真-极度易懂但全面的 WORD2VEC

作者: 骰子AI

2022-4

WORD2VEC

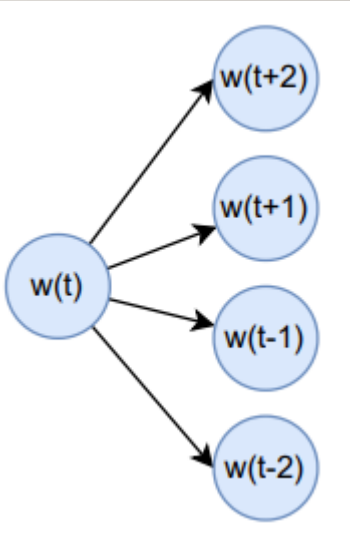
- Word2vec指用特征向量表示单词的技术，且每两个词向量可计算余弦相似度表示它们之间的关系。
- 实现的算法手段：
  - Skip-Gram (跳元模型)
  - CBOW (Continues Bag of Words, 连续词袋)
- 算法优化方法：
  - 负例采样
  - 层序Softmax(Hierarchical Softmax)

SKIP-GRAM

- Skip-Gram的初步理解：用一个词去预测其周围的词。假设窗口大小为5，目前窗口中的词是$[w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}]$，则Skip-Gram的任务即可描述为用$w_t$生成$w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$。
- 训练过程：
  1. 随机初始化词汇数量的中心词向量v与背景词向量u。当某个词作为中心词参与计算时，取中心词向量，用v_c表示；当某个词作为背景词时则取背景词向量，用u_o表示。
  2. 通过中心词c生成背景词o的概率可描述为：

$$P(o|c) = softmax(u_o \cdot v_c) = \frac{e^{u_o v_c}}{\sum_{i \in V} e^{u_i v_c}}$$

  3. 损失函数采取交叉熵损失函数。



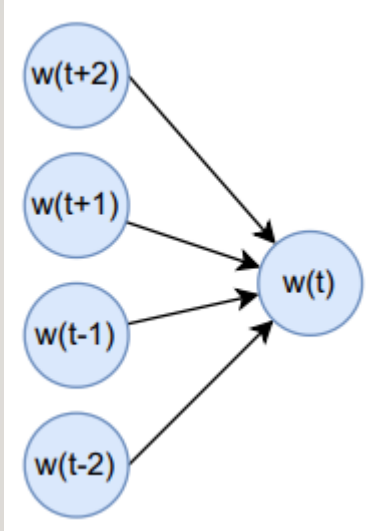
CBOW

- CBOW的初步理解：用周围的词预测中心测。假设窗口大小为5，目前窗口中的词是$[w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}]$，则CBOW的任务即可描述为用$w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$，生成$w_t$。
- 训练过程：
  1. 随机初始化词汇数量的中心词向量v与背景词向量u，当某个词作为中心词参与计算时，则取中心词向量，用v_c表示，当某个词作为背景词时则取背景词向量，用u_o表示。
  2. 设窗口大小为m，则周围包含索引的背景词向量可表示为$\mathrm{U_o} = \{u_{o-m} \dots u_{o-1}, u_{o+1} \dots u_{o+m}\}$，生成中心词c的概率可描述为：

$$P(c|\mathrm{U_o}) = softmax\left(v_c \cdot u_{U_o}\right)$$

$$u_{U_o} = \frac{1}{2m}\sum_{i=-m}^{m} u_i$$

  3. 损失函数采取交叉熵损失函数。



负采样

- Skip-Gram或CBOW都属于Sotfmax多分类预测，且类别数目是整个词典的大小。负采样正是为了优化这一计算开销。负例指的是不与中心词c同时出现在同一窗口的词。
- 训练时可看做是二分类预测，中心词c与背景词o同时出现在同一窗口的概率可描述为：

$$P = (D = 1|c, o) = Sigmoid(u_c u_o) = \frac{1}{1 + e^{-u_c u_o}}$$

- 而中心词c与任意噪音背景词o'不出现在同一窗口的概率可描述为：

$$P = (D = 0|c, o') = 1 - Sigmoid(u_c u_{o'}) = 1 - \frac{1}{1 + e^{-u_c u_{o'}}}$$

- 以Skip-Gram为例，通过中心词c生成背景词o 的概率$P(o|c)$可近似为:

$$P(o|c) = P(D = 1|c, o) \prod_{k=1}^{k} P(D = o|c, o_k')$$

- 所以负采样后Word2vec的任务可转变为以Sigmoid为最终计划的二分类任务。
- 负采样的方式可以完全随机，或者根据某个概率分布，例如根据整个文档词频做概率分布。
- 设负例的数量为k，则时间复杂度可由原先的o(n) 降为 o(k+1)。

层序SOFTMAX

- 首先构建一个二叉树，用叶子节点表示所有的词汇，节点的深度是$\log_2$(节点数量)。
- 以Skip-Gram为例，通过中心词c生成背景词o 的概率$P(o|c)$可近似为，二叉树的根节点走到节点o的概率，用数学语言可描述为：

$$P(o|c) = \prod_{j=1}^{deep-1} \sigma([\![n(o,j+1) = leftChild(n(o,j)]\!] \cdot u_{n(o,j)} v_c)$$

  - σ表示Sigmoid函数。
  - $n(o,j)$表示走到节点o的第j个中间节点。(j由根节点起始)
  - $u_{n(o,j)}$即表示中间节点$n(o,j)$的向量表示， v_c表示的是中心词c的向量。
  - $[\![n(o,j+1) = leftChild(n(o,j)]\!]$表示某种运算，若节点$n(o,j+1)$是节点$n(o,j)$的左边节点,则输出1，否则输出-1。
- 时间复杂度由原先的o(n) 降为了 o(log(n))。

- 假设要以中心词c生成图中的背景词w3为例，则：

$$P(w3|c) = \sigma(u_{n(w3,1)} v_c) \cdot \sigma(-u_{n(w3,2)} v_c) \cdot \sigma(u_{n(w3,3)} v_c)$$