NLP 与深度学习 - 理论篇

从 n-gram 到 Attention

Junchen Feng

Generation AI

January 4, 2025

课程简介

课程目标

- 了解 NLP 从传统 n-gram 到深度学习（word2vec、Attention 等）的核心思路

课程简介

课程目标

- 了解 NLP 从传统 n-gram 到深度学习（word2vec、Attention 等）的核心思路
- 掌握 Skip-gram 的直观含义、词向量"`king - man + woman = queen`" 著名例子

课程简介

课程目标

- 了解 NLP 从传统 n-gram 到深度学习（word2vec、Attention 等）的核心思路
- 掌握 Skip-gram 的直观含义、词向量"`king - man + woman = queen`" 著名例子
- 理解 Seq2Seq、LSTM 与 Attention/Transformer 的基本原理

示例句子

示例句子:

"I met my friend Sarah yesterday. She told me about her new research project. It might revolutionize the AI field, according to the professor who worked with her last year."

- **代词指代**: "She" 和"her" 指的是 Sarah; "It" 指"her new research project"

示例句子

示例句子:

"I met my friend Sarah yesterday. She told me about her new research project. It might revolutionize the AI field, according to the professor who worked with her last year."

- **代词指代**: "She" 和"her" 指的是 Sarah; "It" 指"her new research project"
- **省略/隐含信息**: "the professor" 暗示需要背景，可能在更前面或上下文中

示例句子

示例句子:

"I met my friend Sarah yesterday. She told me about her new research project. It might revolutionize the AI field, according to the professor who worked with her last year."

- **代词指代**: "She" 和"her" 指的是 Sarah; "It" 指"her new research project"
- **省略/隐含信息**: "the professor" 暗示需要背景，可能在更前面或上下文中
- **长距离依赖**: "her" 多次出现，需要模型能记住之前提到的实体"Sarah"

n-gram 模型

1. **核心思路:**
   - 通过统计固定大小的词/字符序列出现的频率来表达语言规律

2. **概率公式（以 n-gram 为例）:**

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$$

3. **局限:**

n-gram 模型

1. 核心思路:

- 通过统计固定大小的词/字符序列出现的频率来表达语言规律
- 例如，对句子"I met my friend Sarah"，在 2-gram 中出现"I met"，"met my"，"my friend"，"friend Sarah" 等

2. 概率公式（以 n-gram 为例）:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$$

3. 局限:

n-gram 模型

1. 核心思路:

- 通过统计固定大小的词/字符序列出现的频率来表达语言规律
- 例如，对句子"I met my friend Sarah"，在 2-gram 中出现"I met"，"met my"，"my friend"，"friend Sarah" 等

2. 概率公式（以 n-gram 为例）:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$$

3. 局限:

- 当句子变得复杂、长度变长（尤其含代词指代），n-gram 往往无法捕捉远距离词的关系，容易出现数据稀疏

n-gram 模型

1. 核心思路:

- 通过统计固定大小的词/字符序列出现的频率来表达语言规律
- 例如，对句子"I met my friend Sarah"，在 2-gram 中出现"I met"，"met my"，"my friend"，"friend Sarah" 等

2. 概率公式（以 n-gram 为例）:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1}, \ldots, w_{i-n+1})$$

3. 局限:

- 当句子变得复杂、长度变长（尤其含代词指代），n-gram 往往无法捕捉远距离词的关系，容易出现数据稀疏
- 这就是为什么前面那个多重指代的句子里，n-gram 很难" 知道" She 和 Sarah 是同一个人

word2vec 与词向量概念

1. 为什么要用词向量？

- 传统方法中，词是离散 ID，没有”距离”概念；无法度量语义相似度

2. 著名例子:

```
"king - man + woman = queen"
```

什么是连续向量空间？

- 点击查看 TensorFlow Embedding Projector 可视化演示

word2vec 与词向量概念

1. 为什么要用词向量？

- 传统方法中，词是离散 ID，没有”距离”概念；无法度量语义相似度
- **词向量将词映射到连续向量空间，语义相似的词更靠近**

2. 著名例子:

```
"king - man + woman = queen"
```

什么是连续向量空间？

- 点击查看 TensorFlow Embedding Projector 可视化演示

word2vec 与词向量概念

1. 为什么要用词向量？

- 传统方法中，词是离散 ID，没有”距离”概念；无法度量语义相似度
- **词向量将词映射到连续向量空间，语义相似的词更靠近**

2. 著名例子:

```
"king - man + woman = queen"
```

- word2vec 能捕捉到单词间的语义关系：性别、身份等

什么是连续向量空间？

- 点击查看 TensorFlow Embedding Projector 可视化演示

Skip-gram 目标函数

1. **Skip-gram 核心想法:**
   - 给定一个目标词 (w)，去预测它上下文中的词
   - 用这个过程来学习" 若两个词经常一起出现，则它们的向量应相似"
2. **数学形式 (示意):**

$$\max \sum_{(w,c)} \log P(c \mid w)$$

其中

$$P(c \mid w) = \frac{\exp(\mathbf{v}_c \cdot \mathbf{v}_w)}{\sum_{c'} \exp(\mathbf{v}_{c'} \cdot \mathbf{v}_w)},$$

$$\mathbf{v}_w = w\text{的向量}, \quad \mathbf{v}_c = c\text{的向量}$$

3. **直观解释:**
   - 当"Sarah" 与"She" 在文本中常常成对出现，Skip-gram 会调整它们的向量，让 $\mathbf{v}_{\text{Sarah}}$ 和 $\mathbf{v}_{\text{She}}$ 内积更大，提高 $P(\text{She} \mid \text{Sarah})$
   - 这样，" 相互关联高" 的词向量距离会更近

什么是 RNN（循环神经网络）？

1. **基本概念**
   - **动机**：普通的前馈网络无法很好处理序列数据（如文本、语音），因为输入是**有序**且长度可变的。
   - **循环结构**：RNN 在每个时间步（t）接收当前输入 x_t 和上一个时间步的隐状态 h_{t-1}，更新当前隐状态 h_t。

$$h_t = f\big(W \cdot [\,x_t,\ h_{t-1}\,] + b\big)$$

   其中 f 通常是非线性激活函数（如 $\tanh$、ReLU 等），$[\cdot]$ 表示向量拼接。
2. **优势**
   - 能够在序列的**时间维度**上”记住”或”传递”信息。
   - 对语言序列、语音信号等有天然的适配性。
3. **局限**
   - 当序列过长时，RNN 中会出现**梯度消失或梯度爆炸**问题，导致难以记住”远处”的信息。

RNN 运行流程示意-1

1. 以句子序列为例

- 对句子"Sarah wants to show her project"：
  1. x_1 = "Sarah" 的词向量 $\rightarrow h_1$
  2. x_2 = "wants" 的词向量 $\rightarrow h_2$
  3. .....
  4. x_5 = "project" 的词向量 $\rightarrow h_5$
- **每一步** t:

$$h_t = \mathrm{RNNCell}(x_t,\, h_{t-1})$$

2. 隐状态传递

- h_t 不仅和当前输入 x_t 有关，也跟历史隐状态 h_{t-1} 强相关。
- 这让网络对序列上下文有了" 记忆"，比如知道"Sarah" 在第 1 个词时出现。

RNN 运行流程示意-2

3. 输出层

- 如果是语言模型，用 h_t 去预测下一个词；
- 如果是分类任务（如情感分析），也可能只用最后一个 h_T。

简单示意图：

$$x_1 \rightarrow [\text{RNNCell}] \rightarrow h_1 \rightarrow \ \ldots \rightarrow h_{T-1} \rightarrow [\text{RNNCell}] \rightarrow h_T$$

LSTM

1. **为什么需要 LSTM？**
   - RNN 在长序列中梯度容易消失，难以记住开头关键信息（如"Sarah"）
   - LSTM 通过"门"机制，能在更长距离保留上下文
2. **核心公式（简示）：**

$$f_t = \sigma(\dots), \quad i_t = \sigma(\dots), \quad C_t = f_t C_{t-1} + i_t \tilde{C}_t, \dots$$

   - 避免过多细节，聚焦概念掌握

在我们的例子中...

- 在"Sarah ⋯⋯She"相距较远的情况下，LSTM 可以将最初"Sarah"的信息保存在单元状态 C_t 中较长时间

Attention 机制 - 核心概念

1. **为什么需要 Attention？**
   - LSTM 在非常长距离依赖上仍会有困难
   - **Attention**: 让模型在生成某个词时，显式地" 查看" 输入序列中最相关的部分

2. **Q、K、V 具体解释:**
   - **Query (Q)**: 解码器（或自注意力）当前正在生成的查询向量
   - **Key (K)、Value (V)**: 编码器输出（或序列中每个 token）的键和值向量
   - Attention 分数 $= \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$
   - 通过分数加权" 值" 向量，聚焦最相关内容

3. **优势:**
   - 并行计算，且可针对不同位置进行加权
   - 捕捉远程依赖更灵活

Attention 演示 (QKV 计算)

示例:

句子: "Sarah wants to show her project to the professor"

- **Query**: Decoder 正在生成"her", 想找到先行词; Q 可以是"her" 的隐藏向量
- **Key/Value**: 对应编码器对各词的向量表示: ["Sarah", "wants", "to", "show", "her", "project", "to", "the", "professor"]

演示计算 (简化成 1 维):

1. 假设 Q, K, V 均为 1 维向量:

$$Q(\text{"her"}) = 2.0, \quad K(\text{"Sarah"}) = 1.5, \; K(\text{"her"}) = 2.0, \ldots$$

2. 注意力得分 $= Q \times K$ (再除以 $\sqrt{d_k}$, 此处 $d_k = 1$):

$$\text{Score("Sarah")} = 2.0 \times 1.5 = 3.0, \quad \text{Score("her")} = 2.0 \times 2.0 = 4.0$$

3. `softmax` 归一化得到注意力权重
4. 加权合并 V 向量: $\sum_j \alpha_j \times V_j$

BERT：双向 Transformer 预训练模型-1

1. BERT 是什么？

- **全名**：Bidirectional Encoder Representations from Transformers
- **提出者**：Devlin et al., 2018/2019
- **核心思路**：基于 Transformer 的 Encoder 部分进行双向预训练；在海量文本上学习语言理解能力，随后可微调（fine-tuning）到各种下游 NLP 任务。

2. 训练目标

- **Masked Language Modeling (MLM)**：随机 `mask` 掉部分单词（如 15%），让模型预测被 `mask` 的词；模型需学会双向关注上下文。
- **Next Sentence Prediction (NSP)**：判断两句话是否前后相邻，从而学习上下句关系。
  - 许多后续工作（如 RoBERTa）取消了 NSP，但 MLM 仍是核心。

BERT：双向 Transformer 预训练模型-2

3. BERT 与长距离依赖

- BERT 使用多层 Transformer Encoder，自注意力机制允许在任意位置直接关注序列中的其他词。
- 因为是双向，能同时从前后文获取信息，对上下文理解更出色。

(可配图/示意)：

- **BERT 结构图**：多层 Encoder 堆叠，每层包含多头自注意力和前馈网络。
- **MLM 例子**："The [MASK] is good." → 模型预测被遮住的单词"food"。

BERT 架构示意图

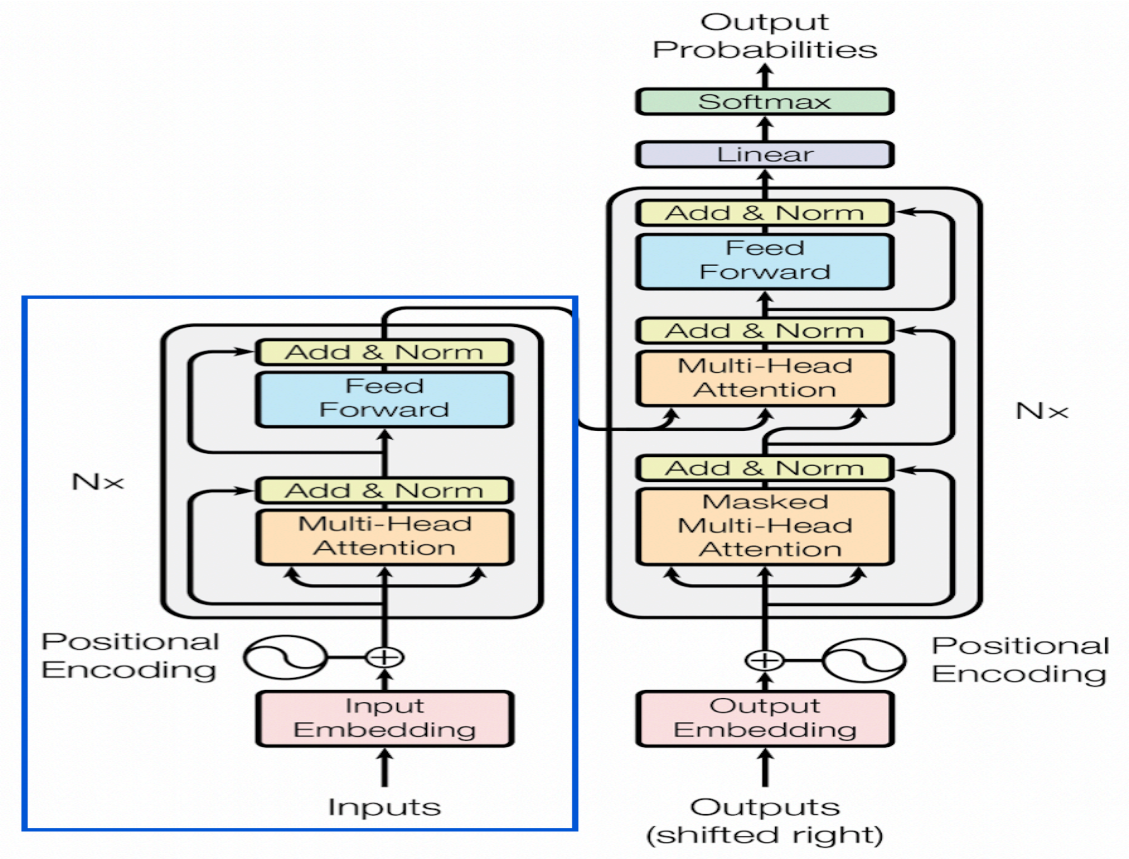

Figure: BERT 模型架构图

总结

1. **回顾**
   - **n-gram**: 简单统计，长距离依赖困难
   - **word2vec**: 词向量让"she" 和"Sarah" 在语义空间更接近
   - **LSTM**: 门机制保留较长依赖信息，但仍有一定局限
   - **Attention**: 显式对句子各位置" 加权关注"，并行化好
   - **BERT**: 双向 Transformer 预训练模型，能捕捉长距离依赖，开箱即用
2. **预告**
   - 将用 IMDB 情感分析数据进行实践
   - 比较传统 NLP，简单 RNN 和 BERT 的表现

参考文献 & 感谢聆听

1. **关键论文:**
   - **n-gram/语言模型**: Shannon, C. E. (1948). *A mathematical theory of communication.*
   - **word2vec**: Mikolov, T. et al. (2013). *Distributed Representations of Words and Phrases and their Compositionality.* NIPS.
   - **Attention**: Bahdanau, D. et al. (2014). *Neural Machine Translation by Jointly Learning to Align and Translate.* ICLR.
   - **Transformer**: Vaswani, A. et al. (2017). *Attention Is All You Need.* NIPS.