

基于 Spark 的协同过滤算法并行化研究

陆俊尧, 李玲娟

(南京邮电大学 计算机学院, 江苏 南京 210023)

摘要:协同过滤算法在推荐系统中应用广泛。但是随着数据量的爆炸式增长,协同过滤算法所需的计算量也随之增长。针对传统的单机集中式计算已无法满足推荐系统的实时性和扩展性要求的问题,基于主流的大数据平台 Spark 在迭代计算以及内存计算方面的优势,设计了一种基于项目的协同过滤算法在 Spark 上的并行化方案。该方案利用 RDD 并行化计算的特点,通过合理设计 RDD 算子来实现对物品间相似度计算过程和评分计算过程的并行化,同时采用了 RDD 的缓存机制以及 Spark 中的广播变量来对一些重要的计算资源进行缓存与分发,从而提高计算速度。用 MovieLens 公开数据集对基于 Spark 平台的并行化 Item-Based 协同过滤算法的性能进行测试,结果表明该并行化协同过滤算法在准确性以及时效性方面均有较好的表现。

关键词:协同过滤;Spark 平台;并行化;基于项目

中图分类号:TP301

文献标识码:A

文章编号:1673-629X(2019)01-0085-05

doi:10.3969/j.issn.1673-629X.2019.01.018

Research on Parallelization of Collaborative Filtering Algorithm Based on Spark

LU Jun-yao, LI Ling-juan

(School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract: Collaborative filtering algorithm is a widely used in the recommendation system. However, with the explosive growth of the amount of data, the amount of computing required by the collaborative filtering algorithm also increases. The traditional centralized computing of single machine has not been able to meet the requirements of the real-time and expansibility of the recommended system. Based on the advantages of the mainstream big data platform Spark in iterative computing and memory computing, we design a parallelization scheme of item-based collaborative filtering algorithm based on Spark. Based on the parallelization characteristics of RDD, it realizes the parallelization of items' similarity calculation and score calculation by reasonably designing RDD operator, at the same time using the cache mechanism of RDD and broadcast variables of Spark to cache and distribute some important computing resources, so as to improve the calculation speed. The performance of parallel item-based collaborative filtering algorithm based on Spark platform is tested by MovieLens dataset. The results show that this parallel collaborative filtering algorithm performs well in accuracy and timeliness.

Key words: collaborative filtering; Spark platform; parallelization; item-based

0 引言

推荐系统的目的是为用户进行精准高效的信息推送,在现有的推荐技术中,基于协同过滤的推荐技术是最为成功的。“协同过滤”^[1]一词最早是由 GlodBerg 等在 90 年代中期开发推荐系统 Tapestry^[2]时提出的,并且在后来得到了广泛的研究和应用。协同过滤推荐算法主要分为基于用户间相似度的 (User-Based)^[3-5] 协同过滤推荐算法和基于项目间相似度的 (Item-Based)^[6-9] 协同过滤推荐算法。前者基于用户之间的

相似度,为目标用户推荐其相近用户感兴趣的项目;后者计算项目之间的相似度,从而为目标用户提供与其感兴趣项目相似度较高的项目。

由于与日益增加的用户数量相比,项目的数量是相对稳定的。所以,基于项目间相似度的协同过滤算法可以有效地减少计算量,提高推荐的时效性。但是面对爆炸式增长的数据量,协同过滤算法的计算效率仍面临着挑战。

作为新一代大数据计算平台,Spark 具有基于内存

收稿日期:2018-01-27

修回日期:2018-05-27

网络出版时间:2018-09-21

基金项目:国家自然科学基金(61302158,61571238)

作者简介:陆俊尧(1993-),男,硕士研究生,研究方向为推荐系统;李玲娟,教授,通讯作者,研究方向为数据挖掘、信息安全、分布式计算。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20180920.1536.038.html>

计算的特性,同时也支持对加载到内存中数据的反复查询,非常适合大数据场景下的数据计算,已成为当今最为火热的大数据计算框架之一^[10]。

为了提高协同过滤乃至推荐系统的效率,研究了 Item-Based 协同过滤算法在 Spark 平台上的并行化方案,并通过实验对该方案的准确性以及时效性进行了验证。

1 Item-Based 协同过滤算法原理分析

在基于协同过滤的推荐系统中,最关键的部分就是用户-项目评分矩阵 $A(m,n)$,它由数量为 m 的用户集合 $U = \{u_1, u_2, \dots, u_m\}$ 对数量为 n 的项目集合 $I = \{i_1, i_2, \dots, i_n\}$ 的评分构成,用户 u 对项目 i 的评分用 R_{ui} 表示,推荐则基于评分矩阵按照评分高低进行。而基于协同过滤的推荐算法的任务就是完善和精确这个评分矩阵。

Item-Based 协同过滤算法完善用户-项目评分矩阵的过程大致如下:首先计算项目间的相似度,得到相似度矩阵 $\mathbf{Sim}_{n \times n}$;然后在完善用户对未浏览项 i 的评分时,根据相似度矩阵 $\mathbf{Sim}_{n \times n}$ 选取 K 个与 i 相似度最高的项目来组成最近邻居集 KNN_i ;再基于最近邻居集 KNN_i 利用评分公式进行计算,得到预测评分。

(1) 项目间相似度的计算方法。

定义 1:对 $\forall i \in I$,定义项目-评分矩阵 $A_{m \times n}$ 中对应于 i 的列为项目 i 的评分向量,记为 U_i 。

定义 2:对 $\forall u \in U$,定义项目-评分矩阵 $A_{m \times n}$ 中第 u 行对应于用户 u 的评分向量,记为 I_u 。

Item-Based 协同过滤算法中项目间相似度的计算主要涉及到用户对项目的评分向量,计算方法有标准的余弦相似度、修正的余弦相似度、相关相似度等。其中修正的余弦相似度计算方法如式 1 所示。

该方法是通过将用户对于项目的评分减去用户评分均值来表现用户对具体项目的评分与大众评分的差异性。

$$\cos^{\text{adjusted}}(i, j) = \frac{\sum_{u \in U} (R_{ui} - \bar{R}_u) * (R_{uj} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{ui} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{uj} - \bar{R}_u)^2}} \quad (1)$$

其中, \bar{R}_u 表示用户 u 评分的均值。

基于相似度计算公式进行项目间相似度计算,生成项目相似度矩阵 $\mathbf{Sim}_{n \times n}$ 。该矩阵由项目集合 $I = \{i_1, i_2, \dots, i_n\}$ 两两间的相似度 $\text{Sim}(i, j)$ 组成。

(2) 评分计算方法。

预测用户对于未评分或者未浏览项目 i 的评分的方法包括加权相似度预测方法和 Park 采用的预测方法等,其中 Park 采用的预测方法的计算公式如下:

$$P_{u,i} = \bar{R}_i + \frac{\sum_{j \in KNN_i} \text{sim}(i, j) (R_{uj} - \bar{R}_j)}{\sum_{j \in KNN_i} |\text{sim}(i, j)|} \quad (2)$$

基于最近邻居集 KNN_i ,利用评分公式得到目标用户对于项目 i 的评分,进而完善用户-项目评分矩阵,最后为用户推荐评分较高的项目。

2 Spark 大数据计算平台

Spark 是新一代大数据计算平台的代表,由 UC Berkeley 的 AMPLab 实验室提出。相较于传统大数据计算平台 Hadoop^[11],Spark 基于内存进行数据计算,原生语言采用了更为精炼简洁的 Scala,不仅可以进行数据的离线计算也可以进行实时计算,不仅可以利用自带的资源调度框架运行 (Standalone),还可以利用 Hadoop 的资源调度框架 Yarn、Mesos 运行,具有速度快、易用、适用范围广、可扩展等特点。

Spark 的核心机制包括弹性分布式数据集 RDD 和分布式运行架构等。RDD 是 Spark 中最为基本的数据抽象,代表一个由可分区、不可变、内部元素可并行化计算的集合。首先, RDD 由分区组成,分区是存取数据、进行计算的基本单位,其数量可由用户按需求自定义。在进行计算时, Spark 会根据分区中数据的物理位置进行计算的迁移,从而减少网络中数据的传输,提升运算速度,分区的存在同时还提升了计算过程中的并发度。其次,在 Spark 中,程序计算过程的本质就是不同 RDD 之间的转换,计算过程中的中间值与最终结果均被保存于一系列的 RDD 之中。每个 RDD 都可以通过持久化操作被存储于内存或者磁盘上,而且 RDD 间类似于有向无环图 (DAG) 的转换依赖关系也会被保存起来。当计算过程中发生错误导致 RDD 中数据丢失时,若该 RDD 进行过持久化操作,则可以直接进行调用恢复;否则,根据 RDD 中的依赖关系,从前往后重新进行计算,这在很大程度上提高了 Spark 的容错性。

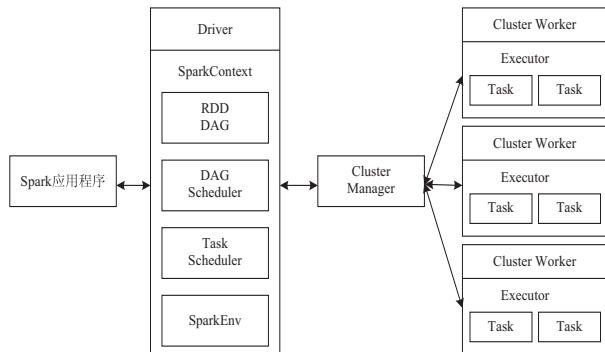


图 1 Spark 的分布式运行架构

Spark 先进的分布式运行架构如图 1 所示。当用户向 Spark 集群提交一个计算任务时,驱动程序 (Driver) 会向资源管理器 (Cluster Manager) 申请资源。当

资源分配完成后,Spark 便开始在工作节点(Worker)启动多个任务执行器(Executor),然后等待 Driver 将规划完成的计算任务集合(即 RDD 间转换的步骤,类似于一个有向无环图 DAG)发送到 Worker 上。最终计算完成后,Worker 将结果发回到 Driver。

3 基于 Spark 的协同过滤算法的并行化方案设计

由于 Item-Based 协同过滤算法在计算过程中需要对数据进行反复的迭代计算,而 Spark 的 RDD 机制正好可以契合这个需求,因此,文中基于 RDD 转换设计了基于 Spark 的 Item-Based 协同过滤算法的并行化方案,同时利用 Spark 中 RDD 缓存的特性来对一些计算消耗量极大的中间结果进行存储,采用 Spark 的广播变量的机制减少计算过程中的网络数据传输^[12]。方案的具体过程如下:

(1) Spark 的配置与数据源的读取。

Spark 的驱动器程序会读取相关的配置文件生成 SparkConf 对象,然后基于该对象进一步生成 SparkContext 对象去连接 Spark 集群。而 Spark 的计算流程就是通过读取外部数据源或者 Spark 内存中的数据将其转换为源 RDD,然后利用 RDD 的算子操作进行不同 RDD 间的转换,最终得到结果 RDD。在计算过程中,一个 RDD 分区就会生成一个计算任务。如果 RDD 分区数量与 Spark 集群为程序分配的计算资源不匹配,会导致 Spark 的并行化计算效率降低。因此,需要对源 RDD 的分区数量进行合理调整。

(2) 多节点并行化执行 Item-Based 协同的过滤算法。

基于上文的分析,Item-Based 协同过滤算法的执行过程被分为两大部分:项目间相似度计算以及评分计算。具体过程如下所述,其中 Step1 ~ Step5 为项目间相似度计算过程,Step6 ~ Step7 为评分计算过程。

Step1:Item-Based 协同过滤算法需要涉及到用户对项目评分的历史记录,需要其中的用户编号 $userId$ 、项目编号 $itemId$ 、用户对项目的评分 $rate$ 三个字段。因此,先对原始数据进行预处理生成包含这三个字段的源文件。然后读取源文件,进行字段切分,转换成格式为 $(userId, itemId, rate)$ 的元组组成的 RDD_{source}。

Step2:采用修正的余弦相似度进行项目间相似度的计算,会涉及到每个用户的评分均值。因此,获取 RDD_{source} 中的 $userId$ 和 $rate$ 字段,用 reduceByKey 操作将具有相同 $userId$ 的元组聚合到一起以后,用 avgRateCalculate() 这个自定义的平均分计算方法计算用户评分均值并存入 RDD_{avgRate}。RDD_{avgRate} 由格式为 $(userId, avgRate)$ 的元组组成。作为分布式计算框架,Spark 默

认会为多个并行操作中所用到的同一个变量分别进行发送,这会导致计算过程中网络上存在大量的数据传输,影响计算效率。针对该问题,文中采用 Spark 的广播变量机制来提高计算效率,将有关数据封装成广播变量的数据分发到各个计算节点进行保存且只发送一次,后续计算节点需要用到该数据时直接从本地获取,不依赖网络传输。由于 RDD_{avgRate} 中的数据在后续计算过程中会被反复查询,因此先用 collect 操作把 RDD_{avgRate} 中的数据从各 Worker 节点汇总到 Driver 中,用 toMap 操作转化为 Map 集合 Map_{avgRate},再将其作为广播变量广播到各 Worker 节点。

Step3:获取 RDD_{source} 中的 $userId$ 、 $itemId$ 、 $rate$ 字段,将 $itemId$ 与 $rate$ 用“-”符号连接成“项目-评分”字段。再利用 reduceByKey 操作将具有相同 $userId$ 的元组聚合到一起,构成存储用户历史评分信息的 RDD_{histRate}。RDD 由格式为 $(userId, itemId_1 - rate_1, itemId_2 - rate_2, \dots, itemId_N - rate_N)$ 的元组构成, N 为编号为 $userId$ 的用户评论过的项目数量。由于后续的评分计算过程中同样还涉及到对用户历史评分的查询,因此利用 toMap 操作转化为 Map_{histRate} 后,再将其封装为广播变量,分发到各个 Worker 节点。

Step4:基于 RDD_{histRate},对于每个 $userId$,将其对应的由“ $itemId - rate$ ”字段组成的集合进行两两间类似笛卡尔积的合并。先将格式为 $(userId, itemId_1 - rate_1, itemId_2 - rate_2, \dots, itemId_N - rate_N)$ 的元组利用 map 操作变成由格式为 $(itemId_i - itemId_j, rate_i - rate_j)$, $i, j \in [1, N]$ 的元组组成的 List 集合。接着利用 flatMap 操作将 List 集合中元素取出,再用 reduceByKey 操作进行聚合,最终得到被用户同时评论过的两个项目的评分汇总,存放于 RDD_{rateSum} 之中。RDD_{rateSum} 由格式为 $(itemId_i, itemId_j, rate_i - rate_j)$, $i, j \in [1, N]$ 的元组组成。

Step5:以 RDD_{rateSum} 和 Map_{avgRate} 为参数,利用自定义的函数 similarityCalculate() 进行评分计算,最终得到包含所有项目之间相似度的结果 RDD_{similarity}。该 RDD 由格式为 $(itemId_i, itemId_j, Sim(i, j))$, $i, j \in [1, N]$ 的元组组成,其中 M 为项目的总数量。由于相似度的计算消耗大量资源,故采取 Spark 中 RDD 的缓存机制将 RDD_{similarity} 缓存起来,避免后续计算中因数据丢失而产生的重复计算。由于后续计算过程中也要对 RDD_{similarity} 中的数据反复查询,因此也将 RDD_{similarity} 封装成广播变量发送到每一个 Worker 节点。

Step6:对目标项目 i 进行预测评分的计算。基于 RDD_{similarity},利用 filter 操作,将 RDD_{similarity} 中 key 值包含项目 i 的元组过滤出来,然后将其转化为 List 集合,对 List 集合按照相似度值进行从大到小的排序,最后获

取排序后的 List 的前 K 个值,组成项目 i 的最近邻居集合 KNN_i , KNN_i 也是一个 List 集合。

Step7:以 KNN_i 以及 $Map_{histRate}$ 为输入参数,利用自定义函数 $rateCalculate()$ 进行预测评分的计算。该函

数使用 Park 采用的预测方法作为计算公式进行评分计算,最终获得用户 u 对未评分的项目 i 的预测评分 R_{ui} 。

以上过程中涉及到的 RDD 转换流程如图 2 所示。

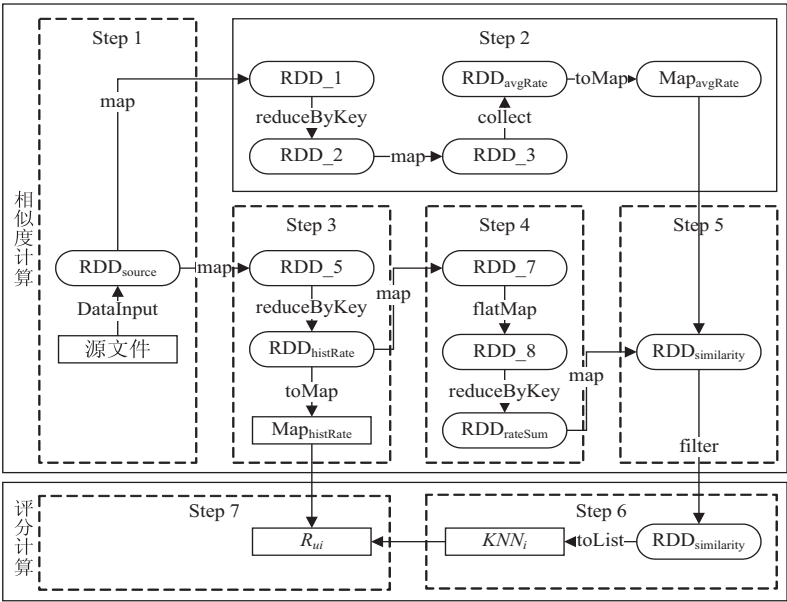


图 2 RDD 的转换流程

4 实验与结果分析

为了测试和验证基于 Spark 的 Item-Based 协同过滤算法的准确性与时间效率,采用 MovieLens 数据集对基于 Spark 的并行 Item-Based 协同过滤算法进行性能测试。首先利用不同比例的测试集和训练集对该算法的准确性进行测试。然后,分别在单机和 Spark 集群上运行相同规模的数据集,进行算法时效性的对比实验。用 Scala 语言进行算法实现。

(1)实验环境和数据。

实验配置的 Spark 环境包含了三个节点,一台驱动节点 Master,两台执行节点 Worker。每个节点的 CPU 版本信息为 Intel CORE i5-4210H,每个 CPU 都拥有两个处理单元,硬盘的读写速度为 600.00 MB/s, Master 节点配有 6 G 内存,其余 Worker 节点为 4 G。Spark 版本为 1.6.1;Spark 运行的操作系统为 CentOS 6.5;Java 版本为 JDK1.7.0_13;Scala 版本为 2.10.4。

实验使用了 GroupLens Research 提供的 MovieLens 数据集^[13],该数据集提供了三种大小不同的数据集,分别为 100 k,1 M 和 10 M。实验采用了大小为 100 k 的数据集,其中包含 943 位用户对 1 682 部电影共计 10 万条评分记录。记录中包含 user id、item id、rating、timestamp 四个字段,分别表示用户编号、电影编号、电影评分、评分时间戳。

(2)评分预测准确性测试。

衡量推荐系统推荐准确度^[14]的最重要指标就是

其评分预测准确性,通常采用均方根误差(RMSE)和平均绝对误差(MAE)来计算,两者可反映出真实值和预测值之差,两者的值越小,说明评分预测的准确性就越高。

实验采用不同比例的训练集和测试集,利用 MAE 值进行准确度计算。结果如表 1 所示。

表 1 不同的训练集和测试集比例下的 MAE 值

指标	6 : 4	7 : 3	8 : 2	9 : 1
MAE	0.852 4	0.831 2	0.815 9	0.794 3

可以看出,运行在 Spark 集群上的 Item-Based 协同过滤算法的 MAE 值较小,而且 MAE 值随着训练集比例的提高而降低,即历史数据越丰富则推荐准确度也越高,这体现出了大数据计算框架的重要性,因为它可以处理更多的历史数据,从而带来更高的精确度。

(3)算法执行时间测试。

分别在单机和 Spark 集群上针对算法运行时间做了测试实验。训练集和测试集比例为 9 : 1,数据集样本数量逐渐递增。实验结果如图 3 所示。

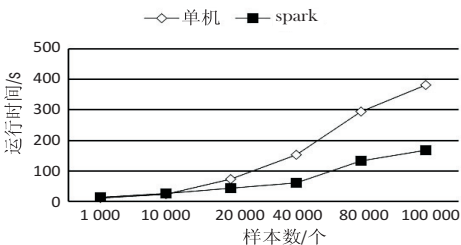


图 3 基于单机和 Spark 集群的运行时间

由实验结果可知,在数据量较小时,由于 Spark 的启动需要消耗大量资源以及时间,因而无法体现并行化算法在时间效率方面的优势,但随着数据量的增加,其时间效率明显提升。

5 结束语

设计了一种 Item-Based 协同过滤算法在 Spark 集群中的并行化方案,并通过基于 MovieLens 数据集的实验结果证明,在应对大规模数据处理时,基于 Spark 的并行化 Item-Based 协同过滤算法,不仅可以保证评分的准确性,而且算法执行速度更快,可以提高推荐系统的时效性。

参考文献:

[1] GOLDBERG D,NICHOLS D,OKI B M,et al. Using collaborative filtering to weave an information tapestry[J]. Communications of the ACM,1992,35(12):61-70.

[2] 冷亚军,陆青,梁昌勇. 协同过滤推荐技术综述[J]. 模式识别与人工智能,2014,27(8):720-734.

[3] BELLOGÍN A,CASTELLS P,CANTADOR I. Neighbor selection and weighting in user-based collaborative filtering[J]. ACM Transactions on the Web,2014,8(2):1-30.

[4] 李涛,王建东,叶飞跃,等. 一种基于用户聚类的协同过滤推荐算法[J]. 系统工程与电子技术,2007,29(7):1178-1182.

[5] 何哲. 基于用户聚类的推荐算法研究[J]. 科技创业月刊,2017,30(10):135-136.

[6] 邓爱林,左子叶,朱扬勇. 基于项目聚类的协同过滤推荐算法[J]. 小型微型计算机系统,2004,25(9):1665-1670.

[7] KIM B M,LI Q,PARK C S,et al. A new approach for combining content-based and collaborative filters[J]. Journal of Intelligent Information System,2006,27:79-91.

[8] 邓爱林,朱扬勇,施伯乐. 基于项目评分预测的协同过滤推荐算法[J]. 软件学报,2003,14(9):1621-1628.

[9] TIAN X H. PCIB:a new algorithm for item-based collaborative filtering recommendations[C]//Proceedings of 2014 international conference on artificial intelligence and industrial application. [s. l.]:Advanced Science and Industry Research Center,2014:9.

[10] 杨志伟. 基于 Spark 平台推荐系统研究[D]. 合肥:中国科学技术大学,2015.

[11] 赵娟,程国钟. 基于 Hadoop、Storm、Samza、Spark 及 Flink 大数据处理框架的比较研究[J]. 信息系统工程,2017(6):117.

[12] 唐振坤. 基于 Spark 的机器学习平台设计与实现[D]. 厦门:厦门大学,2014.

[13] 徐新瑞,孟彩霞,周雯,等. 一种基于 Spark 时效化协同过滤推荐算法[J]. 计算机技术与发展,2015,25(6):48-55.

[14] 李成,冯青青. 推荐系统准确度衡量方案—引入权重概念[C]//工业设计研究. 出版地不详:出版者不详,2017:269-275.

(上接第 84 页)

tual data[C]//Proceedings of the 9th international natural language generation conference. [s. l.]:Association for Computational Linguistics,2016:143-152.

[27] MEI H,BANSAL M,WALTER M R. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment[C]//Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics;human language technologies. [s. l.]:Association for Computational Linguistics,2016:720-730.

[28] LEBRET R,GRANGIER D,AULI M. Neural text generation from structured data with application to the biography domain[C]//Proceedings of the 2016 conference on empirical methods in natural language processing. [s. l.]:[s. n.],2016:1203-1213.

[29] REITER E,SRIPADA S,HUNTER J,et al. Choosing words in computer-generated weather forecasts[J]. Artificial Intelligence,2005,167(1-2):137-169.

[30] CHEN D L,MOONEY R J. Learning to sportscast;a test of grounded language acquisition[C]//Proceedings of the 25th

international conference on machine learning. Helsinki, Finland:ACM,2008:128-135.

[31] DAHL D A,BATES M,BROWN M,et al. Expanding the scope of the ATIS task;the ATIS-3 corpus[C]//Workshop on human language technology. [s. l.]:[s. n.],1994:43-48.

[32] GATT A,KRAHMER E. Survey of the state of the art in natural language generation;core tasks,applications and evaluation[J]. Journal of Artificial Intelligence Research,2018,61:65-170.

[33] GKATZIA D,MAHAMOOD S. A Snapshot of NLG evaluation practices 2005-2014[C]//Proceedings of the 15th European workshop on natural language generation. [s. l.]:Association for Computational Linguistics,2015:57-60.

[34] REITER E,ROBERTSON R,OSMAN L. Types of knowledge required to personalise smoking cessation letters[C]//Joint European conference on artificial intelligence in medicine and medical decision making. [s. l.]:[s. n.],1999:389-399.

[35] PAN S J,YANG Q. A survey on transfer learning[J]. IEEE Transactions on Knowledge and Data Engineering,2010,22(10):1345-1359.