计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2014.20131335
51(12):2724-2732, 2014

基于分布内存的层次短语机器翻译并行化算法

赵 博 黄书剑 戴新宇 袁春风 黄宜华
(计算机软件新技术国家重点实验室(南京大学) 南京 210023)
(江苏省软件新技术与产业化协同创新中心 南京 210023)
(bhoppi@outlook.com)

Parallel Algorithm for Hierarchical Phrase Machine Translation Based on Distributed Memory Storage

Zhao Bo, Huang Shujian, Dai Xinyu, Yuan Chunfeng, and Huang Yihua
(*State Key Laboratory for Novel Software Technology* (*Nanjing University*), *Nanjing* 210023)
(*Collaborative Innovation Center of Novel Software Technology and Industrialization*, *Nanjing* 210023)

Abstract In recent years, in order to improve the accuracy of SMT (statistical machine translation) system, massive corpus has been widely applied to train language and translation models. As the scale of the language and translation models increase, computing performance becomes a challenging issue for SMT, which makes existing single-machine translation algorithms and systems difficult to complete the computation in time, especially when dealing with online translation. In order to overcome the limitations of single-machine translation decoding algorithm and improve the computing performance of large-scale SMT toward a practical online translation system, this paper proposes a distributed and parallel translation decoding algorithm and framework by adopting a distributed storage and parallel query mechanism upon both the language and translation models. We develop a hierarchical phrase parallel decoder by using a distributed memory database to store and query large-scale translation and language model tables. To further improve the speed of parallel decoding, we also make three additional optimizations: 1) Transform the synchronous rules in translation model table and the Trie data structure of language model table into a Hash indexed key-value structure for use in the distributed memory database; 2) Modify the cube-pruning algorithm to make it suitable for batch query; 3) Adopt and optimize the batch query for language model and translation model tables to reduce the network overhead. Our implemented algorithm can achieve fast decoding of SMT based on large-scale corpus and provide excellent scalability. Experimental results show that, compared with the single-machine decoder, our parallel decoder can reach 2.7 times of speedup for single sentence translation and reach 11.7 times of speedup for batch translation jobs, achieving significant performance improvement.

Key words statistical machine translation; hierarchical phrase; language model; translation model; parallel decoding; distributed memory

摘 要 近年来,为了提高统计机器翻译系统的准确性,普遍应用海量语料训练出大规模语言模型和翻译模型.而模型规模的不断增大,给统计机器翻译带来了突出的计算性能问题,使得现有的单机串行化翻译处理难以在较快的时间内完成计算,该问题在处理联机翻译时更为突出.为了克服单机机器翻译算

收稿日期:2013-10-12;**修回日期**:2014-01-08

法在这方面的局限性,提高大规模统计机器翻译处理的计算性能,面向一个实际的联机翻译系统,提出了一个分布式和并行化翻译解码算法框架,对整个大规模语言模型和翻译模型同时采用分布式存储和并行化查询机制,在此基础上进一步研究实现完整的翻译解码并行化算法.研究实现了一个基于分布式内存数据库的层次短语并行化机器翻译解码器,该解码器使用分布式内存数据库存储和查询大数据量的翻译模型表和语言模型表,克服了传统的机器翻译系统所面临的内存容量和并发度方面的限制.为了进一步提高并行解码速度,还研究实现了另外3项优化技术:1)将翻译模型表的同步规则和Trie树结构的语言模型表转化为基于内存数据库的"键-值"结构的Hash索引表的方法;2)对Cube-Pruning算法进行了修改使其适用于批量查询;3)采用并优化了批量查询方式减少语言和翻译模型查询时的网络传输开销.所提出的解码算法实现了基于大规模语料统计机器翻译时的快速解码,并具备优异的系统可扩展性.实验结果表明:与单机解码器相比,单句翻译速度可提高2.7倍,批量翻译作业的总体解码性能可提高至少11.7倍,实现了显著的计算性能提升.

关键词　统计机器翻译;层次短语;语言模型;翻译模型;并行化解码;分布内存

中图法分类号　TP391

统计机器翻译自从20世纪90代初由IBM提出,发展至今已经大行其道,其实用价值和研究热度远远大于传统机器翻译.近年来,大规模语料的应用给统计机器翻译带来了新的机遇和挑战.Google的研究结果表明,随着训练语料和语言模型的规模增大,统计机器翻译系统的翻译准确性会逐步提高[1].

统计机器翻译主要分为线下的语言模型和翻译模型训练阶段以及线上的翻译解码阶段.但是,翻译模型、语言模型和解码器作为统计机器翻译系统的3个主要组成部分,在处理海量语料时都会产生计算性能问题,使得现有的单机串行化翻译处理难以在较快的时间内完成计算,这种翻译性能问题在处理实时联机翻译时更为突出.

首先,在翻译模型和语言模型的构建阶段,处理TB级别的语料时,使用传统的单机处理方法进行训练需要耗时一周左右[2].在线下的翻译模型和语言模型的训练阶段,针对大规模语料的训练,文献[1-2]实现了基于MapReduce的翻译模型和语言模型构建的并行化算法,大大缩短了模型训练的耗时.但由于本文主要讨论线上翻译解码阶段算法的并行化,因此,这部分内容不在本文中展开讨论.

在通常的解码过程中,翻译一篇短文往往需要对语言模型进行数百万次的查询[3],对翻译模型的查询也接近这个数量级;因此,在实时联机翻译过程中,针对巨大的翻译模型表和语言模型表的查询效率问题,成为影响联机翻译解码过程性能的主要瓶颈.传统的单机机器翻译系统在数据存储和计算能力上的局限性,严重制约了解码过程中对大规模语言模型和翻译模型数据的存储和快速查询能力,如内存容量限制导致无法载入全部数据表内容、处理器核心数限制导致解码时并发度低等等.

目前在统计机器翻译领域,绝大多数研究工作主要还是致力于提高翻译精度,而采用分布式和并行化处理实现实时联机翻译的研究工作很少.

为了提高传统的单机翻译算法的速度,有一些针对语言模型查询性能的优化工作.文献[4]提出了采用布隆过滤器来处理超大规模的语言模型,但该方法会压缩数据量,从而降低了准确率;文献[5]通过对语言模型查询的左右状态的缓存,避免查询回退时的重复查询;文献[3]结合缓存和Hash索引提高了查询效率,但Hash索引由于需要额外的空闲空间导致空间浪费.但这些优化还是基于单机的实现,难以实现基于大规模语料的分布式和并行化翻译解码处理.

在翻译解码阶段,有少数研究工作考虑了使用分布式语言模型、以便能使用大规模语料和语言模型来提高翻译精度.针对大规模N元语言模型的存储和查询,文献[6]提出了分布式语言模型,并采用了2阶段的解码方式:第1阶段先使用单机翻译解码器得到初步的翻译结果,然后在第2阶段中用分布式语言模型对单机解码器的翻译结果进行重排序;文献[7]改进了这种2阶段的解码过程,使得语言模型的查询更有效.这2项研究工作主要是为了使用大规模语言模型来提高其对于评测数据集的翻译精度,因而其研究目的并不是为了解决联机翻译的计算性能问题.针对并行化快速解码,文献[5]提到其采用并行计算和分布式语言模型结合的方式,对解码过程进行加速,但在论文中并没有具体地描

述其实现方式.这3项研究工作有一个共同的问题,即都仅仅是针对机器翻译评测的目的,为提高评测数据集翻译精度对语言模型采用了分布处理,而并没有对翻译模型表也作分布式处理;它们针对评测数据集对翻译模型规则进行了过滤,仅仅保留了与评测数据有关的翻译模型数据,然后在单机上完成对翻译模型表的查询处理,过滤后的翻译模型数据比实际联机翻译所需要的翻译模型数据要低两个数据级.因此,这种针对评测目的所采用的局部分布处理方法并不是为解决计算性能问题所进行的真正意义上的分布式和并行化联机翻译研究工作.

针对目前机器翻译领域极少关注联机翻译性能问题的研究现状,本文着重研究面向实时联机翻译的分布式和并行化解码算法.与上述研究工作仅使用有限的分布式语言模型提高评测数据集翻译精度的目标不同,本文的研究目标是面向一个实际的联机翻译系统,研究提出一个完整的分布式和并行化翻译解码算法框架,对整个大规模语言模型和翻译模型同时采用分布式存储和并行化查询机制,在此基础上进一步研究实现完整的翻译解码并行化算法.为了解决单机机器翻译系统在计算性能和扩展性方面的问题,本文研究实现了一个基于分布内存的并行化统计机器翻译算法和系统,吸收已有的分布式语言模型架构,将其扩展后也用于分布式翻译模型,有效地实现了基于大规模语料统计机器翻译时的快速解码.为了完成并行化处理,在语言模型和翻译模型的组织上,将翻译模型表的同步规则和Trie树结构的语言模型表转化为分布内存数据库Redis可用的"键-值"结构;在解码处理阶段,为了进一步提高解码处理的速度,我们研究改进了Cube-Pruning算法,并研究实现了一种优化的批量查询方式以降低网络传输开销.我们同时结合了一些语言模型查询性能的优化,用于语言模型的本地缓存,即使用Trie树索引的本地缓存保存语言模型的若干临时状态值,用以降低解码时的网络传输开销.最后通过若干对比实验,展现了我们的并行化解码器比单机解码器拥有更好的性能和系统可扩展性.在统计机器翻译领域的研究文献中,由于目前尚未见到与本文研究目标相同的研究工作报道,因此本文的研究工作具有显著的创新性.

1 基于层次短语的统计机器翻译

统计机器翻译中的翻译模型,其发展经历了基于词的模型、基于短语的模型和基于句法的模型3个阶段.基于句法的模型是最复杂、也是翻译质量最好的模型,是当下的研究热点[8].我们的研究将考虑采用Chiang提出的层次短语模型[9],这是一种基于形式化句法的统计翻译模型,目前被广泛使用.

统计机器翻译的一个核心问题就是定义统计翻译模型,即一个源语言句子 s 翻译成一个目标语言句子 t 的概率 $P(t|s)$.基于层次短语的统计机器翻译使用对数线性模型进行描述[10].具体地,给定一个源语言句子 s,定义 s 的译文 $\hat{t}$ 如下:

$$\hat{t}(s)=\arg\max_t P(t|s)=\arg\max_t \prod_{i=1}^{m}[\phi_i(t,s)]^{\lambda_i}, \tag{1}$$

其中,$\phi_i(t,s)$ 是取值为概率的特征函数,λ_i 是 ϕ_i 的加权系数.对式(1)两边取对数,得:

$$\hat{t}(s)=\arg\max_t \sum_{i=1}^{m}\lambda_i h_i(t,s).$$

其中,$h_i(t,s)=\ln\phi_i(t,s)$.在各种特征函数中,最重要的2个特征函数为翻译模型和语言模型,下面分别介绍.

1.1 翻译模型

翻译模型刻画了源语言到目标语言词汇间的对应关系.基于层次短语的统计机器翻译通过概率同步上下文无关文法(probabilistic synchronous context-free grammar, PSCFG)来表示这种对应.PSCFG可以表示为四元组$\langle S,T,U,R\rangle$,其中 S 是源语言终结符集合,T 是目标语言终结符集合,U 是源语言和目标语言共享的非终结符集合,R 是同步规则集合.同步规则的形式为

$$X\rightarrow\text{diplomatic relations}\Leftrightarrow\text{外交 关系}, \tag{2}$$

$$X\rightarrow\text{one of } X\Leftrightarrow X\text{ 之 一}, \tag{3}$$

$$X\rightarrow\text{have } X_1 \text{ with } X_2\Leftrightarrow\text{和 } X_2 \text{ 有 } X_1. \tag{4}$$

一个同步规则由规约符号与一对对等的源语言和目标语言的符号串组成,其中源语言串和目标语言串中的非终结符一一对应.由于同步规则是从双语语料库抽取自动抽取得到的,为了防止同步规则数量过多,一般规定1个同步规则中最多包含2个非终结符.上述的3个同步规则示例分别有0,1,2个非终结符.例如上述的规则式(4)表示,源语言中的"have X_1 with X_2"形式的短语,可以规约为非终结符 $X(X\in U)$,同时对应的翻译为"和 X_2 有 X_1".规则中的"X_1"和"X_2"是非终结符,代表下一层被嵌套的短语.另外,每个规则 d 都有一个对应的权值 w_d,大小为该规则出现的概率的对数值.

当一个句子整体被规约为一个非终结符“X”时,这个句子会由同步规则层层嵌套形成类似句法树的结构,并由同步规则的目标语言端推导得出句子的译文.由此可定义该句子的翻译模型特征函数如下:

$$H(s,t)=\sum_{d\in D(s,t)} w_d,$$

$$D(s,t)=$$

{同步规则 $d|d$ 在 s 翻译为 t 的规约过程中被使用}.

$H(s,t)$的含义是该规约过程的概率的对数值.一个良好的原句子与译文的对应关系,必然由一个良好的、合理的规约过程得出,因此 $H(s,t)$较大.

所有的同步规则及其权值从双语语料库抽取、统计得到,存储在翻译模型表中.

1.2　语言模型

如果一个特征函数的取值与源语言无关,即特征函数 $L_j(t)=h_i(t,s)$,则称该特征为语言模型.语言模型刻画了译文自身的性质,可以直观地理解为,$L_j(t)$越大,表示译文越流利,或者说译文更符合人们的日常使用习惯.

最常用的语言模型是 N 元语言模型.它表示词序列的概率,即已有 $N-1$ 个词的序列时预测下一个词的出现概率.设用 w_m^n 表示词序列 $w_m w_{m+1}\cdots w_n$,$w_i(m\leqslant i\leqslant n)$为词序列中的单词.则该词序列的 N 元语言模型计算公式为

$$P(w_m^n)=\prod_{i=m}^{n} P(w_i|w_{i-N+1}^{i-1}).$$

由于 N 元语言模型的状态空间的稀疏性,计算该概率时常常需要进行回退.对于 $P(w_m^n)$的计算,如果上述连乘积中的某一项在语言模型表中不存在,其语言模型计算公式为

$$P(w_i|w_{i-N+1}^{i-1})=$$

$$\begin{cases}\rho(w_{i-N+1}^{i}), & \text{若 } \rho(w_{i-N+1}^{i}) \text{存在},\\ \beta(w_{i-N+1}^{i-1})\times P(w_{i-N+2}^{i}), & \text{否则},\end{cases}$$

其中,$\rho(w_{i-N+1}^{i})$是预先计算并存储在语言模型表中的概率对数值,$\beta(w_{i-N+1}^{i-1})$表示词序列 w_{i-N+1}^{i-1}的回退系数.回退系数的计算有多种方法,如较简单的、适合大规模语料训练的 Stupid Backoff,准确度高但计算较复杂的 Kneser-Ney 平滑算法等等[1].回退的意义在于将 N 元语言模型用 $N-1$ 元语言模型和回退系数的乘积表示.$N-1$ 元语言模型的计算同样遵循上述过程(连乘积和回退),这是一个递归的计算过程.

词序列的概率值和回退系数都是通过对单语言的规范语料统计得到,并存储在语言模型表中.

1.3　解码过程及其并行化

基于层次短语的解码过程类似于传统的句法分析过程,是一个 Chart-Parsing 算法.它是自底向上不断地使用同步语法规则对源语言句子进行规约,并在规约过程中伴随着剪枝,以防止上层结点的组合项过多.翻译一个句子的基本过程如下:

1) 首先从跨度为 1 的短语开始,应用层次短语的句法规则进行规约与翻译,生成上层结点;

2) 计算每个短语的权值(特征函数值的加权和),并保留 n-best;

3) 然后计算跨度为 2 的短语,重复上述步骤 1),2);

4) 不断扩展跨度,直到跨度为 N(N 为句子包含词的个数)时,其翻译就是整个句子的译文.

以上就是解码的大致过程,具体过程比较复杂,请参见文献[9].从上述过程可以看出,生成每层的结点时,需要对下层的所有短语组合进行枚举,因此会有大量的翻译模型的查询操作.每个短语又会有许多不同的翻译方式,因此语言模型的查询量更大.在有大数据的数据表时,这些查表操作会成为解码过程的瓶颈.因此,我们需要寻找好的分布存储和并行计算框架,能够很好地存储这些巨大的数据表,并支持快速的并行化查表操作.

有若干种将 N 元语言模型集成到解码器中的方法,包括重排序法、直接集成法和 Cube-Pruning 算法等[9].在以往的并行化解码器实现中,常见的做法是,先在单机上仅使用翻译模型表或者再加上小规模的语言模型表进行初步的粗翻译,然后使用多机的分布式语言模型表对这些“翻译假设”进行重排序[6].这种方法翻译速度很快,但准确性较差,并不比单机系统有显著提高.这里要将准确性较好的 Cube-Pruning 算法应用到我们的基于分布内存数据库的系统中.

另外,除了快速的查表操作,在解码过程生成每层结点时,如何并行化地快速生成结点,我们也在进一步研究当中.

2　基于分布式内存的并行化解码器

我们提出了一个新的基于分布式内存数据库系统 Redis 的并行化解码器框架,并在通用的分布式并行计算集群上实现来取代专用服务器上的单机解码器.

Redis 是一个开源的、高性能的“键-值”数据库,其数据全部存放在内存中,并提供了数据持久化功

能. 它支持多种数据结构,我们这里仅用到链表、Hash 表 2 种. Redis 的架构是简单的单机 C/S 架构,但我们可以用一些分布式算法使它能够用于分布式存储,例如简单 Hash、一致性 Hash 等等.

为了便于表示,规定如下的记号来表示键值对的数据结构:

1) 用$\langle k,v\rangle$表示一个键值对,其中 k 为"键",v 为"值";

2) 若"值"是一个链表,用$[a,b,\cdots]$表示,其中链表中的项依次为 $a,b,\cdots$;

3) 若"值"是一个 Hash 表,用$\{a,b,\cdots\}$表示,其中 Hash 表的项依次为 $a,b,\cdots$.

我们的并行框架采用分布式计算和并行计算相结合的方式. 集群的一部分结点将用于基于 Redis 的分布式数据存储,我们将翻译模型表、语言模型表使用一致性 Hash 将各个数据表项映射到集群的不同结点的 Redis 中. 解码器使用相同的映射函数,在翻译过程中通过网络远程从 Redis 中请求数据. 解码器自身具备缓存机制,它将从 Redis 获取的一部分数据保存在其中,如果以后查询的数据在本地缓存中存在,则直接从中获取,免去了网络查询的过程.

集群的另一部分结点用于同时运行多个解码器进行并行翻译,有 2 种工作方式:1)用于处理大量的分散翻译任务,比如提供在线翻译服务. 在这种方式下,当多个翻译请求(每个翻译请求为一个句子)到达时,每个句子由反向代理向集群内部的某个轻负载结点分配一个翻译任务,该机器启动一个解码器进程,负责翻译此句子. 2)用于处理批量式的翻译作业,比如翻译一篇文章. 这种工作方式基于 MapReduce 并行框架框架实现,由 MapReduce 框架将文章分割为句子组,每个 map 端负责处理一个句子组,map 端将会调用解码器完成翻译任务,最后由 reduce 端汇总结果.

基于 Redis 分布内存的并行化解码器框架如图 1 所示,其中数据结点、并行解码结点和管理结点只是职责上的划分,实际上它们可能是同一个物理结点.

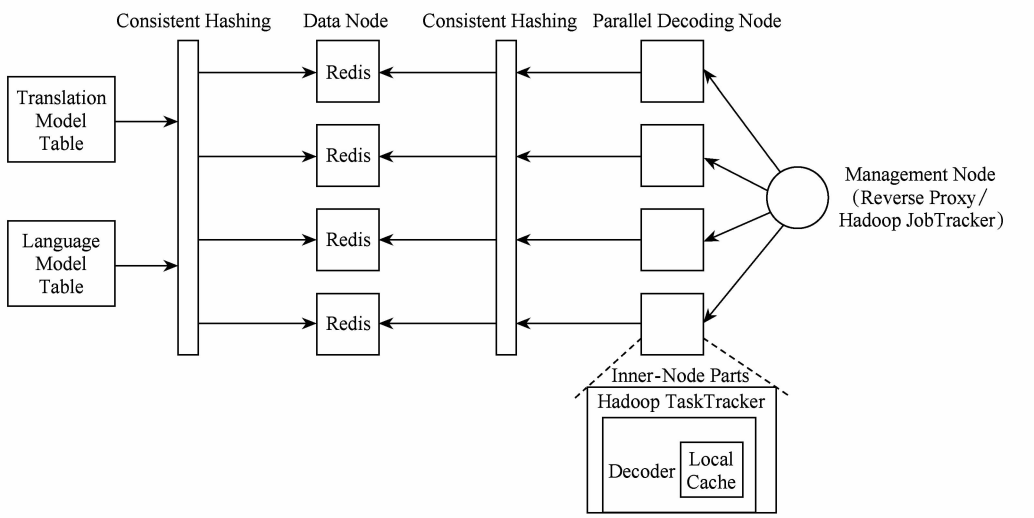


Fig. 1 Parallel decoder architecturebased on distributed memory of Redis.

图 1 基于 Redis 分布内存的并行化解码器框架

这个并行框架充分利用了集群的大容量分布式内存,大幅提高了并发度,而且可以在需要时增加集群结点以扩充系统,因而具有很强的系统扩展性,很好地克服了单机解码器系统难以扩展的缺陷.

采用分布式内存数据库存储数据后,由于每个结点存储的数据量变小(为总数据量的 N 分之一),且各结点可以同时进行检索,因此查表速度能够变快.

另外,我们还提供了数据表的 HBase 接口,以防全内存时内存不足. HBase 是一个基于 HDFS 的分布式数据库,它比 Redis 速度慢,但数据在磁盘上,可以提供几乎无限大的容量. 提供基于 HBase 的存储选项可以用于支持极大规模翻译语料时的翻译处理过程.

2.1 数据表的"键-值"结构

Redis 是一个"键-值"数据库,这个索引结构决定了我们可选的数据结构,即单层的 Hash 索引结构. 我们在把单机的数据表移植到 Redis 上时,为了适合这个结构,需要对数据格式进行重新定义.

对于层次短语的翻译模型表,我们采用和目前单机解码器类似的做法. 将所有的同步规则按照源语言串分组,源语言串相同的同步规则合并入同一组,将该组的源语言串作为"键",所有的目标语言串

及其权值等合并为一个链表作为“值”.

例如假设有如下的一些同步规则分为一组:

X→美国 的 X⇔X of the U.S.,

X→美国 的 X⇔X of America,

X→美国 的 X⇔X in the U.S.,

⋮

我们将“美国 的 X”作为键,所有的候选翻译项构成的链表“[X of the U.S.,X of America,X in the U.S.,…]”作为“值”.一个候选翻译项包含若干元素,如目标语言词串、一系列权值等等,而 Redis 链表的每一项只能是一个字节串,不可再分,因此需要我们将这些元素序列化为字节串后进行存储.

对于语言模型表,目前单机上的语言模型多用 Trie 树索引,我们需要把这种树形结构变为扁平的“键-值”结构.由 Trie 树结构得到扁平的“键-值”结构的方法为:遍历 Trie 树,到达任何一个非根结点时,生成一个串联该结点所有的祖先结点(靠近根结点的在前)得到的符号串为“键”、该结点为“值”内一个 Hash 表项的键值对.

例如存储“one of the best”的语言模型数据时,Trie 树会构造成“one—of—the—best”的 4 层树,one 为根结点.遍历该树,会生成“〈one:{of}〉”、“〈one of:{the}〉”和“〈one of the:{best}〉”的 3 个键值对.Trie 树结构转换为“键-值”结构的过程如图 2 所示:

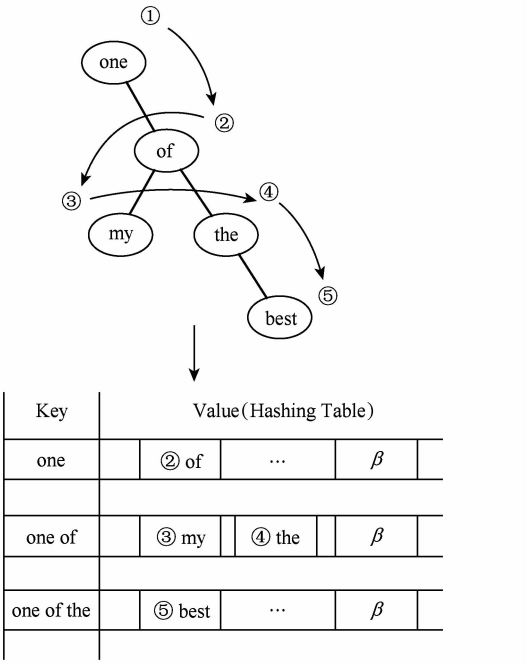


Fig. 2　General process of converting Trie tree structure into Key-Value structure.

图 2　Trie 树结构转化为键-值结构过程示意

除此之外,每个“键”的 Hash 表中还保存有一个特殊的值,即键对应词串的回退系数.

这样把词串的前一部分作“键”,最后一个词作“值”的方式,相对于把整个词串当作“键”,“值”中仅存放一个特征值的方式,相当于使用了二级索引,降低了查询时键的 Hash 值发生碰撞的概率.并且后文中将提到的解决语言模型回退的批量查询方式,在这种方式下能够减少 1 倍的查询次数(具体见 2.2 节).缺点是 Hash 索引所需的索引空间变大,浪费了更多的内存容量,但因为我们的框架拥有较好的可扩展性从而使该缺点变得次要.

2.2　批量查询

单机解码算法查询数据表时是“即需即查”的,每当有数据表的查询请求时都会立即处理.但在分布式系统中,网络传输的时间开销是内存的上千倍,所以这样做将大幅降低翻译处理的性能.翻译一句话需要查表数千次,为了减少查表的网络开销,我们采用了批量查询方式,将大量查询请求合并为一个请求.

由于解码过程所用的 Chart-Parsing 算法是一种动态规划算法,层与层之间具有依赖关系,而同层之间相互独立,所以我们的策略就是将同一层中的所有查询累积为一个批量查询请求,并在扩展下一层之前将这个批量查询请求发送出去.这样,翻译一句话只需要进行 N 次网络传输,其中 N 为句子包含词的个数.

为了能够使用批量查询方式,需要对 Cube-Pruning 算法作出一些修改.原有的 Cube-Pruning 算法是“即需即查”的,翻译模型和语言模型的查询混杂在一起.现在采用了批量查询的方式后显然无法这样做,因为在得到所有的翻译模型的查询结果之前无法预知需要进行哪些语言模型的查询.

将原有的 Cube-Pruning 算法的单层计算过程修改为 2 个阶段:1)单独处理翻译模型查询;2)单独处理语言模型查询.

1) 第 1 阶段

① 累积该层所有的翻译模型表的查询请求;

② 发送请求并取回结果,扩展上层结点;

③ 不考虑语言模型,对上层结点进行初次排序;

④ 取出 k-best.

2) 第 2 阶段

① 枚举 k-best 的所有语言模型表查询,将这些查询累积为一个批量查询请求;

② 发送请求并取回结果;

③ 重排序,再取 n-best.

上述过程中要保证 $k \geqslant n$. 这里我们取 $k=2\times n$.

文献[11]也对 Cube-Pruning 算法进行了类似地修改,但没有上述过程第 2 阶段的步骤③,而是将取回的结果放入一个堆中,得到了 Cube-Growing 算法的单层计算过程. Cube-Growing 算法和 Cube-Pruning 算法的本质区别是 Cube-Pruning 算法自底向上生成翻译假设,而 Cube-Growing 算法是自顶向下的. Cube-Growing 算法具备惰性求值的特性,避免了很多不必要的计算,因此速度较 Cube-Pruning 更快,但同时它的这一"即需即算"特性无法进行批量查询,因此难以应用到分布式系统中.

语言模型的回退需要额外的查表操作,同样地,在单机解码器中也是"即需即查"的方式,我们这里需要对其进行修改. 在批量查询下,为了避免回退带来的网络传输开销,需要使用"预取"的方式,即在一个批量查询请求中将 1～N 元的查询(包括概率值和回退系数)都一次性放进去. 例如我们要查询"we can do it"时,还同时要将"can do it","do it","it"的查询一同放进去,这样再计算回退的值时数据全部都在本地,无需经过网络查询.

在 2.1 节所使用的语言模型表结构中,我们使用了二级索引的方式,该方式给这种"预取"的查询方式减少了一半的查询次数,原因是词串的概率值和其回退系数位于同一个"键"下. 例如 ρ(we can do it)和 β(we can do)都在键"we can do"下,使用 Redis 的多 Hash 值查询命令一次就可以得到这两个值.

另外,我们将查询得到的语言模型数据以及计算得到的一些语言模型状态值,以 Trie 树索引保存在本机内存中作为缓存. 以后再查询这些值时,直接从缓存中取得,无需网络查询,从而在一定程度上也加快了速度. 缓存的容量是受限的,当保存的条目过多时,会按照一定的策略淘汰掉部分数据.

3 实验及结果

在引言部分中提到,目前机器翻译领域主要还是致力于研究更好的翻译算法并提高翻译的精度,几乎很少关注作为联机实时翻译应用时翻译算法的计算性能问题,尤其是面对大量数据时整个系统的计算性能瓶颈及系统可扩展性问题. 就我们所调查的相关工作研究文献看,目前在现有工作中尚未找到与本文研究目标相同、具有可比性的分布式和并行化联机翻译算法和系统.

此外,不同的解码算法本身有很大的差异,会导致并行化时的方法和框架差别很大,加之各家积累了很多年的解码算法大多很复杂并且不公开,因此,实际研究中也很难去重现不同的解码算法、设计出相应的分布式和并行化算法、并进行实验比较.

基于以上的实际原因,我们在设计实验时,为了能较好说明并行化带来的速度提升,仿照了文献[5]中测试并行化性能的实验做法,即使用我们的单机解码算法与分布式算法进行实验对照,比较其单句翻译速度和翻译作业批处理时的翻译速度. 另外,在此基础上,我们还分析了机器翻译系统不同的并发度对翻译速度造成的影响,以此来评价系统的计算性能瓶颈和可扩展性.

我们以基于层次短语模型的单机解码器作为基本的对比实验系统[12]. 并行解码器采用和单机系统完全一致的解码算法、数据集和参数,并且没有采用任何针对自然语言自身特性的性能优化. 这样做的目的是保证在 BLEU 得分基本一致的前提下,排除语言相关的优化对性能的影响,从而比较 2 个系统的计算性能和可扩展性.

单机解码器运行于专用服务器上,处理器 32 核 64 线程、主频 2.0 GHz、内存容量 128 GB. 我们的并行解码器运行在集群上,使用其中 30 个结点用于数据存储,17 个结点用于运行解码器进行并行翻译.

测试数据集来自 NIST 2012. 通过对中英双语语料的训练,得到了大小约 25 GB 的翻译模型文件和 4 GB 的语言模型文件. 将其转换后装入 Redis,在 30 个结点上占用的总内存量为:翻译模型 23 GB,语言模型 12 GB. 可以看出,"键-值"结构的翻译模型和原先大小基本一致甚至略小,而语言模型的大小是原先的 3 倍.

需要指出的是,这里的实验所使用的数据表大小几乎已经达到了单机解码器所在的专用服务器的容量上限,但在我们的并行化解码器所在集群上还远没有达到容量限制,在 30 个数据结点上平均每个结点仅占用 1.2 GB 左右的内存. 在将来面临更大规模的语料数据集时,单机的解码器将无法直接使用,需要采取一些优化措施,比如数据压缩、更节省空间的数据结构、过滤不相关数据等等,而并行化解码器将能够轻松应对.

对测试集中给定的 878 个测试语句进行翻译,得到在不同线程数(每个线程代表一个解码器)同时进行翻译的条件下,平均每个语句翻译耗时的对比结果如图 3 所示:

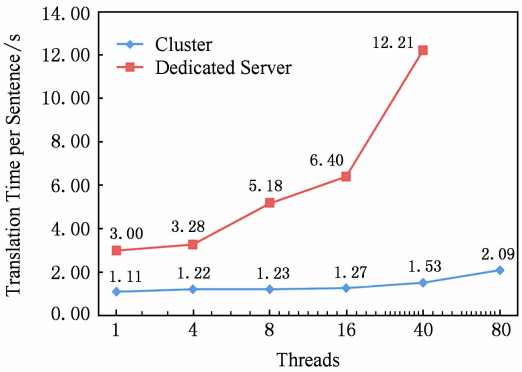


Fig. 3　Comparison for time cost per sentence.

图 3　单句平均耗时对比

首先是计算性能的对比.在单线程情况下,单机解码器平均翻译每个句子需要 3 s;我们的并行化解码器平均每句耗时约 1.1 s,解码速度是单机的 2.7 倍.在其他各个并发度级别下,我们的并行化解码器平均每句的性能也都同样保持了领先.

然后是可扩展性对比.实验结果显示,随着线程数量增加,单机解码器性能下降较快,在 80 线程时已经无法工作;而我们的并行化解码器在 80 个线程时仍保持较好的性能.

翻译完成测试集的 878 个测试语句需要的总时间如图 4 所示.可以看出,在并发度较低(线程数少于 16)时,单机解码器的系统可扩展性瓶颈还没有显现出来,并行解码器的翻译作业速度提升主要来自于单句翻译速度的提升,平均性能是单机解码器的 270%～420%.在并发度较高时,单机解码器的 I/O 瓶颈造成了其翻译效率下降明显,而并行解码器保持了良好的可扩展性,平均性能达到单机解码器的 500%～800%.

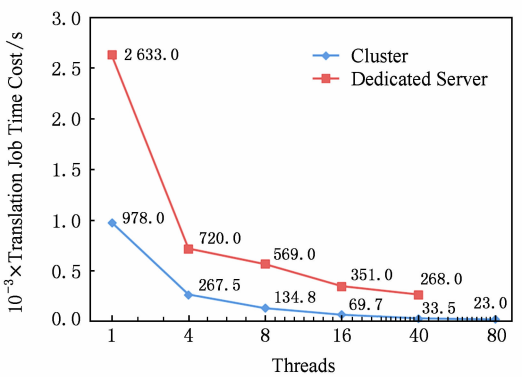


Fig. 4　Comparison for total time cost of batch translation job.

图 4　批量任务总耗时对比

完成整个翻译作业的最短用时,单机解码器需要 268 s,而并行化解码器需要 23 s,速度至少比单机解码器快 11.7 倍.事实上,并行化解码器在 80 线程下进行翻译时,集群的总体负载不算很高,因此如果继续提高并发度,理论上翻译速度还有提升的空间;而单机解码器在 40 线程下负载已经接近饱和,速度很难再有提高.我们没有作进一步的实测,因此在此不作比较.

综上所述,我们的系统采用了前述的分布式计算和并行计算结合的方式后,完成大批量翻译任务的总体性能很好,相比于单机算法速度有显著的提高.

4　结　　语

机器翻译领域主要致力于提高翻译精度,很少关注实时联机翻译计算性能问题,尤其是面对大量数据时的计算性能瓶颈及系统可扩展性问题.为此,面向一个实际的联机翻译系统,本文研究提出一个完整的分布式和并行化翻译解码框架,对大规模语言模型和翻译模型同时采用分布式存储和并行化查询机制,在此基础上进一步研究实现完整的翻译解码并行化算法.由于目前尚未见到与本文研究目标相同的研究工作报道,因此本文的研究工作具有显著的创新性.

为了解决单机机器翻译系统在计算性能和扩展性方面的问题,本文研究实现了一个基于分布式内存数据库的层次短语并行化机器翻译解码器.该解码器使用分布式数据库 Redis 来存储语言模型和翻译模型表,并采用并行化解码算法和模型表查询方式,有效克服了传统的机器翻译系统面对大数据量的翻译模型表和语言模型表时内存容量和并发度的限制,以及模型解码时的查表性能瓶颈.通过对翻译模型表的同步规则按照源语言串分组的方式,并将 Trie 树结构的语言模型表转化为扁平的 Hash 索引表的方法,研究实现了基于 Redis 的"键-值"结构的翻译模型表和语言模型表的组织和存储,在此基础上,为了能够进行批量查询以降低网络传输开销,我们对 Cube-Pruning 算法和批量查询方式进行了改进和优化.实验结果表明:并行化解码器的平均单句解码速度达到单机解码器的 2.7 倍,对于批量翻译作业的总体解码性能可提高至少 11.7 倍,显著提升了大规模机器翻译的计算性能,并实现了基于大规模语料机器翻译处理时良好的系统可扩展性.

参 考 文 献

[1] Brants T, Popat A C, Xu Peng, et al. Large language models in machine translation [C] //Proc of the 2007 Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Stroudsburg, PA: ACL, 2007: 858-867

[2] Lin J, Dyer C. Data-Intensive Text Processing with MapReduce [M]. San Francisco, CA: Morgan and Claypool Publishers, 2010

[3] Pauls A, Klein D. Faster and smaller n-gram language models [C] //Proc of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: ACL, 2011: 258-267

[4] Talbot D, Osborne M. Smoothed Bloom filter language models: Tera-scale LMs on the cheap [C] //Proc of the 2007 Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Stroudsburg, PA: ACL, 2007: 468-476

[5] Li Zhifei, Khudanpur S. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance [C] //Proc of the 2nd Workshop on Syntax and Structure in Statistical Translation. Stroudsburg, PA: ACL, 2008: 10-18

[6] Zhang Ying, Hildebrand A S, Vogel S. Distributed language modeling for *n*-best list re-ranking [C] //Proc of the 2006 Conf on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2006: 216-223

[7] Venugopal A, Zollmann A, Stephan V. An efficient two-pass approach to synchronous-CFG driven statistical MT [C] //Proc of Human Language Technologies 2007: The Conf of the North American Chapter of the Association for Computational Linguistics. Stroudsburg, PA: ACL, 2007: 500-507

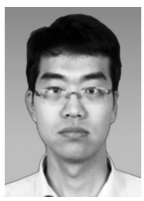
[8] Liu Qun. New progress in machine translation research [J]. Contemporary Linguistics, 2009, 11(2): 147-157 (in Chinese)
(刘群. 机器翻译研究新进展[J]. 当代语言学, 2009, 11(2): 147-157)

[9] Chiang D. Hierarchical phrase-based translation [J]. Computational Linguistics, 2007, 33(2): 201-228

[10] Och F J, Ney H. Discriminative training and maximum entropy models for statistical machine translation [C] //Proc of the 40th Annual Meeting on Association for Computational Linguistics. Stroudsburg, PA: ACL, 2002: 295-302

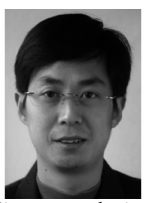
[11] Huang L, Chiang D. Forest rescoring: Faster decoding with integrated language models [C] //Proc of the 45th Annual Meeting of the Association of Computational Linguistics. Stroudsburg, PA: ACL, 2007: 144-151

[12] Xi Ning, Zhao Yinggong, Tang Guangchao, et al. NJU-NLP technical report for the 7th China workshop on machine translation [C] //Proc of CWMT2011. Beijing: Chinese Information Processing Society of China, 2011: 74-80 (in Chinese)
(奚宁, 赵迎功, 汤光超, 等. 南京大学第七届机器翻译研讨会评测技术报告[C] //机器翻译研究进展——第七届全国机器翻译研讨会论文集. 北京: 中国中文信息学会, 2011: 74-80)



Zhao Bo, born in 1988. PhD candidate in the Department of Computer Science and Technology, Nanjing University. His main research interests include text mining and big data parallel processing.

Huang Shujian, born in 1984. Received his PhD in computer science from Nanjing University. Assistant researcher in the Department of Computer Science and Technology, Nanjing University. His research interests include natural language processing, information extraction, machine translation, etc.



Dai Xinyu, born in 1979. Received his PhD degree in computer science from Nanjing University in 2005. Associate professor of the Department of Computer Science and Technology, Nanjing University. His research interests include on machine translation and information retrieval.

Yuan Chunfeng, born in 1963. Received her master degree in computer science from Nanjing University in 1987. Professor of the Department of Computer Science and Technology, Nanjing University. Senior member of China Computer Federation. Her research interests include Web data mining, text mining, information retrieval, and big data parallel processing.

Huang Yihua, born in 1962. Received his PhD degree in computer science from Nanjing University in 1997. Professor of the Department of Computer Science and Technology, Nanjing University. Senior member of China Computer Federation. His research interests include Web data mining, text mining and semantic analysis, RDF database, and big data parallel processing.