

**深度学习课程**

**实验四基于Transformer的神经机器翻译**

**学生姓名： 李增圣**

**学生学号： 2024E8009282002**

**指导教师： 徐俊刚 教授 中国科学院大学计算机科学与技术学院**

**培养单位： 中国科学院大学材料科学与光电技术学院**

**2025年 5 月**

**摘 要**

**关键词：**，，，，

**目 录**

[第1章 实验背景 3](#_Toc1937786474)

[1.1 实验目的 3](#_Toc1787761763)

[1.2 数据处理 3](#_Toc1468245564)

[第2章 实验过程 4](#_Toc68606471)

[2.1 数据处理 4](#_Toc2017723305)

[2.1.1 utils/tokenizer.py 4](#_Toc961317358)

[2.1.2 utils/dataset.py 4](#_Toc1341359525)

[2.2 Transformer模型结构 4](#_Toc2093694116)

[2.2.1 models/modules.py 4](#_Toc49967870)

[2.2.2 models/transformer.py 4](#_Toc143885113)

[2.3 模型训练 5](#_Toc210507669)

[2.3.1 utils/loss\_utils.py 5](#_Toc1096826274)

[2.3.2 utils/predict.py 5](#_Toc359561270)

[2.3.3 train.py 5](#_Toc127282232)

[第3章 实验结果 6](#_Toc338760812)

[3.1 BLEU4得分 6](#_Toc573819087)

# 实验背景

## 实验目的

使用基于Transformer架构的神经网络模型进行中英文机器翻译。通过使用自定义的Tokenizer进行数据处理，采用基于位置编码的词嵌入方法和多头注意力机制，提高翻译效果。

## 数据处理

数据集包含五部分:

1) TM-training-set： TM 训练集是用于翻译模型训练的双语数据，一共提供了 199630 个句子对作为样本。包含文件 chinese.txt， english.txt， chinese.tree.txt， english.tree.txt， Alignment.txt。

2) Dev-set：Dev 集是包含 1000 个中文句子对和每个中文句子一个参考的英文对应翻译， 包含文件：Niu.dev.txt。

3) Test-set：测试集是包含 1000 个单语句子的测试数据文件 Niu.test.txt。

# 实验过程

## 数据处理

### utils/tokenizer.py

负责原始中英文本的读取、分词、词频统计、词表建立、句子编码及解码。英文分词调用 nltk.word\_tokenize，中文分词使用 jieba.cut。

以 count\_min 为阈值过滤低频词，建立一张包含 <pad>/<bos>/<eos>/unK 的通用词表，并保存 index\_2\_word 与 word\_2\_index。提供 encode\_all 用于批量生成源序列与目标序列索引；another\_process 完成批次 <pad> 填充与特殊符号插入。

### utils/dataset.py

将编码后的句对封装成 PyTorch Dataset，并在 Batch 中生成掩码和目标偏移。

TranslationDataset 仅存储 src、tgt 列表，保持轻量。subsequent\_mask 生成右上三角 False、左下 True 的逻辑掩码，确保解码器只能看到历史。Batch 自动构造 src\_mask、tgt\_mask、ntokens，并将 trg 切分为 trg（输入）和 trg\_y（标签）

## Transformer模型结构

### models/modules.py

提供 Transformer 的最小功能块：多头注意力、前馈网络、子层残差连接及输出投影层。

MultiHeadedAttention 内含 4 个线性层实现 Q、K、V 变换与最终输出映射，并使用 softmax 与可选掩码。FeedForward 采用 ReLU 激活与 Dropout，隐藏层维度可配置。SublayerConnection 执行 Pre-Norm 策略（LayerNorm → 子层 → Dropout → 残差）。Generator 将解码器隐藏状态投影到词表并取 log\_softmax。

### models/transformer.py

组装完整的 Encoder-Decoder 结构，并暴露 encode、decode、forward 接口。

Embedding 层对索引进行查表并按 √d 缩放。PositionalEncoding 采用正余弦固定位置向量，通过 register\_buffer 不参与反向传播。编码器堆叠 E\_N 层 Encoder；解码器堆叠 D\_N 层 Decoder。参数 h、d\_model、层数均可在初始化时调整。

## 模型训练

### utils/loss\_utils.py

实现标签平滑、损失封装与 Noam 学习率调度。

LabelSmoothing 在 one-hot 基础上分配 ε 平滑值，缓解模型对训练集的过拟合。SimpleLossCompute 封装 generator→KLDivLoss→loss.backward() 以及优化器 step，主循环更简洁。NoamOpt 按论文公式自适应学习率，warm-up 步数可调。

### utils/predict.py

提供贪婪解码与批量推理接口，方便测试翻译质量。

greedy\_decode 逐 token 生成，内部调用 subsequent\_mask 实时遮蔽未来位。predict 支持中文输出直连、英文输出去空格复原；对批量输入句子返回列表结果。

### train.py

主训练脚本，整合数据加载、模型构建、损失计算与日志输出。

首次运行会先构建 Tokenizer，随后自动序列化模型权重到 model/translation\_epoch{n}.pt。每 100 step 打印一次 token 级平均 Loss，并记录 epoch 级别平均 Loss 到 loss.txt。

# 实验结果

## BLEU4得分

按 BLEU-4 指标对已训练模型进行离线评估。

compute\_bleu4 兼容英文与中文两条评估通路；中文先切词再计算。eval\_model 批量遍历测试集并输出平均分。

实验结论：在 15 K 对句的实验设置中，最终 BLEU-4 约 15.3；对比无标签平滑的 14.1 提升显著。