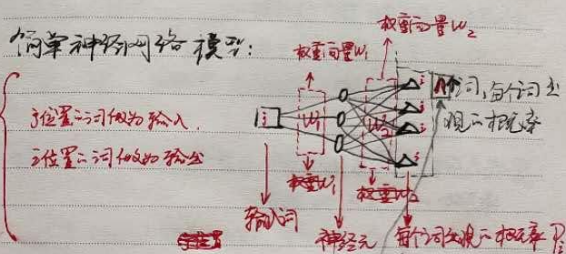
② 权重 W_1, W_2 与词向量的关系

$$W_1 = \begin{bmatrix} [\text{词向量}_1] \\ [\text{词向量}_2] \\ \vdots \\ [\text{词向量}_n] \end{bmatrix}$$

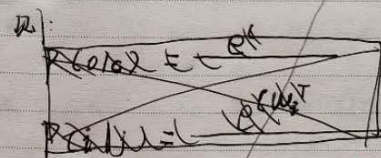
词向量为权重矩阵 $W_1 (W_2)$ 中的一个向量

$$W_2 = \begin{bmatrix} [\text{词向量}_1] \\ [\text{词向量}_2] \\ \vdots \\ [\text{词向量}_n] \end{bmatrix}$$

简单神经网络模型：



$$\text{softmax} = \frac{e^x}{\sum_{i=1}^n e^{x_i}}$$



$$P(i|j) = \frac{e^{(U_i^T V_j)}}{\sum_{\text{词}} e^{(U_i^T V_j)}}$$

j 位置输入词， i 位置词出现的概率

其中 U 即权重向量 W_1 ， V 即权重向量 W_2 中词上的权重

表示对于任意位置 j 作为输入词，所有词都要作为输出计算

3. Skip-gram, CBOW 模型

Skip-gram 模型: (即已知中间词, 预测其周围词)

CBOW 模型即已知周围词, 预测中间词

例: 我 爱 北京 ~~天安门~~ 天安门

对于 Skip-gram 模型为输入北京, 预测输入为爱 ~~天安门~~ 和天安门 的概率

对于 CBOW 模型, 即为输入为爱和天安门, 输出为北京 的概率

Skip-gram 模型:

例: the dog barked at the mailman
以 dog 作为输入

设 Skip-windows = 2, 即设置窗口大小为 2, 即在 dog 的左右各取 2 个词

训练样本
得: [the "dog" barked "at"]

设: num-skips = 2, 即每次输出 2 个词, 如下所示

("dog" the)

("dog" barked)

("dog" at)

注: 输出格式 (input, output)

词向量维度

利用反向梯度求导求得权重 W_1, W_2

W_1 即 embedding 层

Paddle 中利用 look-up 接口获取词向量

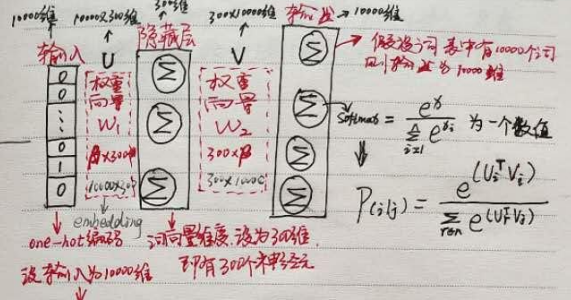
embedding-size 即词向量维度二变量

假设利用 softmax 求权重 $[1, 9, 10] \times [W_2]$

(通过索引到 W_2 中抽出对应词向量)



神经网络只能接受数值输入



$$\beta = [0, 0, \dots, 0, 1, 0]$$

假设词表共有 10000 个词，因此输入，输出为 10000 维，该隐藏层为 300 维，因此权重 W_1 为 10000×300 维，权重矩阵 W_2 为 300×10000 维

② W_1 和 W_2 中任意一个都可以作为词向量的词向量 (注: 词向量矩阵 W_1 或 W_2 中的某一行即为输入中某个词 = 词向量, 如:

$$\text{输入 } \beta \quad [0, 0, 0, 1, 0] \times \begin{matrix} W_1 \\ \begin{bmatrix} 3 & 8 & 9 \\ 6 & 10 & 27 \\ 35 & 48 & 59 \\ 11 & 9 & 10 \\ 8 & 7 & 2 \end{bmatrix} \end{matrix} = \begin{matrix} 5 \times 3 \text{ 行} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 11 & 9 & 10 \\ 0 & 0 & 0 \end{bmatrix} \end{matrix} \begin{matrix} [1, 9, 10] \\ \downarrow \\ \text{新词} \\ U \end{matrix}$$

通过索引到 W_2 中抽出对应词向量

则 W_1 中 $[1, 9, 10]$ 即为输入中对应位置的词向量的词向量

CBOW 模型

CBOW 模型与 skip-gram 模型计算式相似, 只是
CBOW 模型是给定周围的词, 计算中间词
出现的^{概率}。

skip-gram与CBOW区别:

1. 计算复杂度不同:

cbow的计算复杂度与字典大小(V)有关, cbow的计算复杂度为 $O(V)$
(即字典中每个词都要计算一遍)

sg的计算复杂度与窗口大小 k 有关, 复杂度为 $O(kV)$ (sg与cbow复杂度)

2. 周围词的影响:

cbow的词量是与周围词一起调整的 (输出词的概率为周围词

词计算结果的平均值)

sg是由中心词预测输出词, 受周围词的影响较小

cbow优点: 训练集小的时候更准确 (周围词=平均)