Skip-Gram

Yeats William

July 2024

1 Introduction

这是关于 Skip-Gram 的简要笔记

2 Language Model

语言模型的主要任务是预测接下来将要出现的词语.
其中的两个主要任务如下, 计算全局出现概率和计算第 n 个词的出现概率.

- A language model is a probability distribution over a sequence of words
  - Compute joint probability of a sequence of words:

$$P(W) = P(w_1, w_2, \cdots, w_n)$$

  - Compute conditional probability of an upcoming word w_n:

$$P(w_n \mid w_1, w_2, \cdots, w_{n-1})$$

- Assumption: the probability of an upcoming word is only determined by all its previous words

  1.

$$\begin{aligned} P(\text{Never, too, late, to, learn}) = {} & P(\text{Never}) \times P(\text{too} \mid \text{Never}) \times P(\text{late} \mid \text{Never, too}) \\ & \times P(\text{to} \mid \text{Never, too, late}) \times P(\text{learn} \mid \text{Never, too, late, to}) \end{aligned}$$

  2.

$$P(\text{learn} \mid \text{Never, too, late, to}) = \frac{P(\text{Never, too, late, to, learn})}{P(\text{Never, too, late, to})}$$

- Language Model

$$P(w_1, w_2, \cdots, w_n) = \prod_i P(w_i \mid w_1, w_2, \cdots, w_{i-1})$$

2.1 N-gram model

N-gram 模型是一种用于预测序列中下一个词的概率语言模型. 它基于这样一个假设, 即一个词的概率只取决于它前面的 $n-1$ 个词. 这被称为马尔可夫假设.

2.1.1 定义

N-gram 是从给定文本或语音样本中连续出现的 n 个项的序列. 1-gram 模型称为 unigram 模型, 2-gram 模型称为 bigram 模型, 3-gram 模型称为 trigram 模型, 依此类推.

2.1.2 数学公式

在 N-gram 模型中, 词序列 $W = w_1, w_2, \cdots, w_n$ 的概率表示为:

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$$

例如, 在 bigram 模型中 ($n = 2$), 词序列的概率近似为:

$$P(W) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_2) \times \cdots \times P(w_n \mid w_{n-1})$$

2.1.3 训练 N-gram 模型

为了训练 N-gram 模型, 我们从大规模语料库中计算条件概率. 条件概率 $P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$ 的估计公式为:

$$P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1}) = \frac{\text{Count}(w_{i-(n-1)}, \cdots, w_{i-1}, w_i)}{\text{Count}(w_{i-(n-1)}, \cdots, w_{i-1})}$$

其中 $\text{Count}(w_{i-(n-1)}, \cdots, w_{i-1}, w_i)$ 表示序列 $w_{i-(n-1)}, \cdots, w_{i-1}, w_i$ 在训练语料库中出现的次数, 而 $\text{Count}(w_{i-(n-1)}, \cdots, w_{i-1})$ 表示序列 $w_{i-(n-1)}, \cdots, w_{i-1}$ 出现的次数.

2.1.4 优点和缺点

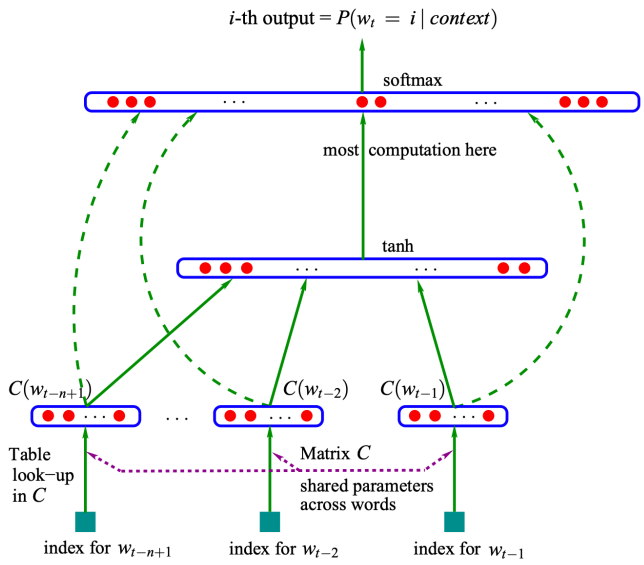
优点:

- 简单性: N-gram 模型易于理解和实现.
- 效率高: 对于小的 n 值, 它们计算效率很高.

缺点:

- 数据稀疏性: N-gram 模型在 n 值较大时会遇到数据稀疏性的问题.
- 上下文有限: 它们只考虑固定窗口的上下文, 可能无法捕捉文本中的长距离依赖关系.

A Neural Probabilistic Language Model



这个架构图展示了一个神经网络语言模型的架构，用于计算在给定上下文的情况下，下一个词出现的概率。以下是图中各部分的详细解释：

词嵌入（Word Embeddings）

图中的 $C(w_{t-n+1})$、$C(w_{t-2})$、$C(w_{t-1})$ 表示通过查表（table look-up）将每个词转换成向量表示。词嵌入表（Matrix C）是共享的，即所有词共享同一个嵌入矩阵。

隐藏层（Hidden Layer）

词嵌入后的向量输入到一个隐藏层。隐藏层的主要计算通过 *tanh* 激活函数完成。

输出层（Output Layer）

隐藏层的输出通过一个 *softmax* 层，计算当前上下文下每个可能的词的概率分布。*softmax* 层的输出即是模型预测的概率。

计算流程（Computation Flow）

- 词的索引输入查表，得到词向量（红色圆点表示向量维度）。
- 词向量输入到隐藏层，经过 *tanh* 激活函数进行非线性变换。
- 隐藏层的输出通过 *softmax* 层，得到每个词的概率分布。

公式表示（Mathematical Representation）

图上方的公式表示模型的目标，即计算在给定上下文 *context* 下，词 w_t 为某个词 i 的概率 $P(w_t = i|context)$。

例子

我们可以通过一个具体的例子来解释这个语言模型的工作流程。

假设我们有一个句子片段：”The quick brown fox jumps over the lazy”。我们的目标是预测下一个词。为了简化，假设我们的上下文长度为 3，即我们用前三个词来预测下一个词。

词嵌入（Word Embeddings）

假设我们选择的上下文词是”brown fox jumps”。首先，我们将这些词通过嵌入矩阵 C 转换为向量：

- $C(brown) \rightarrow$ 向量表示 V_{brown}
- $C(fox) \rightarrow$ 向量表示 V_{fox}
- $C(jumps) \rightarrow$ 向量表示 V_{jumps}

隐藏层（Hidden Layer）

这些词向量输入到隐藏层，假设隐藏层是一个简单的全连接层加上 *tanh* 激活函数。计算过程如下：

$$H = \tanh(W_1 \cdot V_{brown} + W_2 \cdot V_{fox} + W_3 \cdot V_{jumps} + b) \tag{1}$$

其中 W_1, W_2, W_3 是权重矩阵，b 是偏置向量。

输出层（Output Layer）

隐藏层的输出 H 输入到 *softmax* 层，计算每个可能的下一个词的概率分布。假设词汇表包含”dog”, ”cat”, ”horse”, ”lazy”, ”quick” 等词。

$$\textit{softmax}(H) = P(w_t = i|context) \tag{2}$$

其中 i 是词汇表中的每个词。

最终输出（Final Output）

模型输出一个概率分布，例如：

- $P(lazy) = 0.6$
- $P(dog) = 0.1$
- $P(cat) = 0.1$

- $P(horse) = 0.1$
- $P(quick) = 0.1$

因此，模型预测下一个词最可能是”lazy”。

例子的总结

通过这个例子，我们可以看到模型如何通过嵌入层将词转换为向量，再通过隐藏层进行非线性变换，最后通过 *softmax* 层计算出每个词的概率分布。模型的输出即是给定上下文下，每个词作为下一个词的概率。

Word2Vec

Word2Vec 是一种用于自然语言处理（NLP）的词嵌入技术，由 Google 的 Tomas Mikolov 及其团队在 2013 年提出。它通过神经网络将词汇表中的每个词表示为一个实数向量，使得语义上相似的词在向量空间中彼此接近。Word2Vec 有两个主要模型：CBOW（Continuous Bag of Words）和 Skip-gram。

主要模型

1. **CBOW（Continuous Bag of Words）**：CBOW 模型预测给定上下文中的中心词。换句话说，它通过已知的上下文词来预测目标词。这种方法适合处理小数据集，因为它能够更快地收敛并且效果较好。

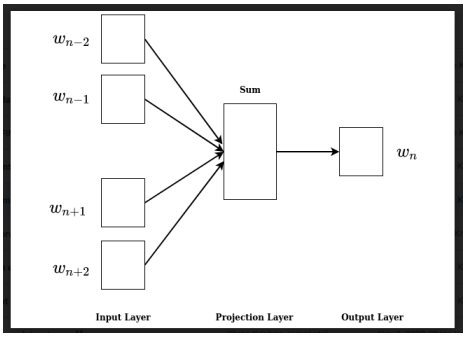


Figure 1: CBOW 模型

2. **Skip-gram**：Skip-gram 模型预测给定目标词的上下文词。它通过已知的目标词来预测周围的上下文词。这种方法在大数据集上效果较好，因为它能够更好地捕捉词与词之间的关系。

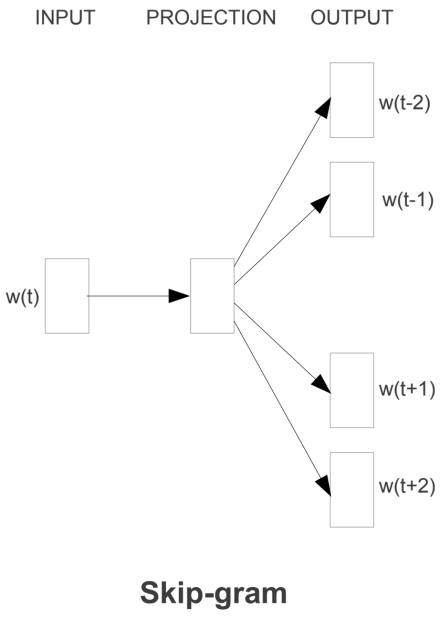


Figure 2: Skip-gram

工作原理

Word2Vec 的工作原理可以概括为以下几个步骤：

1. **词向量初始化**：初始化词汇表中每个词的向量为随机值。
2. **训练神经网络**：使用 CBOW 或 Skip-gram 模型，通过训练数据不断调整词向量，使得模型能够准确预测上下文词或目标词。损失函数通常是负采样（Negative Sampling）或分层 Softmax（Hierarchical Softmax），以提高训练效率。
3. **输出词向量**：训练完成后，输出神经网络的隐藏层权重作为词向量。这些向量在向量空间中保留了词之间的语义关系。

优点

- **高效性**：Word2Vec 使用浅层神经网络进行训练，比传统的基于矩阵分解的方法（如 LSA 和 PCA）更高效。
- **可扩展性**：Word2Vec 可以处理大规模语料库，生成高质量的词嵌入。
- **语义相似性**：生成的词向量能够捕捉词与词之间的语义关系，在很多 NLP 任务中表现出色。

应用

Word2Vec 生成的词嵌入在许多 NLP 任务中都有广泛的应用，例如：

- **文本分类**：将文本表示为词向量的组合，用于训练分类模型。
- **信息检索**：通过计算查询词与文档词向量的相似性，提高检索效果。
- **机器翻译**：作为词表示的输入，提高翻译质量。
- **情感分析**：分析文本中的情感倾向。

2.2 Coding

这里举的是我使用 skip-gram 对 text8 进行操作的一个代码示例, 代码中具有详细注释, 地址如下: `https://github.com/yeatscircle/SkipGram`