南京大学

研 究 生 毕 业 论 文

（申请硕士学位）

论 文 题 目　大规模分布式统计机器翻译离线模型训练方法与系统

作 者 姓 名　杨文家

学 科 、专 业　计算机科学与技术

研 究 方 向　大数据并行处理技术

指 导 教 师　袁春风 教授

2017 年 5 月 20 日

学　　号：MG1433074

论文答辩日期：2017 年　5 月　27 日

指 导 教 师 ：　　（签字）

南京大学申请硕士学位论文

大规模分布式统计机器翻译
离线模型训练方法与系统

作者： 杨文家

专业： 计算机科学与技术

研究方向： 大数据并行处理技术

指导教师： 袁春风 教授

南京大学计算机科学与技术系

2017 年 5 月

Offline Model Training Method and System for Large-Scale Distributed Statistical Machine Translation

Presented By

Wenjia Yang

Supervised by

Prof. Chunfen Yuan

A DISSERTATION

FOR THE APPLICATION OF MASTER DEGREE

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND TECHONOLOGY OF NANJING UNIVERSITY

May 2017

声　明

本人声明所呈交的论文是我个人在导师指导下、在南京大学及导师提供的研究环境（含标明的项目资助）下作为导师领导的项目组项目整体的组成部分而完成的研究工作及取得的研究成果。

除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

南京大学及导师所有权保留：送交论文的复印件，允许论文被查阅和借阅；公布论文的全部或部分内容；可以采用影印、缩印或其它复制手段保存该论文。

学生签名：　　日期：

导师签名：　　日期：

Declaration

I make a declaration here that the thesis submitted is composed of the researching work by myself and its corresponding researching results finished as a constituent part of the whole project in the project team lead by my advisor. The thesis is completed with the guidance of my advisor, and under the researching circumstances offered by Nanjing University and my advisor (including the project support indicated).

The thesis does not include other people's researching results ever published or composed, except that are specially annotated and acknowledged somewhere in the article. Any contribution made to the research by my working partners is declared explicitly and acknowledged in the thesis.

Nanjing University and the advisor retain the copyright as follows: submitting the copies of the thesis, allowing the thesis to be consulted and borrowed; publicizing the whole or part of the thesis' content; keeping the thesis by photocopy, microcopy or other copy methods.

Author Signature：　　Date：

Advisor Signature：　　Date：

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：大规模分布式统计机器翻译离线模型训练方法与系统

计算机科学与技术 专业 14 级硕士生姓名： 杨文家

指导教师（姓名、职称）： 袁春风 教授

随着信息技术的快速发展和全球学术、人文的广泛交流，一方面用于机器翻译的平行语料数据集呈现出爆炸性增长的趋势，另一方面实际生活中对机器翻译服务的应用场景也日益增多。统计机器翻译的翻译质量很大程度上取决与平行语料数据集的大小。然而随着语料数据集的增加，传统单机机器翻译系统的模型训练耗时急剧增长，这严重制约了统计机器翻译模型的深入研究及其应用推广。因此，研究大规模分布式统计机器翻译系统具有很大的研究意义和实用价值。

现有的分布式统计机器翻译工具存在模型并行训练性能和可扩展性不足的缺陷，并且难以支持完整的机器翻译离线模型训练流程。事实上，设计实现高效的分布式统计机器翻译离线模型的训练流程存在诸多困难。首先，统计机器翻译中繁多、复杂的任务对计算资源的需求各不相同，这使得在有限的硬件资源下设计高效的并行化算法具有一定的挑战性。其次，模型训练过程中产生的大量 I/O 和网络通信开销，会严重影响模型的并行化效率。最后，若对原始训练数据集或数据集分区中数据倾斜问题的处理不当，容易拖慢整个模型的训练流程。针对上述难点问题，本文在分析现有研究工作的不足和并行化统计机器翻译模型难点的基础上，基于广为使用的分布式数据并行计算平台研究并实现了一个完整、高效、弹性可扩展的大规模分布式统计机器翻译离线模型训练系统，为机器翻译的模型研究和应用服务提供了有利支撑。本文主要研究工作内容与贡献点如下：

（1）针对耗时较长的离线模型训练部分，分析和研究了每个模型或算法的处理流程，完成在大规模场景下的并行化模型训练方法与算法的实现。其中，词对齐模型部分包括预处理和相应的词对齐模型的并行化训练，翻译模型包含三种翻译模型的并行化训练，语言模型支持四种概率平滑算法进行并行化训练。

（2）系统负载均衡和网络通信优化。在词对齐模型中设置合理的分块阈值，在保证并行效率的前提下，降低 I/O 操作和节点间网络通信带来的开销。在翻译模型训练中对原始数据进行数值化处理，降低中间数据集的规模，以及系统负载

和网络通信量，从而提高模型整体训练效率。

（3）按照训练流程中需要多次模型参数的特点，研究实现了使用 join 算子的并行化最大似然估计优化算法。主要有两种优化策略，一是广播小表到分布式大表，避免全局 join；二是对两个待 join 的分布式大表使用相同的 partitioner，使得它们内部数据预先满足相同的划分规则，从而避免执行过程中的数据 shuffle。

（4）针对模型训练中存在的数据倾斜问题，研究实现了两种优化策略。一是适当提高模型训练并发度（或重分区）；二是使用两阶段聚合策略，首先对初始 key 添加随机前缀，使其均匀地分布到各个计算节点。然后在每个节点进行局部聚合，最后去掉前缀进行全局聚合。

（5）最后，基于广为使用的分布式数据并行计算平台 Spark 设计实现了上述大规模分布式统计机器翻译离线模型训练的原型系统 Seal，并在大规模数据集下进行了性能评估与分析。实验结果表明，Seal 的并行训练性能优于现有的单机和分布式统计机器翻译离线模型训练工具，同时还具有更好的可扩展性。

关键词：统计机器翻译；分布式机器翻译模型；分布式数据并行计算平台

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Offline Model Training Method and System for Large-Scale Distributed Statistical Machine Translation

SPECIALIZATION: Computer Science and Technology

POSTGRADUATE: Wenjia Yang

MENTOR: Prof. Chunfen Yuan

With the rapid development of information technology and wide-ranging academic and cultural exchanges, on the one hand, parallel corpus for machine translation show explosive growth, on the other hand, the demand for machine translation is also increasing. The quality of Statistical Machine Translation (SMT) is largely determined by the scale of the parallel corpus. However, with the increasement of the parallel corpus, the training time of model in typical stand-alone machine translation system increases sharply, which immensely restricts the algorithm research and application of SMT. Therefore, the research of large-scale distributed machine translation system has great research significance and practical value.

Existing distributed SMT tools usually have some inevitable problems, such as the inferior parallel performance and scalability, incomplete SMT offline training pipeline. In fact, it is extremely difficult to implement an efficient distributed SMT offline training pipeline. Firstly, SMT system has complicated tasks require different computing resources, making it hard to design efficient parallel algorithms. Secondly, there exist massive I/O and network node communications overhead in model training, which greatly affects the model parallel efficiency. Thirdly, it is easy to slow down the entire training process if the data skew problems in model training are not properly handled. For above difficulties, by analyzing the deficiency of existing research work and the difficulty of distributed machine translation model, this paper implements a complete, efficient, flexible and scalable distributed SMT offline training pipeline, which provides favorable support for the model research and application of SMT. The main contents and contributions of this paper are as follows:

(1) This paper analyzes each model in SMT offline training pipeline, and

implements the process of large-scale distributed training. Word alignment model part includes parallel training of preprocessing and corresponding word alignment model. Translation model part mainly contains three different parallel translation models. Distributed language model training can support four different probability-smoothing algorithms.

(2) System balancing and network communication optimization. In the word alignment training, we set the proper block threshold in data preprocessing. In the translation model training, the training data is processed numerically to cut down the scale of the intermediate data. Owing to these measures, we reduce the system load and network traffic, and meanwhile improve the training efficiency of the model.

(3) Considering the multiple times parameters estimation in model training, this paper optimize the join operation based parallel Maximum Likelihood Estimation algorithm. There are two optimization strategies. First one is broadcasting a smaller table to a large distributed table to avoid global join. The second one is using the same partition function for two distributed tables, making their records in advance to meet the same partition rules in order to avoid the data shuffle in the execution process.

(4) For data skew problems in model training, two methods are studied and implemented. One is to enhance the degree of model training parallelism (or repartition data). The other is to employ the two-stage aggregation strategy. Firstly, extending the key with a random prefix in order to make the records evenly distributed. Then aggregating in each node. Finally, removing the prefix for global aggregation.

(5) Finally, based on the widely used distributed data-parallel computing platform Spark, this paper implements a large-scale distributed SMT offline training pipeline prototype system called Seal, and conducts the system performance comparison analysis under massive corpus. Experiments show that, the parallel performance of Seal is superior to existing standalone and distributed machine translation training tools, meanwhile, Seal has better scalability.

Keywords: Statistical Machine Translation; Distributed Machine Translation Model；Distributed Data-Parallel Computing Platform

目 录

插图清单

表格清单

第一章 绪论

1.1 研究背景和意义

随着信息技术的迅猛发展以及其全球学术、人文的广泛交流，各类语言的文本数据规模呈现出爆炸性增长。日益广泛的跨语言交流对语言间的自动化翻译（也称机器翻译）技术提出了非常迫切的需求[1]。机器翻译是利用计算机自动搜索出从一种自然语言（一般指源语言）到另一种自然语言（一般指目标语言）的最可能转换的过程[2]。

统计机器翻译作为自然语言处理研究中一个热点研究领域，可以满足大多数场景下的翻译需求。作为一种本质上依赖统计模型的方法，统计机器翻译的翻译质量很大程度上取决于平行语料数据集的大小。Brants 等人[3][4]的研究表明，对于一些简单模型，运用更大的训练数据集会使得现有机器翻译系统的翻译质量得到很大提高。如图 1-1 中所示，坐标横轴表示语料库数据规模的增长变化，左侧的坐标纵轴为模型训练时间，右侧的坐标纵轴为翻译质量自动评价指标 BLEU(Bi-Lingual Evaluation Understudy, BLEU)[5]。从图中可以看出，在单机统计机器翻译系统中，随着平行语料库规模不断增大，翻译质量不断提高，但是机器翻译模型训练时间也不断增大[4]。

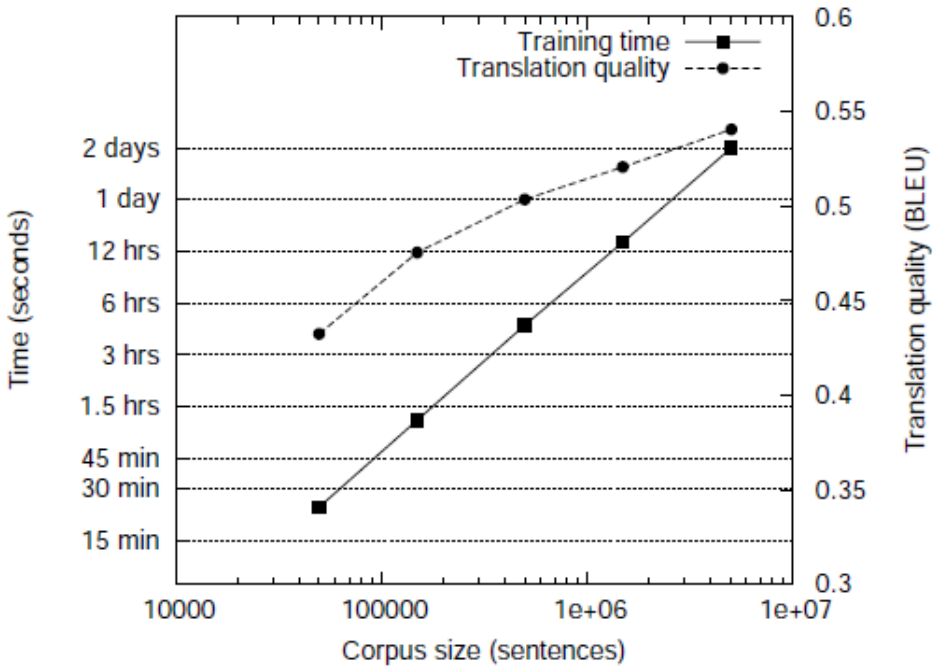


图 1-1 语料库大小变化时模型训练耗时与翻译质量的关系（引自文献[4]）

虽然大规模语料库有益于机器翻译质量的提升，但是在可接受时间范围内利用单机系统完成机器翻译模型的训练存在巨大技术挑战。随着平行语料数据集的增长，现有单机机器翻译系统的模型训练耗时会显著地增加。例如，当前广泛使用的单机统计机器翻译系统 Moses[6]在一台 CPU 频率为 3 G HZ 的机器上，使用 75 万句对的德语-英语平行语料（包含约 1500 万个词，词对齐模型训练使用 GIZA++ [7]）训练基于短语的翻译模型，总共需要耗时 26 个小时左右[8]。尽管 MGIZA++ [9]结合多线程技术对机器翻译系统中最耗时的词对齐模型训练部分进行了优化，然而对于一个更大的数据集（包含约 1000 万句对）仍然需要至少一周以上的时间[10]。对于机器翻译模型本身的研究而言，过大的数据集会使得模型训练时间变长，从而导致机器翻译算法研究的更新迭代周期变长；对于机器翻译应用而言，现实中需要每隔一段时间在新的数据集上训练模型，训练耗时过长会导致机器翻译服务的线上更新周期变长，影响应用服务质量和用户体验。因此，平行语料规模的增加虽然能够提升机器翻译质量，但是它存在的模型训练耗时较长问题严重制约了统计机器翻译模型的研究与应用推广。

随着计算机硬件和并行化技术的飞速发展，多核/众核处理器和广泛使用的分布式集群提供了更多计算资源，大数据时代涌现了多种基于数据并行化（Distributed Data-Parallel）的计算平台和分布式处理系统，例如广泛使用的 MapReduce[11]和 Spark[12]。这些平台和系统为用户屏蔽了底层系统实现细节，如资源调度、数据同步，节点通信等，用户只需要按照其提供的 API 编写并行化程序。基于这些分布式数据并行计算平台，研究者们提出了一系列大规模分布式统计机器翻译的模型训练方法和工具[4][10][13]，相比传统的单机机器翻译系统，这类方法较好地克服了单机资源不够等缺陷，从而能够使用大数据集训练机器翻译模型。

虽然现有的分布式统计机器翻译模型训练方法和工具包括了机器翻译离线模型训练中的基本模块，不过还存在以下几点重要问题：1) 模型的并行训练性能不高。现有的分布式统计机器翻译模型训练工具在大数据集下的训练效率低下，使得模型的并行训练性能和扩展性较差。2) 难以支撑完整的机器翻译离线模型训练流程。例如，现有并行化翻译模型都是基于短语翻译模型，缺少层次化短语翻译模型和基于句法翻译模型的并行化训练。3) 未处理好分布式并行计算平台的制约。机器翻译包含诸多复杂且需要迭代的流程，而 MapReduce 本身存在计

算效率上的缺陷，这类工具未进行很好的处理，导致机器翻译模型并行化训练效率低下。

设计实现一个完整、高效的并行化统计机器翻译离线模型训练系统具有很高的研究意义和实用价值。事实上，利用现有分布式数据并行化计算平台实现统计机器翻译模型的高效训练具有一定挑战性，主要存在以下几种困难：1) 统计机器翻译中复杂和繁多的处理模块的任务类别不仅包含计算密集型，还有 I/O 密集型以及两者复合型。这需要设计合理的并行化算法以高效地利用集群资源。2) 随着数据集规模的增加，翻译模型训练中抽取翻译规则的规模也会急剧增长，使得在参数评估阶段存在大量的数据传输及 I/O 操作，严重影响分布式模型训练的效率。3) 翻译模型和语言模型的并行化训练过程中，原始训练数据集或数据分区中数据倾斜问题处理不当会使得某些计算节点存在较大延迟，从而拖慢整个模型训练流程。

在分析了现有研究工作的不足和并行化机器翻译离线模型训练难点的基础上，本文将使用当前广为使用的数据并行化计算平台 Spark 构建一个完整、高效的分布式统计机器翻译离线模型训练系统，为机器翻译模型的研究和机器翻译的实际应用提供有利支撑。

1.2 现有相关工作

统计机器翻译经历多年发展后，涌现了诸多重要的研究工作和模型训练工具。经典的统计机器翻译框架包括离线模型训练和在线解码两个过程。本文重点关注比较耗时的离线模型部分，具体包括词对齐模型，翻译模型和语言模型三大模块。下面从现有单机模型训练工具和分布式模型训练工具两个方面来介绍分析现有研究工作。

1.2.1 单机机器翻译研究工作

词对齐模型方面，目前普遍使用的词对齐模型工具是 GIZA++ [7]，包括 IBM Model 1 ~ IBM Model 5 以及 HMM 词对齐模型[14] (Hidden Markov Model，HMM)。Gao 等人发现 GIZA++ 中没有利用多核资源，提出了充分利用多 CPU 核优势的单机多线程词对齐模型训练工具 MGIZA+ 和多机多进程词对齐模型训练 PGIZA++ [9]。翻译模型方面，使用比较广泛的是基于短语的翻译模型[15]，基于

层次短语翻译模型[16]和基于句法的翻译模型[17]。语言模型广泛使用 KenLM[18]来进行语言模型训练。近期 Nikolay 等人[19]在 GPU 上应用 B+树来实现 N-gram 语言模型的多路高效查询。然而，这些传统的单机工具由于单机资源的限制，在大数据集上存在较长的训练耗时等问题。

1.2.2 分布式词对齐研究工作

分布式词对齐模型方面，Dyer 等人[4]基于 MapReduce 框架完成了两个词对齐模型（IBM Model 1 与 HMM Model）的并行化训练。Gao 等人[10]提出基于 MapReduce 框架的 Chaski，其中包括预处理（词聚类、共线文件生成等）和相应词对齐模型的并行化训练。词对齐模型并行化训练时，需预先将 MGIZA++ 部署到每个计算节点和设定模型训练序列。第一步在数据分片所在计算节点调用 MGIZA++ 得到词对齐参数的加权统计量，第二步获取所有节点本轮生成的词对齐模型训练结果进行参数的统一正则化更新。按照指定的模型训练反复执行以上两步来得到最终的词对齐结果。然而由于数据分块划分不合理和未处理好 MapReduce 框架本身的制约，Chaski 在大规模数据集下训练词对齐模型的并行训练性能不高。Cadigan 等人[20]基于 Spark 框架研究实现了并行化词对齐模型训练工具 GISA，不过由于 GISA 中仅包含 IBM Model 1 难以满足实际的机器翻译需求。此外 IBM 系列词对齐模型和 HMM 词对齐模型通常都是使用 EM 算法框架[21]训练，在 E-Step 获得词对齐统计量，在 M-Step 完成词对齐参数的统一正则化。因此，也有很多基于分布式数据并行化计算平台上研究实现的并行化 EM 算法[22][23]。

1.2.3 分布式翻译模型研究工作

分布式翻译模型方面，Dyer 等人[4]基于 MapReduce 框架完成了短语翻译模型的分布式训练。Gao 等人[10]研究并实现了并行化短语翻译模型训练工具 Chaski，结合 1.2.2 节中介绍的分布式词对齐模型，形成了一个比较完整的分布式离线翻译模型训练工具。Brodnik 等人[24]基于 MapReduce 框架实现了基于规则依赖的翻译和基于统计的机器翻译的并行化训练，并量化分析了并行化机器翻译模型训练中的系统吞吐量和节点扩展性。Zollmann 等人[25] 基于 MapReduce 框架完成了句法翻译模型的并行化训练，首先从包含词对齐关系的平行语料中抽取概率上下文同步无关文法（Probability of Synchronization Context-Free Grammar,

PSCFG）和翻译规则，然后对翻译表参数调优，最后使用现有的解码器进行解码。Andreas 等人[26]基于 MapReduce 框架实现了句法翻译模型的并行化训练，同时给出了 N-Best 翻译候选规则。Weese 等人[27]提出了一个基于层次化短语翻译模型训练结果的大规模分布式解码器。Tomar 等人[28] 基于 MapReduce 框架，在句法翻译模型的并行化训练中引入内存块替换策略，来加速翻译模型的训练流程。然而这些分布式翻译模型训练工具，一方面没有完整的包括主流翻译模型的并行化训练（仅实现了一种或者两种翻译模型的并行化训练），另一方面在大数据集下存在模型并行训练性能不高和扩展性较差的问题。

1.2.4 分布式语言模型研究工作

分布式语言模型方面，Brants 等人[13]基于 MapReduce 框架完成了在超大规模数据集下 N-gram 语言模型的并行训练。实验结果表明，在语料数据集足够大时，传统计算复杂度较高的概率平滑方法对模型精度的提升不明显，针对这个问题他们提出了一种简洁高效的概率平滑算法。Allam 等人[29]基于 MapReduce 框架和分布式非关系型数据库 HBase[30]，研究了在大规模数据集下 N-gram 语言模型的并行训练和最终结果存储组织方式。Yu 等人[31]基于 MapReduce 框架并行化训练 N-gram 语言模型，并利用 HBase 进行 N-gram 语言模型的存储和概率查询等。Uszkoreit 等人[32]基于 MapReduce 框架，讨论在 N-gram 稀疏的情况下，使用词聚类结果来提高语言模型的精度。Emami 等人[33]利用 MPI 实现了 N-gram 语言模型的并行化训练，并提出了一种高效的方法来估计语言模型参数的后退权重。Tejas Patil 等人[34]基于 Spark 编程框架进行大规模分布式语言模型训练，并提出了一种高效的数据分片方式来提高语言模型的训练效率。另外，随着人工智能和深度学习技术的快速发展，出现了一系列基于神经网络和深度学习的语言模型[35]。 同现有的分布式翻译模型训练工具一样，现有的分布式语言模型训练工具也存在模型并行训练性能不高和扩展性较差的问题。

1.3 本文的研究内容和主要贡献点

在分析了现有研究工作的不足和并行化统计机器翻译离线模型训练难点的基础上，本文基于分布式数据并行计算平台研究并实现了一个完整、高效、弹性可扩展的分布式统计机器翻译离线模型训练系统。本文主要贡献可总结为以下几点：

1）针对经典统计机器翻译框架中训练耗时较长的离线模型训练，本文通过对每个模型的处理流程和算法进行分析，研究实现了在大规模场景下的并行化模型训练方法和算法。其中，词对齐模型部分包括预处理和相应词对齐模型的分布式训练；翻译模型包括短语翻译模型，层次化短语翻译模型和句法翻译模型的分布式训练，并支持翻译模型参数的概率平滑；语言模型支持使用四种概率平滑算法进行并行化训练。

2）研究实现了系统负载和网络通信优化。词对齐模型中设定合理的分块阈值，在保证并行效率的前提下，降低 I/O 操作和网络节点间的通信带来的开销。翻译模型中对原始数据进行数值化处理，并在最后一步进行还原处理。经过数值化处理可以显著地降低中间结果集的数据规模，以及系统负载和网络通信量，从而提高模型训练效率。

3）依据训练流程中需要多次估计模型参数的特点，研究实现了基于 join 算子的并行化最大似然估计优化算法。主要采用两种优化策略：一是广播小表到每个计算节点，在分布式大表的每个分区中获取广播的小表来避免全局 join；二是对两个待 join 的分布式大表使用相同的自定义（或 Spark 自带的）partitioner，使得它们内部数据预先满足相同的划分规则，从而避免执行过程中的数据 shuffle。

4）针对模型训练中存在的数据倾斜问题，研究实现了两种优化策略。一是使用并行化编程框架自带的接口适当提高模型训练的并发度（或重分区）。二是使用两阶段聚合策略，首先对原有的 key 添加随机前缀（或后缀）来扩展 key 的数量，使得数据均匀分布到每个计算节点，然后在每个节点进行局部聚合，最后再去掉添加的前缀来进行全局聚合。

5）最后基于广为使用的分布式数据并行计算平台 Spark，设计实现了大规模分布式统计机器翻译离线模型训练的原型系统 Seal，并在大规模数据集下进行了性能评估与分析。实验结果表明，词对齐模型方面，Seal 的并行训练性能比 Chaski 平均提高 5 倍。翻译模型方面，短语翻译模型的并行训练性能相较 Chaski 平均提高 5~8 倍，层次化短语翻译模型和句法翻译模型的并行训练性能相较现有分布式工具平均提高 13~21 倍。语言模型方面，Seal 的并行训练性能相较现有的分布式语言模型平均提高 7~9 倍。同时，Seal 还具有更好的可扩展性。

1.4 本文的组织结构

本文的组织结构如下：

第一章，绪论。首先说明本文研究工作的背景和研究意义，阐述了现有研究工作存在的不足以及并行化统计机器翻译离线模型训练的难点，最后总结本文的主要贡献点。

第二章，相关背景知识。主要介绍统计机器翻译中离线模型训练部分的基本理论知识。

第三章，分布式统计机器翻译离线模型训练方法。主要介绍分布式统计机器离线模型训练的具体实现细节。

第四章，大规模分布式统计机器翻译模型训练系统的研究与实现。主要介绍大规模分布式统计机器翻译离线模型训练原型系统 Seal 的整体框架和设计细节。

第五章，实验效果评估。主要介绍使用大规模数据集在分布式环境下对 Seal 中模型的并行训练性能进行评估与分析。

第六章，总结与展望。主要对本文的主要工作进行总结，并讨论下一步要展开的工作。

第二章 相关背景知识

本章首先给出统计机器翻译的组成以及单机机器翻译系统 Moses 的离线模型训练流程，扼要说明离线模型中的相关理论知识。

2.1 统计机器翻译简介

第一个具有完备数学理论基础的统计机器翻译模型是 Brown 等人的 IBM 系列模型[3]。统计机器翻译被广泛认为是一个噪声信道方法，如图 2-1 所示。若文本串 T 为最终译文，使用某个特定噪声信道后得到源语言串 S，即源语言串 S 是文本串 T 的一段特殊编码，机器翻译就是将 S 转换成 T 的过程[2]。

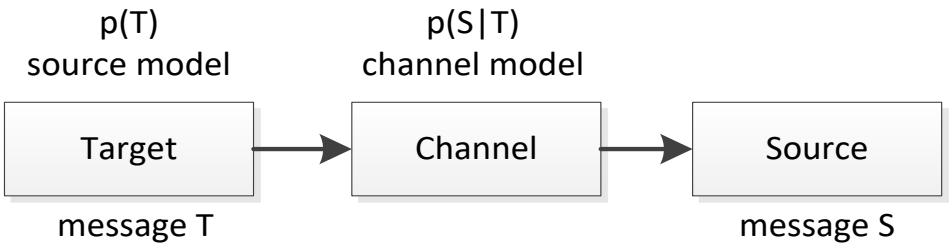


图 2-1 噪声信道模型

按照这种看法，目标语言中的全部文本串都有概率成为源语言中的译文，如式（1）由贝叶斯公式可得：

$$T = \arg\max_T P(T) \times P(S \mid T) \tag{1}$$

其中，语言模型 $P(T)$ 仅与目标语言相关，反映文本 T 在目标语言中存在的合理程度。翻译模型 $P(S \mid T)$ 与两个方向的语言皆相关，反映目标语言文本 T 和源语言文本 S 互为译文的概率。

图 2-2 为经典统计机器翻译系统框架图。输入一个源语言句子，先将源语言句子分割成很多翻译单元来得到解码器的输入（如句法翻译系统中翻译单元表示包含句法信息的翻译规则对），然后解码器利用语言模型与翻译模型的训练结果对源语言文本进行解码得到译文，最后将其转化为目标语言文本。因此统计机器翻译问题一般拆分为三个研究点：

(1) 语言模型 $P(T)$ 参数计算，反映一个句子的合理程度。

(2) 翻译模型 $P(S \mid T)$ 参数计算，反映两个句子互翻译的概率。

(3) 翻译解码，搜查最优的翻译译文。

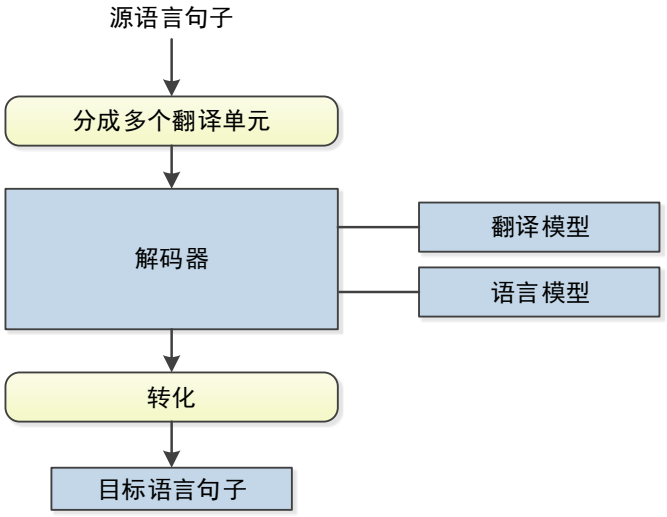


图 2-2 经典统计翻译系统框架

翻译模型的训练依赖于平行句对间的词对齐关系，因此需要先训练词对齐模型再进行翻译模型的训练。一般将词对齐模型、翻译模型和语言模型的训练看作是统计机器翻译系统中的离线模型训练部分。本文重点介绍离线模型训练部分，而解码过程，本文不对其进行介绍。图 2-3 所示为单机机器翻译工具 Moses[6]离线模型训练部分的流程，其中翻译模型为短语翻译模型，使用单机工具 GIZA++ [7]训练词对齐模型。整个流程图包含 10 个步骤，接下来的几节会按照图 2-3 所示的机器翻译系的统流程来进行介绍。

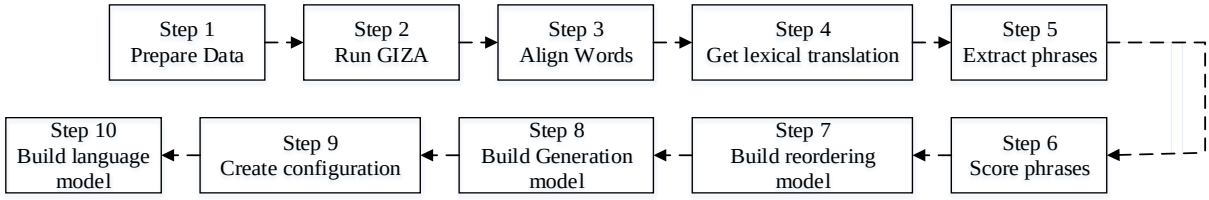


图 2-3 Moses 翻译系统处理流程

2.2 最大似然估计

统计机器翻译中通常使用最大似然估计方法来估计模型参数。如词对齐模型训练中参数的更新，翻译规则概率估计以及语言模型中的概率计算。因而这里对最大似然估计给出扼要说明。最大似然估计（Maximum Likelihood Estimation，MLE）是数据分析理论中估计未知参数的基本方法之一，其假定要计算的参数服从统一的先验分布。

假设要估计的概率分布为θ，其概率质量函数或概率密度函数为f_θ。MLE的目标是寻找要估计参数θ的最优估计值θ^*，使得在给定多个观测变量的采样$X_1=x_1, X_2=x_2,...X_n=x_n$下的概率最大，如式（2）所示：

$$l(\theta^*)=\arg\max f_\theta(x_1,x_2,..,x_n\mid\theta) \tag{2}$$

则每个观测变量的最大似然估计为式（3）所示：

$$f(x_i\mid\theta^*)=\frac{m}{N} \tag{3}$$

其中，x_i为变量X_i的观测量，m为观测量x_i在语料中的频数，N为变量X_i全部值的频数和。

这里概要说明最大似然方法在统计机器翻译中的应用。公式（4）为翻译模型中词翻译概率的估计，其表示双语平行语料中源语言单词到目标单词的互翻译概率。在公式（4）中为源语言方向单词w_e翻译为目标语言方向单词w_f的概率，其中$count(*)$为观测变量在平行语料的频数。

$$P_{MLE}(w_f\mid w_e)=\frac{count(w_f,w_e)}{\sum_{w_f'}count(w_e)}=\frac{count(w_f,w_e)}{count(w_e)} \tag{4}$$

2.3 词对齐模型

词翻译模型（即词对齐模型）虽然已不是主要的统计机器翻译模型，但是现在广泛使用的翻译模型（如短语翻译模型）在训练时都需要平行句对间的词对应关系。因此词对齐模型训练是统计机器翻译系统中不可或缺的过程。本节对词对齐模型的基本背景知识进行扼要说明。

给定双语平行句对(f_1^J,e_1^I)，f_1^J表明源端语言文本有J个词，e_1^I表明目标端语言文本有I个词。句对间的词对齐映射函数为α，如式子（5）所示：

$$\alpha\subseteq A_I^J=\{(j,i)\mid j\in[1,J],i\in[0,I]\} \tag{5}$$

其中$i=0$为源语言句子中个别单词可能与目标语言中的词没有映射关系，比如一些连接词。如公式（6）所示，翻译概率是全部可能词对齐关系的概率总和。

$$P(f_j^J\mid e_i^I)=\sum_{\alpha\in A_I^J}P(f_1^J,\alpha_1^J\mid e_1^I)=\sum_{\alpha\in A_I^J}P(\alpha_1^J\mid e_1^I,f_1^j)\prod_{j=1}^{J}P(f_j\mid e_{a_j}) \tag{6}$$

IBM 系列共有 5 个模型，即 IBM Model 1 ~ IBM Model 5[1]。从模型时空间复杂度上看，这 5 个模型的时空间复杂度表现出递增关系；从模型参数的包含关系看，为依次包含（即 $1\subseteq 2\subseteq 3\subseteq 4\subseteq 5$ ）。因此它们的训练顺序一般为：$1\rightarrow 2\rightarrow 3\rightarrow 4\rightarrow$。这 5 个模型可以概括为表 2-1。

表 2-1 IBM Model 1~5 概括

模型	描述
Model 1	Lexical translation
Model 2	Added absolute alignment
Model 3	Added fertility model
Model 4	Added relative alignment model
Model 5	Fixed deficiency problem

在 IBM Model 1 词对齐模型中的参数只有词翻译概率，如式（7）：

$$P(f,\alpha\,|\,\mathrm{e})=\prod_{(i,j)\in\alpha}P(f_i\,|\,e_j) \tag{7}$$

其中 f 和 e 分别为源端和目标端文本，(i, j) 为源端文本 f 中位置为 i 的词翻译为目标端文本 e 中位置为 j 的词（位置表示下标）。IBM Model 2 在 IBM Model 1 之上，扩展词在句子中的绝对位置关系。如公式（8）所示。

$$P(f,\alpha\,|\,e)=\prod_{(i,j)\in\alpha}P(f_i,e_j)\bullet Q(i\vee j,l_e,l_f) \tag{8}$$

公式（8）中平行句对的长度各自为 l_e,l_f，$Q(i\vee j,l_e,l_f)$ 则表示词位置间的对齐概率。IBM Model 3~5 中都增加“fertility model”，来对应一个词映射到多个词的可能性。HMM Model[14]中改进了 IBM Model 2 中使用词之间绝对位置关系的不足，可以得到更好的词对齐效果。

词对齐模型一般使用 EM[21]算法在双语平行数据集上实施多轮训练。如图 2-4 为使用 EM 算法框架的词对齐模型训练流程图。每轮 E-Step，估计平行句对 (f_1^J,e_1^I) 中局部最有（或所有）可能的词对齐参数，得到参数的加权统计量；每轮 M-Step，对本轮 E-Step 中得到的词对齐参数进行正则化更新。虽然 IBM Model

[1] IBM Models. https://en.wikipedia.org/wiki/IBM_alignment_models

1-2 能够得到所有可能的词对齐信息，不过其他模型抽取所有可能的词对齐信息的计算复杂度是 NP 难。因此一般使用 Viterbi 算法[36]来搜索可能的词对齐结果。

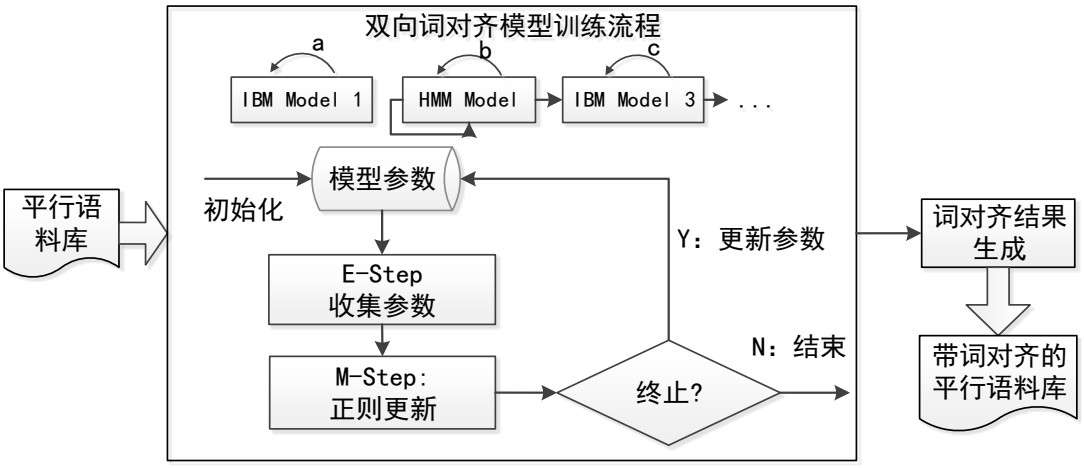


图 2-4 使用 EM 算法的词对齐模型训练流程图

2.3.1 现有的单机词对齐模型训练工具

在约翰霍普金斯大学(Johns Hopkins University)提出包括 IBM 系列词对齐模型训练的工具 GIZA [37]，其优化版本 GIZA++ [7]是现在如今普遍的词对齐模型训练工具。使用 GIZA++进行词对齐模型训练时，模型间的计算顺序为 1→HMM→3→4（不使用 Model 2 是因为其对齐效果不明显，而 Model 5 在训练时非常耗时且对翻译质量提升微小）。一般需要源语言到目标语言方向的词对齐训练和目标语言到源语言方向的词对齐训练，最后再对两个方向的词对齐结果合并得到最终的词对齐结果。然而 GIZA++ 在较大规模数据集下的模型训练是非常耗时[4][8]。因此出于对性能等方面的考虑，Gao 等人结合多线程技术提出优化后的词对齐模型训练工具 MGIZA++ [9]，其可以较大地提高词对齐模型的训练效率。

2.3.2 单机词对齐模型训练示例

以中-英语词对齐模型训练为例，给定中文（源语言）句子 S=“澳洲 是 少数 与 北韩 建交 的 国家”，英文（目标语言）句子 T=“Australia is one of a few countries that has relationship with North Korea”。使用 MGIZA++ 来获取词对齐结果，下面为具体的训练流程。

第一步，预处理生成 MGIZA++ 训练词对齐模型的所需的文件，这些中间结

果主要由 plain2snt、snt2cooc 和 mkcls[1]三个程序产生。

其中 plain2snt 程序对双语平行语料进行数值化处理，产生后缀为*.*vcb* 和*.*snt* 两种文件。其中*.*vcb* 文件格式为 (*ID* ||| *word* ||| *count*) 。如在 ch.vcb（en.vcb）中有“2 澳洲 4”中，2 为词“澳洲”对应的编码，4 为“澳洲”在语料中的计数。*.snt 文件格式为“平行语句在语料中出现的次数 ||| 中文句子字符串 ID ||| 英语句子字符串 ID”。如在 ch_en.snt（en_ch.snt）中有“1 ||| 2 3 4 5 6 7 8 9 ||| 2 3 4 5 6 7 9 10 11 12 13 14 15”，1 为该句对在语料中出现的次数，“2 3 4 5 6 7 8 9”对应中文句子“澳洲是少数与北韩建交的国家”。

snt2cooc 程序生成双语单词共线表，其包含在平行语料数据集中所有可能出现的单词对（形成单词对的前提是源语言对应的单词和目标语言对应的单词必须在同一个句对中出现过）。文件后缀为*.cooc，格式为：源语言词 ID 目标语言词 ID，如在 ch_en.cooc 中有“2 3\n3 3…”。

mkcls 程序产生词聚类文件。这个过程的耗时与词典的大小、设定的聚类个数和迭代轮数有关。产生后缀为*.classes 和*.classes.cats 的两种文件。其中*.classes 的文件格式为“语料中的词 词的类别 ID”，如在 ch.classes（en.classes）中有“澳洲 34\n 是 25”, 34 为词“澳洲”的类别 ID。*.classes.cats 的文件格式为“类别 ID 对应的词组”，如在 ch.classes.cats(en.classes.cat)中有“34 澳洲，东、记住，…”。

第二步，源语言到（中文）到目标语言（英语）的词对齐模型训练，设定每个模型的迭代轮数，训练完成后得到词对齐结果，如图 2-5 所示。

```
# Sentence pair (1) source length 8 target length 14 alignment score : 9.42911e-25
  Australia is one of a few countries that has relationship with North Korea
  NULL ({ }) 澳洲 ({1}) 是 ({2}) 少数 ({6}) 与 ({9}) 北韩 ({12 13}) 建交 ({9 10 11}) 的 ({ })
国家 ({7 })

……
```

图 2-5 中文到英语的词对齐结果

同样可得目标语言（英语）到源语言（中文）的词对齐结果，如图 2-6 所示。

[1] http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/mkcls.html

```
# Sentence pair (1) source length 14 target length 8 alignment score : 9.90936e-12
  澳洲 是 少数 与 北韩 建交 的 国家
  NULL ({ }) Australia ({1}) is ({2}) one ({3}) of ({3}) a ({3}) few ({3}) countries ({8}) that
({4}) has ({4 6}) relationship ({6}) with ({1}) North ({5}) Korea ({5})
……
```

图 2-6 英语到中文的词对齐结果

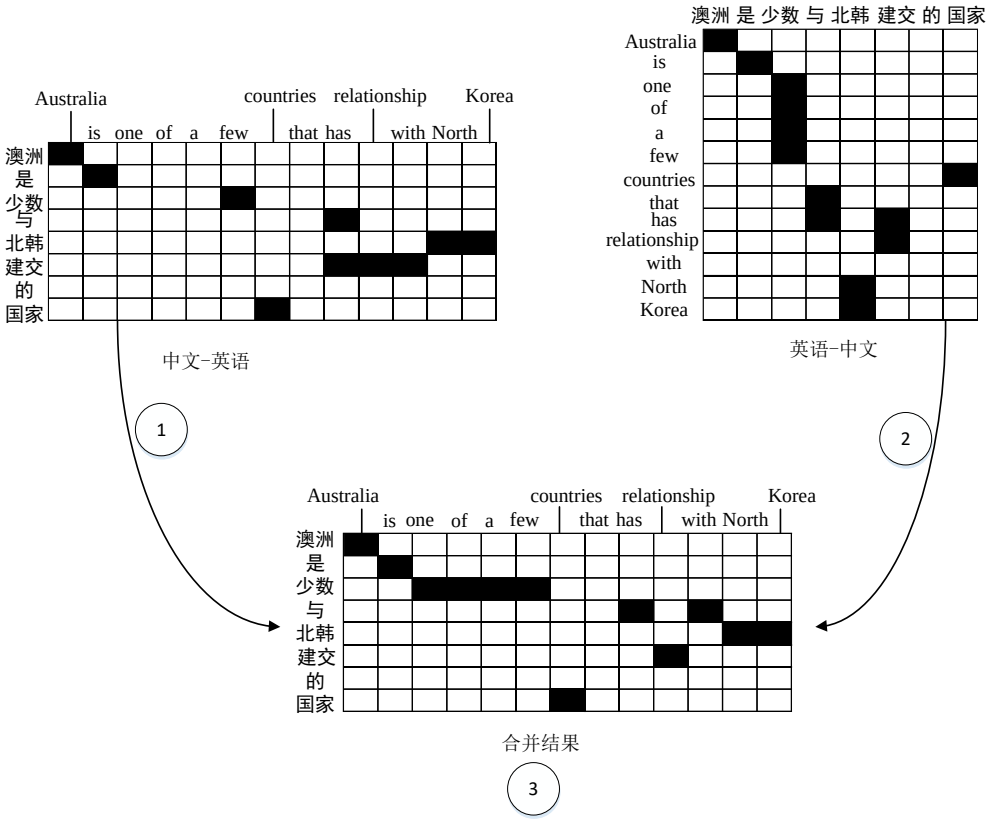


图 2-7 MGIZA++词对齐合并结果

第三步，合并双向的词对齐结果。最终的词对齐结果需要将两个方向对齐结果进行组合，因为单向的对齐精确度较低。合并双向的词对齐结果有很多启发式的方法。MGIZA++ 中默认的合并策略为“ *grow – diag – final* ”[1]。如图 2-7 为合并过程示例图，图中实心空格表示对齐。合并后得到最终的词对齐结果，如“0-0 1-1 2-1 2-3 2-4 2-5 3-8 3-10 4-11 4-12 5-9 7-6”。

[1] http://www.statmt.org/moses/?n=FactoredTraining.AlignWords

2.4 翻译模型

翻译模型为两个语言方向的句子互为译文的概率。本文将在双语平行语料中训练得到的翻译知识表示统称为翻译规则（短语、层次化短语和带有句法信息的短语和层次化短语），翻译规则可以看作是源语言和目标语言间翻译知识表示间的映射关系。如图 2-8 所示为机器翻译的金字塔，翻译规则的粒度从单词到短语再逐渐变为层次化短语，随后又结合了上下文和句法结构等信息。很多学者在层次化短语翻译模型的基础上将语言的句法信息引入到翻译模型中，涌现了一系列基于句法的翻译模型，这些模型可以归为三类：串到树（String to Tree, S2T）、树到串（Tree to String, T2S）和树到树（Tree to Tree, T2T），其中树表示一端的语言句子的语法信息树，串为本来的句子。因此本节将分别介绍短语翻译模型，层次化短语翻译模型和句法翻译模型。

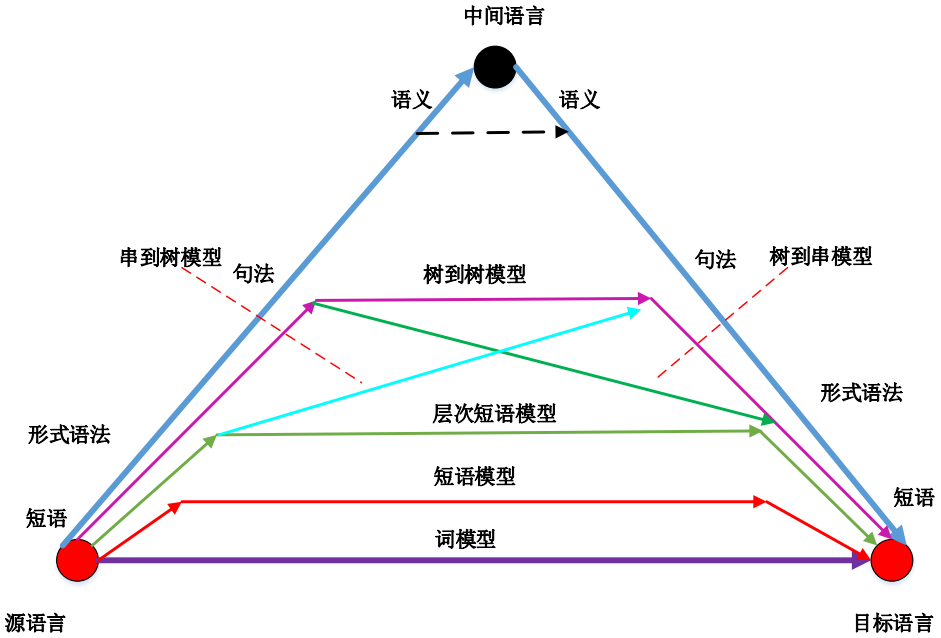


图 2-8 机器翻译金字塔

2.4.1 短语翻译模型

在 Brown 等人[3]的词翻译模型基础上，后续研究人员发现词翻译模型存在以下缺陷。1. 词翻译模型中存在一对多和一对空的映射，这不但降低了翻译质量同时也提高模型的训练难度。2. 以单词为翻译粒度时很难在中文等类似语言中确定词之间的边界。3. 实际翻译任务中，容易造成翻译结果中每个词对应的翻译结果正确，但翻译顺序却不对的情况。为了解决上述缺陷，Philipp 等人[15]提出

以多个连续单词构成的短语为整体的翻译模型，这样目标语言句子中的任何一个非空短语都可以映射到源语言句子中的一个非空短语。Philipp 等人开发的基于短语翻译模型的翻译系统 Moses 成为学术界公认的翻译基准系统。很多研究学者发现，平行句对中互为翻译的短语对并不是严格按照句子的顺序对齐，因此需要对短语对之间的翻译顺序进行调整。Tillmann 等人[38]将短语对间的顺序总结为三种：单调(monotone, m)、交叉(swap, s)和间断(discontinuous, d)。

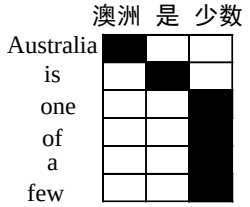
短语翻译模型中一般有五个需要估计的翻译规则参数。分别是反向短语翻译、反向短语对词汇权重、正向短语翻译概率、正向短语对词汇权重和短语对惩罚度（默认为自然常数 2.178 或短语对计数的倒数，正向为源端语言方向到目标端语言方向）。

公式（9）为正向短语对翻译概率，其中 $C(f,e)$ 为语料中短语对 (f,e) 出现的频数，$\sum_{f'} C(f',e)$ 为翻译规则中目标短语为 e 的所有短语对总频数。同理可得另一个方向的短语对翻译概率。

$$P(f \mid e) = \frac{\mathrm{C}(f,e)}{\sum_{f'} \mathrm{C}(\mathrm{f}',\mathrm{e})} \tag{9}$$

公式（10）为正向短语对词汇翻译权重，其中 (f,e) 为从平行语料中抽取的翻译短语对，α 表示短语对间的词对齐映射关系。n 为源语言短语翻译规则对的长度，$w(f_i \mid e_j)$ 为源语言短语翻译规则 f 中的第 i 个词翻译为目标语言短语翻译规则 e 中第 j 个词的词翻译概率。图 2-9 为正向词汇化权重计算过程示例。

$$P_w(f \mid e,\alpha) = \prod_{i=1}^{n} \frac{1}{|\{\mathrm{j} \mid (\mathrm{i},\mathrm{j}) \in \alpha\}|} \sum_{\forall (i,j)\in\alpha} w(\mathrm{f}_i \mid \mathrm{e}_j) \tag{10}$$



$$P_w(\bar{f} \mid \bar{e},\alpha) = P_w(f_1, f_2, f_3 \mid e_1,e_2,e_3,e_4,e_5,e_6,\alpha)$$
$$= w(f_1 \mid e_1) \times w(f_2 \mid e_2) \times \frac{1}{3}(w(f_3 \mid e_3) + w(f_3 \mid e_4) + w(f_3 \mid e_6))$$

图 2-9 正向词汇化权重计算示例

2.4.2 层次化短语翻译模型

虽然短语翻译模型较好处理了词翻译模型的不足，不过短语翻译模型还是存在以下问题。一方面，包含词较少的短语需要调整翻译顺序，另一方面，包含词较多的短语容易产生稀疏。为此 Chiang[16]等人以同步上下文无关文法（Synchronization Context-Free Grammar, SCFG）为理论指导，在翻译规则中增加一些 SCFG 规则来解决上述问题。在 SCFG 结构中将重写包含词对齐信息的翻译规则，如下所示。

$$X \rightarrow <\gamma, \alpha, \sim>$$

X 表示非终结符，γ 和 α 由终结符和非终结符构成，$\sim$ 表示 γ 和 α 中非终结符间的单射函数。翻译规则由一个连接符开始，在每一词连接中用单一部分文法规则来重写两个相关的非终结符。抽取出的包含 SCFG 翻译规则如下所示：

$$X \rightarrow < X_1\text{之一} \mid \text{one of } X_1 >$$

层次化短语显著地扩充了翻译规则的数量，不但有效解决了短语较长时的稀疏问题，而且也解决了长度较短的短语在长距离上的调序问题。然而在大规模平行语料数据集中抽取的翻译规则会呈现爆炸性增长，严重影响模型训练效率。同时层次化短语翻译模型本身对抽取的翻译规则不进行任何约束，这对翻译的解码过程和翻译单元的组织存储提出了很大的技术挑战。

2.4.3 句法翻译模型

从整个模型训练耗费的时间和翻译规则占用空间来看，层次化短语翻译模型比短语翻译模型差；从译文质量上来看，层次化短语翻译模型优于短语翻译模型。这两种翻译模型中对于翻译规则的约束可以看作是基于词对齐信息的边界约束和基于同步上下文无关文法的语法规则约束。

虽然引入更多的句法信息可以一定程度上提升翻译质量，然而模型训练耗时、模型结果难以存储以及解码时的不匹配等问题日益成为约束句法翻译模型发展的主要因素，所以需要翻译规则进行限制和惩罚。Libin Shen 等人[17]提出一种基于目标端句法的翻译模型，其抽取的翻译规则中源语言端表示不变，而在目标语言端中加入句法依存树的信息，并且限定目标端的规则必须为符合特定语法的

依存结构。这样不但可以清除原有一些坏的（没用的）的翻译规则，而且会使得后续翻译规则的解码更加高效。他们将目标端句法结构树归纳为：符合语法规则（well-formed）的依存结构和不符合语法规则（ill-formed）的依存结构。解码时可以选择过滤不符合条件的规则。符合语法规则的依存结构又可以分为“fixed”依存结构和“floating”依存结构。句法翻译模型在短语或者层次化短语抽取的同时，在目标端规则中加入句法树依赖信息（从不同的双语平行句对中抽取的相同的翻译规则，其目标端的句法树依赖信息可能会不同），并用式（11）进行惩罚。

$$P(\mathrm{st}_i \mid \mathrm{dt}, \mathrm{C}) = \frac{Count(\mathrm{st}_i, \mathrm{dt})}{\sum_i Count(\mathrm{st}_i, \mathrm{dt})} \tag{11}$$

其中 dt 表示包含目标端句法树依赖信息的翻译规则，st_i 表示翻译规则 dt 的一个句法依赖状态，C 为上下文信息。$\mathrm{Count}(\mathrm{st}_i, \mathrm{dt})$ 表示在给定翻译规则 dt，在目标端句法依赖状态为 st_i 翻译规则出现的频数。

2.4.3 翻译模型平滑

在上述翻译模型中，计算翻译规则的参数概率时，都是应用最大似然来进行计算。然而一般训练数据集规模有限，抽取的翻译规则不能覆盖所有可能的情况，此时就会产生零概率。

为了尽可能消除零概率问题，就需要参数概率进行平滑处理。Good-Turing[39]平滑，其原理上是根据真实频次来获得期望频次。在短语翻译模型中，依据训练语料中短语对的真实频次来获得期望频次的计算，如公式（12）：

$$C_g(f, e) = \frac{(\mathrm{C}(\mathrm{f}, \mathrm{e}) + 1) * \mathrm{n}_{c+1}}{n_c} \tag{12}$$

其中 f 和 e 分别为源端短语和目标端短语，$C(f, e)$ 为源端和目标端短语对共同出现频数，n_c 是所有短语对中出现c次的短语对频数，$C_g(f, e)$ 为短语对的期望频数。如式（13）为计算源端到目标端方向的短语对翻译概率。

$$P(f \mid e) = \frac{C_g(\mathrm{f}, \mathrm{e})}{\sum_e C_g(f, e) + P(f) * \mathrm{n}_1} \tag{13}$$

式（13）中，$P(f) = \frac{C(f)}{\sum_{f'} C(f)}$，$C(f)$ 是源语言端短语 f 出现的频数，n_1 是短语对中仅出现一次的短语对个数。同样可以获得目标端到源端方向的短语对翻译概率。

Knerser 和 Ney 等人[40]提出的 Kneser-Ney（KN）平滑方法是目前使用最为广泛的概率平滑方法，Chen 等人[41]又提出了修正的 Kneser-Ney 平滑方法（Modified Kneser-Ney smoothing, MKN）。

2.4.5 翻译模型训练基本流程

前面几个小节中分别介绍了短语翻译模型、层次化短语翻译模型、句法翻译模型以及模型参数平滑。本小节将以单机短语翻译模型的训练来说明翻译模型的基本流程。如图 2-10 为短语翻译模型的训练流程。可以分为以下几步：词翻译概率计算、翻译规则抽取和模型参谋估计等。

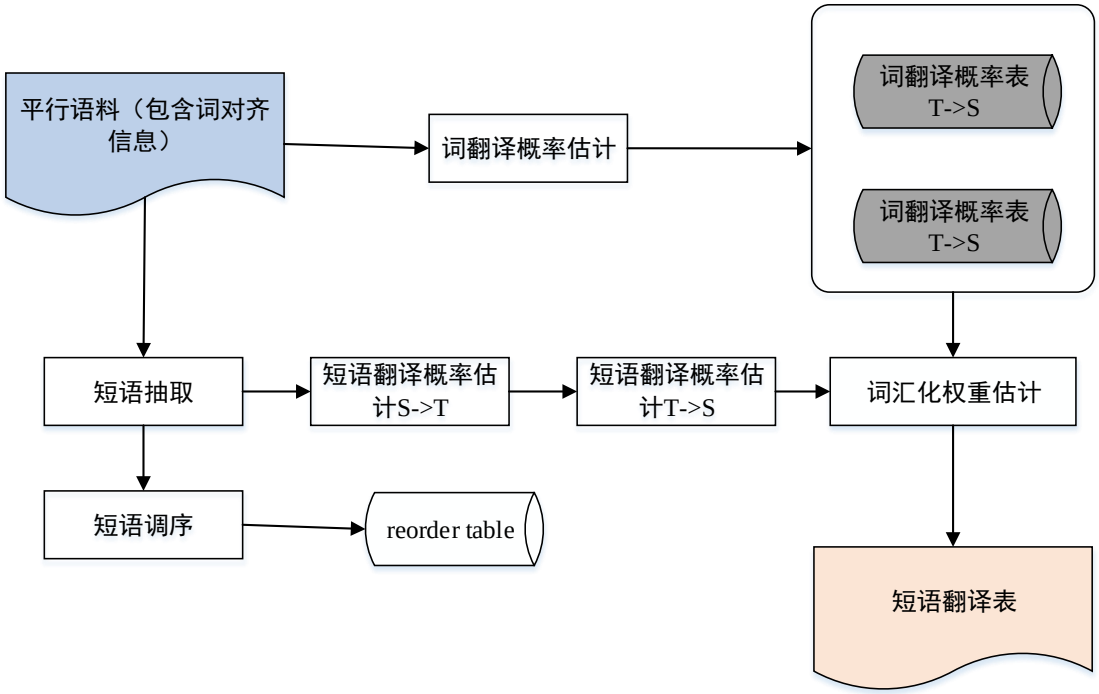


图 2-10 单机短语翻译模型训练流程

首先计算双向的词翻译概率（后续的模型参数估计阶段会使用，S->T 表示源语言到目标语言的词翻译概率表，T->S 表示目标语言到源语言的词翻译概率表）；然后从包含词对齐关系的平行语料中按照指定的匹配策略[42]来抽取短语对。以 2.3.2 节示例中使用的示例结果为例，抽取的短语对形如“少数 国家 ||| one of a few countries ||| 0-0 0-1 0-2 0-3 1-4”。最后计算两个方向的短语对的翻译概率和词汇权重，另外在抽取短语对后可以同时进行短语调序表的训练。

2.5 语言模型

语言模型在统计机器翻译和其他自然语言处理研究方向都占据十分重要的地位。语言模型反映了一个句子产生的可能性和合理程度。例如英语中“One of a few countries”肯定要比“One of countries a few”出现的概率更大。在统计机器翻译系统中训练目标端语言模型主要有两个目的，一是选择正确的翻译顺序，如上述示例中会选择概率更大的翻译顺序；二是翻译规则的选择，若一个词与多个词存在翻译关系时，语言模型就需要根据现有的上下文等信息选择更加符合语境的翻译。语言模型是对整个句子来构建概率模型，其中 N-gram 语言模型是一种广泛使用的模型，其通过估计相互近邻的单词出现的概率来构建模型，N 就是连续 N 个词。N-gram 的数学依据是马尔科夫链假设，其将估计一个句子的概率分解为预测每个词出现的条件概率。形式化地，给定词序 $W = w_1, w_2, ..., w_n$，根据链式法则其出现的概率分解为式（14）：

$$P(\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_n) = P(\mathrm{w}_1)\mathrm{P}(\mathrm{w}_2 \mid \mathrm{w}_1)...\mathrm{P}(\mathrm{w}_n \mid \mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_{n-1}) \quad (14)$$

由马尔科夫假设，一个词出现的可能性只与前面 n-1 词相关联，表示预测一个词的概率需要用到前面 n-1 个单词。以 2-gram 示例一个句子的概率为式（15）：

$$P(\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_n) \approx \prod_{i=1}^{n} P(\mathrm{w}_i \mid \mathrm{w}_{i-1}) \quad (15)$$

N-gram 语言模型中计算一个词产生的概率，同样使用最大似然来进行估计。不过由于数据稀疏性有些 N-gram 在训练集没有出现，而在测试集中出现，从而导致整个句子的概率为 0。因此在 N-gram 语言模型中对概率进行平滑。Good-Turing 平滑方法原理上是根据需要估计参数的真实频次来获得期望频次的方法。而 Kneser-Ney 平滑方法和修正的 Kneser-Ney 平滑方法（Modified Kneser-Ney smoothing, MKN）很大程度上解决了 Good-Turing 平滑对频次为 0 的高阶 n 元文法给予相同概率的缺陷，并且还能够有效地结合低价文法的信息，对不同的 n 元文法有更好的区分度。因此后面两种平滑方法更为常用。使用 MKN 平滑进行估计，如式（16）：

$$P_{MKN}(\mathrm{w}_i \mid \mathrm{w}_{i-n+1}^{i-1}) = \frac{\max\{\mathrm{C}(\mathrm{w}_{i-n+1}^{i}) - \mathrm{D}(\mathrm{C}(\mathrm{w}_{i-n+1}^{i})), 0\}}{\mathrm{C}(\mathrm{w}_{i-n+1}^{i-1})} + \beta(\mathrm{w}_{i-n+1}^{i-1}) * P_{MKN}(\mathrm{w}_i \mid \mathrm{w}_{i-n+2}^{i-1}) \quad (16)$$

其中 $C(\mathrm{w}_{i-n+1}^{i})$ 表示 N-gram 出现的频次，$\beta(\mathrm{w}_{i-n+1}^{i-1})$ 称之为回退权重，而

$D(C(w_{i-n+1}^{i}))$的计算如式（17），D 是 N-gram 中的常量。

$$D(c)=\begin{cases}0, if\ c=0\\ D_1, if\ c=1\\ D_2, if\ c=2\\ D_3, if\ c>2\end{cases},\quad D_1=\frac{n_1}{n_1+2n_2},\quad D_2=1-2*D_1*\frac{n_3}{n_2},\quad D_{3+}=1-2*D_1*\frac{n_4}{n_3} \tag{17}$$

其中，N_c的计算如式（18）：

$$\begin{aligned}N_1(w_{i-n+1}^{i-1}\bullet)&=|\{w_i:C(w_{i-n+1}^{i})=1\}|\\ N_2(w_{i-n+1}^{i-1}\bullet)&=|\{w_i:C(w_{i-n+1}^{i})=2\}\\ N_{3+}(w_{i-n+1}^{i-1}\bullet)&=|\{w_i:C(w_{i-n+1}^{i})>2\}\end{aligned} \tag{18}$$

后退权重$\beta(w_{i-n+1}^{i-1})$的计算如公式（19）所示：

$$\beta(w_{i-n+1}^{i-1})=\frac{D_1N_1(w_{i-n+1}^{i-1}\bullet)+D_2N_2(w_{i-n+1}^{i-1}\bullet)+D_{3+}N_{3+}(w_{i-n+1}^{i-1}\bullet)}{C(w_{i-n+1}^{i-1})} \tag{19}$$

公式（16）是个递归计算式，其终结与一元文法，如公式（20）所示。

$$\begin{aligned}P_{MKN}(w_i)&=\frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}\\ N_{1+}(\bullet w_i)&=|\{w_{i-1}:c(w_{i-1}w_i)>0\}|\\ N_{1+}(\bullet\bullet)&=\sum_{w_i}N_{1+}(\bullet w_i)\end{aligned} \tag{20}$$

在计算 MKN 平滑时，是从低阶文法开始计算，即先计算 1 元文法，再计算 2 元文法，依次类推。参数概率平滑方法虽然在一定程度解决模型的稀疏性问题，然而也增加了模型的计算复杂度。Brants 等人[13]通过实验表明训练语料集规模的增多，复杂概率平滑算法对模型性能的提升不再明显，因此他们提出一种相对简单的称之为 Stupid Backoff 的平滑方法，如式（21）：

$$P(w_i\mid w_{i-n+1}^{i-1})=\begin{cases}\frac{C(w_{i-n+1}^{i})}{C(w_{i-n+1}^{i-1})}, & if\ \ C(w_{i-n+1}^{i})>0\\ \alpha*P(w_i\mid w_{i-n+2}^{i-1}), & \text{otherwise}\end{cases} \tag{21}$$

其中α为回退因子，实验中取经验值$\alpha=0.4$。语言模型广泛使用 Perplexity[43]来自动评价语言模型的好坏，Perplexity 值越小表明语言模型越好，反之越大则表明语言模型越差。

2.6 大数据处理平台

本节主要介绍本文中使用的两种大数据处理平台：分布式文件系统和分布式数据并行化计算平台。

2.6.1 分布式文件系统 HDFS

分布式文件系统 HDFS[44]（Hadoop Distributed File System，HDFS）是如今大数据生态中重要组成部分，提供用户海量数据的存储功能，并且还能够为分布式应用提供快速的数据存取速度和可靠的容错保障。HDFS 中对数据进行分区，将数据分片存储在集群中的多个节点，并使用冗余副本在多个节点备份（默认副本数为 3）。这样做的目的主要有：1. 使用多副本来加速大数据应用中的数据存取速度。2. 保证不会由于集群中某些节点的宕机而丢失数据。3. 判断数据在网络传输时的正确性。

如图 2-11 为 HDFS 的主要架构图。主要包含 Namenode 和 Datanode 两个进程。Namenode 进程存在 Master 机器上，负责分布式文件系统中所有元数据（如整个系统的文件目录结构，文件名字与数据分块的映射关系等）和操作日志。Datanode 进程运行在 Slave 机器上，提供实际的数据管理和应用读写请求。

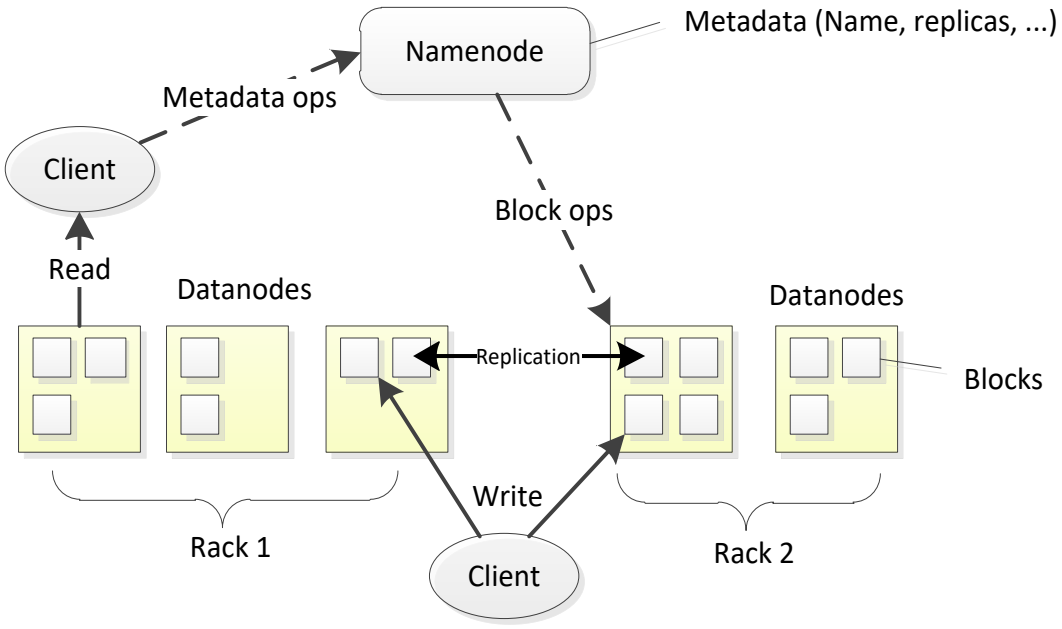


图 2-11 HDFS 框架图

2.6.2 Spark 计算框架

Apache Spark 是 UC Berkeley AMP Lab 开源的分布式数据并行计算平台。其中弹性分布式数据集 RDD（Resilient Distributed Dataset）[12]是 Spark 的核心。用户可以选择将多次使用的 RDD 持久化存储到内存，同时 Spark 中的 DAG (Directed Acyclic Graph, DAG)执行逻辑尽可能避免了作业迭代过程中对文件系统的多次读写，提高了分布式应用程序的并行化计算效率。Spark 相比 MapReduce[11]框架，提供了更加丰富的编程接口。Spark 的这些特点使得其更加适合类似于机器翻译或者机器学习这种任务。

Apache Spark 采用主从式架构，主要由 Master 和 Worker 两部分组成。图 2-12 所示为 Spark Job 的运行流程，Job 运行过程中存在若干进程。Driver 是执行应用程序的入口，来负责程序执行过程中的协调调度，Driver 启动后会连接 Spark 的 Master 节点，将 RDD 的转换[1]（transformation）和操作发送到每个 Worker 节点。Worker 进程是运行在每个计算节点，在从节点上可以根据计算资源情况启动多个 Executor 进程。Executor 进程执行调度分发的 task，Executor 中使用一段独立的内存空间来缓存用户指定的数据。Spark 作业执行完成后可以选择将最终的结果写入到 HDFS 中进行持久化存储。

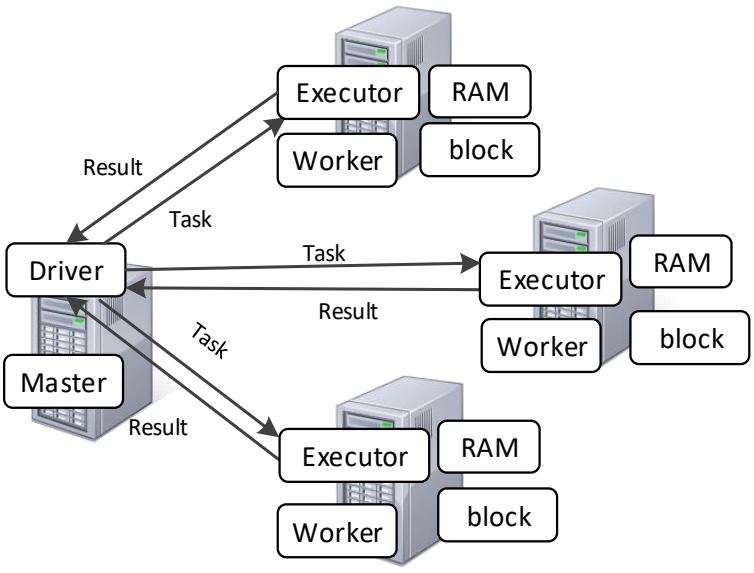


图 2-12 Spark job 运行流程

1. http://spark.apache.org/docs/2.1.0/programming-guide.html

2.7 本章小结

本章开始介绍了统计机器翻译的经典框架，说明了统计机器翻译系统的组成和离线模型训练的一般训练流程，介绍了常用的最大似然估计方法。然后介绍了词对齐模型的基本方法，以及现有的单机词对齐模型工具和词对齐模型的训练示例。接下来介绍了三种不同的翻译模型。随后介绍了 N-gram 语言模型以及一些概率平滑方法。最后简单介绍了本文中的使用分布式文件系统 HDFS 和分布式数据并行计算平台 Spark。

第三章 分布式统计机器翻译离线模型的训练方法

在第二章的理论知识基础上，本章节将对统计机器翻译中离线模型的分布式训练进行详细介绍，并针对每个模型提出了性能优化策略。

3.1 并行化最大似然估计算法

在统计机器翻译模型的训练过程中，模型参数的估计通常使用最大似然估计方法。如词对齐模型中的参数正则化，翻译模型中词翻译概率计算和翻译规则的参数估计，以及语言模型的概率计算等。因此本节将以词翻译概率来说明使用 join 算子的并行化最大似然估计算法。

Algorithm 1： Join-Based MLE Calculate wordTranslationPro

```
1. Input: dataPath is HDFS path of parallel corpus with aligned info
2. Output: wordProbT2S is word translation probabilities
3. Begin
4.      //get statistics of word pair
5.      termPairs ← sc.textFile(dataPath, partNum)
6.                  .flatMap(pair (src ||| tgt, 1))
7.                  .reduceByKey(pair have same key).persist()
8.      // maximum likelihood estimation
9.      edgeCount ← termPairs.mapPartitions(pair (src, count))
10.                 .reduceByKey(pair have same src)
11.     wordProS2T ← termPairs
12.                 .map(pair with the same key)
13.                 .join(edgeCount)
14.                 .map(pair (src ||| tgt, countPair / totalSrc)
15. return wordProS2T
16. end
```

算法 1 为在分布式数据并行计算平台 Spark 上，使用 join 算子计算词翻译概率的并行化最大似然估计算法。算法 1~2 行中指定输入和输出等参数，输入为带有词对齐信息的双语平行语料，输出即为词翻译概率表；第 4 行中算子 textFile 将输入文件从 HDFS 加载到 RDD 中；第 6~7 行来对词对出现频次累加得到 RDD *termPairs*，键值对形如 $(src|||tgt, count)$，并将其 persist 到内存中（因为接下来这个 RDD 需要频繁使用）；第 9~10 行得到源端的边缘频次 RDD *edgeCount*，记录格式为 $(src, totalSrc)$；第 11~14 行中，先对 RDD *termPairs* 的键值对进行调整，记录格式为 $(src, src|||tgt|||count)$，然后使用 join 算子与 *edgeCount* 连接获得 RDD *wordProS2T* 为源端单词翻译为目标单词的概率表。

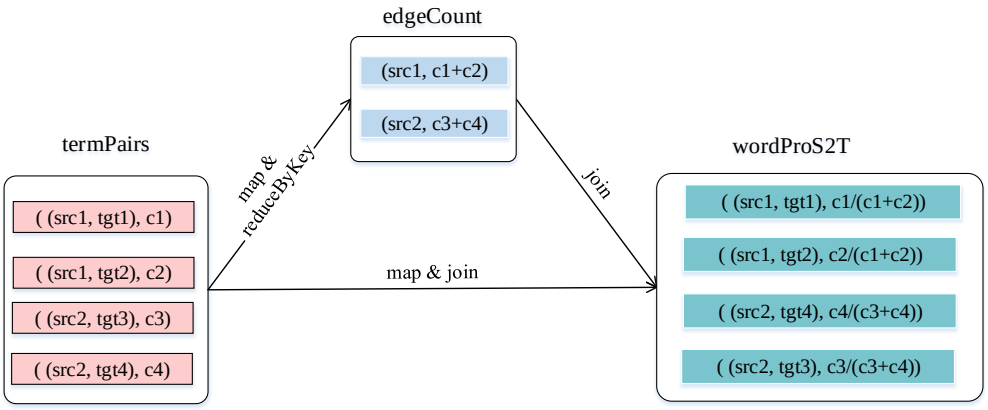


图 3-1 使用 join 算子估计词翻译概率的最大似然估计算法

图 3-1 为使用 join 算子估计词翻译概率的最大似然估计过程的示例图。RDD *termPairs* 作为输入，通过算子 map 和 reduceByKey 得到包含源语言端单词的边缘频次 RDD *edgeCount*，然后改变 *termPairs* 中数据的 key/value 格式，将其和 *edgeCount* 进行 join 就得到最终的词翻译概率表。

3.2 分布式词对齐模型的训练

本文中分布式词对齐模型训练时预先将单机词对齐工具 MGIZA++安装在每个计算节点。首先对训练语料进行预处理（包括词聚类和按照指定分块阈值生成数据分片）。然后在指定模型训练序列后，第一步在数据分片所在计算节点调用 MGIZA++得到词对齐参数的加权统计量；第二步统一正则化更新所有节点在本轮生成的词对齐模型参数。反复执行以上两步来完成两个方向的词对齐模型训练。最后进行合并两个方向的词对齐结果。如图 3-2 为分布式词对齐模型的训练流程图。

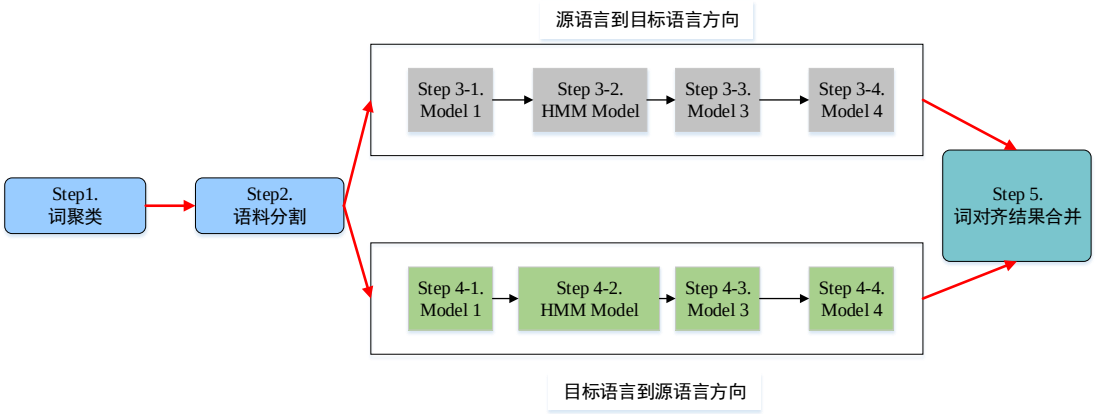


图 3-2 分布式词对齐模型训练过程

图 3-2 给出了分布式词对齐模型训练的具体流程，本文之后会按照此流程进行具体的介绍，整个流程总共包含 5 步。第一步和第二步主要是对 2.3.2 节中介绍的 mkcls、plain2snt.out 和 snt2cooc.out 三个程序的并行化处理，此外对语料数据按照指定的分块阈值大小分片。第三步和第四步中，首先为 MGIZA++ 的调用生成的运行配置文件(设定模型训练序列和输入和输出文件等参数)；然后并行地在每个数据分片所在的 Worker 上调用 MGIZA++ 进行词对齐模型训练；最后按照指定的策略合并两个方向的词对齐信息（同样要为调用 MGIZA++ 生成运行说明文件），默认的合并策略为“ $grow-diag-final$ ”。

3.2.1 预处理和语料分割

本节对 2.3.2 节中介绍的 mkcls 程序的处理过程进行并行化处理，图 3-3 为源语言方向的词聚类流程，同样可以获得目标语言方向的中间结果。

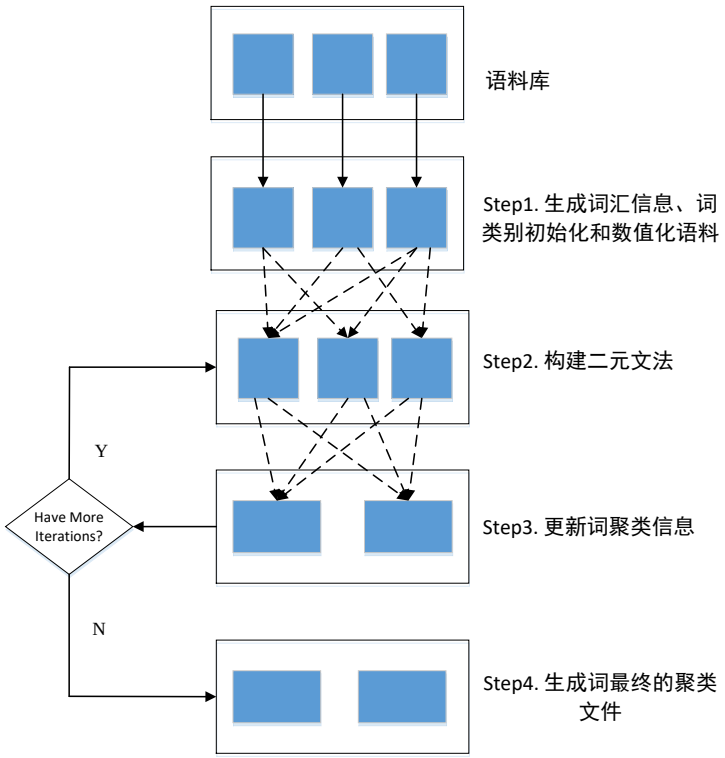


图 3-3 单向词聚类处理流程

（1）生成词汇信息、词类别 ID 初始化和语料数值化。首先得到词频统计 *WordCount*，格式为 $(srcWord, count)$。然后对词出现频率排序，按照频率大小对每个词编号和初始化类别 ID，得到两个 RDD *srcID* 和 *clsMap_0*，格式分别为：$(srcWord, srcId)$ 和 $(srcId, clsId, count)$， *srcId* 为词编码， *clsId* 为词类别编码。最

后用上面得到的词典对语料库数值化，得到 RDD *srcDcorpus*。

（2）构建二元文法。利用上一步生成的 *clsMap_0* 和 *srcDcorpus*，来生成 *2-gram* RDD，键值对格式为 $(\text{srcId1}|||\ srcId2\ |||\ clsId2, count)$。

（3）更新词聚类信息。利用上一步得到的 *2-gram* 对词的类别信息进行更新。这里按照 Franz 等人[37]提出方法对词的类别信息进行更新，得到更新后的词聚类信息 *clsMap_1*。反复执行（2）、（3），直到满足终止条件。

（4）生成词聚类文件。使用 join 连接(2)、(3)中的 RDD *srcID* 和 *clsMap_3* (假设迭代次数为 3）得到最终词聚类 *srcClasses*，格式为 word ||| clsId。

同样，本文对 2.3.2 节中的 plain2snt.out 和 snt2cooc.out 也进行了并行化训练。考虑到 MGIZA++ 为单机工具，若语料规模较大会出现训练耗时较长等问题。本节对训练语料按照指定的阈值 splitLimit 进行分割，一方面可以加快单个节点上模型训练速度，另一面可以利用数据并行优势进行并行化训练。如图 3-4 所示为本文中语料分割流程图，完成后得到所有的数据和分片数目 SplitNum。

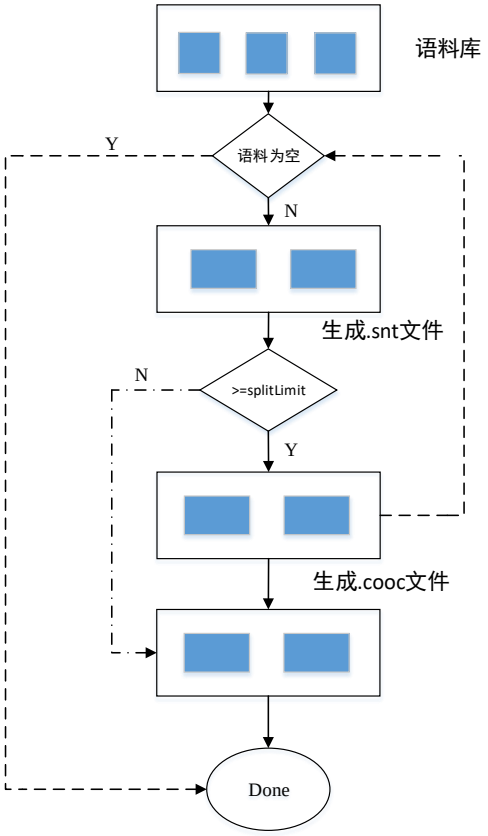


图 3-4 语料分割流程图

3.2.2 两个方向的词对齐模型训练

如图 3-5 是分布式词对齐模型调用框架图。需要预先设定词对齐模型的训练参数，如模型迭代训练序列等。多个词对齐模型的默认训练顺序是 1→HMM→3→4，比如给定词对齐模型迭代序列 111*11*HHH*HH*H*33*3*4*4 (其中数字 1 表示 IBM Model 1，出现三次表示这轮需要迭代三次，字母 H 表示 HMM Model，数字 3 表示 IBM Model 3，数字 4 表示 IBM Model 4，字符*表示进行参数的并行正则化更新)。本文 2.3 节介绍过使用 EM 算法进行词对齐模型的训练流程图，分布式词对齐模型训练同样使用 EM 算法流程处理。如图 3-5 所示，每轮 E-Step 并行计算平台 Spark 在语料分块所在机器调用 MGIZA++ 获取隐含变量词对齐当前轮次的参数加权统计量。这样做的目的有两个：1. 运用数据并行计算平台在数据并行上的特点，在多个节点训练来提高程序的并行度；2. 最大程度地发挥 MGIZA++ 多线程处理优势来提高模型的训练效率。在 M-step，将上一步在每个计算节点产生的词对齐参数估计值从本地加载到 HDFS 进行统一的参数正则化。如此反复执行以上两步迭代地完成指定的模型训练序列。

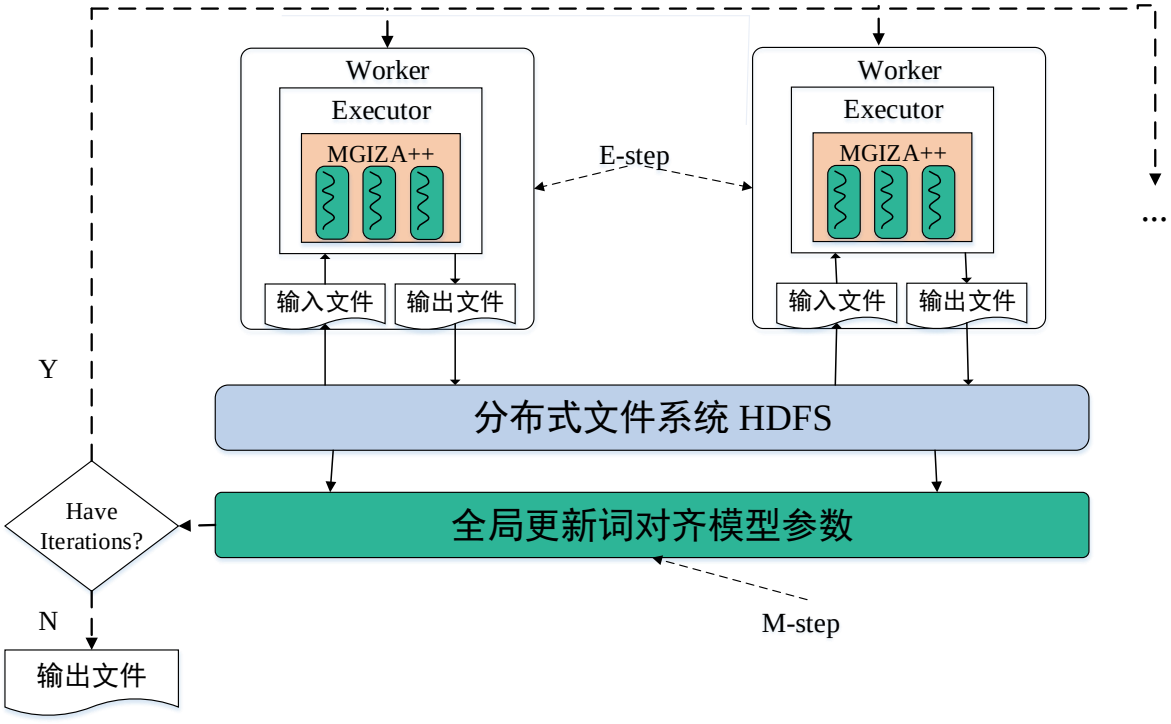


图 3-5 分布式词对齐模型调用架构图

在每个分割后的语料分区训练词对齐模型之前，需要事先为调用 MGIZA++ 生成运行时配置文件（MGIZA++ 需要知道输入文件，训练那些模型，使用 CPU

核数，输出文件信息等参数）。算法 2 为调用 MGIZA++ 生成运行配置文件。

Algorithm 2： buildRuningConfig

```
1. Input: corpusInfo(contains splitNum)
2. Output: running config file
3. function buildRuningConfig
4. begin
5.      tasks ← Vector[String] //分区说明
6.      for i from 0 to splitNum-1
7.          cmdsheet ← (new CommandSheet())
8.          .setExecutable(mgizabinary).setConf()//MGIZA++执行参数
9.          .setInputFile().setOutputFile()//输入输出参数
10.         tasks[i] ← encodeObject(cmdsheet)//编码
11.     end
12.  return tasks
13.  end
```

其中第 1 行指定语料分割信息，如存储路径、分区数目等；第 2 行为输出配置文件的路径；第 7 行为配置文件生成调用命令对象；第 8-9 行设定 MGIZA++ 中二进制文件的路径及运行时参数；第 9 行设定 MGIZA++ 的输入文件路径和输出路径；第 10 行对执行命令对象进行编码，生成当前分区的运行时配置文件。之后 Spark 在调用 MGIZA++ 时，使用 decodeObject(scheduleInfo)进行解码即可获取 MGIZA++ 的运行参数。第 12 行返回所有数据分配的运行时配置文件。

Algorithm 3： *Invoking_Mgizapp*

```
1. Input: corpusInfo, config File
2. Output: unidirectional alignment result
3. function Invoking_Mgizapp
4. begin
5.   arr ←trainingSequence.split("\\*")
6.   while(i ← arr.length)
7.       tasks ← buildRuningConfig (splitNum,arr[i])//调用执行说明
8.       counts = spark.parallelize(tasks, splitNum)
9.          .partitionBy(new HashPartitioner(splitNum))//重分区
10.           .foreachPartition(part.foreach(elem => {
11.             val cmd = decodeObject(elem).asInstatnceOf[CommandSheet]//说明解码
12.             externalOpts.callMgizapp(cmd)   }))
13.      new Normalize(tasks).normalize(counts, parallism, src2tgt)//Normalize
14.      i+=1
15.     end while
16. end
```

算法 3 为 Spark 调用 MGIZA++ 的整个过程，算法第 1~2 行指定语料文件的

路径、生成的配置文件信息和模型训练结束后存储结果的位置。第 5 行分析模型的训练序列，第 7 行调用函数 *buildRuningConfig* 为数据分片生成运时配置文件。第 8 行通过算子 parallelize 将任务序列加载到 RDD 中。第 9~10 行优化数据分区，使得配置文件均匀分配到计算节点。算法 12~13 行在语料分片所在的分区调用 MGIZA++ 进行词对齐模型的训练，每轮训练完后将得到的结果从本地文件系统拷贝到 HDFS 中。第 14 行加载上一步拷贝到 HDFS 中的词对齐估计值来进行参数的统一并行正则化更新。

3.2.3 词对齐结果的合并

最后合并上面获得的双向的词对齐信息，MGIZA++ 中默认的合并方法为“*grow – diag – final*”详见 2.3.2 节。Spark 调用 MGIZA++ 实施词对齐结果的合并也需要预先生成运行时配置文件。为。与算法 3 类似，不同的是需要指定合并策略、两个词对齐结果的位置等信息。算法 4 来生成执行词对齐合并需要的运行时配置文件。

Algorithm 4： *buildMergeConfig*

```
1. Input: bidirectional alignment info, splitNum
2. Output: mergeConfigFile
3. function buildMergeConfig
4. setSymalbin Location
5. setSymalMethod //设定合并策略
6. analysis(setSymalMethod) //分析合并策略
7. begin
8.     tasks ←Array[String]
9.     for i until splitNum-1
10.        cmd ← (new CommandSheet())
11.            .setExecutable(mgizabinary)
12.            .addInitFile(srcAlign, srcAlignLocal, true) //单向的词对齐
13.            .addInitFile(tgtAlign, tgtAlignLocal, true)
14.            .setConf().setOutputFile() //合并结果
15.        tasks[i] ← encodeObject(cmdsheet)
16.    end
17. return tasks
18. end
```

算法 4 为生成合并双向词对齐信息运行配置文件的过程。第 1 行为双向的词对齐信息和语料分割数目；第 2 行为生成的合并配置文件；第 4 行设定 MGIZA++中执行合并策略可执行文件的路径；第 5 行指定合并策略；第 6 行分析设定词对齐合并策略；第 10~11 行为配置文件生成命令对象并设置 MGIZA++ 的可执行文件路径。第 12 行指定源端语言到目标端语言的词对齐结果信息；第 13 行指定目标端语言到源端语言的词对齐结果信息；第 14 行设定合并后结果的

输出路径。第 15 行对执行对象进行编码，形成当前分区的运行时配置文件。

3.2.4 分布式词对齐模型优化

在 3.2.1 节中对平行语料数据按照分块阈值 splitLimit 来划分训练数据。分块阈值大小设定对词对齐模型的训练耗时起着非常重要的作用。若分块阈值太大，则当前语料上得到的分块数目将越少且每个数据分片较大，此时会导致两个问题：1. 由于程序的并发度不高，无法充分发挥 Spark 数据并行计算平台的优点；2. 过大的数据分片会使得 MGIZA++ 的训练时间更长。若分块阈值太小，则当前语料上得到的分块数目将越多且每个数据分片的较小，此时程序的并发度会较高高同时单个 task 执行的速度也越快。由于 MGIZA++ 和 Spark 通过 HDFS 进行模型参数的交互，越多的分区将导致更频繁的 I/O 操作和网络节点通信开销。本文中对分块大小的设定进行了多次验证，发现在较大的平行语料（大于 GB）上将分块阈值设定为 128 MB（HDFS BlockSize 的大小默认为 128 MB，保持不变），而在平行语料规模较小，为保证并发度将分块阈值设定为 64 MB（或 32MB）。语料分块大小的优化对比实验见第五章的实验部分。

3.3 分布式翻译模型的训练

本节将分别介绍短语翻译模型[15]，层次短语翻译模型[16]和基于句法的翻译模型[17]三种翻译模型的分布式训练。因为这三种翻译模型的分布式训练流程大致相似，因为这里先给出一个翻译模型并行化训练的处理流程，之后的章节将说明每个翻译模型在并行化训练时的不同点。

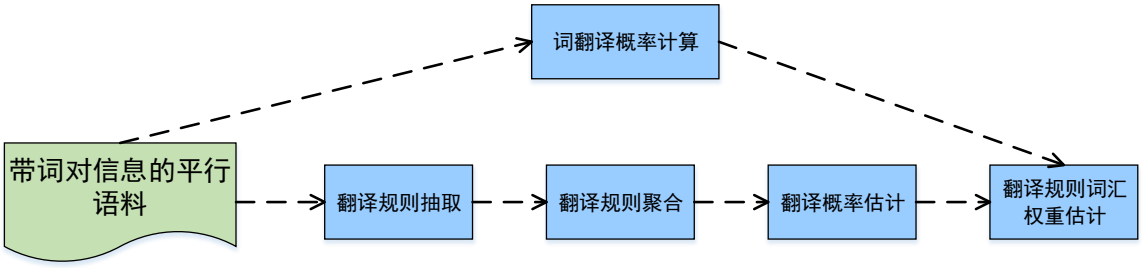


图 3-6 分布式翻译模型训练的处理流程

如图 3-6 为分布式翻译模型训练的处理流程图，包括词翻译概率计算，翻译规则抽取和翻译规则聚合，翻译概率估计和词汇权重估计。

（1）词翻译概率计算。翻译规则的词汇权重估计阶段需要使用词翻译概率

表，3.1 节中说明了并行化最大似然估计算法，在此不重复介绍。图 3-7 为 Spark 上估计词翻译概率的 RDD 世袭流程。图中方框代表 RDD，其中的文字为 RDD 中的数据示例说明，淡红色代表需要缓存到内存中的 RDD，方便后续阶段加载。使用 Spark 算子 saveAsTextFile 将词翻译概率 *wordProS2T* 和 *wordProT2S* 持久化存储到 HDFS，便于后续的阶段使用。

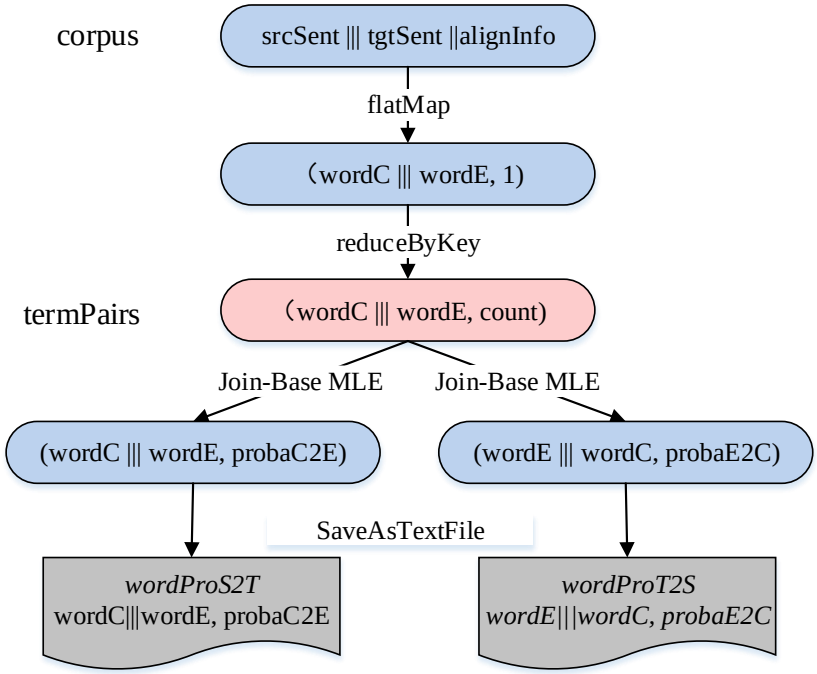


图 3-7 词翻译概率估计流程图

（2）翻译规则的抽取和两阶段聚合。因为平行句对间不存在依赖关系，是互相独立的，所以这里利用 flatMap 算子在多个计算节点并行地抽取翻译则。如图 3-8 为翻译规则抽取和聚合的 RDD 世系图。

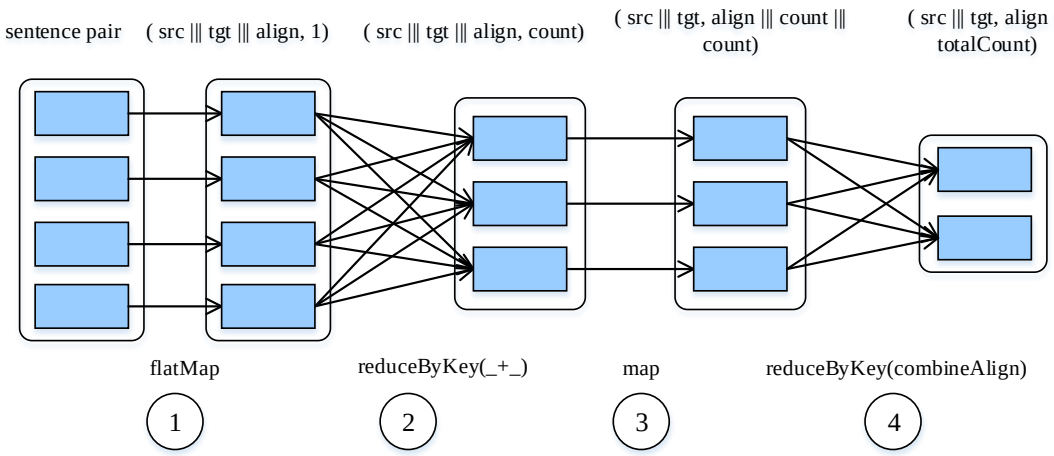


图 3-8 翻译规则抽取与聚合 RDD 世系图

如图 3-8 所示，第一步利用 flatMap 算子按照特定的匹配规则[42]抽取翻译规则，抽取后的翻译规则形式 $(src \mid\mid\mid tgt \mid\mid\mid align,1)$，这里 *src* 为源端翻译规则，*tgt* 为目标端翻译规则，$align$ 为源端翻译规则和目标端翻译规则间的词对齐关系。第二步为第一阶段翻译规则聚合，使用 Spark 中的 reduceByKey 算子聚合所有具有同一源端翻译规则、目标端翻译规则且词对齐关系相同的翻译规则，得到 RDD 中每个记录的格式为：$(src \mid\mid\mid tgt \mid\mid\mid align, count)$。

在抽取出的翻译规则中，源端和目标端规则对一般具有不同的词对齐关系。这里有两种策略：同一翻译规则对保存所有的词对齐信息和同一翻译规则对只保存出现频数最高的词对齐关系。在第一种策略中，保存同一翻译规则对的全部词对齐信息，一方面翻译模型精度的提升微乎其微，另一方面在大规模数据下会造成翻译规则的急剧增加。因此本文中采用第二种策略：同一翻译规则对只保留出现频数最高的词对齐信息。第三步使用 map 操作对上一步中的更改数据表示，得到新的数据格式形式 $(\text{src} \mid\mid\mid \text{tgt}, \text{align} \mid\mid\mid \text{count} \mid\mid\mid \text{count})$；第四步为第二阶段翻译规则聚合，使用 reduceByKey 算子中对相同翻译规则中具有不同词对齐关系的规则对合并。算法 5 为对相同翻译规则对间不同词对齐关系的合并过程。

Algorithm 4： *combineAlign*

```
1. Input: rule A and rule B are values of translation rules with different alignments
2. Output: combined results
3. function combineAlign
4. begin
5. val value = ruleA.totalCount + ruleB.totalCount
6. if ruleA.maxCount > ruleB.maxCount
7.     return ruleA.align ||| ruleA.maxCount ||| value
8. else
9.     return ruleB.align ||| ruleB.maxCount ||| value
10. end if
11. end
```

对两个键值对进行分析，保存出现频数最高的词对齐信息，得到新的数据形如 $(src \mid\mid\mid tgt, align \mid\mid\mid \max Count \mid\mid\mid totalCount)$，$totalCount$ 表示相同翻译规则不同词对齐信息的累加频次，$\max Count$ 表示相同翻译规则对中出现最多的词对齐关系的频次，$align$ 表示对应出现频次最多的词对齐信息。

（3）翻译概率估计与翻译规则词汇权重计算。翻译规则概率估计使用最大似然方法，3.1 节已介绍了基于 join 的最大似然的并行化。翻译规则词汇权重估计依赖词翻译概率表。首先将词翻译表广播到每个节点，在每个节点获取广播的

词翻译概率表；然后用 mapPartitions 算子和词汇化权重公式来估计词汇权重；最后生成包含翻译规则对、翻译概率以及词汇化权重等信息组成的翻译规则表。

3.3.1 短语翻译模型

抽取短语对时，首先设定抽取短语的必要参数，之后使用特定的匹配规则来并行地抽取短语对；图 3-9 所示是 Spark 抽取短语规则的 RDD 世袭图，其中每个矩形框标识了 RDD 中的数据示例说明，其中的灰色方框表示 RDD，其中的文字为数据示例说明。经通过短语抽取和短语聚合阶段之后的翻译规则。

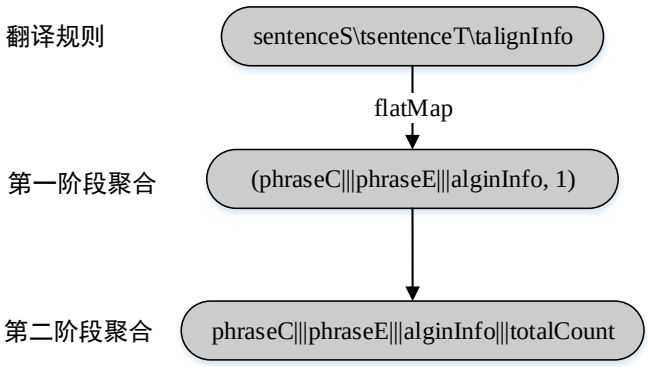


图 3-9 短语对抽取与聚合 RDD 世系图

短语翻译概率需要估计两个方向，翻译概率的估计同样使用最大似然估计方法。图 3-10 为短语翻译概率估计与词汇权重估计的 RDD 世系图。图中每个矩形中的示例了对应 RDD 中的记录格式说明，红色的方框为缓存到内存中的 RDD，便于后续阶段使用。

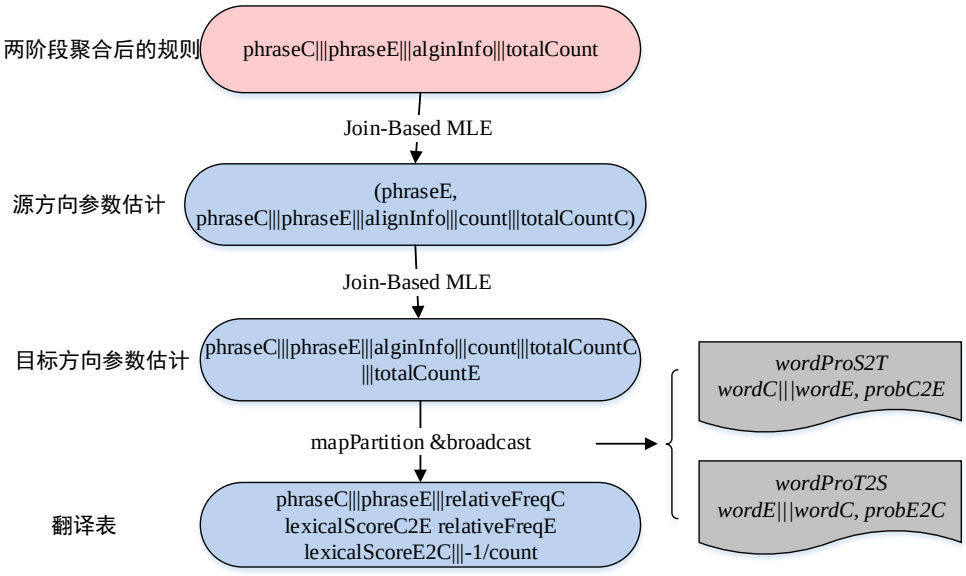


图 3-10 短语翻译模型参数估计流程图

图 3-10 为短语对词汇权重的估计示 RDD 世系图。其中对两阶段聚合后的翻译规则使用 mapPartitions 算子来完成双向的词汇化翻译权重的估计。本文中加载词翻译等共享参数的方式有两种，第一种是通过广播的方式在每个计算节点共享参数；第二种通过读写 HDFS 获取参数，即从 HDFS 中读取参数。最后生成的源端到目标端的翻译规则表，表中的数据说明和 2.4.1 节介绍的一样，共有 5 个概率值，这里不再介绍。

3.3.2 层次化短语翻译模型

层次化短语翻译模型的训练同样适用于 3.3 节中提到的分布式翻译模型的训练流程。因为层次短语翻译模型的分布式训练过程与短语翻译模型大致相似，下面仅介绍不同之处。

（1）翻译规则抽取。抽取的翻译单元是层次化短语。。从包含词对齐信息的平行语料中挖掘符合要求的规则集合主要有两个环节：第一个环节，从平行句对中挖掘符合要求的规则对；第二个环节，搜索所有包括小短语的规则，使用非终结符表示长度较短的小短语。

（2）翻译规则参数估计。与短语翻译模型参数概率计算类似，然而层次化短语翻译模型中抽取出来的翻译规则的规模要比短语翻译模型中抽取的短语规模大很多。表 3-1 为在 LDC 语料集上短语翻译模型和层次化短语翻译模型中规则聚合前的结果集的规模和两阶段压缩效果对比。

表 3-1 短语和层次短语两阶段压缩效果对比

句对数 (百万)	**短语 (maxLen = 3， GB)**			**层次（cutoff=1， GB）**		
	聚合前	聚合后	压缩比	聚合前	聚合后	压缩比
2	1.5	1.0	0.66	19	4.8	0.25
4	3.5	2.4	0.68	49	12.1	0.26
6	4.9	3.5	0.71	77	22.4	0.29
8	8.7	5.6	0.64	123	33.2	0.27

从表 3-1 中看出在最大抽取短语长度设定为 3 时，短语翻译模型中未聚合的短语规则大小为几个 GB，其他数据规模下可以发现短语翻译模型中抽取的规则在聚合前后整体规模压缩效果不是很明显。而对于层次化短语翻译模型，尽管在模型训练过程中使用剪枝技术来减少规则数量，但是未聚合的层次短语集的大小

随着训练数据规模的增加呈现出急剧地增长。同时从表 3-1 可以可以看出在两阶段压缩过程中分布式层次化短语翻译模型相较分布式短语翻译模型取得了更好的规则压缩比。

3.3.3 句法翻译模型

本节介绍分布式句法翻译模型的并行化训练。句法翻译模型训练之前使用斯坦福大学提供的 Stanford Parser 来抽取目标端句子的句法依赖和词性标注信息，并将这些信息整合到平行句对中。翻译规则的抽取时和前面介绍的两种模型一样，不过目标端规则添加句法依赖信息。在翻译规则聚合阶段，增加了目标端规则句法依赖信息的聚合。

在 2.4.3 节介绍了对目标端规则句法树信息状态的估计。其在进行概率计算时使用了整个翻译规则对，然而句法树状态的估计仅和目标端的信息有关，引入源语言端规则可能会对状态的估计产生影响，因此对公式（10）进行修改。

$$P(\mathrm{st}_i \mid \mathrm{dt}, \mathrm{C}) = \frac{Count(st_i, dt_e) + P_e(st_i)}{\sum_i (Count(st_i, dt_e) + P_e(st_i))} \tag{22}$$

公式（22）中 dt_e 表示抽取的翻译规则中的目标端，$Count(st_i, dt_e)$ 为给定目标语言规则 dt_e 时，句法树状态为 st_i 的出现次数。由于句法信息非常稀疏，容易导致零概率出现，因此这里对其做了简单的概率平滑处理，$P_e(st_i)$ 为对句法状态 st_i 的平滑因子，其计算为公式（23）。

$$P_e(\mathrm{r}) = \beta * \frac{Count_e(\mathrm{r}) + 0.1}{\sum_r Count_e(r)} \tag{23}$$

其中 β 属于 0-1 之间，实际取经验值 0.2。

3.3.4 翻译模型平滑

对 3.1 节提出的使用 join 算子的并行化最大似然估计算法进行修改后，图 3-11 所示为使用 join 算子的并行化短语翻译模 GT 平滑的 RDD 世系图。起始数据源 *rdd_wordPair*，格式为 $((src,tgt),count)$ 的 key/value 结构，其中 *src* 和 *tgt* 分别表示源语言单词和目标语言单词，count 表示在整个语料库中这两个单词对出现频数和。其他两种概率平滑策略 KN（Kneser-Ney，KN）和 MKN（Modified Kneser-Ney smoothing, MKN）的并行化类似 GT 方法，这里不再赘述。

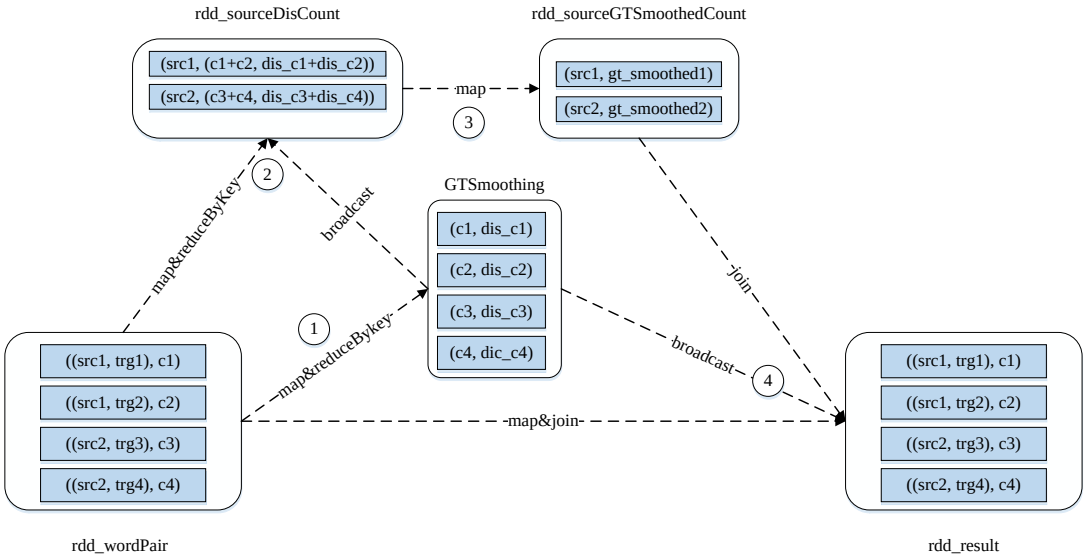


图 3-11 基于 join 算子的 GT 平滑 RDD 世系图

3.3.5 分布式翻译模型优化

本节介绍在训练翻译模型时使用的优化策略，目的是从整体上来减少翻译模型的训练耗费的时间。翻译模型在给定的平行语料数据集上抽取翻译规则时，抽取出的翻译规则集合一般要远大于原始的数据集。一是，3.3 节介绍的翻译规则两阶段压缩、翻译概率以及词汇化权重估计等过程中，存在多次 shuffle 类型的操作，使得集群计算节点间存在巨大的网络通信开销，同时由于翻译规则难以全部 cache 到内存，大量的中间结果需要读写本地磁盘，导致 I/O 和网络通信开销增多。二是，由于翻译模型训练时对原始训练数据集或数据集分区中数据倾斜问题处理不当，容易使得某些计算节点的存在较大延迟，而集群中模型的训练效率又取决于数据分片最大所在的计算节点的处理速度，从而会拖慢整个模型的训练流程。因此本节针对对翻译模型并行化训练时存在的问题提出 3 种优化策略，同时将在 5.3 节在大规模数据集下进行优化效果评估。

（1）对整个平行语料进行数值化处理。在生成词编码时，首先按照词出现的频率进行排序，之后再对词赋予编码（整型数值）。这样做的目的是为频繁出现的词赋予更小的编号，使得出现较多的短语拥有更短的编码长度。然后利用得到的词编码信息对平行语料进行数值化处理。最后在完成整个模型的训练之后，再利用词编码信息对翻译规则进行还原处理。

（2）根据模型训练流程中需要多次估计模型参数的特点，优化使用 join 算子的并行化最大似然估计算法。一般 Spark 中的 join 算子很可能造成数据 shuffle，而在数据并行计算平台上尽可能减少或避免网络和 I/O 开销是非常重要的。本文采取两种优化方案，主要是处理两个场景：一个小表和一个大表的 join 和两个大表的 join (这里的“表”指的是要操作的 RDD)。针对一个小表和一个大表 join 的情况，首先 collect 小表到 driver 端，然后将其广播到每个计算节点进行共享。之后在分布式大表的每个数据分区获取广播的小表来避免全局 join。针对两个分布式大表的 join，主要是对两个待 join 的大表使用相同的自定义 partitioner 方法（或 Spark 中自带的 partitioner）来使得它们内部数据预先满足相同的划分规则，这样两个表中需要连接的记录会预先分配到同一个节点，join 时直接在本地完成计算。首先对两个分布式大表通过 partitionBy 算子应用相同的 partitioner 来产生同一数据分区方式，然后使用 action 操作来触发表中数据的重新划分。最后两个表进行 join 时计算都在本地节点，从而避免执行过程中的数据 shuffle。

（3）数据倾斜优化。由于训练数据分布或数据划分方式导致的数据倾斜问题的存在，本文使用两种策略进行优化。第一种，使用 Spark 并行化编程框架自带的接口 repartition（或使用 repartitionAndSortWithinPartitions，需要重定义分区函数且 RDD 类型必须为 PairRDD，通过重新划分数据分片来使得数据分布更加均匀）来适当提高程序并行度。第二种，使用两阶段聚合策略，首先对原有记录的 key 添加随机前缀（或后缀）进行扩展，使得记录可以均匀地分布到每个计算节点。然后在每个节点做一次局部聚合，最后去掉之前添加的前缀进行全局聚合。通过以上措施，来解决模型训练过程中单个 task 处理数据量过多的问题。图 3-12 为第二种策略的示意图。

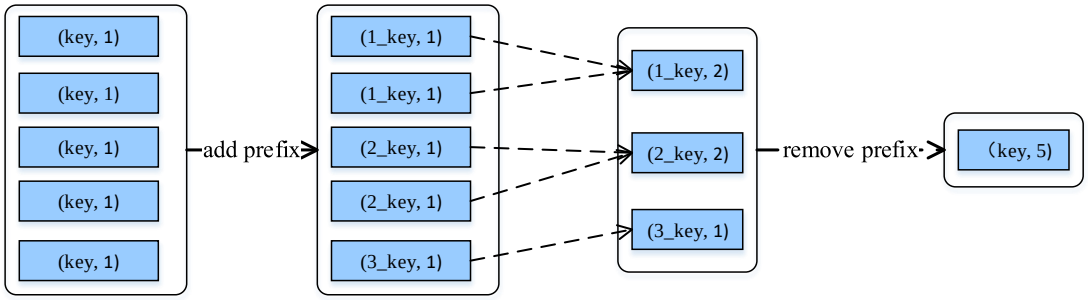


图 3-12 数据倾斜的两阶段聚合策略

3.4 分布式语言模型的训练

本节主要介绍 N-gram 语言模型在分布式数据并行计算平台上的分布式训练流程。如图 3-13 所示为分布式 N-gram 语言模型的训练的流程图。模型构建流程分为三个主要部分。第一词典构建，如图 3-13 中标识的 1-2。第二步：生成 N-gram，如图 3-13 中标识的 3-4。第三步：将平滑方法应用到语言模型中，如图 3-13 中标识的 5-6。本文中对语言模型的平滑使用四种方法，分别是 Good Turing（以下简称 GT）、Kneser-Ney 平滑（以下简称 KN）、Modified Kneser-Ney 平滑（以下简称 MKN）和 Stupid BackOff 平滑（以下简称 GSB），这些平滑方法的介绍见 2.5 节。构建时在对数空间进行语言模型概率估计，这样做的目的有两个：一是避免多次概率相乘造成浮点数下溢；二是加法的运算效率要优于乘法。

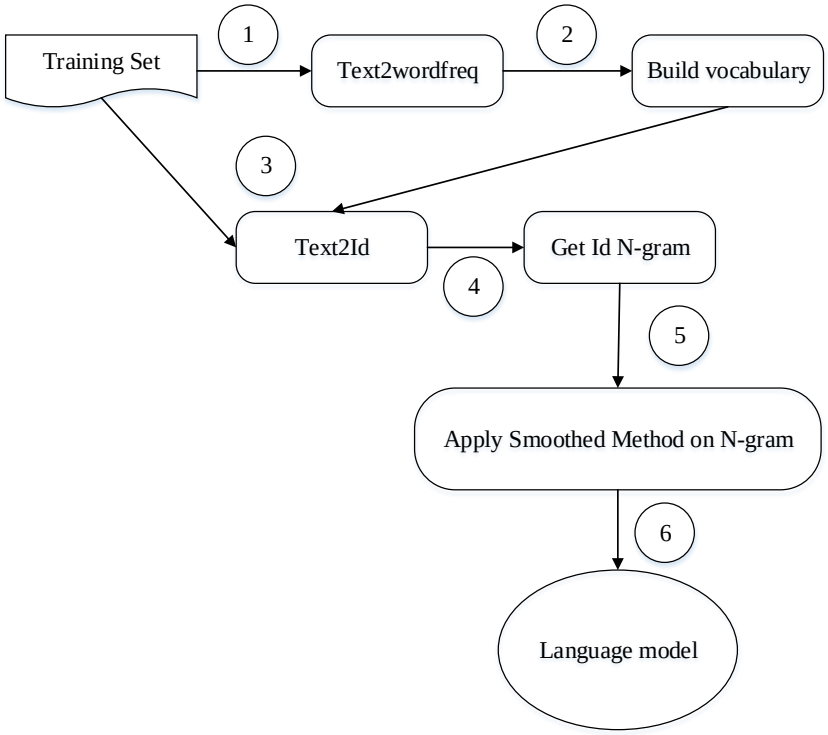


图 3-13 分布式 N-gram 语言模型训练流程图

3.4.1 构建词典和 N 元文法生成

N-gram 语言模型中构建词典主要目的为对训练集合进行数值化。这样使得之后生成的 N-gram 为 ID（整型数组）字符串，尽可能对中间结果进行压缩。在对词赋予编号之前，先按照词出现的频率对其进行排序，之后按照频率进行编号。对频繁出现的词赋予更小的编号，就能够使得频繁出现的 N-gram 可以得到有效的变长编码。如图 3-14 为 Spark 构建词典时的 RDD 世系图。

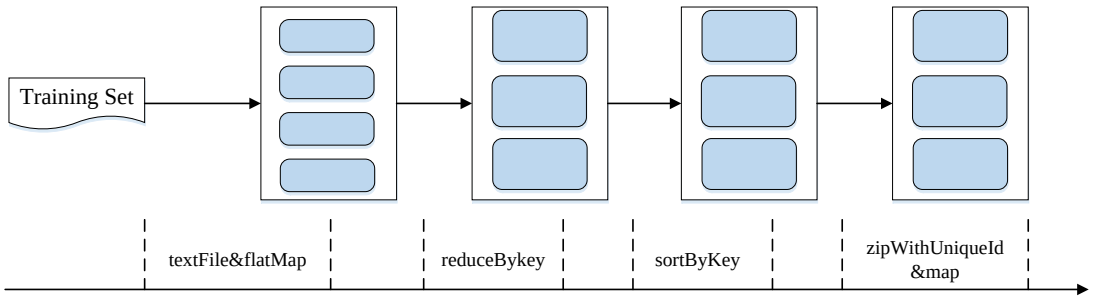


图 3-14 词典构建 RDD 世系图

在抽取之前对每个句子进行简单的处理，在每个句子的句首增加<s>，句尾增加</s>来标识句子的结束和开始，对这两个新增的符号分别赋予编码 0 和 1。首先将词典广播到每个计算节点，然后对训练文本进行数值化处理，最后根据设定的 n 值生成得到 RDD *N-gram*，格式为（n-gram, count）, count 表示这个 n-gram 在训练集合中出现的次数。

3.4.2 语言模型训练

这节介绍如何将 MKN 平滑方法应用到 N-gram 语言模型中，其他三种平滑方法的处理流程与其大致相似，MKN 平滑算法的理论知识在 2.5 节进行了详细介绍。

首先对 2.5 节中的式（16）~（20）进行归类，设定 α 是要统计的模式串。算子 $C(\alpha)$ 为模式串 α 出现的次数，算子 D_n 是一个和 N-gram 相关的统计量，算子 $N_1(\alpha\bullet)=|\{w:C(\alpha w)=1\}|$，表示观测的模式串 α 后面紧接的词只出现一次的个数，同样可以计算算子 $N_2(\alpha\bullet)$、 $N_{3+}(\alpha\bullet)$。

对低阶一元文法的处理稍微有点特殊，式（16）~（20）归纳为算子 $N_{1+}(\bullet\alpha)=|\{w:C(\alpha w)>0\}|$ 表示模式串 α 前面的词的种类个数，其中 $N_{1+}(\bullet<s>)=0$，因为<s>本来就是每个句子的第一个字符。算子 $N_{1+}(\bullet\bullet)$ 为训练集合中所有 2-gram 的种类，是个常量。

这样对 MKN 平滑方法的计算就分割成几个主要算子。首先从低阶文法开始计算，即先计算 1 元文法，在计算 2 元文法，依次类推。如表 3-2 所示，为计算 MKN 平滑是的在不同的计算阶段需要的中间结果，Y 表示需要进行计算，N 表示不需要，其中 $i\in\{1,2,3\}$。

表 3-2 MKN 中不同阶段需要计算的算子

Operator	Lowest-Order	Higher-Order	Highest-Order
$C(\alpha)$	Y	Y	Y
$N_{1+}(\bullet\alpha)$	Y	Y	N
D_i	N	Y	Y
$N_i(\alpha\bullet)$	N	Y	Y

MKN 平滑的算法主要分三步。首先计算最低阶的结果，之后计算中间阶的结果，最后计算最高阶的结果（这样是因为高阶 n-gram 概率的计算依赖于低阶 n-gram 的结果）。以 3-gram 的计算为例进行简单介绍。假设需要计算 $P_{MKN}(w_3 \mid w_1 w_2)$，第一步计算最低阶 $P_{MKN}(w_3)$ 的概率，算子 $N_{1+}(\bullet w_3)$ 使用 *2-gram* 来计算，先用 map 数据格式，即数据格式从 $(w_2 w_3, count)$ 转化为 $(w_3, w_2 w_3)$，然后利用 *reduceByKey* 算子进行聚合统计得到 $N_{1+}(\bullet w_3)$。第二步计算 $P_{MKN}(w_3 \mid w_2)$，其回退权重 $\beta(w_2)$ 利用 2-gram 进行，然后与 $P_{MKN}(w_3)$ 进行 join 得到 $P_{MKN}(w_3 \mid w_2)$。第三步计算 $P_{MKN}(w_3 \mid w_1 w_2)$ 与第二步类似。图 3-15 为计算 MKN 平滑的 RDD 世系图。

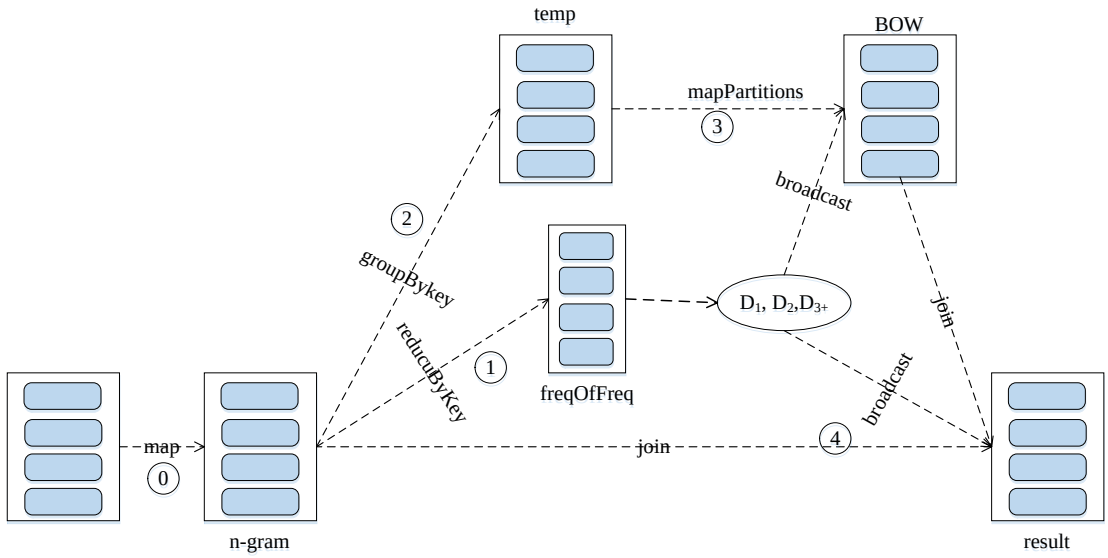


图 3-15 MKN 计算的 RDD 世系图

3.4.3 分布式语言模型优化

语言模型与翻译模型在并行化训练时存在类似特点：需要多次使用并行化最大似然估计算法和容易出现数据倾斜。因此语言模型中采用类似与 3.3.5 节翻译模型优化策略中的（2）和（3）。不过语言模型中平滑概率的计算依赖 w_{i-n+1}^{i} 和

w_{i-n+1}^{i-1}（n≥2），可以发现它们前面两个词是相同。所以对分布式语言模型中使用 join 算子的并行化最大似然估计方法进行优化（3.3.5 节中的优化策略 2），在自定义的 partitioner 中使用前两个词的哈希值来设计分区函数，这样使得 N-gram 在重新划分后可以提高后续参数概率估计阶段的计算效率。语言模型的优化效果在 5.3 节进行实验评估与分析。

3.5 本章小结

本章主要介绍机器翻译离线模型中每个模型的分布式训练流程。首先对介绍多个模型训练中多次使用的并行化最大似然估计算法。然后介绍了分布式词对齐模型的训练方法，包括对训练前的预处理过程和相应的词对齐模型的并行化方法。另外通过设定合理数据分块阈值来进一步提高词对齐模型的并行训练性能。接下来介绍了分布式翻译模型的训练方法，分别讨论了短语翻译模型、层次化短语翻译模型和句法翻译模型的并行化，并针对翻译模型训练时存在的问题提出了三种性能优化策略。最后介绍分布式语言模型的并行方法，重点介绍了 MKN 平滑算法的并行化，为提高语言模型并行训练性能也提出了两种优化策略。

第四章 大规模分布式统计机器翻译系统的设计与实现

在前面的分布式机器翻译模型离线训练的基础上，本文设计和实现了一个完整、高效、弹性可扩展的分布式统计机器翻译离线模型训练的原型系统，并将其命名为 Seal，本章将介绍其主要整体组成框架，模块的工程细节等详细信息以及常用的参数说明。

4.1 系统整体框架

图 4-1 为本文提出的大规模分布式统计机器翻译离线模型训练原型系统 Seal 的整体架构组成图。图中最上层为第三章实现的三个分布式模型训练模块（词对齐模型、翻译模型和语言模型）。中间两层分别为分布式数据并行化计算平台 Spark 和分布式文件系统 HDFS，其中 Spark 为整个系统提供计算支持，HDFS 为整个系统提供数据存储和参数同步，在每个节点上安装本地单机词对齐模型训练工具 MGIZA++ 。

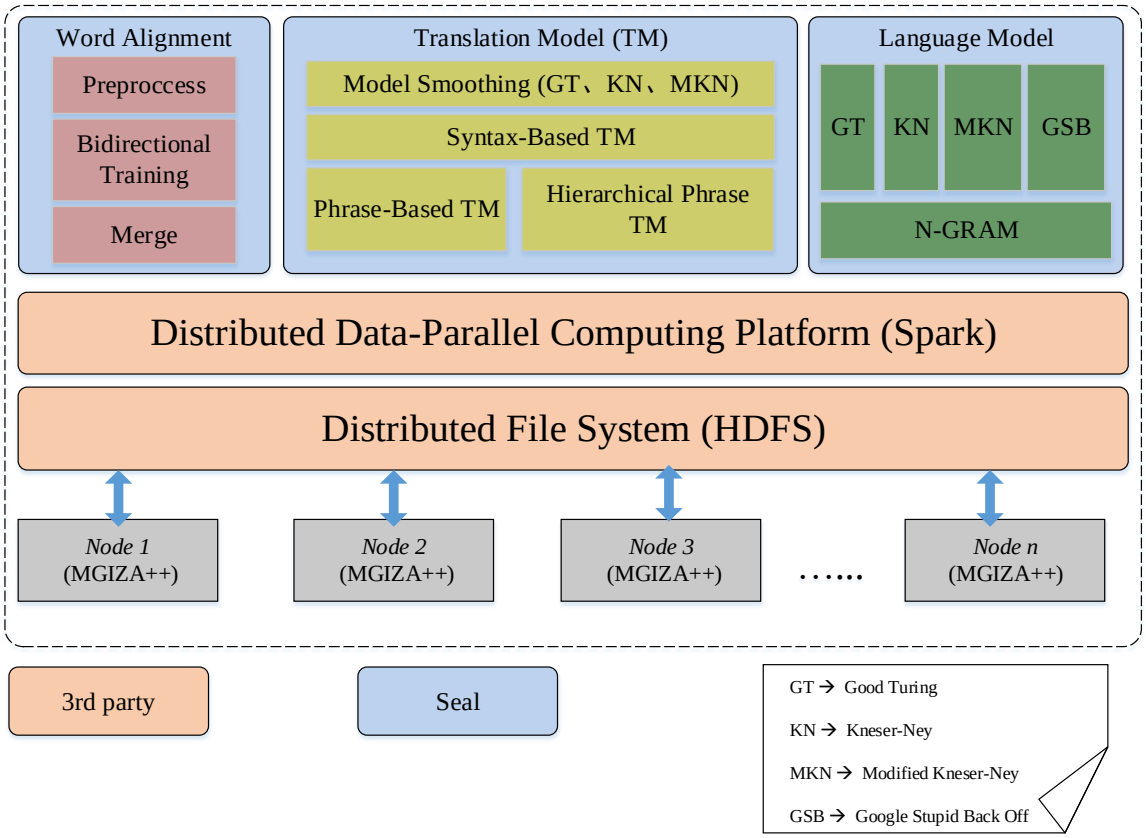


图 4-1 Seal 的整体框架图

在并行化模型训练部分，分布式词对齐模块主要包括三个环节：预处理、双向的词对齐模型和最后的结果合并的并行化训练；分布式翻译模型包括短语翻译模型、层次化短语翻译模型、基于句法的翻译模型和翻译模型的平滑等并行化训练；分布式语言模型可以使用四种不同的平滑算法来并行化训练（2.5 节）。

在本文提出的分布式机器翻译离线模型训练原型系统 Seal 中，分布式持久化存储层采用文件系统 HDFS。Seal 系统中 HDFS 的作用主要有两个：1. 数据的持久化存储。机器翻译模型的输入语料和最终输出结果都保存在分布式文件系统 HDFS 中，此外一些重要的中间结果（如词编码表等）也保存在 HDFS 中，这样便于 Spark 高效地进行模型训练。2. 模型参数更新同步。如本文提出的分布式词对齐模型中，在 E-Step 中，通过数据并行计算平台 Spark 在数据分片所在计算节点调用 MGIZA++ 获取隐含变量词对齐的最大似然估计值；在 M-Step，将上一步在每个计算节点产生的词对齐参数统计值从本地加载到 HDFS 进行统一的参数正则化。

4.2 系统的基础模块功能与接口设计

本小节将介绍 Seal 中的主要功能和基础模块的主要类关系说明。本文在广为使用的数据并行计算平台上构建大规模分布式统计机器翻译离线模型的训练系统，设计目标主要有：构建一个完整的分布式机器翻译离线模型训练系统，提高每个模型的分布式训练效率，提供弹性高效的扩展性。主要包含三个模块：

（1）分布式词对齐模型模块。在词对齐模型训练前需要对双语平行语料进行预处理，得到 MGIZA++ 所需的必要文件（详情见 2.3.2），本文对并行化处理了预处理过程。词对齐模型中将 MGIZA++ 作为词对齐模型训练的中介，通过 Spark 数据分区所在分区调用 MGIZA++，利用 HDFS 对模型参数统一更新。

（2）分布式翻译模型训练模块。Seal 中实现短语翻译模型和层次化短语翻译模型的分布式构建，在这两个模型的基础上通过在目标端（或源端）翻译规则中增加句法依赖信息来构建基于句法翻译模型的并行化训练，此外还提供三种平滑策略优化翻译模型参数。

（3）分布式语言模型训练模块。Seal 中实现了 N-gram 语言模型的分布式构建，同时为了减少数据不足导致的零概率产生，本文在 N-gram 的基础上应用了四种不同的概率平滑方法（平滑方法详情见 2.5 节），Seal 中语言模型并行化训

练时默认的平滑策略为 MKN（Modified Kneser-Ney smoothing, MKN）。

4.2.1 词对齐模型模块设计

图 4-2 为词对齐模型训练中预处理流程部分的主要类关系图。预处理模块有两个接口组成：wordClustering 和 wordAlignPrep，其中 wordClustering 中的方法 mkcls 产生词聚类信息，wordAlignPrep 中的方法 corpusOpts 过滤平行语料中过长的句子（默认长度设定为 100），方法 splitCorpus 会按照指定的分块大小和词聚类信息来生成 2.3.2 节介绍的 snt 和 cooc 文件。

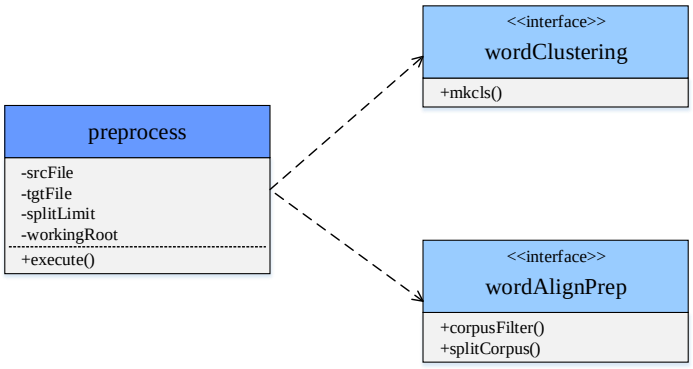


图 4-2 词对齐预处理模块主要类关系图

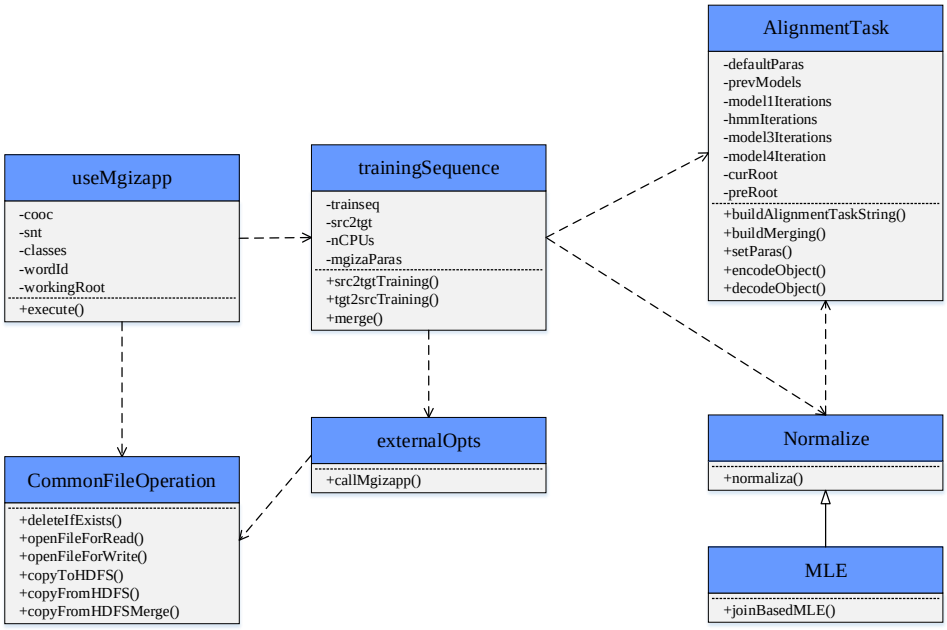


图 4-3 分布式词对齐模型的模块主要类关系图

图 4-3 为 Seal 中词对齐模型的模块主要类关系图。类 useMgizapp 为词对齐

模型训练程序的核心类，使用类 trainingSequence 解析词对齐模型训练序列 trainSeq，再通过类 AlignmentTask 为 MGIZA++ 生成运行时配置文件。而 externalOpts 类中的方法 callMgizapp 来在数据分片所在的计算节点调用 MGIZA++ 进行词对齐模型的训练，获取隐含变量词对齐的最大似然估计值。类 NormalizeTask 将本地的词对齐参数加载到 HDFS 进行统一参数正则化。完成双向的模型训练之后再调用 trainingSequence 类中 merge 方法来合并词对齐结果。

4.2.2 翻译模型模块设计

图 4-4 为分布式翻译模型部分的主要功能类关系图，类 translationModel 为程序的主类，其中可以设定输入和输出的路径、翻译规则抽取的类型、句法类型、规则长度等参数，同时可以指定训练的模型类型（短语翻译模型、层次化短语翻译模型和句法翻译模型）。类 modelTraining 完成翻译规则压缩、句法信息的获取、翻译模型参数的估计，此外还提供参数平滑处理（提供三种平滑方法，见 2.4.3 节），类 modelTraining 主要依赖两个接口，接口 extractFromAlignSentece 从带有词对齐信息的平行句对中来抽取翻译规则，接口 translationUnit 提供模型参数和句法参数的估计。参数的估计为使用 join 算子的并行化最大似然估计算法，在 3.1 节已经进行了详细介绍。

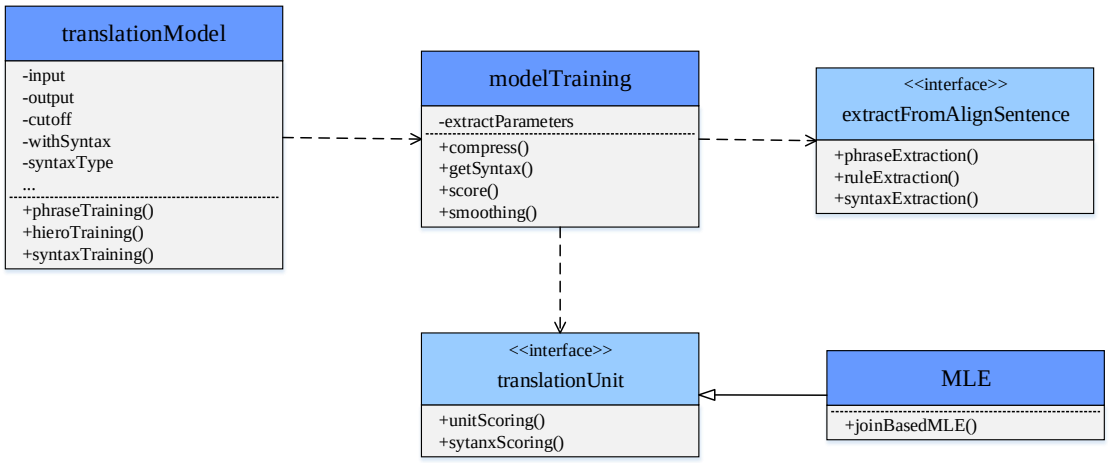


图 4-4 分布式翻译模型的模块主要类关系图

4.2.3 语言模型模块设计

图 4-5 为分布式语言模型的模块主要类关系图。类 languageModel 为程序的核心类，在其中设定输入和输出的路径、N-gram 文法的长度以及参数平滑信息等。类 modelGenerate 提供语言模型的并行化训练。其中方法 buildDict 在单语语

料中来完成词典的构建，方法 genNGram 来生成 N-gram 文法，方法 smoothing 中完成最终模型的生成。类 modelGenerate 依赖两个接口，接口 partitionFun 为自定义的 RDD 分区函数，本文使用 N-gram 文法中的最前面两词来进行分区，可以最大程度保证数据分区分布更为平衡。接口 modelSmoothing 中提供四种不同的参数平滑，默认平滑策略为 MKN（Modified Kneser-Ney smoothing, MKN）。

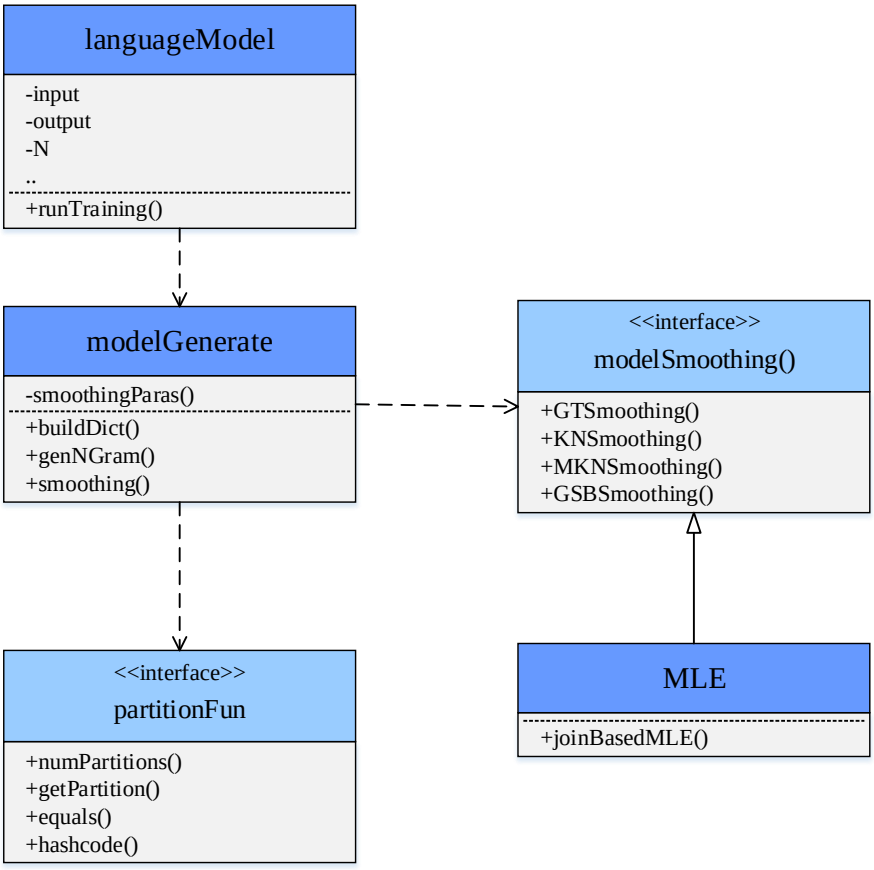


图 4-5 分布式语言模型的模块主要类关系图

4.3 系统参数配置说明

本文提出的离线模型并行训练原型系统 Seal，在使用过程存在较多的训练参数。因此为实现在不同场景、不同模型下的参数灵活配置以及模型选择。本文中使用统一的配置文件来管理所有的模型训练参数，使用 typesafe[1]对配置文件解析来获取参数值。Seal 中的参数类型可以分为四类：语言模型参数、翻译模型参数、语言模型参数和 Spark 本身所需参数。

[1] https://github.com/typesafehub/config

表 4-1 列出 Seal 中主要的参数列表，并对每个参数的含义进行了简单说明，其他不是非常重要的参数未列出。配置文件的语法采用 JSON 格式[1]，每个模块的参数配置是独立的，可以方便修改配置文件来适应不同的模型训练。

表 4-1 Seal 中的主要参数列表

参数名	参数含义
source	源语言文件路径
Target	目标语言文件路径
rootDir	Seal 工作目录
splitLimit	词对齐预处理中分块阈值大小
trainSeq	词对齐模型训练序列
nCPUs	MGIZA++使用的 CPU 数目
mgizabinary	MGIZA++可执行文件的路径
extractType	抽取的翻译规则类型
maxSpan	翻译规则的最大长度
cutoff	翻译规则过滤阈值（可选）
withSyntax	是否需要对规则增加句法信息
syntaxType	句法信息的类型
smoothingType	平滑方法选择
N	N-gram 的长度（马尔科夫阶）
parallelism	Spark 默认的并发度
registrationRequired	是否需要使用 Kyro 序列化方式
compressType	存储文件的压缩格式
javaOpts	JVM 参数

4.4 本章小结

本章主要说明本文提出的大规模分布式统计机器翻译离线模型训练的原型系统 Seal 的整体组成框架，以及设计方面的工程性细节。首先介绍了 Seal 的整体框架图，并对框架中的每个组件进行概要说明。然后介绍 Seal 中的三个模块的主要模块和类关系图进行说明。最后，对原型系统 Seal 中使用的主要参数进行了简单说明。

[1] http://www.json.org/

第五章 实验效果评估

本章主要介绍使用大规模数据集在分布式环境下对 Seal 中三个模型的并行训练性能进行验证与分析。不仅对 Seal 本身的并行化训练性能进行验证分析，也与现有的分布式模型训练工具进行并行化训练性能对比。以下是要进行的实验分析概要。

(1) 5.2 节中将对 Seal 中的每个模型的并行训练性能与现有的分布式机器翻译训练工具进行对比分析。

(2) 5.3 节中将分别评估在大规模数据集下第 3 章中对词对齐模型的优化效果(3.2.4 节)，对翻译模型的优化效果(3.3.5 节)，和对语言模型的优化效果（3.4.3 节)。同时对优化后的每个模型的并行训练性能分别与现有的单机和分布式训练工具进行对比分析，除此之外还需要对优化后每个模型的数据规模可扩展性进行分析。

(3) 5.4 节中将分别评估 Seal 中每个模型的计算节点可扩展性。

(4) 5.5 节将从统计机器翻译离线模型训练的整个流程进行性能对比分析。

5.1 实验设置

在实验数据集的选择上，本文选用包含八百万中-英双语平行句对的 LDC[1] (Linguistic Data Consortium)和包含一千五百万的中-英双语平行句对的联合国平行语料库(United Nations Parallel Corpus, UNPC)[46]。实验数据集以百万行为单位（缩写 M），如 LDC-1 M 表示 LDC 数据集中选取 100 万行来进行测试。本文在实验中使用 12 个计算节点和 1 个主节点组成的分布式环境，每个节点的软硬件信息如表 5-1 所示。

同时为对比在大规模数据集下本文提出的系统 Seal 与现有分布式统计机器翻译模型离线模型训练工具的并行训练性能差异，本文为 Seal 中每个并行化模型选用相应的对比对象（另外本文也选用单机工具，主要是为了能够直观对比分布式模型的训练耗时）。

[1] https://www.ldc.upenn.edu/

(1) 词对齐模型。本文选用分布式词对齐模型训练工具 Chaski 作为性能对比对象，选用单机词对齐训练工具 MGIZA++ 作为参考。

(2) 翻译模型。短语翻译模型选用 Chaski 和基于 MapReduce 框架的 MR-phrase 作为性能对比对象；层次化短语和句法翻译模型分别选用 MR-hiero、MR-syntax 为性能对比对象；选用单机短语翻译模型训练系统 Moses 作为参考。

(3) 语言模型。分布式语言模型选用基于 MapReduce 框架实现的语言模型 MR-LM，选用单机语言模型训练工具 KenLM 作为参考。

表 5-1 计算节点软硬件环境信息

Configuration	**Property**
CPU	2 x Intel® Xeon(R)E5-2620 v2@2.10GHZ 15M cache 6-core 12-Thread
Disk	SAS controller 6TB raid0
Memory	64 GB
Network	Gigabit Network
Operating System	Redhat Enterprise Server 7.0
JDK Version	1.8.0
Spark Version	2.1.0
Hadoop Version	2.7.3
Moses Version	2.0
MGIZA++ Version	0.7.0

5.2 分布式机器翻译模型的并行训练性能对比

本节将在数据集 LDC 和 UNPC 分别对比分析 Seal 中词对齐模型、翻译模型和语言模型的并行训练性能。

5.2.1 词对齐模型并行化性能

图 5-1 为 Seal 和 Chaski 在 LDC 和 UNPC 两个数据集下的词对齐模型并行化训练耗费的时间对比。图中坐标横轴表示不同的数据规模，坐标纵轴表示词对齐模型训练耗时（包括预处理、两个方向的词对齐和最后的合并）。Seal 和 Chaski 的分块阈值大小都采用固定值 512 MB。随着数据规模的增加，分布式词对齐 Seal 的并行训练性能相比 Chaski 平均提高 2~3 倍左右。

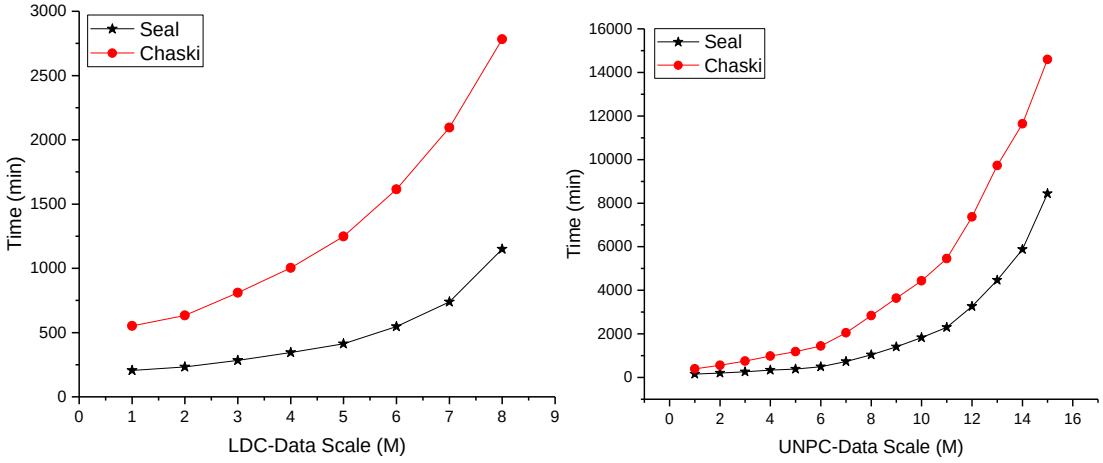


图 5-1 分布式词对齐模型并行训练性能对比

5.2.2 翻译模型的并行训练性能

本节将分别使用 LDC 和 UNPC 数据集下，将 Seal 的分布式翻译模型中短语翻译模型、层次化短语翻译模型和句法翻译模型的模型训练耗时与现有系统进行对比分析。

（1）**分布式短语翻译模型**

图 5-2 为分布式短语翻译模型 Seal-phrase 和现有分布式短语翻译模型 Chaski、MR-phrase 的并行化模型训练耗费时间对比。图中坐标横轴表示不同数据规模，坐标纵轴表示短语翻译模型训练耗费时间。抽取长度设为 10（两端各为 5），Seal 的并行训练性能相比 Chaski 平均提高 3~4 倍，相较 MR-phrase 并行训练性能平均提高 4~5 倍。

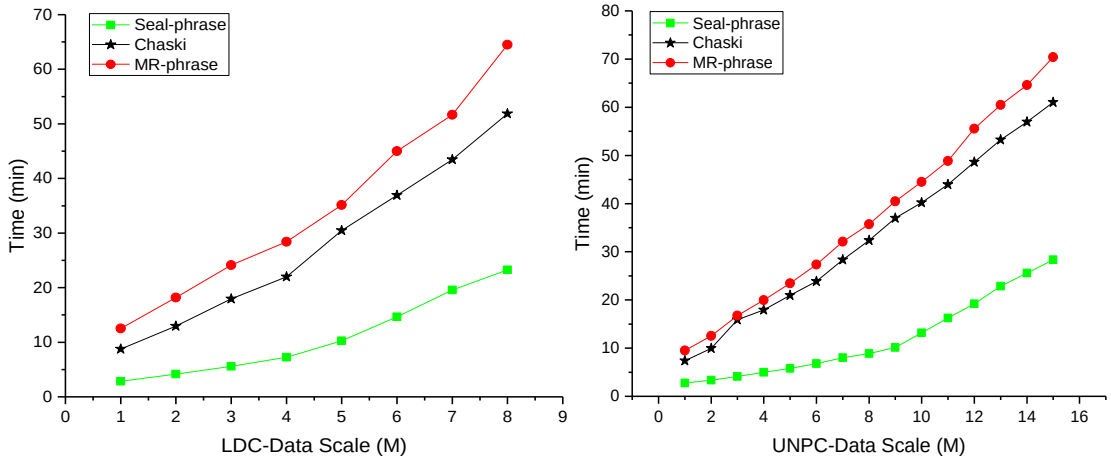


图 5-2 分布式短语翻译模型训练耗费时间对比

（2）分布式层次化短语翻译模型

图 5-3 为分布式层次短语翻译模型 Seal-hiero 与 MR-hiero 的并行化训练耗费时间对比，其他参数与短语翻译模型一样，规则剪枝阈值为 1。图中坐标横轴为不同的数据规模，坐标纵轴表示层次化短语翻译模型训练所用时间。Seal-hiero 的并行训练性能相较 MR-hiero 平均可以达到 7-8 倍的加速比。

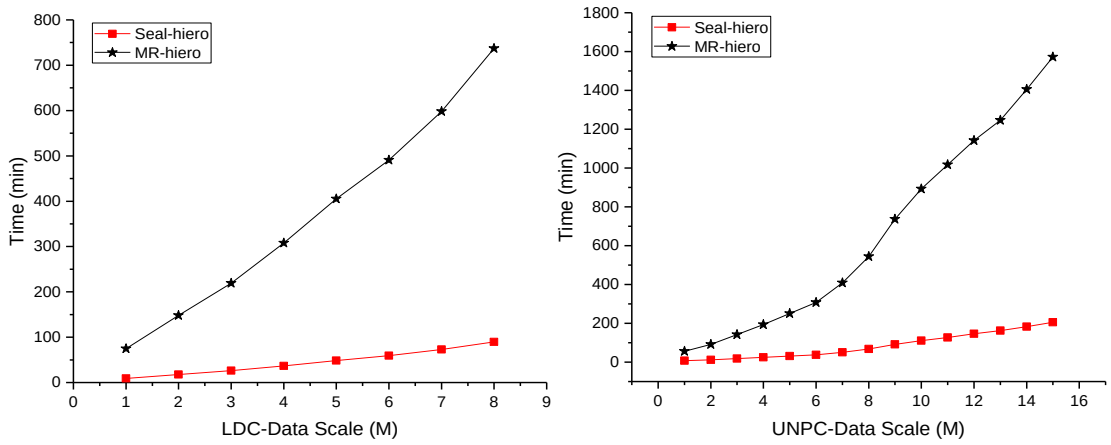


图 5-3 分布式层次化短语翻译模型并行训练性能对比

（3）分布式句法翻译模型

图 5-4 为分布式短语句法翻译模型 Seal-syntax-phrase 与 MR-syntax-phrase 并行训练性能对比。图中坐标横轴表示数据规模的变化，坐标纵轴表示句法翻译模型训练消耗的时间。短语句法翻译模型中，Seal-syntax-phrase 的并行训练性能相较 MR-syntax-phrase 平均提高 4 倍左右。

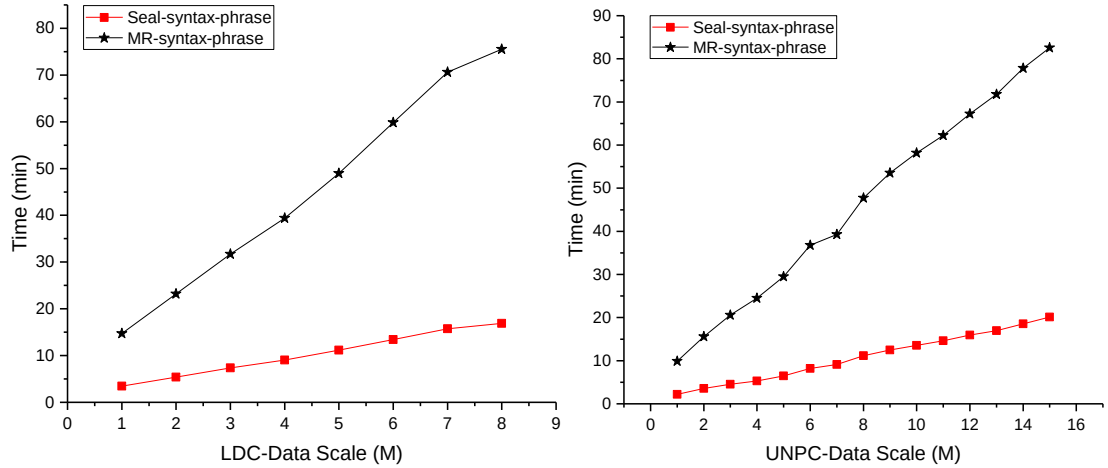


图 5-4 分布式短语句法翻译模型并行训练性能对比

图 5-5 为分布式层次短语句法翻译模型 Seal-syntax-hiero 和 MR-syntax-hiero 的并行化训练耗费时间对比。图中坐标横轴表示数据规模的变化，坐标纵轴表示句法翻译模型训练耗时。层次短语句法方面 Seal-syntax-hiero 的并行训练性能相较 MR-syntax-hiero 能达到平均 6-7 倍的加速比。

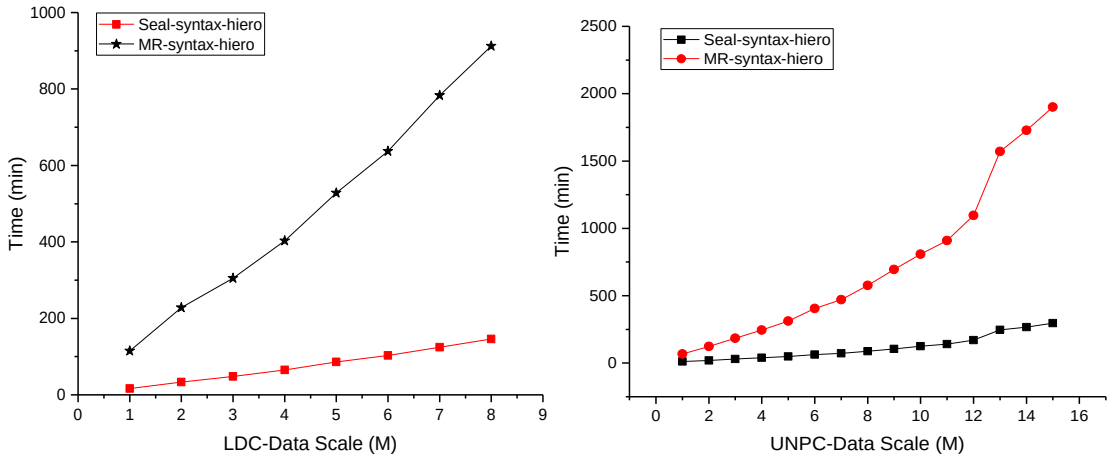


图 5-5 分布式层次短语句法翻译模型并行训练性能对比

5.2.3 语言模型的并行训练性能

图 5-6 为在 LDC 和 UNPC 数据集下（这里选用英语），Seal-LM 与 MR-LM 使用 4 种平滑算法的语言模型的并行训练性能对比。其中 GT 为 Good Turing 平滑，KN 为 Kneser-Ney 平滑，MKN 为 Modified Kneser-Ney 平滑，GSB 表示 Google Stupid Backoff 平滑。图中坐标横轴表示不同的数据规模，坐标纵轴表示模型训练所用时间。Seal-LM 相较 MR-LM 的并行训练性能平均提高 5 倍左右。

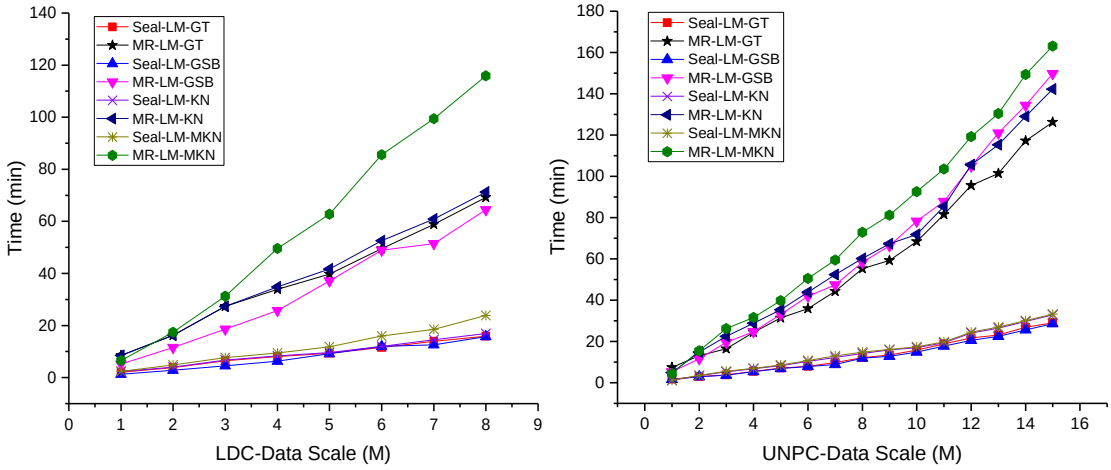


图 5-6 分布式语言模型并行训练性能对比

5.3 分布式机器翻译模型优化效果评估

本节对第三章中分析对每个模型设计实现的优化策略的效果。分别为 3.2.4 节分布式词对齐模型的分块优化策略，3.3.5 节分布式翻译模型的三种性能优化策略，3.4.3 节分布式语言模型的性能优化策略。

5.3.1 词对齐模型分块优化效果评估

图 5-7 为在数据集 LDC-1 M 和 UNPC-5 M（M 表示百万行）下的分块阈值对词对齐模型的并行训练性能影响。图中坐标横轴代表分块阈值，坐标纵轴代表不同分块阈值词对齐模型训练所用时间 (HDFS Block Size 为 128 MB，实验中保持不变)。在一百万行数据规模中分块阈值设定 32 MB 并行化性能最优。在 5 百万行数据集下分块阈值设置为 128 MB 并行化性能最优。

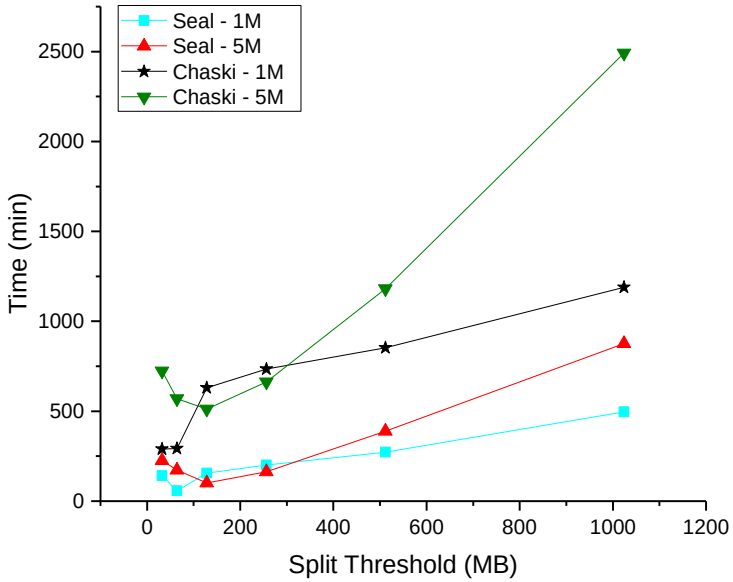


图 5-7 分块阈值对词对齐模型并行训练性能的影响

在 5 百万规模的数据集下，分块大小设定为 128 MB，词对齐模型 Seal 相较 Chaski 性能平均提高 5 倍左右，而其他阈值设定下平均只有 2~3 倍的加速比。分块阈值对并行化词对齐模型的影响主要集中在并发度和单机处理速度两个方面。分块阈值越小分块数目越多，意味并行度越高并且单机处理的速度越快；分块阈值越大分块数目越少，并行粒度也越低并且单机的处理速度越慢。总结来说，通过设定合适分块阈值可以很大地提高词对齐模型的并行化训练性能。

图 5-8 为经过分块阈值优化后的 Seal 和 Chaski 在 LDC 和 UNPC 词对齐模型并行化性能对比，另外单机词对齐模型训练工具 MGIZA++ 只测试到 500 万规

模。图中坐标横轴为不同的数据规模，坐标纵轴为模型训练耗费时间。实验中 Seal 与 Chaski 设置同样的分块阈值，优化后 Seal 的并行训练性能相较 Chaski 平均提高 5 倍左右，相较单机词对齐模型训练系统 MGIZA++ 平均提高 40 倍以上。总体上，本文提出的并行化词对齐模型相较 Chaski 有更好的并行训练性能和数据规模可扩展性。

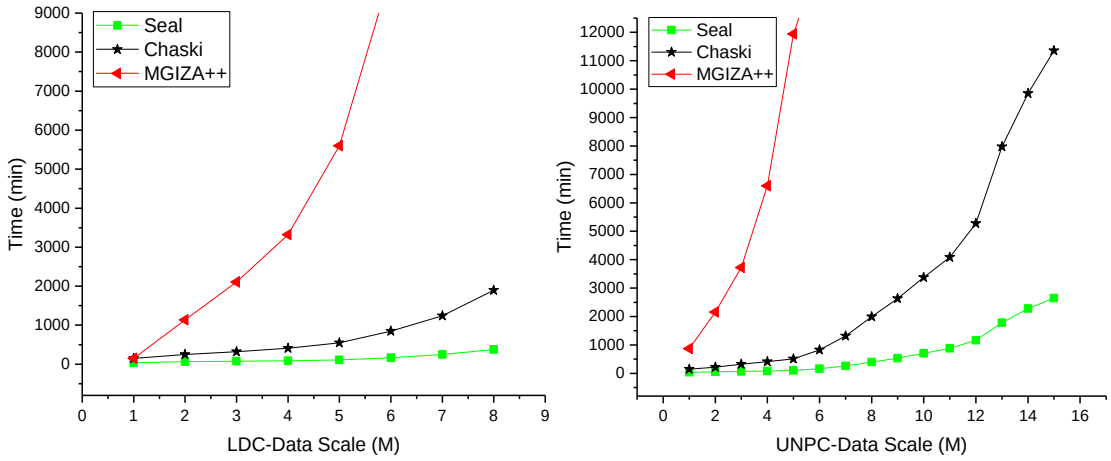


图 5-8 优化后的分布式词对齐并行训练性能对比

5.3.2 翻译模型优化效果评估

本节对 3.3.5 节针对分布式翻译模型提出的三种性能优化策略的实际优化效果在 LDC 和 UNPC 两个数据集下进行分析验证。同时对优化后的短语翻译模型、层次化短语翻译模型和句法翻译模型的并行训练性能和数据规模可扩展性与现有系统进行对比分析。

（1）分布式短语翻译模型优化性能评估

图 5-9 为短语翻译模型 Seal-phrase 经过数值化处理（记为 Encode）、join 优化（记为 Join-optimize）和数据倾斜优化（记为 Skew-optimize）后的并行训练性能对比。图中坐标横轴为不同的数据规模，坐标纵轴为短语翻译模型训练耗时，短语抽取的其他参数设置保持不变。使用数值优化后翻译模型的并行性能相比未进过优化前（Seal-phrase）平均提高 1.7 倍左右；使用 join 优化模型的并行性能相较数值化后平均提高 1.2 倍左右；使用数据倾斜优化后模型的并行性能相较 join 优化后平均提高 1.1 倍左右。综合三种优化策略，优化后 Seal-phrase 并行训练性能相比优化前平均提高 2 倍以上。可以看出 3.3.5 节提出的优化策略均可以有效地减少短语翻译模型的并行化训练耗时。

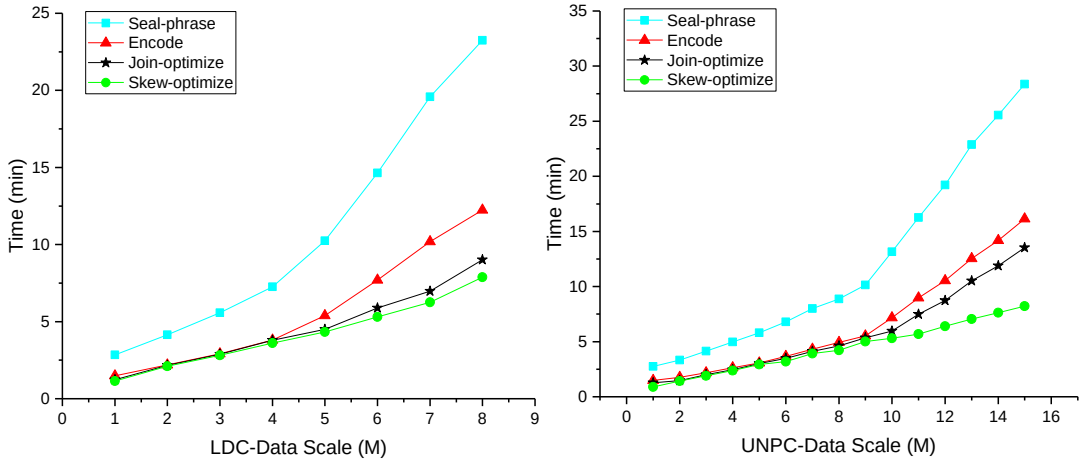


图 5-9 短语翻译模型优化性能对比

图 5-10 为优化后的 Seal-phrase 和 MR-phrase 和 Chaski 的并行模型训练耗时对比，本文也和单机短语翻译模型训练工具 Moses 也进行对比。图中坐标横轴表示数据规模的变化，坐标纵轴表示短语翻译模型训练耗费的时间。经过优化短语模型 Seal-phrase 的并行性能相较 Chaski 平均提高 6-7 倍，相较 MR-phrase 平均提高 8 倍左右，相比单机短语翻译模型训练工具训练性能平均提高 100 倍以上。从图中看出随着数据规模增加，优化后 Seal-phrase 相较现有的分布式短语翻译模型拥有更好的并行训练性能和数据规模扩展性。

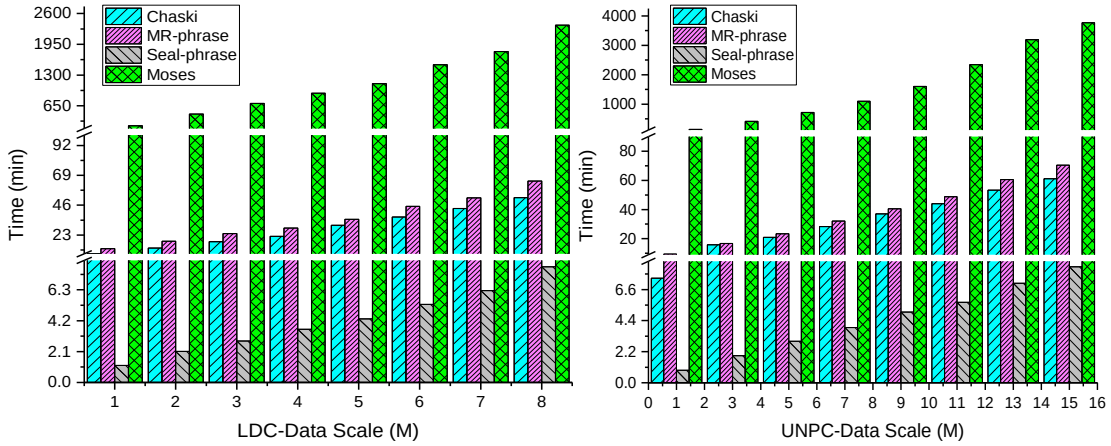


图 5-10 优化后的分布式短语翻译模型并行训练性能对比

（2）分布式层次化短语翻译模型优化性能评估

图 5-11 为层次化短语翻译模型 Seal-hiero 经过数值化处理、join 优化和数据倾斜优化后的并行训练性能对比。图中坐标横轴为不同数据规模，坐标纵轴表示模型训练所用时间。使用数值优化后的并行训练性能相比优化前平均提高 1.8 倍左右；join 优化后的并行训练性能相比数值化后平均提高 1.2 倍左右；数据倾斜

优化的并行训练性能比 join 优化后平均提高 1.1 倍左右。总体上，经过优化后 Seal-hiero 的并行训练性能相比优化前平均提高 2~3 倍。因此 3.3.5 的优化方案均可以减少层次化短语翻译模型并行化训练所用时间。

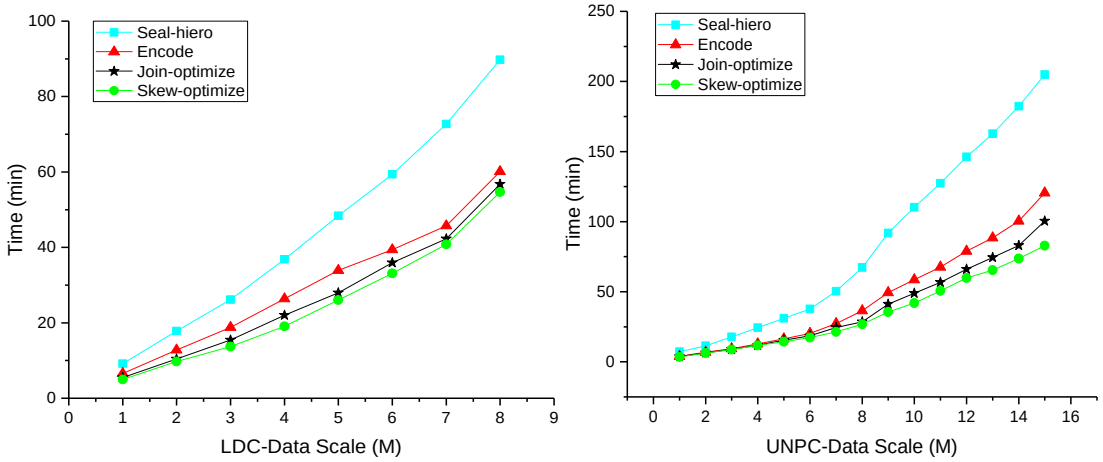


图 5-11 层次化短语翻译模型优化性能对比

图 5-12 为优化后的层次化短语翻译模型 Seal-hiero 与 MR-hiero 的性能对比，图中坐标横轴为数据规模的变化，坐标纵轴为模型训练耗时。优化层次化短语模型 Seal-hiero 的并行训练性能相较 MR-hiero 平均提高 15-21 倍。随着数据规模增长，层次化短语翻译模型抽取的规则单元数量远大于短语翻译模型，本文研究实现优化策略的作用也更加明显。从图中看出优化后的 Seal-hiero 呈现出更短的模型训练耗时和数据规模扩展性。

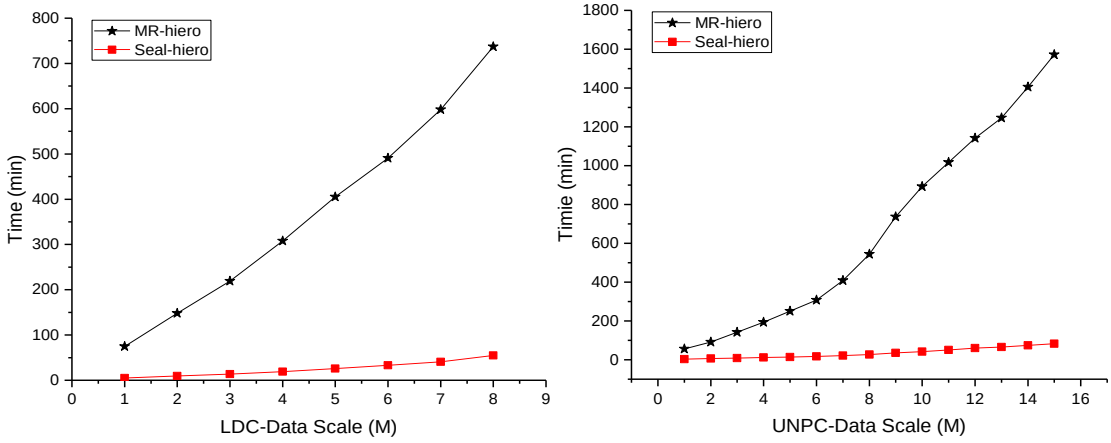


图 5-12 优化后的层次化短语翻译模型并行训练性能对比

（3）分布式句法翻译模型优化性能评估

图 5-13 到图 5-14 为短语句法翻译模型 Seal-syntax-phrase 和层次化短语句法翻译模型 Seal-syntax-hiero 经过优化后的并行训练性能对比。图中坐标横轴表示

不同的数据规模变化，坐标纵轴表示句法翻译模型训练耗时。优化后的短语句法翻译模型 Seal-syntax-phrase 的并行训练性能相较优化前平均提高 2 倍左右，优化后的层次短语句法翻译模型 Seal-syntax-hiero 相较优化前平均提高 2-3 倍。可以看出这些优化策略均能有效地减少句法翻译模型的并行训练所用时间。

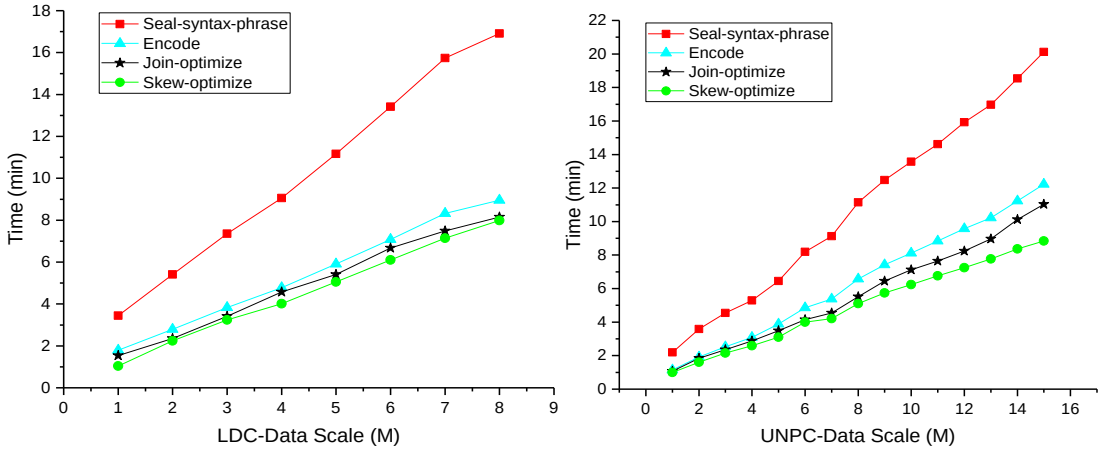


图 5-13 短语句法翻译模型优化性能对比

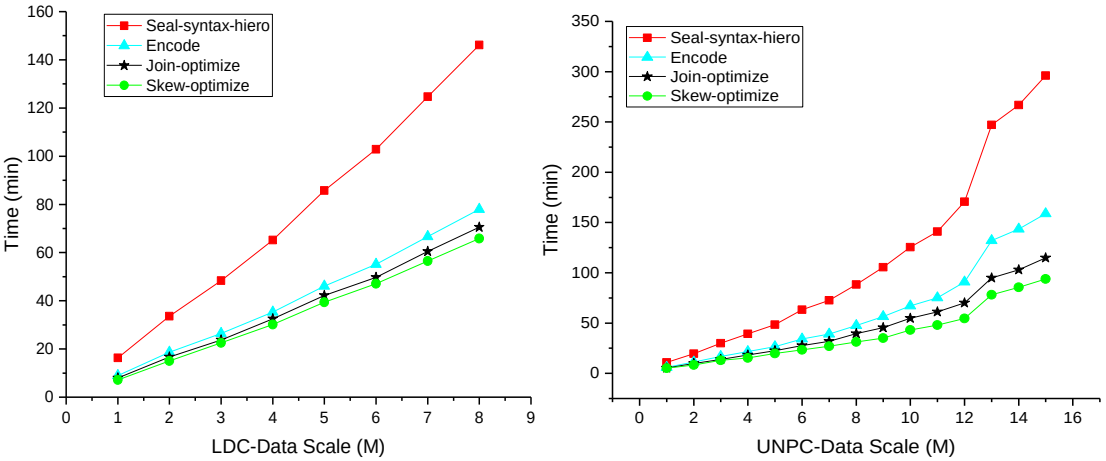


图 5-14 层次化短语句法翻译模型优化性能对比

图 5-15 为优化后的分布式短语句法翻译模型 Seal-syntax-phrase 与 MR-syntax-phrase 的并行训练性能对比，图 5-16 为优化后分布式层次短语句法翻译模型 Seal-syntax-hiero 与 MR-syntax-hiero 的并行训练性能对比。图中坐标横轴表示数据规模的变化，坐标纵轴表示模型训练耗费的时间。优化 Seal-syntax-phrase 相比 MR-syntax-phrase 模型可以平均达到 9 倍左右的加速比，优化 Seal-syntax-hiero 相比 MR-syntax-hiero 模型并行训练性能平均提高 13-20 倍。从图 5-15 到图 5-16 可以看出，Seal 中无论短语句法翻译模型还是层次短语句法翻译模型，相比

现有系统都具有更短的模型训练耗时和更好的数据规模扩展性。

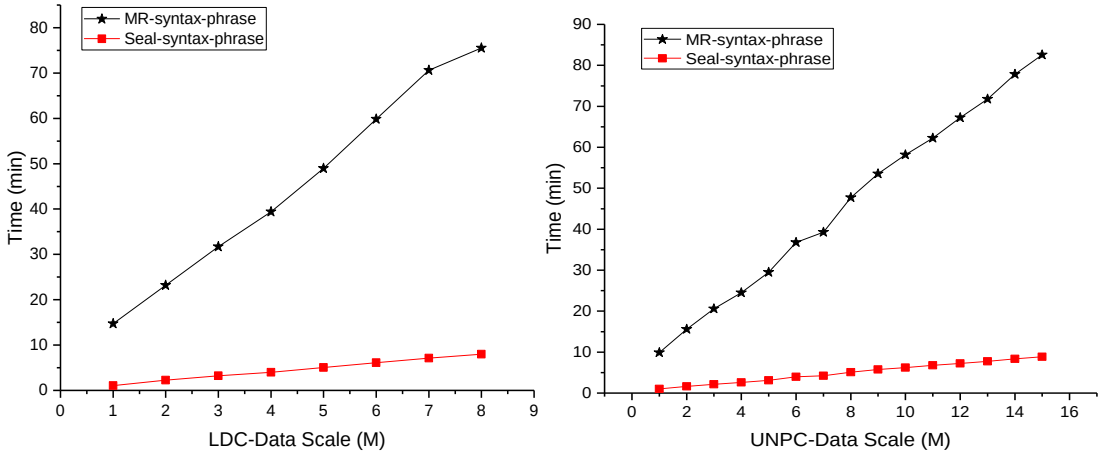


图 5-15 优化后的短语句法翻译模型并行训练性能对比

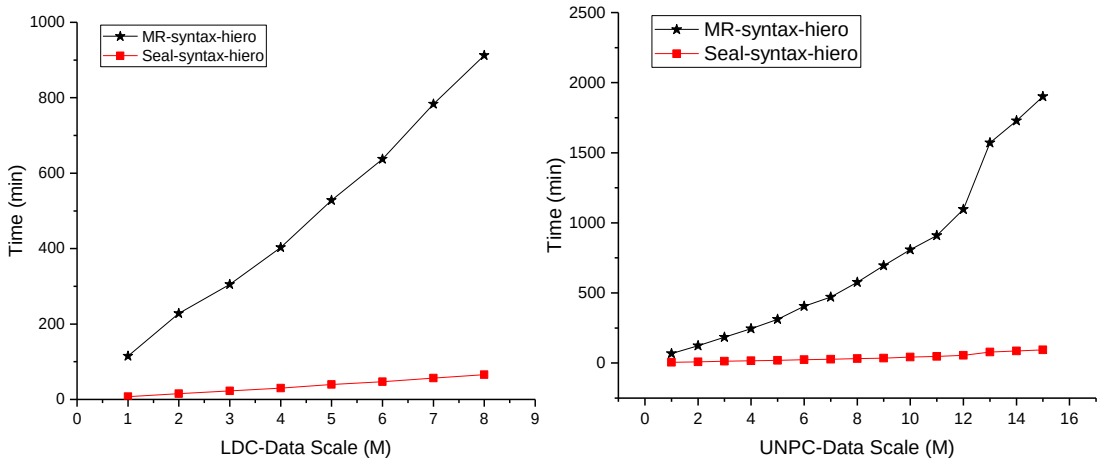


图 5-16 优化后的层次短语句法翻译模型并行训练性能对比

（4）分布式翻译模型概率平滑性能评估

Seal 中的翻译模型在参数概率估计时都可以选择不同的概率平滑方案，简单起见，这里只对 Seal 中短语翻译模型采用平滑时的并行训练性能与现有的短语翻译模型采用同样概率平滑方案时的并行训练性能进行对比分析。图 5-17 为 Seal-phrase 中短语翻译模型采取三种概率平滑的并行化训练性能与 MR-phrase 对比，图中坐标横轴表示数据规模的变化，坐标纵轴表示模型训练耗费的时间。其中，GT 表示 Good Turing 平滑，KN 表示 Kneser-Ney 平滑，MKN 表示 Modified Kneser-Ney 平滑（见 2.4.3 节）。从图中可以看出，随着数据规模的不断变化，Seal 中短语翻译模型采取的三种概率平滑方法的时间变化曲线相较 MR-phrase 都较为平缓，模型的并行训练耗时更短和更好的数据规模扩展性。

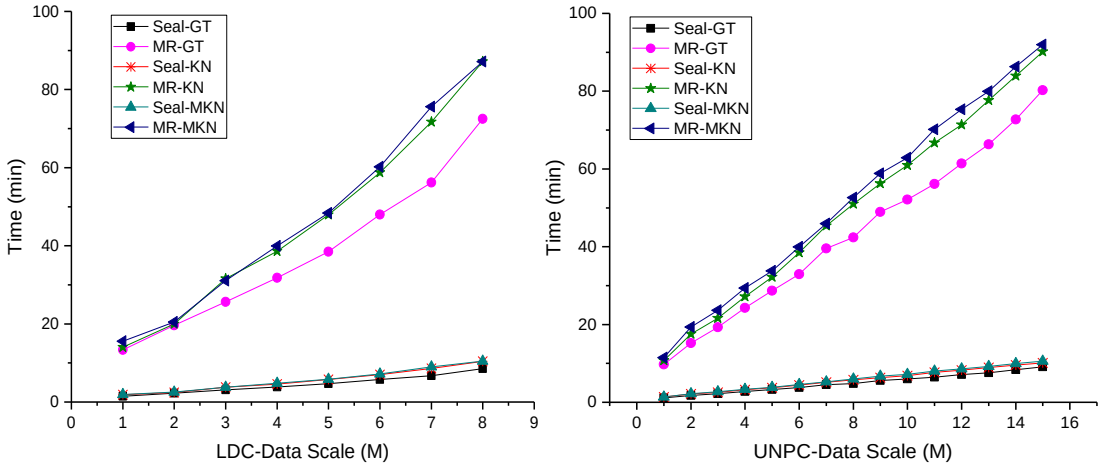


图 5-17 短语翻译模型平滑并行训练性能对比

5.3.3 语言模型优化效果评估

本节对 3.4.3 节针对语言模型提出的性能优化策略优化效果进行分析验证。图 5-18 为采用 MKN（Modified Kneser-Ney Smoothing）平滑的语言模型。经过 join 优化（Join-optimize）和数据倾斜优化（Skew-optimize）后的并行训练性能与现有系统的对比。图中坐标横轴为不同的数据规模，坐标纵轴为语言模型训练所用时间。经过 join 优化后的并行训练性能相较原来模型的性能平均提高 1.8 倍左右。经过数据倾斜优化后的并行性能相较 join 优化后平均提高 1.5 倍左右，综合两种优化策略相较优化前可以达到 2-3 倍的性能加速比，可以看出 3.4.3 节中对分布式语言模型提出的优化策略均能有效地减少模型的并行化训练所耗费的时间。

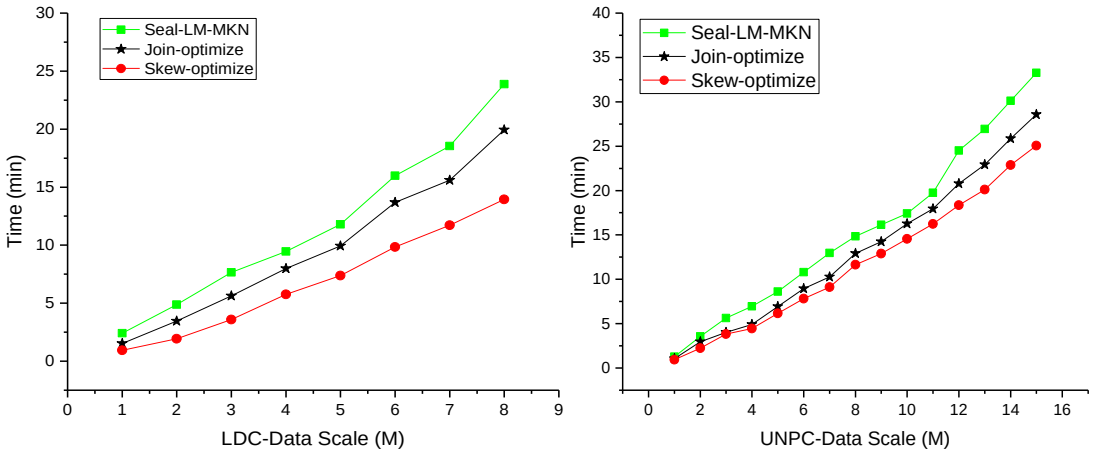


图 5-18 MKN 语言模型优化性能对比

图 5-19 为 Seal-LM-MKN 和 MR-LM-MKN 语言模型并行训练性能对比，本文也和单机语言模型工具 KenLM 对比。图中坐标横轴为不同的数据规模，坐标

纵轴为模型生成耗费时间。优化后的 Seal-LM-MKN 相较 MR-LM-MKN 并行训练性能平均提高 8 倍左右，相较 KenLM 的训练耗时平均提高 20~100 倍左右。总体上，Seal-LM 相比 MR-LM 具有更好的并行训练性能和数据规模可扩展性。

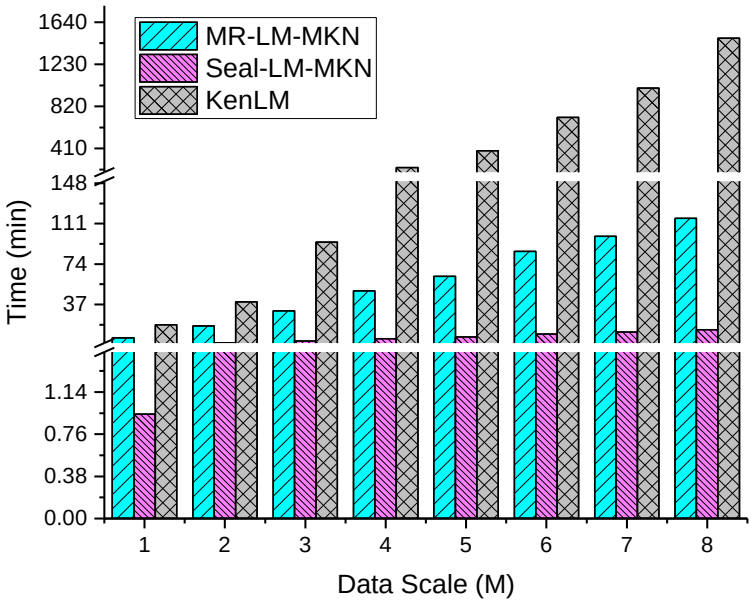


图 5-19 优化后的 MKN 语言模型并行训练性能对比

图 5-20 为优化的语言模型 Seal-LM 与 MR-LM 的并行训练性能对比。图中坐标横轴为数据规模的变化，坐标纵轴是模型训练耗费的时间，其中 GT 表示 Good Turing 平滑，KN 表示 Kneser-Ney 平滑，MKN 表示 Modified Kneser-Ney 平滑，GSB 表示 Google Stupid Backoff 平滑。经过优化后的语言模型 Seal-LM 的并行化训练性能相较 MR-LM 平均提高 8-9 倍。从图中可以看出，优化后的 Seal-LM 表现出更短的模型训练耗时和更好的数据扩展性。

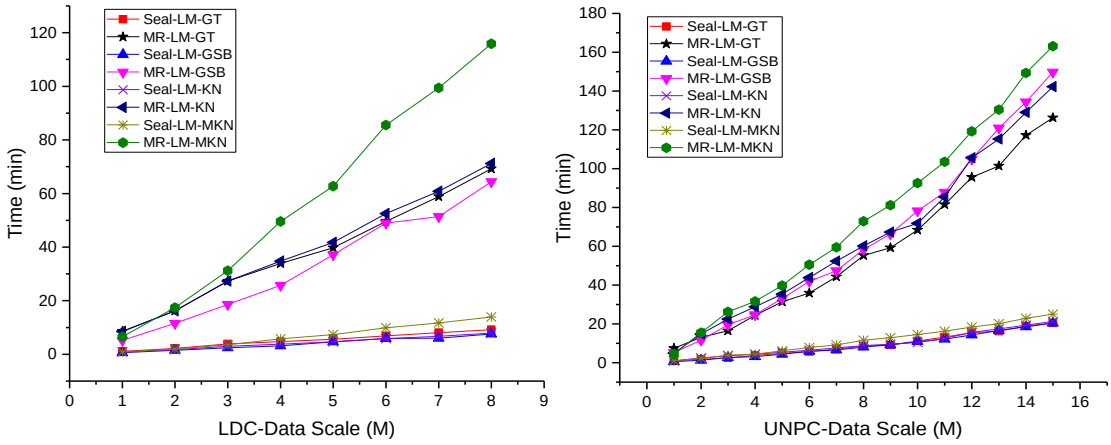


图 5-20 优化后的语言模型并行训练性能对比

5.4 分布式机器翻译模型节点扩展性

由于 5.3 节中讨论了 Seal 中每个模型的数据扩展性，同时分布式模型的节点扩展性也是对并行训练性能的一个重要评价标准。因此本节将分别对分布式词对齐模型，分布式翻译模型和分布式语言模型的节点扩展性进行测试分析。

5.4.1 词对齐模型扩展性

图 5-21 为 Seal 与 Chaski 在相同数据规模下，计算节点数量变化时对词对齐性能影响。坐标横轴表示不同的计算节点个数，坐标纵轴表示模型训练耗费的时间，数据集选用 LDC-5 M 和 UNPC-6 M（M 表示百万行）。从图中可以看出，词对齐模型训练的时间随着计算节点数目的增加不断减小，Seal 的时间变化曲线趋于平缓，相较 Chaski 呈现出近线性的节点可扩展性。

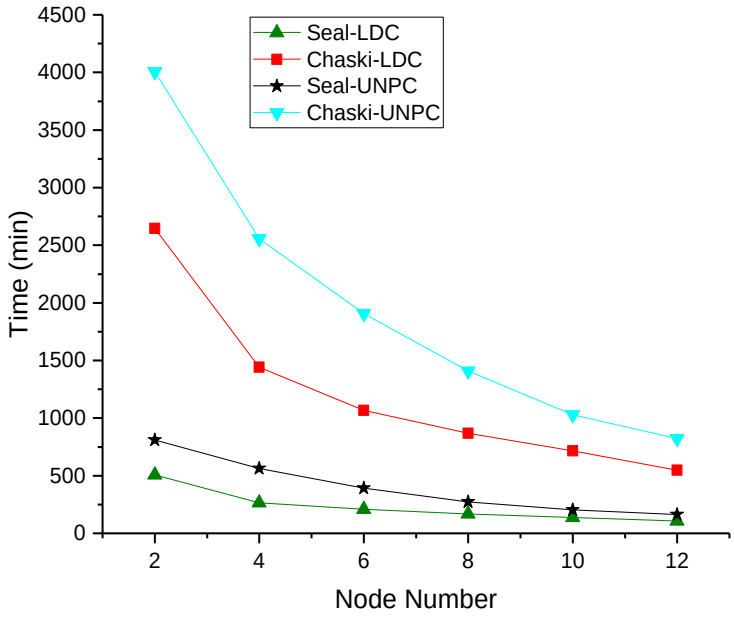


图 5- 21 分布式词对齐模型节点扩展性

5.4.2 翻译模型扩展性

图 5-22 为 Seal-phrase 与 Chaski 和 MR-phrase 在相同数据规模下，计算节点数量变化时对短语翻译模型的并行化性能影响。坐标横轴表示不同的计算节点个数，坐标纵轴表示模型训练耗费的时间，数据集选用 LDC-8 M 和 UNPC-15 M（M 表示百万行）。从图中可以看出，短语翻译模型训练的时间随着计算节点数目的增加不断减小，Seal-phrase 相较 Chaski 和 MR-phrase 都呈现出平稳、近线性的节点可扩展性。

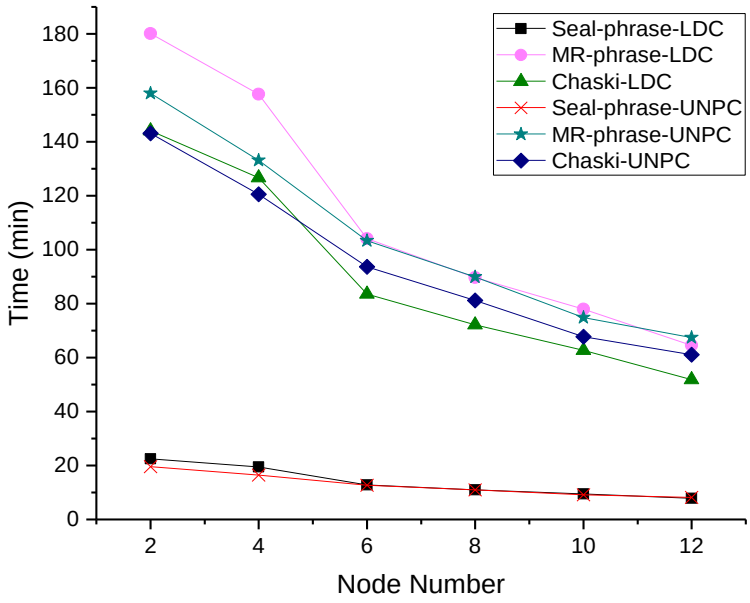


图 5-22 短语翻译模型节点扩展性

图 5-23 为 Seal-hiero 与 MR-hiero 在相同数据规模下，计算节点数量变化时对层次化短语翻译模型的并行化性能影响，坐标横轴表示不同计算节点数量，坐标纵轴表示模型训练耗时，数据集选用 LDC-5 M 和 UNPC-8 M（M 表示百万行）。从图中可以看出，Seal-hiero 相较 MR-hiero 表现出更好的节点可扩展性。

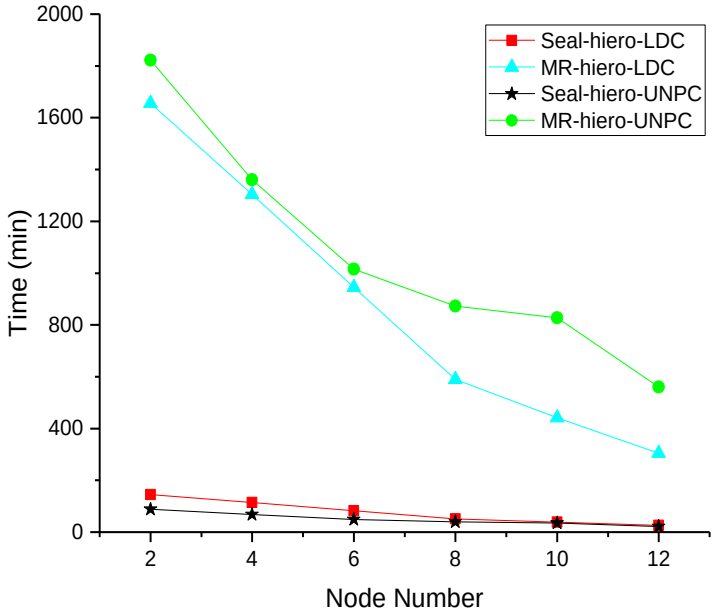


图 5-23 层次化短语翻译模型节点扩展性

图 5-24 为句法翻译模型 Seal-syntax 与 MR-syntax 在相同数据规模下，计算节点个数变化时对层次化短语翻译模型性能影响，图中坐标横轴表示不同的计算节点个数，坐标纵轴表示模型训练耗费的时间，对于短语句法翻译模型数据集选

用 UNPC-15 M（M 表示百万行），对于层次短语句法翻译模型数据集选用 UNPC-8 M。从图中可以看出，随着数据大小的变化 Seal-syntax 相较 MR-syntax 表现出更好的节点可扩展性。

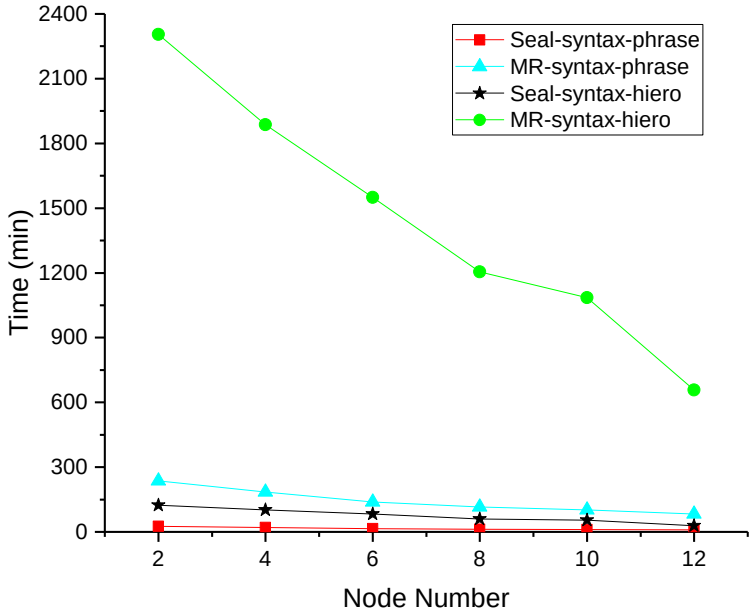


图 5-24 句法翻译模型节点扩展性

图 5-25 为短语翻译模型采取平滑后，在相同数据规模下的计算节点扩展性。图中坐标横轴为计算节点数量的变化，坐标纵轴为模型训练耗费时间。数据集选用 UNPC-15 M（M 表示百万行）。从图中可以看出，随着数据规模的变化 Seal 的短语翻译模型平滑表现出更好的近似线性的节点可扩展性。

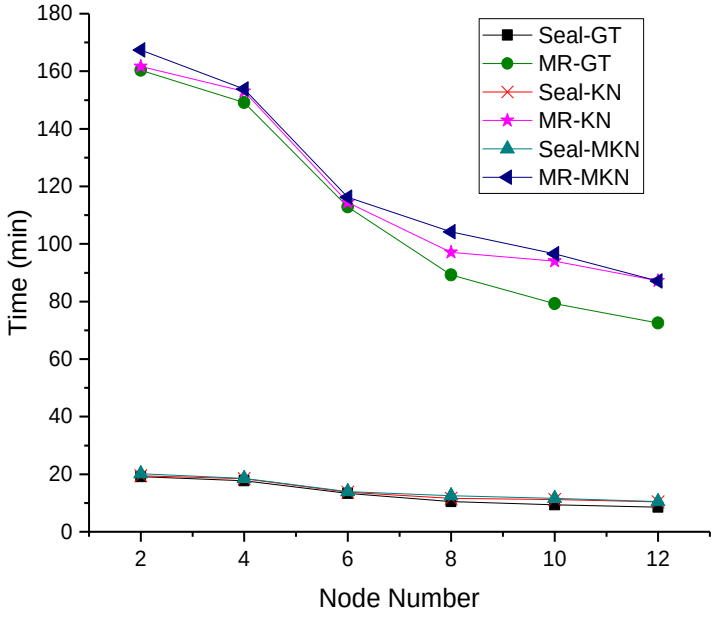


图 5-25 短语翻译模型平滑性能节点扩展性

5.4.3 语言模型扩展性

图 5-26 为 Seal-LM 与 MR-LM 在相同数据规模下，计算节点数量变化时对语言模型性能的影响，坐标横轴表示不同的计算节点数目，坐标纵轴表示模型训练耗费的时间，数据集选用 UNPC-15 M（M 表示百万行）。总体上，语言模型训练的时间随着计算节点数目的增加不断减小，Seal-LM 的时间变化曲线趋于平缓，相较 MR-LM 呈现出更好的、近线性的节点可扩展性。

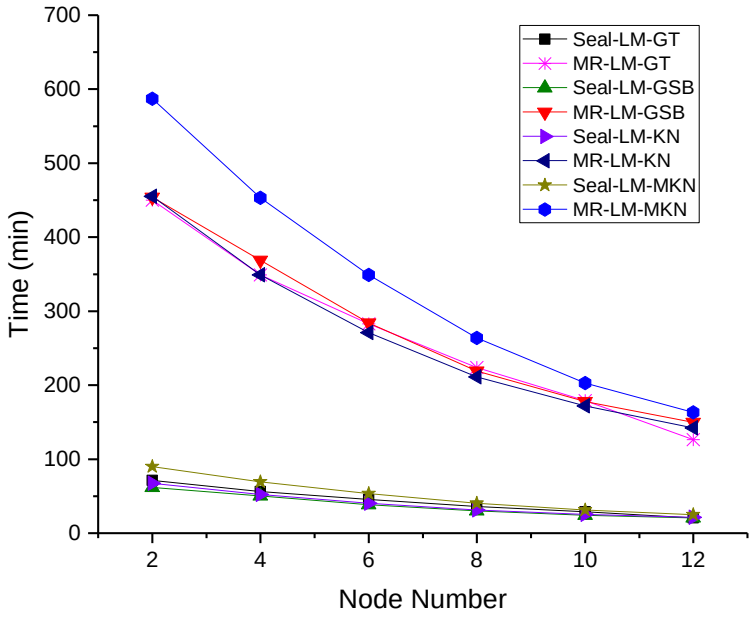


图 5-26 语言模型节点扩展性

5.5 与现有其他系统的对比

本节主要是将统计机器翻译中离线模型训练的整个流程作为整体分别与单机系统和组合的现有分布式系统进行性能对比分析。这里约定词对齐模型训练中采用相同的参数配置（如模型训练序列等），翻译模型以短语翻译模型为基准，参数配置不变，语言模型的参数配置也保持相同。单机系统中选用 MGIZA++ 来训练词对齐模型，选用 Moses 来训练短语翻译模型，选用 KenLM 来训练语言模型。组合的现有分布式工具中（因为现有的分布式机器翻译工具都不支持整个离线模型的训练过程），以分布式词对齐工具 Chaski 作为对比对象，以分布式短语翻译模型系统 Chaski 作为对比，以分布式语言模型 MR-LM-MKN 作为对比对象。数据集选用 LDC 数据集，数据大小范围为 100 万行到 500 万行（这里没有选用更大的数据集是因为其中的有些模型如词对齐在较大的数据集下存在较长的模型训练耗时）。表 5-2 所示为离线模型整体的性能对比。

表 5-2 为统计机器翻译中将整个离线模型作为整体时在不同数据规模下的性能对比，可以看出 Seal 相比组合分布式系统能够平均实现 4-5 倍的加速比，相比现有的单机机器翻译系统平均能达到 22-37 倍的加速比。因此从整体上看，Seal 相较现有的系统具有更少的模型训练耗时，整体性能表现更为优异。

表 5-2 离线模型整体性能对比

	Seal (min)	**Distributed (min)**	**Speedup**	**Standalone (min)**	**Speedup**
LDC-1 M	62	309	4.98	1365	22.02
LDC-2 M	94	479	5.09	2598	27.64
LDC-3 M	130	660	5.07	4076	31.35
LDC-4 M	206	1088	5.28	6641	32.24
LDC-5 M	324	1833	5.65	12086	37.30

最后，为证明 Seal 中对离线模型的并行训练不会很大程度改变最终的翻译质量，本文在 WMT [1](Workshop Machine Translation)上的五个测试集进行了验证，表 5-3 是测试结果。

表 5-3 WMT 英法两个方向的 BLEU 测试指标

测试集	**英语-法语**		**法语-英语**	
	BLEU(11b)	BLEU-cased(11b)	BLEU(11b)	BLEU-cased(11b)
test2007	26.8	22.6	27.4	21.2`
nc-test2007	27.8	24.0	27.4	22.4
newstest2010	22.4	18.3	24.2	18.6
newstest2011	25.0	20.7	26.7	21.2
newstest2013	23.6	20.2	25.6	20.3

表 5-3 中包含了五个测试集合上英语到法语和法语到英语测自动评价指标，可以看出 Seal 中对模型并行化后并不会很大程度上改变翻译精度。

5.6 本章小结

本章主要介绍在大规模数据集下，对 Seal 中三个模型的并行训练性能进行验证与对比。不仅对 Seal 本身的并行训练性能进行验证分析，也与现有的分布式模型训练工具进行对比。本章首先介绍实验设置，包括采用的训练数据集和计算节点的软硬件环境配置。5.2 节中对词对齐模型、翻译模型和语言模型的并行

[1] http://www.statmt.org/wmt17/translation-task.html

训练性能与现有的分布式模型训练工具进行对比分析。接下来，5.3 节中在大规模数据集下对每个模型优化策略的实际效果进行评估。在词对齐模型中，评估 3.2.4 节对词对齐模型提出的优化策略的优化效果，并与 Chaski 和 MGIZA++ 对比分析了优化后词对齐模型的数据可扩展性。在翻译模型中，评估 3.3.5 节中针对三个翻译模型提出的优化策略的实际加速性能，并对优化后的三个模型的并行训练性能与现有的分布式工具进行了对比分析，同时也分析了数据规模可扩展性，另外也对翻译模型中采用平滑方法时的并行训练性能进行了分析。在语言模型中，评估 3.4.3 节针对语言模型提出的优化策略的实际效果，并对优化后模型与现有的工具进行了并行训练性能和数据规模可扩展性分析。5.4 节中对比分析了每个模型的节点扩展性。最后 5.5 节将统计机器翻译离线模型的训练流程作为整体进行了性能对比分析。

第六章 总结与展望

6.1 本文工作总结

随着日益频繁的跨国人文和学术交流，对机器翻译提出了迫切的需求。然而在大规模平行语料数据集下，传统单机统计机器翻译系统模型训练耗时过长，这严重制约了统计机器翻译的模型深入研究及其应用推广。因此，研究大规模分布式机器翻译系统具有很高的研究意义和实用价值。

然而，目前缺乏完整的分布式统计机器翻译离线模型训练流程，如现有并行化翻译模型中，缺少对层次化短语翻译模型和句法翻译模型的并行化训练。同时现有的分布式统计机器翻译工具还存在并行化训练性能不足等问题。而事实上，设计实现高效的分布式统计机器翻译模型的离线训练流程存在一定的挑战性。因此本文在分析现有工作的不足和并行化统计机器翻译离线模型训练难点的基础上，基于广泛使用的分布式数据并行计算平台 Spark 研究并实现了一个完整、高效、弹性可扩展的大规模分布式统计机器翻译离线模型训练系统，为机器翻译模型的研究和实际应用提供有利支撑。本文主要完成了以下工作：

1）在大规模分布式场景下，研究实现了统计机器翻译中耗时较长的离线模型训练方法与并行化算法。其中，词对齐模型部分包括预处理与相应的词对齐模型分布式训练；翻译模型并行化实现了三种不同的翻译模型；语言模型可以使用多种不同的概率平滑算法进行并行化构建。

2）优化模型训练中系统负载均衡和节点间的网络通信。在词对齐模型中设置合适的分块阈值，在保证并行效率的前提下，降低 I/O 操作和节点间网络通信带来的开销。翻译模型中为提高模型训练效率，对训练语料进行数值化处理，减少训练过程中的中间数据集规模。

3）根据训练流程中需要多次估计模型参数的特点，使用两种方法优化基于 join 算子的并行化最大似然估计算法。一是广播小表到每个计算节点，在分布式大表中获取广播的小表来避免全局 join；二是对两个待 join 的分布式大表使用相同 partitioner，使得它们内部数据预先满足相同的划分规则，从而避免执行过程中的数据 shuffle。

4）提出两种优化策略，来改善模型训练中的数据倾斜问题。一是适当提高

模型训练并发度（或重分区）；二是两阶段聚合方法，即添加随机前缀来扩展 key，使其可以均匀地分布到计算节点。然后在每个节点进行局部聚合，最后去掉前缀进行全局聚合。

5）在上述关键技术方法和算法研究的基础上，基于广为使用分布式数据并行计算平台 Spark 设计实现了大规模分布式统计机器翻译离线模型训练的原型系统 Seal，并在大规模数据集下进行了性能评估与分析。实验结果表明，Seal 的并行化训练性能优于现有的分布式和单机统计机器翻译模型训练工具，同时还具有更好的可扩展性。

6.2 下一步工作

下一步工作中本文将会针对以下两点展开：

1）对统计机器翻译框架中的解码部分进行分析和研究。结合目前已经实现的离线模型部分，研究实现在大规模数据集下的高效解码。同时本文提出的分布式统计机器翻译离线模型训练工具 Seal 还存在进一步优化空间。如在模型训练中采取更智能的策略去尽可能防止数据倾斜的发生；在分布式词对齐模块中采取自适应的分块阈值；词对齐模型的预处理过程（词聚类等）使用更好的算法来改善预处理的效果，来最终提升词对齐结果的质量；

2）深度学习的快速发展使得机器翻译的研究迎来新的高潮，涌现了大量基于神经网络的深度机器翻译模型。统计机器翻译和深度神经网络机器翻译都在大规模的数据集上进行训练，本质上都是学习参数模型。统计机器翻译中，需要进行词对齐模型训练，翻译规则的抽取，压缩等过程，最后再基于翻译单元表解码。现在广泛使用的深度神经网络机器翻译可以看作是一个端到端的模型，使用较大的语料集和选择合适的模型参数就可以得到良好的翻译效果。然而目前仍然在精度和计算效率上存在诸多问题。因此本文未来将 Seal 作为基准系统，研究深度学习在机器翻译中的应用，并研究如何和传统的统计机器翻译方法融合，从而进一步提高机器翻译模型的训练效率与模型精度。

参考文献

[1] Williams K. Understanding media theory [J]. 2003.

[2] 刘群. 统计机器翻译综述[J]. 中文信息学报, 2003, 17(4): 2-13.

[3] Brown P F, Pietra V J D, Pietra S A D, et al. The mathematics of statistical machine translation: Parameter estimation [J]. Computational linguistics, 1993, 19(2): 263-311.

[4] Dyer C, Cordova A, Mont A, et al. Fast, easy, and cheap: Construction of statistical machine translation models with MapReduce [C]//Proceedings of the Third Workshop on Statistical Machine Translation. Association for Computational Linguistics, 2008: 199-207.

[5] Papineni K, Roukos S, Ward T, et al. BLEU: a method for automatic evaluation of machine translation[C]//Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002: 311-318.

[6] Koehn P, Hoang H, Birch A, et al. Moses: Open source toolkit for statistical machine translation[C]//Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, 2007: 177-180.

[7] Och F J, Ney H. Improved statistical alignment models[C]//Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2000: 440-447.

[8] Moses Training. [EB/OL].http://www.statmt.org/moses/?n=FactoredTraining.HomePage.

[9] Gao Q, Vogel S. Parallel implementations of word alignment tool[C]//Software Engineering, Testing, and Quality Assurance for Natural Language Processing. Association for Computational Linguistics, 2008: 49-57.

[10] Gao Q, Vogel S. Training phrase-based machine translation models on the cloud: Open source machine translation toolkit Chaski [J]. The Prague Bulletin of Mathematical Linguistics, 2010, 93: 37-46.

[11] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

[12] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2-2.

[13] Brants T, Popat A C, Xu P, et al. Large language models in machine translation[C]//In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. 2007.

[14] Vogel S, Ney H, Tillmann C. HMM-based word alignment in statistical translation [C]//

Proceedings of the 16th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, 1996: 836-841.

[15] Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation[C]//Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. 2003: 48–54.

[16] D. Chiang. A hierarchical phrase-based model for statistical machine translation[C]// Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. 2005: 263–270.

[17] Shen, Libin and Xu, et al. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Mode//Proceedings of the 2008 Association for Computational Linguistics. 2008:577-585.

[18] Heafield K. KenLM: Faster and smaller language model queries[C]//Proceedings of the Sixth Workshop on Statistical Machine Translation. Association for Computational Linguistics, 2011: 187-197.

[19] Bogoychev N, Lopez A. N-gram language models for massively parallel devices[C]//Proc. ACL. 2016: 1944-1953.

[20] Cadigan J, Marton Y. GISA: Giza++ Implementation over Spark by Apache [J]. AMTA 2016, 2016: 13..

[21] 王爱平, 张功营, 刘方. EM 算法研究与应用[J]. 计算机技术与发展, 2009, 19(9): 108-110.

[22] Chu C T, Kim S K, Lin Y A, et al. Map-reduce for machine learning on multicore[C]//NIPS. 2006, 6: 281-288.

[23] Wolfe J, Haghighi A, Klein D. Fully distributed EM for very large datasets[C]//Proceedings of the 25th international conference on Machine learning. ACM, 2008: 1184-1191.

[24] Vičič J, Brodnik A. Increasing the throughput of machine translation systems using clouds [J]. arXiv preprint arXiv:1611.02944, 2016.

[25] Venugopal A, Zollmann A. Grammar based statistical MT on Hadoop [J]. The Prague Bulletin of Mathematical Linguistics, 2009, 91.

[26] Zollmann A, Venugopal A, Vogel S. The CMU syntax-augmented machine translation system: SAMT on Hadoop with n-best alignments[C]//IWSLT. 2008: 18-25.

[27] Weese J, Ganitkevitch J, Callison-Burch C, et al. Joshua 3.0: Syntax-based machine translation with the Thrax grammar extractor[C]//Proceedings of the Sixth Workshop on Statistical Machine Translation. Association for Computational Linguistics, 2011: 478-484.

[28] Tomar A, Bodhankar J, Kurariya P, et al. Translation Memory for a Machine Translation System Using the Hadoop Framework[C]//Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on. IEEE, 2014: 203-207.

[29] Allam T M, Sallam A A, Abdullkader H M. Managed N-gram language model based on Hadoop framework and a Hbase tables[C]//Informatics and Systems (INFOS), 2014 9th International Conference on. IEEE, 2014: PDC-58-PDC-63.

[30] Harter T, Borthakur D, Dong S, et al. Analysis of HDFS under HBase: a facebook messages case study[C]//FAST. 2014, 14: 12th.

[31] Yu X. Estimating language models using Hadoop and HBase [J]. University of Edinburgh, 2008.

[32] Uszkoreit J, Brants T. Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation[C]//ACL. 2008: 755-762.

[33] Emami A, Papineni K, Sorensen J. Large-scale distributed language modeling[C]//Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. IEEE, 2007, 4: IV-37-IV-40.

[34] Patil T, Jing Zheng. Using Apache Spark for large scale language model training [EB/OL].[2017-02-26].https://code.facebook.com/posts/678403995666478/using-apache-spark-for-large-scale language-model-training.

[35] Mikolov T, Deoras A, Povey D, et al. Strategies for training large scale neural network language models[C]//Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on. IEEE, 2011: 196-201.

[36] Forney G D. The viterbi algorithm [J]. Proceedings of the IEEE, 1973, 61(3): 268-278.

[37] Och F J. An efficient method for determining bilingual word classes[C]//Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 1999: 71-76.

[38] Tillmann C. A unigram orientation model for statistical machine translation[C]//Proceedings of HLT-NAACL 2004: Short Papers. Association for Computational Linguistics, 2004: 101-104.

[39] Good I J. The population frequencies of species and the estimation of population parameters [J]. Biometrika, 1953: 237-264.

[40] Kneser R, Ney H. Improved backing-off for m-gram language modeling[C]//Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on. IEEE, 1995, 1: 181-184.

[41] Chen S F, Goodman J. An empirical study of smoothing techniques for language

modeling[C]//Proceedings of the 34th annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1996: 310-318.

[42] Och F J, Tillmann C, Ney H. Improved alignment models for statistical machine translation[C]//Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora. 1999: 20-28.

[43] Manning C D, Schütze H. Foundations of statistical natural language processing [M]. Cambridge: MIT press, 1999.

[44] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system[C]//Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on. IEEE, 2010: 1-10.

[45] Chowdhury M. Performance and scalability of broadcast in Spark [J]. 2014-10-08]. http://www. cs. berkeley. edu/~ agearh/cs267. sp10/files/mosharaf-spark-bc-report-spring10. pdf, 2014.

[46] Ziemski M, Junczys-Dowmunt M, Pouliquen B. The united nations parallel corpus v1. 0[C]//Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC. 2016: 23-28.

致 谢

俯仰之间，三年的校园生活行将完结。首先感谢南京大学计算机系和PASA 实验室的老师和同学们，然后感谢袁春风和黄宜华老师对我的细心指导，最后感谢所有帮助过我的朋友和默默支持我的家人。

首先感谢袁春风老师在这三年时间对我的帮助，袁老师平易近人，和蔼可亲，时时关注我的科研工作进展，对我的科研工作提出了很多宝贵的指导意见。同时在毕业大论文和小论文修改中，袁老师对我的论文进行了细致周到的修改，提出了很多针对性非常强的修改意见，很大程度上提高了我的论文写作水平。袁老师和黄宜华老师不但给我们提供了良好的科研环境，而且还一直谆谆教导我们，教会了我们如何进行科研和很多人生哲理，避免了在短暂的三年内走弯路，他们对工作一丝不苟的精神和态度深深地影响了我，我将会受用终生。

其次，感谢顾荣师兄和周娟师姐，本论文也是在他们研究工作基础上展开的。在和他们的互助合作中，我也学习了很多技术和良好的科研学习方法，可谓收益颇多。另外在这里，我还要感谢仇红剑师兄，在投稿计算机学报的一年时间里，我们的论文经历了多次修改，他主动帮我解决了很多困难。同时也感谢 PASA 的同学们和我的舍友，他们陪我一起度过了三年愉快的科研生活，让枯燥的科研生活多了很多乐趣。此外也感谢实验室的王肇康、汪敏梅、郭晨、陈敏、黄圣斌等同学对我的帮助。

最后，感谢在背后默默支持我的家人们，感谢爸妈对我无私的付出！感谢我所有的朋友们，你们让我的生活不孤单，我会继续奔跑下去。

附 录

研究生期间发表论文

[1] 顾荣，仇红剑，**杨文家**，胡伟，袁春风，黄宜华. Goldfish: 基于矩阵分解的大规模RDF数据存储与查询系统. 计算机学报. 已录用.

[2] 周娟，顾荣，**杨文家**，黄书剑，袁春风，黄宜华. 基于Spark的大规模翻译模型并行化训练算法研究. 中文信息学报. 已录用.

研究生期间申请专利

[1] 黄宜华, 周娟, 顾荣, **杨文家**. 《基于Spark的大规模短语翻译模型的训练方法》. 申请号: 201610346396.4

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名：_____

_____年_____月_____日

论文题名	大规模分布式统计机器翻译离线模型训练方法与系统				
研究生学号	MG1433074	所在院系	计算机科学与技术系	学位年度	2017
论文级别	☐硕士 ☐硕士专业学位 ☐博士 ☐博士专业学位 （请在方框内画钩）				
作者电话	183-5188-1151		作者 Email	yangwen_jia@163.com	
第一导师姓名	袁春风		导师电话	189-5167-9120	

论文涉密情况：

☐ 不保密

☐ 保密，保密期（_____年___月___日 至 _____年___月___日）

注：请将该授权书填写后装订在学位论文最后一页 （南大封面）。