STS-Benchmark: 句子向量系统性对比

作　　者: 龚舒凯, 徐瀚臣, 郑楚睿, 杨紫涵
学　　校: 中国人民大学
指导教师: 吴奔, 吕晓玲

内容目录

1 引言

随着自然语言处理技术的快速发展，文本语义理解已成为人工智能领域的核心研究问题之一。在众多文本理解任务中，句子级别的语义表示学习扮演着至关重要的角色，它直接影响着机器对文本语义信息的捕获和理解能力。有效的句子嵌入表示不仅需要准确编码句子的语义信息，还需要在各种下游任务中表现出良好的泛化性能。

语义文本相似度（Semantic Textual Similarity, STS）任务作为评估句子嵌入质量的重要基准，旨在度量两个句子在语义层面的相似程度。该任务要求模型能够理解句子的深层语义含义，而非仅仅依赖于表面的词汇重叠或句法结构。在实际应用中，准确的语义相似度计算对于信息检索、问答系统、文本摘要、机器翻译等多个领域都具有重要价值。

当前，句子嵌入方法主要面临以下挑战：（1）如何有效捕获句子的深层语义信息而非仅仅依赖词汇层面的相似性；（2）如何在保持计算效率的同时提升模型的表示能力；（3）如何设计适当的训练目标和网络架构以优化句子级别的语义表示。针对这些挑战，研究者们提出了多种不同的解决方案，从传统的基于词向量聚合的方法到现代的基于深度神经网络的预训练模型。

本研究旨在探讨不同句子嵌入方法在语义文本相似度任务中的表现，通过系统性的实验比较分析各种方法的优势和局限性。具体而言，我们的研究问题是：**不同的句子嵌入方法如何有效捕获句子的语义信息，并在语义文本相似度任务中取得良好表现？**为回答这一问题，我们将对比分析传统的基于词向量的直接学习方法与现代的基于预训练语言模型的方法，并在标准的 STS-Benchmark 数据集上进行全面的实验评估。

2 相关工作

句子向量表示学习是自然语言处理领域的重要研究方向，现有方法可以大致分为两大类：直接学习方法和基于预训练语言模型的方法。直接学习方法主要以 Word2Vec 和 GloVe 等经典词嵌入技术为基础，通过简单的聚合策略（如平均池化或加权平均）将词向量组合成句子级别的表示。这类方法计算简单高效，但往往忽略了词序信息和复杂的语义关系，在捕获句子深层语义方面存在局限性。

随着深度学习技术的发展，基于预训练语言模型的方法逐渐成为主流。这类方法又可分为基于 RNN 的模型和基于 Transformer 的模型两个发展阶段。早期的 RNN 系列模型（包括 LSTM、GRU 等）通过序列建模的方式学习句子表示，能够捕获一定的词序信息和长距离依赖，但受限于串行处理的特性，训练效率较低。随后，基于 Transformer 架构的预训练模型（如 BERT、ALBERT、RoBERTa 等）凭借其并行化能力和强大的表示学习能力，在多项 NLP 任务中取得了突破性进展。

然而，直接使用预训练模型获得的句子表示在语义相似度任务中的表现往往不够理想，这主要是因为预训练目标与下游任务存在差异。为解决这一问题，研究者们提出了基于孪生网络结构的优化方法，如 Sentence-BERT 和 SimCSE 等。这些方法通过引入对比学习和专门的训练目标，使模型能够直接学习句子间的相似性关系，显著提升了句子嵌入在语义文本相似度任务中的表现。孪生网络结构的成功表明，针对特定任务设计合适的网络架构和训练策略对于获得高质量句子表示至关重要。

3 模型及方法论

3.1 基于传统方法的句子向量学习

在深度学习方法兴起之前，基于传统机器学习方法的词向量和句子向量学习技术已经取得了显著的成果。本节将重点介绍两种经典的词嵌入方法：Word2Vec 和 GloVe，并探讨如何将词向量扩展到句子向量的表示。

3.1.1 Word2Vec 与 GloVe

Word2Vec Word2Vec 是由 Mikolov 等人于 2013 年提出的一种高效的词向量学习方法[11]，其核心思想是通过上下文信息来学习词的分布式表示。该方法采用浅层神经网络架构，能够有效捕获词汇间的语义和句法关系。

Word2Vec 的基本原理是在文本语料上应用固定大小的滑动窗口，并基于分布式假设——即语义相似的词往往出现在相似的上下文中——来学习词的向量表示。该方法主要包含两种训练模式：

CBOW（Continuous Bag of Words）模式：给定上下文词汇，预测中心词。该模式通过周围词汇的向量表示来预测目标词，适用于处理小型数据集和高频词汇。

Skip-gram 模式：给定中心词，预测其上下文词汇。以句子"Never too late to learn" 为例，当窗口大小设为 5 时，Skip-gram 模型会依次预测每个上下文词，如 $P(\text{too}|\text{Never})$、$P(\text{late}|\text{Never})$ 等。该模式在处理大型语料库和低频词汇方面表现更佳。

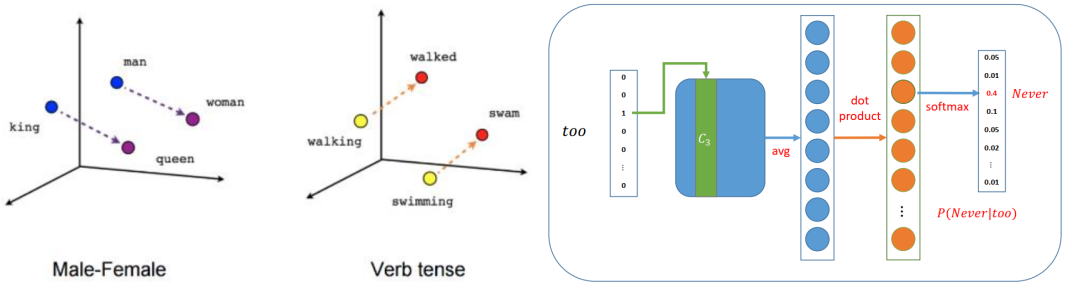


图 1: (a) Word2Vec 语言规律捕捉，(b) Word2Vec 模型结构

Word2Vec 的训练目标是最大化对数似然函数：

$$\mathscr{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \ne 0} P(w_{t+j}|w_t;\theta) \tag{1}$$

等价于最小化平均负对数似然：

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \ne 0} \log P(w_{t+j}|w_t;\theta) \tag{2}$$

然而，标准的 softmax 计算在大规模词汇表上计算成本极高。为解决这一问题，Word2Vec 采用了负采样（Negative Sampling）和层次 softmax（Hierarchical Softmax）等优化技术，显著提高了训练效率。

Word2Vec 学习到的词向量展现出了令人瞩目的语言学规律性。通过向量运算，可以发现诸如性别关系（king - man + woman queen）和动词时态关系（walking - walk + swim swimming）等语义规律，证明了该方法在捕获词汇语义关系方面的有效性。

GloVe　GloVe（Global Vectors for Word Representation）是 Stanford 大学的 Pennington 等人于 2014 年提出的基于全局统计信息的词嵌入方法[12]。与 Word2Vec 的局部上下文窗口方法不同，GloVe 充分利用了语料库中的全局词-词共现统计信息。

GloVe 的核心思想是学习词向量，使得两个词向量的点积等于它们共现计数的对数值。具体而言，对于词汇 i 和 j，GloVe 试图学习向量 w_i 和 $\tilde{w}_j$，使得：

$$w_i^T \tilde{w}_j + b_i + \tilde{b}_j = \log(X_{ij}) \tag{3}$$

其中 X_{ij} 表示词 i 和词 j 在语料库中的共现次数，b_i 和 $\tilde{b}_j$ 分别是对应的偏置项。

GloVe 的损失函数定义为：

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log(X_{ij}))^2 \tag{4}$$

其中 $f(x)$ 是权重函数，用于减少高频词对训练过程的过度影响。该函数通常设计为对低频词赋予较小权重，对中等频率词赋予较大权重，而对极高频词则限制其权重，以平衡不同频率词汇在训练中的贡献。

GloVe 方法的优势在于它既保留了矩阵分解方法利用全局统计信息的优点，又具备了 Word2Vec 等基于上下文的方法在词汇类比任务上的良好表现。实验表明，GloVe 在多个词汇语义评估任务上都取得了优异的性能。

3.1.2　从词向量到句子向量

获得高质量的词向量后，如何将其扩展到句子级别的表示是一个重要问题。传统方法主要采用以下几种策略：

简单平均方法：将句子中所有词向量进行算术平均，得到句子的向量表示。这种方法简单高效，但忽略了词汇的重要性差异和语序信息。

加权平均方法：根据词汇的重要性对词向量进行加权平均。常用的权重计算方式包括：

- TF-IDF 权重：结合词频和逆文档频率来衡量词汇重要性
- 基于频率的权重：根据词汇在语料库中的出现频率分配权重

这些传统方法虽然在计算效率方面具有优势，但在捕获复杂语义关系和长距离依赖方面存在局限性，为后续基于深度学习的句子表示方法的发展奠定了基础。

3.2 基于 RNN 的句子向量学习

3.2.1 RNN, LSTM, GRU

RNN 循环神经网络（Recurrent Neural Network，简称 RNN）[7]是一种专门用于处理序列数据的神经网络架构。其核心特点是循环连接的隐藏层，这种结构使得网络具备了记忆能力，能够捕捉序列中的时序依赖关系。在处理文本、语音、时间序列等数据时，RNN 能够利用序列的上下文信息进行建模，从而为各种自然语言处理和时间序列分析任务提供了强大的支持。然而，RNN 模型也存在一些局限性。在处理长序列数据时，RNN 容易出现梯度消失或梯度爆炸的问题。梯度消失是指在反向传播过程中，梯度值逐渐变小，导致网络难以学习到长距离依赖关系；而梯度爆炸则是指梯度值逐渐变大，导致网络训练不稳定。这些问题使得 RNN 在处理长序列时，句子靠后的单词往往比靠前的单词占据更重要的地位，从而影响整个句子的语义准确度。

LSTM 长短期记忆网络（Long Short-Term Memory，简称 LSTM）[8]是 RNN 的一种改进版本，旨在解决 RNN 在处理长序列时的梯度消失和梯度爆炸问题。LSTM 通过引入门控机制，增加了携带长期记忆的数据流。一个 LSTM 结构单元包含遗忘门（forget gate）、输入门（input gate）和输出门（output gate）三个门限。

- 遗忘门：决定哪些信息需要被丢弃，从而避免无用信息的累积。
- 输入门：决定哪些新信息需要被存储到记忆单元中。
- 输出门：决定哪些信息需要被输出。

这种门控机制使得 LSTM 能够有效保留重要信息，同时丢弃无用信息，从而避免了 RNN 模型中的长期依赖问题。因此，LSTM 在处理长序列数据时的表现通常优于传统的 RNN 模型。然而，LSTM 的结构相对复杂，计算成本较高。

GRU 门控循环单元（Gated Recurrent Unit，简称 GRU）[4]是 LSTM 的一种变体，旨在进一步简化 LSTM 的结构，提高计算效率。GRU 将 LSTM 中的遗忘门和输入门合成了一个单一的更新门（update gate），同时引入了一个重置门（reset gate）。一个 GRU 结构单元包含更新门和重置门。

- 更新门：决定新信息如何更新到当前状态。
- 重置门：决定是否需要忽略之前的隐藏状态。

GRU 模型相比 LSTM 结构更简洁，计算效率更高。然而，由于 GRU 缺少独立的记忆单元，其长时间记忆能力略低于 LSTM。尽管如此，GRU 在许多实际应用中仍然表现出色，尤其是在处理短序列数据时，其性能与 LSTM 相当，但计算成本更低。

模型结构 本研究设计了如图3所示的自训练模型。具体而言，模型可分为句嵌入和计算相似度两个部分。

- 句嵌入：将句子序列通过 RNN 类模型，并将最后一个词的隐藏状态作为句向量输入。对于句子 S_j，计算过程如式5所示。其中，f_{RNN} 表示通过 RNN/LSTM/GRU 模型的计算过程，$h_{n_j}^{(j)}$ 是最后一个词的隐藏状态，作为该句子的向量表示。

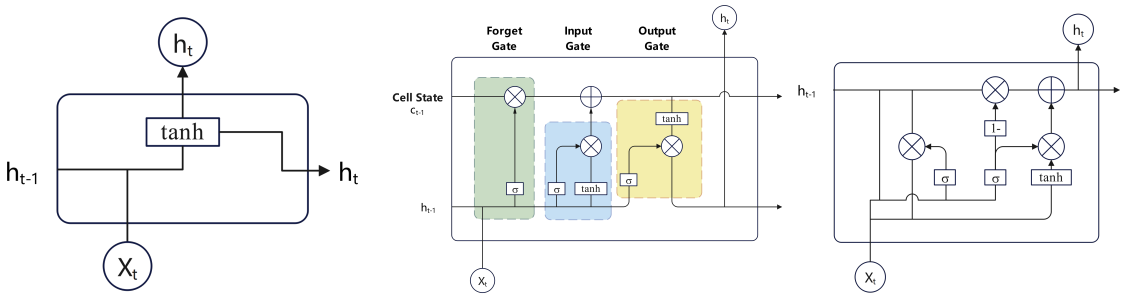


图 2: (a) RNN 结构单元，(b) LSTM 结构单元，(c) GRU 结构单元

- 计算相似度：对于句子 S_1, S_2，将其通过相同的句嵌入模型后得到句向量 $\mathbf{z}_1, \mathbf{z}_2$，将句向量间夹角余弦相似度作为句子相似度输出。训练过程中，将余弦相似度与真实相似度的 MSE 作为损失函数。

$$\mathbf{z}_j = f_{\text{RNN}}(S_j) = h_{n_j}^{(j)} \tag{5}$$

$$\text{cos-sim}(S_1, S_2) = \frac{\mathbf{z}_1^\top \mathbf{z}_2}{\|\mathbf{z}_1\| \|\mathbf{z}_2\|} \tag{6}$$

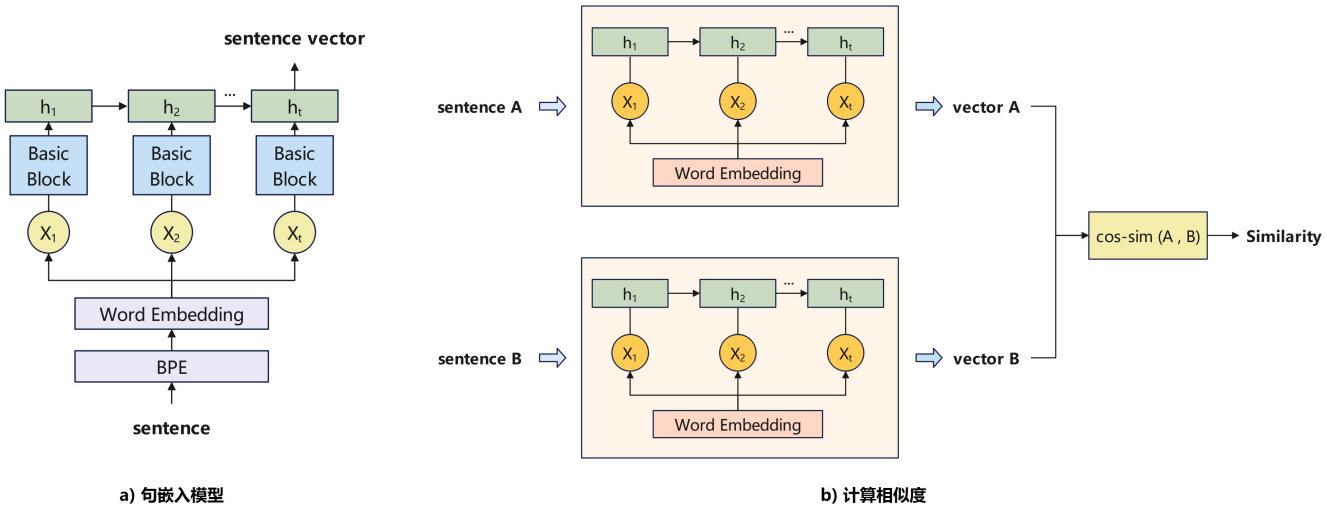


图 3: 自训练 RNN 类模型结构。图 (a) 为句嵌入模型结构示意图，图 (b) 表示计算句子间相似程度的方法

3.2.2 预训练的 RNN 类模型

预训练模型在大规模数据上进行训练，学习到通用的语言表示，能够捕捉语言的结构和语义信息，随后可以通过微调适应特定的任务。本文选取了 2 个基于 RNN 架构的预训练模型 LSTM 和 InferSent，并在 STS-benchmark 数据集上进行微调，以评估它们在句子语义表示方面的性能。

LSTM 预训练模型来自 hugingface 项目"hli/lstm-qqp-sentence-transformer"[2]，该模型的句嵌入部分由一个 LSTM 层和一个池化层构成。LSTM 层负责捕捉序列数据中的时间依赖关系，而池化层用于将 LSTM 输出的序列信息压缩为固定长度的向量，以代表整个句子的语义。

InferSent 是一种用于自然语言处理任务的句子嵌入模型，使用监督学习和自然语言推理 (NLI) 数据集进行训练[5]。其训练方式和模型架构如图4所示。图 (a) 展示了 InferSent 如何通过学习句子对之间的语义关系来生成高质量的句子嵌入；图 (b) 展示了模型句嵌入层的结构，该模型

采用双向 LSTM 网络来捕捉句子中的复杂依赖关系，双向结构允许模型同时从正向和反向处理句子，从而更全面地理解句子的语义。

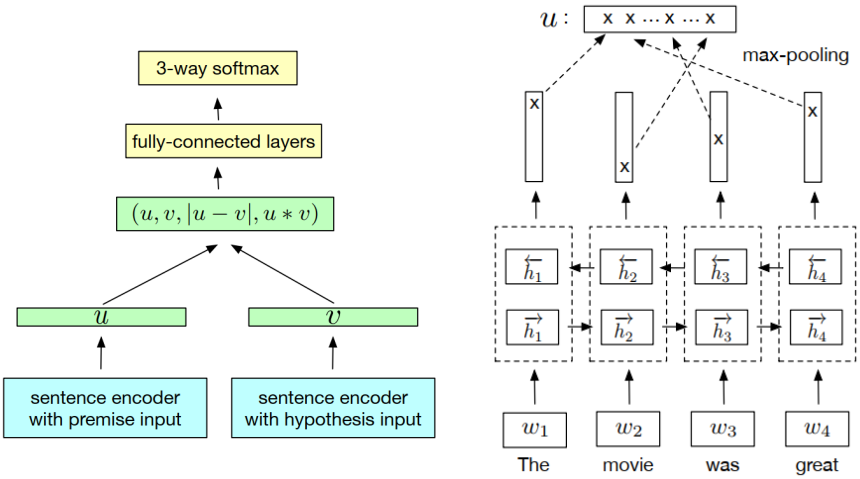


图 4: (a) InferSent 训练方式，(b) 句嵌入模型结构

本文对上述预训练模型的句嵌入层进行全量微调。具体而言，在句嵌入层后添加全连接层以确保句嵌入向量维度的统一，并构建和图3类似的孪生网络结构，该结构由两个共享权重的子网络组成，每个子网络负责处理一个输入句子并生成其句嵌入向量，随后计算余弦相似度。将余弦相似度与真实相似度的 MSE 作为损失函数进行训练。

3.3 基于 BERT 的句子向量学习

3.3.1 BERT

BERT（Bidirectional Encoder Representations from Transformers）[6]是一种预训练语言模型，通过大规模无监督数据进行训练，以获得深层次的语义表示能力。BERT 的设计基于 Transformer 编码器结构，其双向编码能力使其在多种自然语言处理任务中表现卓越。BERT 的预训练包括两个主要任务：掩码语言模型（Masked Language Modeling, MLM）和下一句预测（Next Sentence Prediction, NSP）。图 5 展示了 BERT 的整体结构和训练流程。

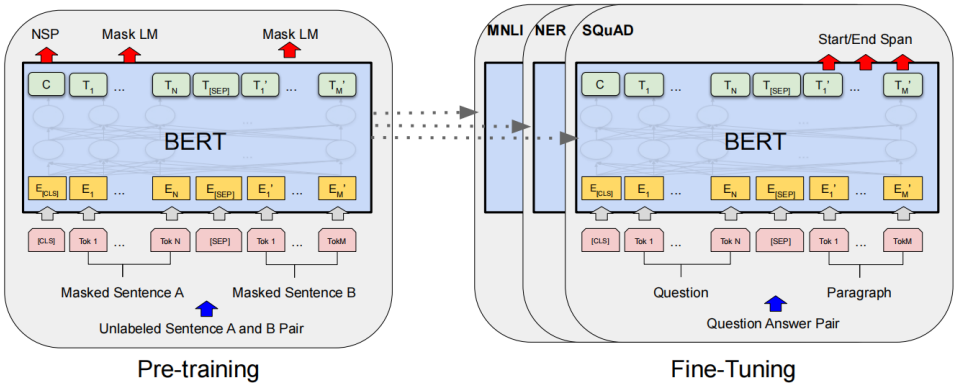


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

图 5: BERT 预训练与微调架构

在 MLM 任务中，输入序列 $X = [x_1, x_2, \ldots, x_n]$ 中的一部分 token 会被随机掩码，记作掩码位置集合 $\mathscr{M}$，目标是预测每个掩码位置上原始的 token。其损失函数定义如下：

$$\mathscr{L}_{\text{MLM}} = -\sum_{i \in \mathscr{M}} \log P(x_i \mid \tilde{X}) \tag{7}$$

其中，$\tilde{X}$ 表示被掩码后的输入序列。

在 NSP 任务中，输入由两个句子片段 (A, B) 组成，目标是判断 B 是否是 A 的真实下一句。该任务的损失函数为：

$$\mathscr{L}_{\text{NSP}} = -[y \log P(\text{IsNext} \mid A, B) + (1-y) \log P(\text{NotNext} \mid A, B)] \tag{8}$$

其中 $y = 1$ 表示 B 是 A 的真实下一句，$y = 0$ 表示不是。

最终总预训练目标为两个子任务之和：

$$\mathscr{L}_{\text{pretrain}} = \mathscr{L}_{\text{MLM}} + \mathscr{L}_{\text{NSP}} \tag{9}$$

在嵌入层结构方面，BERT 的输入嵌入由三部分构成（如图6）：

- **Token Embedding**：每个词对应的词向量；
- **Segment Embedding**：区分句子 A 和 B 的向量；
- **Position Embedding**：编码 token 在序列中的位置信息。

最终输入为这三部分的加和，输入格式为：

$$\text{Input} = [\texttt{[CLS]}, \text{sentence}_1, \texttt{[SEP]}, \text{sentence}_2, \texttt{[SEP]}]$$

BERT 输出每个 token 的表示，句子整体的向量通常取 ‘[CLS]‘ 位置的表示或平均池化形式。

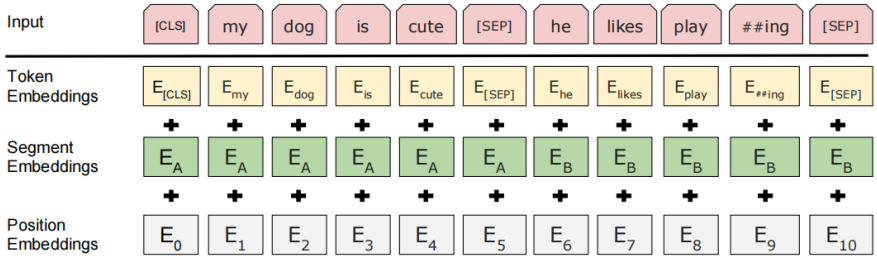


图 6: BERT 嵌入层结构

我们采用一个轻量的 BERT 微调架构对 STS-B 进行建模，训练目标是学习一个句子编码器，使得语义相近的句子对向量间的余弦相似度更高。假设每个 batch 包含 N 对句子样本，模型输入后生成两组句子向量：

$$\mathbf{z}_1 = f_\theta(\text{sentence}_1), \quad \mathbf{z}_2 = f_\theta(\text{sentence}_2)$$

其中 f_θ 为 BERT 编码器，$\mathbf{z}_i \in \mathbb{R}^d$ 为 ‘[CLS]‘ token 对应的输出向量。而句子对之间的语义相似度通过余弦相似度计算：

$$\text{sim}(\mathbf{z}_1, \mathbf{z}_2) = \frac{\mathbf{z}_1 \cdot \mathbf{z}_2}{\|\mathbf{z}_1\| \cdot \|\mathbf{z}_2\|} \tag{10}$$

由于 STS-B 提供的相似度标签为实数 $y \in [0,5]$，我们将其归一化至 $[0,1]$，使用均方误差（MSE）作为损失：

$$\mathscr{L}_{\text{STS}} = \frac{1}{N} \sum_{i=1}^{N} \left(\text{sim}(\mathbf{z}_1^{(i)}, \mathbf{z}_2^{(i)}) - y^{(i)} \right)^2 \tag{11}$$

具体实现中，BERT 模型之后接一个线性投影层以增强表示能力：

$$\mathbf{h}_i = \text{BERT}(\text{input}_i)_{[\text{CLS}]}, \quad \mathbf{z}_i = W\mathbf{h}_i + b \tag{12}$$

其中 $W \in \mathbb{R}^{d \times d}, b \in \mathbb{R}^d$ 为可训练参数。训练细节方面，使用 AdamW 优化器，初始学习率为 2×10^{-5}；将 batch size 设置为 32，最大长度为 128；使用余弦相似度与 MSE 组成的自定义损失，并使用 Pearson 与 Spearman 相关系数作为评估指标；

3.3.2 BERT 衍生模型：ALBERT 与 RoBERTa

ALBERT（A Lite BERT）[9]通过一系列创新改进了 BERT 模型，减少了模型的参数数量，同时保持了强大的性能。主要创新如下：

因式分解嵌入参数化

传统 BERT 在词嵌入矩阵中存在大量的参数。ALBERT 通过将词嵌入矩阵分解为两个较小矩阵：

$$\mathbf{E} = \mathbf{E}_{small} \cdot \mathbf{E}_{factor}$$

从而减少了参数数量，提升了模型的效率。并且 ALBERT 对 BERT 的多层 Transformer 架构

进行了参数共享，使得每一层的参数相同，进一步减少了模型的参数量。

句子顺序预测（SOP）

与 BERT 的 Next Sentence Prediction（NSP）任务不同，ALBERT 引入了句子顺序预测（SOP）任务，要求模型预测句子对的顺序是否正确：

$$\mathscr{L}_{\text{SOP}} = -\log P(\text{Correct Order} \mid A, B)$$

ALBERT 通过同时优化 MLM 和 SOP 任务，有效地提升了模型在句子对嵌入学习中的表现。

训练目标与损失函数方面，ALBERT 的训练范式总体与 BERT 相同，ALBERT 的预训练目标为 Masked Language Modeling（MLM）和 Sentence Order Prediction（SOP）：

$$\mathscr{L}_{\text{MLM}} = -\sum_{i \in \mathscr{M}} \log P(x_i \mid \tilde{X}), \quad \mathscr{L}_{\text{SOP}} = -\log P(\text{Correct Order} \mid A, B)$$

最终的损失函数为：

$$\mathscr{L} = \mathscr{L}_{\text{MLM}} + \mathscr{L}_{\text{SOP}}$$

总体而言，Albert 相较于 BERT 和 RoBERTa 牺牲了部分参数，但是较快地提升了模型训练效率。

RoBERTa[10]则是是对 BERT 的进一步优化，去除了 NSP 任务，采用了动态掩码机制和更长的训练周期。

动态掩码机制

RoBERTa 在每个训练 epoch 中都会重新生成掩码，减少了静态掩码带来的过拟合问题：

$$M^{(epoch)} \sim \text{DynamicMasking}(X)$$

简化的预训练目标

RoBERTa 仅保留了 Masked Language Modeling（MLM）任务，去除了 Next Sentence Prediction（NSP）任务，简化了预训练过程,：

$$\mathscr{L}_{\text{MLM}} = -\sum_{i \in \mathscr{M}} \log P(x_i \mid \tilde{X})$$

并且 RoBERTa 通过延长训练周期（500K 步骤）来提高上下文学习的深度，进一步提升了模型的性能，最终的损失函数为：

$$\mathscr{L} = \mathscr{L}_{\text{MLM}}$$

总体而言，RoBERTa 相较于 BERT 和 ALbert 牺牲了进一步扩大了参数量，模型精度表现优秀，但是训练成本更高。

训练时间复杂度方面，RNN 类模型的前向传播过程是顺序的，每个时间步的计算复杂度为 $O(d^2)$，其中 d 为隐藏层维度。对于长度为 n 的序列，整体计算复杂度为 $O(n \cdot d^2)$。由于其顺序计算的特性，RNN 在处理长序列时容易出现梯度消失或爆炸的问题，导致训练时间较长。LSTM 在 RNN 的基础上引入了三个门控机制（输入门、遗忘门、输出门），每个时间步的计算复杂度为 $O(4d^2)$。整体计算复杂度为 $O(n \cdot d^2)$。由于 LSTM 结构复杂，训练时间较 RNN 更长，但能够更好地捕捉长距离依赖。GRU 合并了 LSTM 中的输入门和遗忘门，简化了结构，每个时间步的计算复杂度为 $O(3d^2)$。整体计算复杂度为 $O(n \cdot d^2)$。GRU 相较于 LSTM，训练时间更短，适用于资源有限的场景。

而 BERT 族中，BERT 采用自注意力机制，每层的计算复杂度为 $O(n^2 \cdot d)$，其中 n 为序列长度，d 为隐藏层维度。对于 L 层的模型，整体计算复杂度为 $O(L \cdot n^2 \cdot d)$。BERT 的训练过程可以并行化，但由于其庞大的参数量和计算量，训练时间较长。ALBERT 通过参数共享和因式分解嵌入矩阵减少了参数量，但计算复杂度与 BERT 相似，为 $O(L \cdot n^2 \cdot d)$。由于参数量减少，ALBERT 在训练时间上优于 BERT。RoBERTa 在 BERT 的基础上移除了下一句预测任务，采用更大的 batch size 和更长的训练时间，计算复杂度与 BERT 相当。RoBERTa 的训练时间较 BERT 更长，但在性能上有所提升。BERT 族模型将时间复杂度从 RNN 线性上升为平方级别。

3.3.3 BERT：MLP+$\sigma(\cdot)$

在传统的 BERT 微调流程中，仅使用单层线性映射（Linear）对 [CLS] 输出进行相似度预测可能导致模型表达能力受限。为了增强非线性建模能力、提升模型拟合复杂关系的能力，本文在原始 BERT 基础上增加了中间全连接层（MLP）与 Sigmoid 激活函数，其动机与理论分析如下：

多层感知机（MLP）通过在 Transformer 输出与回归头之间引入隐藏层，使模型能够学习更高阶的非线性映射。如文献指出，带 ReLU 等非线性激活的 MLP 能够至少构造分段线性函数族，显著提升表达能力，相较于单层线性层具备更强的拟合复杂函数的能力。具体来说，中间引入的线性映射与 ReLU 激活使模型能够捕获语义特征之间的非线性交互效果，从而提升相似度预测的准确性与鲁棒性。

与多分类任务不同，本研究处理的是具有定量意义的回归问题。相似度分值经过归一化处理后处于 $(0,1)$ 区间，引入 Sigmoid 激活有以下优势：

- 强制输出符合标签范围，避免模型预测越界，有助于训练收敛；
- 保证输出端梯度可控，不至于大幅震荡，提高优化稳定性；
- 配合 MSE 损失函数使用时，与标签的对齐更自然，训练效果通常优于无约束输出。

结合上述改进，模型可获得以下潜在提升：

- 表达能力增强：MLP 多隐藏层和非线性激活扩展了模型对复杂语义空间的拟合能力；
- 预测稳定性提升：Sigmoid 输出限制在固定区间，有效约束网络预测结果，减少极端值出现；
- 收敛性更好：非线性层级结构加快模型在训练初期的拟合速度，同时结合 AdamW、MSELoss 稳定优化。

- 对比学习类结构增强：引入 MLP 激活后，能更好提炼 BERT 编码的信息，使预测结构更加合理、性能提升显著。

模型原理方面，令输入文本对为 $\mathbf{x}_1$ 和 $\mathbf{x}_2$，其经过 BERT 编码后的 [CLS] 输出表示为向量 $\mathbf{h} \in \mathbb{R}^d$（本研究 $d = 768$），则加入 MLP 与 Sigmoid 后的预测函数定义如下：

$$\mathbf{z} = \mathbf{W}_1 \mathbf{h} + \mathbf{b}_1, \tag{13}$$

$$\mathbf{u} = \mathrm{ReLU}(\mathbf{z}), \tag{14}$$

$$\hat{y} = \sigma(\mathbf{W}_2 \mathbf{u} + b_2), \tag{15}$$

其中，$\mathbf{W}_1 \in \mathbb{R}^{512 \times d}$、$\mathbf{W}_2 \in \mathbb{R}^{1 \times 512}$ 为可训练参数，$\sigma(\cdot)$ 为 Sigmoid 激活函数。

整体采用基于批次的均方误差（MSE）进行优化，定义如下：

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2, \tag{16}$$

其中 y_i 为真实相似度分数（归一化至 $[0,1]$），$\hat{y}_i$ 为模型预测，N 为批次大小。训练使用 AdamW 优化器，逐步更新 BERT 与 MLP 参数。

3.4 孪生网络

3.4.1 孪生 BERT 网络 (Siamese BERT)

直接使用预训练 BERT 模型进行句子向量语义相似度分析时存在的最大问题是计算开销过大。传统的 BERT 方法通常将词元化的句子 $A \in \mathbb{R}^{s \times d}$ 和句子 $B \in \mathbb{R}^{s \times d}$ 拼接后输入模型，然后通过线性层计算相似度，即

$$\mathbf{z} = \mathrm{BERT}(\mathrm{Concat}(A,B)) \in \mathbb{R}^d, \quad \mathrm{sim}(A,B) = \sigma(\mathbf{W}\mathbf{z}) \in \mathbb{R}.$$

若要对 n 个句子进行两两相似度计算，其时间复杂度高达 $O(n^2(LT^2d + LTd^2))$。其中，L 为 BERT 层数，T 为单个句子（而非拼接后）的 token 长度，d 为隐藏层维度。这种复杂度在处理大规模句子集时效率极低。

为解决上述问题，我们借鉴了 Sentence-BERT[13]的做法，引入了孪生网络（Siamese Network）架构。孪生网络是包含两个或多个共享相同权重和架构的并行子网络。每个子网络独立处理不同的输入，然后将各自的输出用于后续任务，例如比较相似性。这种设计允许模型学习如何将输入映射到某个特征空间，使得相似的输入在该空间中距离较近，不相似的输入距离较远。如 图 7a所示，我们设置预训练的 RoBERTa-base 作为孪生网络的骨架，独立地将每个句子 $A \in \mathbb{R}^{s \times d}$ 和 $B \in \mathbb{R}^{s \times d}$ 分别输入 BERT 模型，并对 BERT 的输出进行平均池化，以生成固定维度的句子嵌入，即

$$\mathbf{z}_1 = \mathrm{MeanPooling}(\mathrm{BERT}(A)) \in \mathbb{R}^d, \ \mathbf{z}_2 = \mathrm{MeanPooling}(\mathrm{BERT}(B)) \in \mathbb{R}^d.$$

句子间的语义相似度则通过这两个嵌入的余弦相似度 $\text{sim}(A,B) = \frac{\mathbf{z}_1^\top \mathbf{z}_2}{\|\mathbf{z}_1\|\|\mathbf{z}_2\|}$ 来衡量。模型训练的目标是最小化余弦相似度与真实标签之间的 L2 范数误差，即

$$\mathscr{L} = \|\text{sim}(A,B) - \text{ground truth}\|_2^2.$$

这种架构显著降低了计算复杂度。孪生 BERT 网络的时间复杂度为 $O(n(LT^2d + LTd^2) + \frac{n(n-1)}{2}d)$，其主导项简化为 $O(n^2d)$，因为**计算瓶颈从 BERT 的前向传播转移到了简单的向量点积**。这相较于直接使用 BERT 时 $O(n^2(LT^2d + LTd^2))$ 的复杂度而言，实现了巨大的效率提升。在评估孪生 BERT 网络的性能时，通常使用 Spearman 相关系数 $r_s = 1 - \frac{6\sum_{i=1}^n d_i^2}{n(n^2-1)}$ 来衡量其与人工标注相似度的相关性。

训练过程中，由于 RoBERTa-base 骨架的参数量较小，我们直接将孪生 BERT 网络在 STS-Benchmark 训练集上全量微调 50 个 epoch，再在测试集上评估其在句子向量相似性任务的性能。

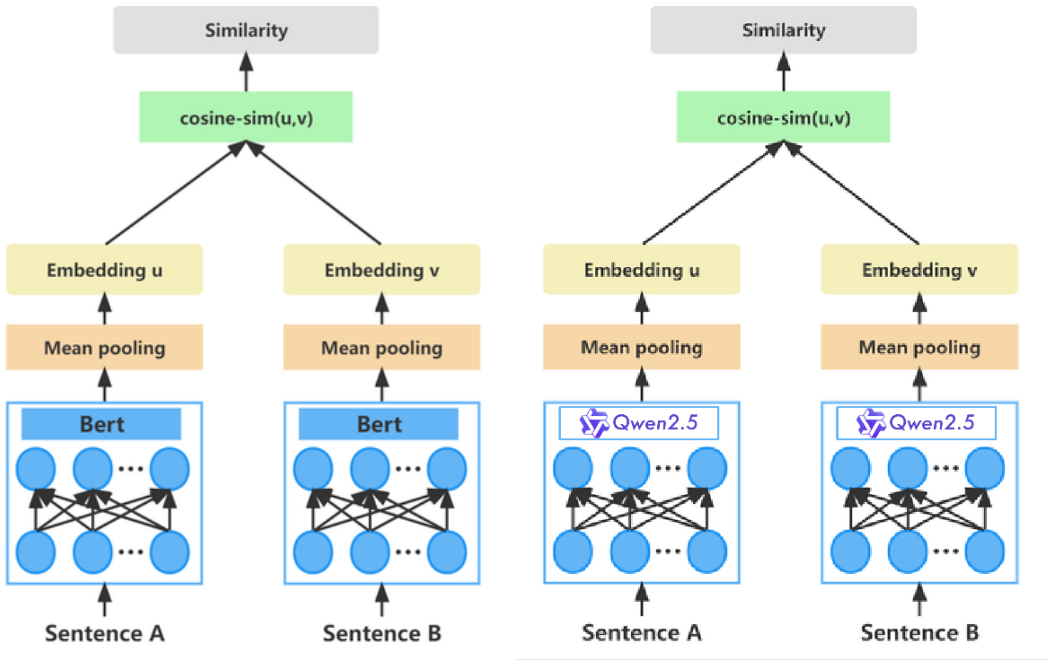
3.4.2 孪生 LLM (Siamese LLM)

受到 Sentence-BERT[13]孪生网络设计的启发，我们构建了孪生 LLM。该模型将 BERT 骨架替换为预训练的 Qwen2.5-0.5B，其具备更强的自然语言处理能力且对硬件负担较小。如图图 7b所示，句子 A 和 B 通过共享权重的 Qwen2.5 模型后，经平均池化转换为固定维度的句子嵌入 $\mathbf{z}_1 = \text{MeanPooling}(\text{Qwen}(A))$ 和 $\mathbf{z}_2 = \text{MeanPooling}(\text{Qwen}(B))$，并通过余弦相似度 $\text{sim}(A,B) = \frac{\mathbf{z}_1^\top \mathbf{z}_2}{\|\mathbf{z}_1\|\|\mathbf{z}_2\|}$ 衡量语义相似度。孪生 LLM 通过最小化预测相似度与真实标签之间的 L2 范数损失进行训练，并使用 Spearman 相关系数进行性能评估。

相比孪生 BERT 网络中 Encoder-only 的 BERT 骨架，孪生 LLM 网络中 Decoder-only 的 LLM 在理论上更具有优越性[1]。首先，Decoder-only 架构中的因果注意力 (Causal Attention) 是下三角矩阵，保证了注意力矩阵满秩性，相比之下，Encoder 的双向注意力可能会退化为低秩状态，这可能会削弱模型的表达能力。其次，Decoder-only 架构在预训练时，每个位置所能接触的信息较少，因此预测下一个 token 的难度更高。当模型足够大，数据足够多时，Decoder-only 模型学习通用表征的上限更高。

在实现细节上，Qwen2.5-0.5b 的隐藏层大小为 896，为了统一起见，我们在 Qwen2.5-0.5b 的最后一层后接了一个含 GeLU 激活函数的线性层，从而将句子向量投影到 768 维。在训练过程中，我们采取了两种训练策略，一种是直接将孪生 LLM 在 STS-Benchmark 训练集上全量微调 50 个 epoch，一种是在 STS-Benchmark 训练集上 LoRA 微调 50 个 epoch。

对于 LoRA 微调，我们设置 LoRA 分解矩阵的秩为 $r = 16$，LoRA 激活值的缩放因子为 $\alpha = 32$，对 Qwen2.5-0.5b 的 Q、K、V 层和其他 MLP 层进行微调。我们尝试了使用 PEFT 库和 Unsloth 框架的 LoRA 微调方法，相比 PEFT 的微调框架，Unsloth 在 CUDA 内核和融合操作上进行了一定优化，提供 Gradient Checkpointing 等技术，在显存的占用和训练速度方面优于 PEFT 库。因此，后续的实验中我们主要采用 Unsloth 框架对模型进行微调。



(a) 孪生 BERT(S-BERT) **(b)** 孪生 LLM(S-LLM)

图 7: (a) 孪生 BERT 网络架构，此处我们采用 RoBERTa 作为骨架。(b) 孪生 LLM 网络架构，此处我们采用 Qwen2.5-0.5B 作为骨架。

4 实验结果分析

各方法在句子向量相似度任务上的表现如表 表 1所示，下面我们进行系统性的原因分析。

表 1: 各方法在句子向量相似度任务上的表现

模型名称	验证集损失	Pearson 相关系数 (ρ)	Spearman 相关系数 (r_s)
Word2Vec-Pretrained	0.143	0.347	0.325
GloVe-Pretrained	0.191	0.237	0.274
Word2Vec-SG	0.162	0.472	0.475
Word2Vec-CBOW	0.150	0.262	0.259
GloVe	0.196	0.240	0.239
RNN	0.157	0.212	0.216
LSTM	0.131	0.348	0.352
LSTM-Pretrained	0.053	0.685	0.688
GRU	0.081	0.531	0.529
InferSent	0.042	0.776	0.785
BERT	0.017	0.902	0.898
ALBERT	0.019	0.886	0.885
RoBERTa	0.016	0.914	<u>0.912</u>
BERT (MLP+$\sigma(\cdot)$)	0.018	0.899	0.896
ALBERT (MLP+$\sigma(\cdot)$)	0.024	0.860	0.858
RoBERTa (MLP+$\sigma(\cdot)$)	0.015	<u>0.916</u>	**0.915**
S-BERT	0.034	0.875	0.868
S-LLM (SFT)	0.087	0.871	0.869
S-LLM (LoRA)	0.092	**0.921**	0.903

传统方法 我们首先对基于词向量聚合的传统方法进行了实验评估，主要分为预训练词嵌入和自训练词嵌入两种策略。

预训练词嵌入方法：该方法直接利用预训练的词嵌入（Word2Vec 和 GloVe）计算句子相似度，不进行任何微调操作。如表 1 所示，Word2Vec 在各项指标上均优于 GloVe，其 Spearman 相关系数为 0.325，而 GloVe 的对应指标仅为 0.274。这一结果的原因在于 GloVe 对词频较为敏感，高频词往往在向量空间中占据主导地位，导致在进行简单平均操作时产生有偏的句子表示。相比之下，Word2Vec 基于局部上下文的训练方式产生了分布更加均匀的词嵌入，使其更适合简单的平均池化操作来构建句子表示。

自训练词嵌入方法：为了更好地捕获领域特定的语义关系，我们直接在 STS 语料库上使用不同算法（Skip-gram、CBOW 和 GloVe）训练词嵌入。如表 2 所示，Skip-gram 方法显著优于其他方法，其 Pearson 和 Spearman 相关系数分别达到 0.475，大幅超越 CBOW 和 GloVe。这一优势主要源于 Skip-gram 在处理低频词方面的能力。在语义相似度任务中，句子间的关键区别往往体现在专业术语或低频词汇上，而 Skip-gram 通过预测上下文词汇的方式能够更好地学习这些低频但语义重要的词汇表示，相比之下 CBOW 方法在处理低频词时表现相对较弱。

总体而言，自训练词嵌入方法的性能显著优于预训练词嵌入方法，这表明针对特定任务进行的领域适应性训练对于提升句子语义表示质量具有重要意义。但与此同时，传统方法只是用词向量的平均作为句子向量，因此无法很好捕捉语义信息，也无法有效识别出个别词的变化对语义的颠覆性影响，因此传统方法较后续方法而言效果较差。

RNN 类方法 就 RNN 类模型内部的对比而言，GRU 的表现优于 LSTM，而 LSTM 优于传统的 RNN。此外，预训练模型的整体表现优于自训练模型。传统 RNN 模型难以捕捉跨词的依赖关系，并且表达能力不足，简单的循环结构难以充分捕捉句子中的语义和句法信息。理论上，LSTM 模型由于其复杂的门控机制，应当比 GRU 表现更好，然而在本文实际应用中，LSTM 的表现却不如 GRU，这可能有以下原因：一是过拟合问题，LSTM 模型的复杂结构使其在小规模训练数据上容易过拟合，导致在测试数据上的性能下降；二是 LSTM 在短序列上相对于 GRU 的优势并不明显，而本研究所用数据集中句子长度整体较短，GRU 的简洁性使其在处理短序列数据时具有更好的泛化能力。预训练模型更好的表现可以归功于其预训练数据集的规模和模型的复杂度，例如 InferSent 所使用的 NLI 数据集包含 57 万条数据，大规模的数据使得预训练模型能够学习到更丰富的语言特征和语义信息，从而在微调后表现出色。

与其他模型对比而言，RNN 类模型的表现略优于基于词向量的表示方法，但明显不如基于 Transformer 架构的 Bert 模型和大语言模型。与后两者相比，RNN 类模型存在以下局限性。首先，RNN 类模型难以捕捉超长距离的依赖关系。虽然 LSTM 和 GRU 通过门控机制缓解了梯度消失或爆炸问题，从而比原始 RNN 能更好地处理长序列，但它们仍然是顺序处理信息，在捕捉超长距离依赖方面表现劣于基于自注意力机制的 Transformer 类模型。其次，RNN 类模型对序列长度较为敏感，通常需要对序列进行截断或填充，这可能导致信息丢失或引入噪声，从而影响模型的性能。

BERT 类方法 如表 1所示，BERT 类模型总体而言拟合表现优秀，明显优于 RNN 族模型与 Word2Vec，并且 Pearson 相关系数等指标比较下，$RoBERTa > BERT > ALBERT$。ALBERT 作为 BERT 的改良版，它的表现效果却没有 BERT 的 basemodel 强，主要是因为 ALBERT 模型拆分了

词向量矩阵来降低参数数量，并且允许多个 Transformer 层之间共享部分参数，进一步降低模型大小以减轻过拟合，到达最优拟合的训练轮次更早，对于算力受限的情况更加受用。而 RoBERTa 对比其他两个模型，由于采用的是动态掩码的方法，所以参数量会更大，相关系数更高的同时更容易出现过拟合。所以这三种模型对比下本质是训练精度与算力消耗的权衡。

而加入了 MLP+$\sigma(\cdot)$ 后，BERTbase 在新的结构中并没有进一步提升句义相似度预测的精度与泛化能力。但是在 ALBERT 与 Roberta 上能够体现区别。添加了 MLP 结构后，AlBERT 在 STS-B 上的泛化能力变差，但相反的是 Roberta 的训练精度和泛化能力得到提升。

从 Trainning Trend 来分析，MLP+sigmoid 的结构甚至加重了 Albert 的过拟合现象，train loss 和 valid loss 的差值变大，而且训练趋势的波动性大幅增强；但是从 RoBERTa 的训练趋势来看，在原先训练中存在较为明显的过拟合现象在加入 MLP+$\sigma(\cdot)$ 的架构后也得到了抑制。

从实验结果向下分析模型架构，ALBERT 使用了跨层参数共享机制来减少参数总量。这使得它在参数上更轻量，但也牺牲了一定的特征表达多样性和丰富性。当输出端添加了一个非线性 MLP 结构时，它会试图学习复杂的映射关系，但 ALBERT 提供的特征维度较单一或受限，MLP 可能会“过度拟合”有限的表达，导致泛化能力下降。可能也意味着轻量模型（如 ALBERT）容易在小数据集或特征维度不足的情况下被 MLP 放大噪声。

而 RoBERTa 这个模型没有参数共享，它的模型容量大、特征表达丰富，预训练时也使用了更大的语料。在引入了 MLP+$\sigma(\cdot)$ 后，非线性层可以更好拟合复杂关系，反而提升了 RoBERTa 的表现，并减少了过拟合现象。

从模型的预训练结构分析，ALBERT 在预训练中使用了 Sentence Order Prediction（SOP）而不是 NSP，其句子间关系建模更简单；可能会导致输出向量分布在 STS-B 上更紧凑，加入 MLP+$\sigma(\cdot)$ 后更容易“过拟合”少量差异。而 RoBERTa 则是使用了大规模动态掩码预训练，而是依靠强大语言建模能力，其特征在句对任务中分布更自然，MLP 可起到进一步优化的作用。

孪生网络方法 如表 1所示，尽管对孪生 LLM 的全量微调性能略低于孪生 BERT，但采用 LoRA 微调的孪生 LLM 取得了最优表现。我们认为对孪生 LLM 的全量微调可能削弱了模型在大规模预训练中建立的通用泛化能力，造成了“灾难性遗忘”。而 LoRA 微调避免了对核心预训练知识的破坏，从而能够有效保留模型原有能力的同时，针对 STS-benchmark 数据集进行优化和适应。

5 总结与思考

本研究通过系统性的实验比较了不同句子嵌入方法在语义文本相似度任务上的表现，涵盖了从传统的词向量聚合方法到现代的预训练语言模型等多种技术路线。实验结果表明，模型架构的演进带来了显著的性能提升：传统词向量方法中 Skip-gram 自训练模型达到 0.475 的 Spearman 相关系数，RNN 类模型进一步改善了序列建模能力，而 BERT 类模型凭借 Transformer 架构取得了更大突破，最终孪生网络结构结合 LoRA 微调的大语言模型实现了最优表现。这一发展轨迹清晰地展现了自然语言处理技术从简单统计方法向深度学习，再向预训练大模型的演进历程。

这些发现为句子嵌入方法的选择和优化提供了重要指导。对于资源受限的场景，传统方法仍具有计算效率优势；对于追求最佳性能的应用，孪生网络结合参数高效微调是理想选择。

参考文献

[1] 苏剑林. 为什么现在的 LLM 都是 Decoder-only 的架构？[EB/OL]. 2023. %5Curl%7Bhttps://kexue.fm/archives/9529%7D.

[2] hli/lstm-qqp-sentence-transformer · Hugging Face[EB/OL]. Huggingface.co. 2023 [2025-06-15]. https://huggingface.co/hli/lstm-qqp-sentence-transformer.

[3] CER D, DIAB M, AGIRRE E, et al. SemEval-2017 Task 1: Semantic Textual Similarity - Multilingual and Cross-lingual Focused Evaluation[C/OL]//Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017. https://aclanthology.org/S17-2001.

[4] CHUNG J, GULCEHRE C, CHO K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.

[5] CONNEAU A, KIELA D, SCHWENK H, et al. Supervised learning of universal sentence representations from natural language inference data[J]. arXiv preprint arXiv:1705.02364, 2017.

[6] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[7] ELMAN J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

[8] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[9] LAN Z, CHEN M, GOODMAN S, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations[C/OL]//International Conference on Learning Representations (ICLR). 2020. https://openreview.net/forum?id=H1eX1jStDr.

[10] LIU Y, OTT M, GOYAL N, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach [J/OL]. arXiv preprint arXiv:1907.11692, 2019. https://arxiv.org/abs/1907.11692.

[11] MIKOLOV T, CHEN K, CORRADO G S, et al. Efficient Estimation of Word Representations in Vector Space[C/OL]//International Conference on Learning Representations. 2013. https://api.semanticscholar.org/CorpusID:5959482.

[12] PENNINGTON J, SOCHER R, MANNING C. GloVe: Global Vectors for Word Representation [C/OL]//MOSCHITTI A, PANG B, DAELEMANS W. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, 2014: 1532-1543. https://aclanthology.org/D14-1162/. DOI: 10.3115/v1/D14-1162.

[13] REIMERS N, GUREVYCH I. Sentence-bert: Sentence embeddings using siamese bert-networks [J]. arXiv preprint arXiv:1908.10084, 2019.