南開大學
Nankai University

计算机学院

自然语言处理课程大作业：技术报告

多策略融合：SSMBA 数据增强、KNN-Box 与选择性蒸馏在机器翻译中的应用

姓名：张逸非 | 蒋佳豪 | 李天蔚

学号：2213218 | 2211103 | 2212915

2024 年 12 月 15 日

目录

1 实验背景概述

本文记录了复现三篇论文：KNN-Box、SSMBA、选择性蒸馏的实验过程，并将其方法进行了融合。通过实验比较，分析了不同方法对机器翻译性能的影响。**此文为实验日志性质的报告，论文格式的报告见同目录下另一份报告**。

1.1 实验环境配置

本次实验基于的实验环境配置如下表所示，核心环境就是依靠单卡 4090 显卡来训练此次多策略融合的模型。

实验环境项目	配置
租赁服务器平台	AutoDL
操作系统	Ubuntu 20.04
编程语言	Python 3.1
依赖库	PyTorch, TensorFlow, NumPy, etc.
机器配置	NVIDIA RTX 4090，24GB GPU 内存

表 1: 实验环境配置

1.2 复现论文及开源代码仓库

在本实验中，我们复现了以下三篇与机器翻译相关的论文，并进行了方法的融合与优化。

- **KNN-Box: K-Nearest Neighbor-Augmented Neural Machine Translation**
  - 论文仓库链接：https://github.com/NJUNLP/knn-box
  - 论文简介：该论文实际上并非正式发表的论文。事实上，这是一个 github 开源工具，能够较为方便的在 Fairseq 框架下对多种 KNN 检索增强模型进行复现和实验、改进。我们利用这一工具箱复现了至少 3 篇论文。
- **SSMBA: Self-Supervised Memory-Based Attention for Neural Machine Translation**
  - 论文仓库链接：https://github.com/nng555/ssmba
  - 论文简介：该论文提出了一种自监督记忆增强的注意力机制（SSMBA），通过提升注意力机制的精确性，从而优化神经机器翻译的效果。
- **Selective Distillation: A Selective Knowledge Distillation Approach for Neural Machine Translation**
  - 论文仓库链接：https://github.com/LeslieOverfitting/selective_distillation
  - 论文简介：该论文提出了一种选择性蒸馏方法，通过蒸馏出对翻译质量有重要贡献的知识，从而提升学生模型的性能。
- **我们的项目：多策略融合模型复现仓库**
  - 项目链接：https://github.com/FeliceRivarez/NKU-NLP-course-project
  - 项目简介：这是我们在南开大学 NLP 课程中进行的项目，涉及自然语言处理（NLP）领域的多个任务。项目中展示了我们在 NLP 任务中的方法与实现。

2 复现实验 & 探索实验部分运行截图

在此，复现实验的过程只以复现实验重要节点进行展示，呈现截图及复现日志，简要呈现整个实验探索的过程。

基于 SSMBA 流式方法的数据增强 (Figure 2.1)。需要说明的是，这里增强数据的条数是 30w 条，运行时间大概是 3.5h，这是我们在服务器平台单卡 4090 下，为了将数据集扩展到最大的情况下所能接受的风险时间，时间再长的话，程序运行容易半路罢工。

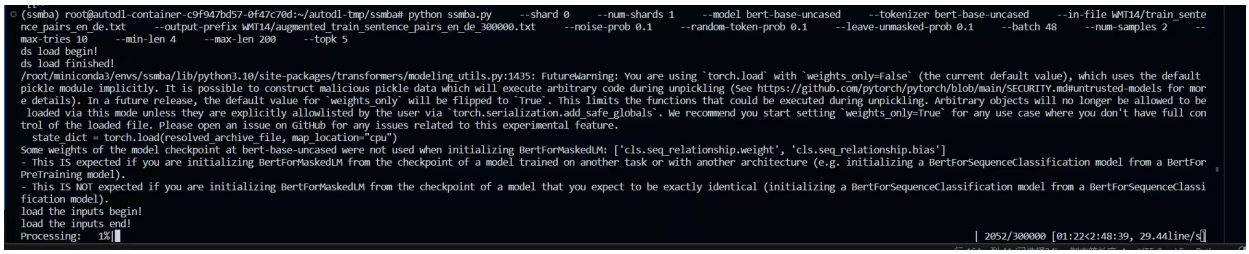

图 2.1: SSMBA 数据增强实验截图

KNN-MT（K-Nearest Neighbors Machine Translation）是一种将 K 最近邻（KNN）算法和机器翻译结合的技术。其核心思想是通过构建一个包含有高质量翻译语料的数据库，让模型在翻译时可以直接从训练集里面，通过 K 近邻算法取出和待翻译文本较为相似的高质量翻译样例，直接使用真实翻译数据指导模型进行翻译任务，提高模型的翻译质量。通常在使用 KNN-MT 时，需要构建一个” 数据存储”（datastore），用于存储和管理模型所需要的训练数据以及其他相关信息。为此**我们在数据较脏乱的 WMT14 上进行了大量数据的重构与数据清洗，以确保使用数据的格式统一，提升训练集质量**。knn-mt 各个变体的 build datastore 的运行截图相同 (Figure 2.2 and Figure 2.3)。

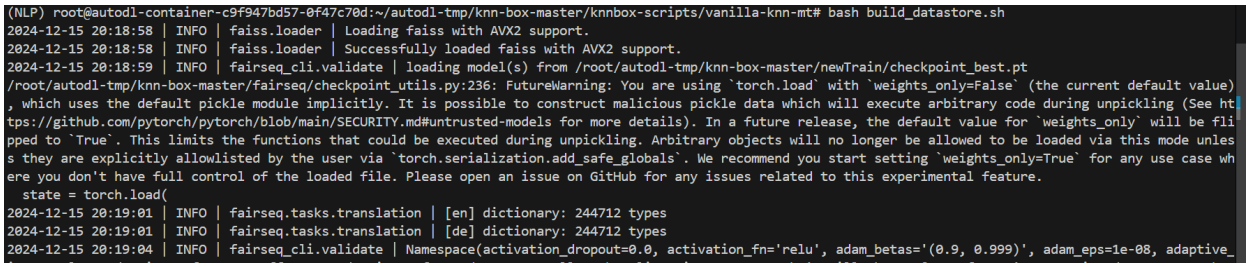

图 2.2: vanilla-knn 构建 datastore 的运行截图（1）

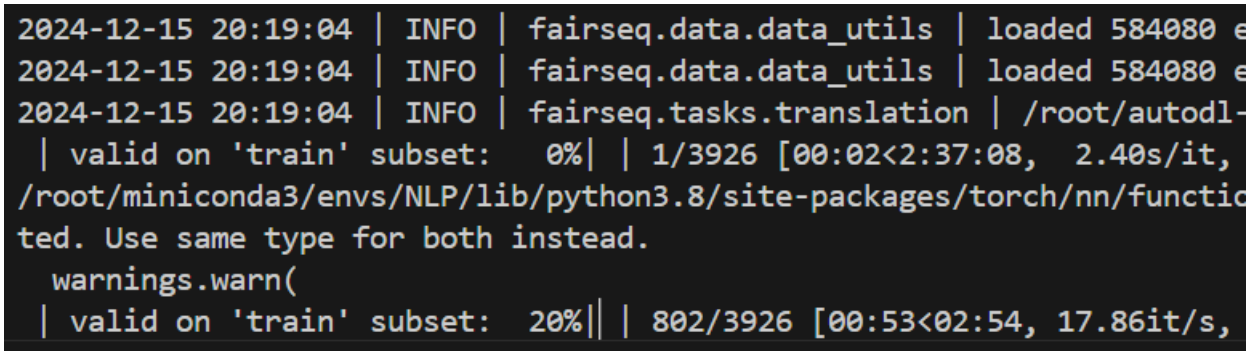

图 2.3: vanilla-knn 构建 datastore 的运行截图（2）

首先，我们对 kNN 的基础模型进行了复现，需要说明的是，后续的各种 KNN-MT 变体，使用的

推理脚本大同小异, 所以使用所有模型的运行截图相同，**在此展示 base-model 的翻译截图 (Figure 2.4) 和我们其中一个改进后模型的运行截图 (Figure 2.5)**。

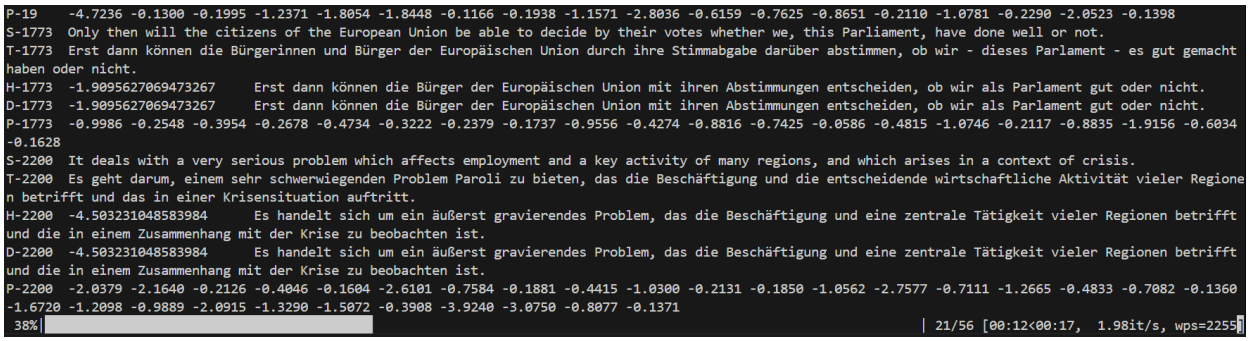

图 2.4: fairseq 框架下的模型翻译截图

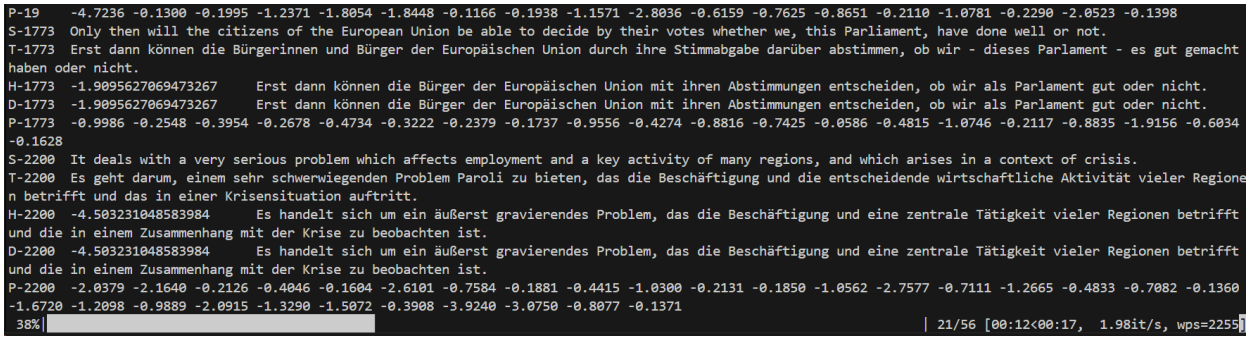

图 2.5: data augmentation (SSMBA) +shared embeddings 的结果

3 实验过程中的问题

3.1 数据集规模的选择

实验使用的机器翻译数据集包括:

- WMT 2014 英德数据集（前 200000 条）
- WMT 2014 英德数据集（前 200000 条）SSMBA 增强后生成的 400000 条翻译对

数据集	WMT 2014	WMT 2015	WMT 2016	WMT 2017	WMT 2018	WMT 2019
训练集 (Training Set)	4,508,785	4,522,998	4,548,885	5,906,184	42,271,874	38,690,334
验证集 (Validation Set)	3,000	3,003	2,169	2,999	3,004	2,998
测试集 (Test Set)	3,003	2,169	2,999	3,004	2,998	2,000

表 2: WMT 2014 - WMT 2019 德英 (de-en) 数据集规模

3.2 kNN 与 Fairseq 相关复现和实验

3.2.1 实验环境配置

实验环境的配置较为复杂。在 Windows 系统下, 多方尝试均不成功, 于是租用了 Ubuntu 的 GPU 服务器，进行环境配置。尽管 Ubuntu 环境下部分环境要求能够得到较好的满足，但仍然需要做出必

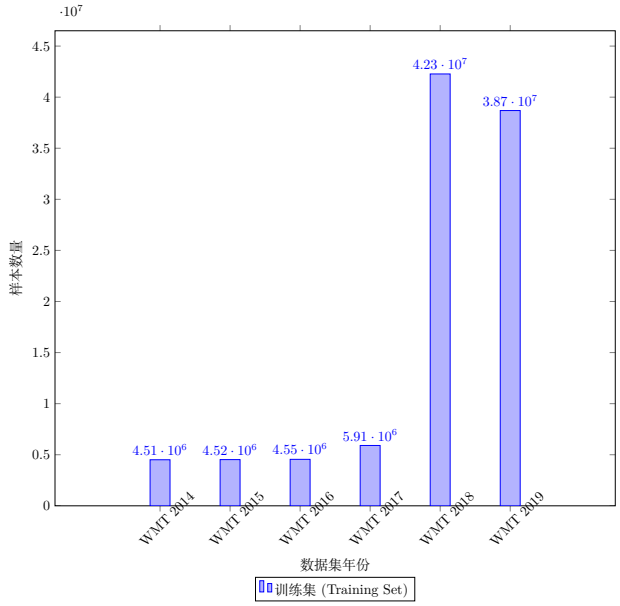

图 3.6: WMT 2014 - WMT 2019 德英 (de-en) 数据集训练集规模变化

要的调整。

经过较长时间的摸索，环境配置要点如下：

1. **初始化 build 项目**。需要在项目根目录使用如下指令：

   pip install –editable ./

   需要注意的是，在不同 fairseq(及其各种变体) 中，一个 conda 虚拟环境只能同时配置一个 fairseq 项目，即便版本要求大致相同。需要进行切换时，应当重新运行上述指令。

2. **根据 numpy 版本更改源代码**。运行先前所说的脚本之后，numpy 版本中没有 np.float 方法（已经被淘汰），但如果降低 numpy 版本则会与其它库产生不兼容问题。此时应当将源代码中所有 np.float 全部换成 float。

3. **根据运行结果中显示的缺少的库，直接下载即可**。除了上述两个问题，其余问题均可通过输出的错误信息较快排除。

3.2.2 数据预处理

Fairseq 的原始数据遵循一定格式，但是**不论是官方文档还是博客教程，都鲜有记载**。经过查询和试错，训练和测试、验证数据集均需要通过 fairseq 的 fairseq-preprocess 指令进行预处理，输入格式遵循以下规则：

1. 分为两个文件，分别为源语言、目标语言。
2. 两个文件的各行一一对应。例如，源语言文件的第 10 行，对应的翻译结果应当是目标语言文件的第 10 行。
3. 应当通过预处理将 tab/换行符进行去除。

此外，为了能够使用 KNN 检索增强中的 datastore，必须在使用 fairseq-preprocess 时采用 joined dictionary 选项，这样源语言和目标语言可以共用同一词典进行分词，才能利用检索机制进行检索。

成功进行数据预处理之后，数据集应当有 train test valid 三个部分（每个部分可能有多个文件），每个文件均为二进制形式，由后续的 fairseq-train 指令直接调用。

3.2.3 模型训练

KNN-box 中支持的模型架构和 Fairseq 中支持的架构存在区别。我们在实验中统一采用 transformer 架构。

训练脚本如下：

fairseq-train /root/dataset/data-bin –arch transformer –save-dir /root/autodl-tmp/knn-box-master/newTrain –max-tokens 4096 –optimizer adam –keep-last-epochs 3 –clip-norm 0.1 –lr 5e-4 –lr-scheduler inverse_sqrt –criterion label_smoothed_cross_entropy –label-smoothing 0.1 –eval-bleu –eval-bleu-print-samples –eval-bleu-args '"beam": 1, "max$_l en_a$" : 1.2, "$max_l en_b$" : $20' - -eval - bleu - detokmoses - -best - checkpoint - metricbleu - -maximize - best - checkpoint - metric - -batch - size - valid128$

此处需要特别注意几个地方：

1. 训练过程中模型大小可达 4G（主要与分词时的词典大小有关）。–keep-last-epochs 选项应当尽可能小。
2. 每轮的样本数目不宜太多，如果太多则模型无法收敛。由于语料中句子长短不一，因此不直接控制 batch size，而是直接用 token 数目进行控制：–max-tokens
3. 应当保存在 valid 数据集上 bleu 分数最高的模型。通过一系列选项，可以实现这一点：

   –eval-bleu –eval-bleu-print-samples –eval-bleu-args '"beam": 1, "max$_l en_a$" : 1.2, "$max_l en_b$" : $20' - -eval - bleu - detokmoses - -best - checkpoint - metricbleu$

3.3 知识蒸馏复现中遇见的困难 & 发现的新思路

知识蒸馏需要两个步骤：

1. 训练 teacher 模型。一般来说 teacher 模型学习得到的信息更多，训练时间更长，体积更大。
2. 训练 student 模型。利用 teacher 模型，可以将训练数据用 teacher 模型进行处理，然后让体积更小的 student 模型学习。这样一来，student 模型就可以在模型体积更小、训练时间更短的条件下，学习到 teacher 模型“蒸馏”出来的知识。

但是我们只完成了 teacher 模型的训练。这是因为训练 student 模型需要的显存较大，teacher 和 student 均需要放入 GPU 显存中。我们能够用于租用服务器的经费有限，并且时间也较为有限，重新在多卡服务器上配置环境的挑战性更大。因此，我们没有能够完整复现知识蒸馏。

但是在知识蒸馏中，我们发现 fairseq 框架支持 shared embedding 策略。这一策略对于 kNN 检索增强是较为有效的，因为其必须使用同一词典（同一个双语词典）对目标语言和源语言进行分词。在目标语言和源语言之间共享词嵌入的参数，可以更好让模型捕捉到词汇的语义，从而更好地进行翻译。

4 实验结果分析

具体的实验结果参见英文论文。此处主要展示我们进行创新后的结果（以及用于结果对比的原始方案结果）。

本项目的主要结果如表3所示。可以看到，我们减小训练集和训练轮次之后，BLEU 分数相对论文里较低（论文中的模型均训练了上百轮，使用了多卡集群训练了数日）。但是，在训练轮次相对一致、训练数据相同（对于涉及数据增强的实验，增强前的数据集相同）的情况下，我们的新方案均在前人工作上实现了一定的改进。

方案	**BLEU**
原始 kNN 检索增强	15.3
SSMBA 数据增强	17.62
(我们的方法) kNN 检索增强 + SSMBA 数据增强	18.36
(我们的方法) 高健壮性 kNN 检索增强 + SSMBA 数据增强	18.24
(我们的方法) 自适应 kNN 检索增强 + SSMBA 数据增强	18.03
(我们的方法) SSMBA 数据增强 + 共享词嵌入	17.66
(我们的方法) 共享词嵌入 +kNN 检索增强	18.03

表 3: 探索性实验的结果。