基于序列感知的推荐算法研究

摘　要

随着互联网信息的不断增长，推荐系统已成为克服信息过载的有效策略。推荐系统作为帮助人们从浩如烟海的信息中发现自己所需的重要工具，在许多网络应用中被广泛集成，不仅辅助人们做出选择还帮助服务商提升了交易量，推荐系统的效用不容小觑，已成为信息社会日常生活的重要组成部分。推荐算法作为推荐系统背后的核心力量，支撑着推荐系统的表现。因此研究泛化能力强鲁棒性好的推荐算法一直是数据挖掘领域的热点。在该领域内的研究通常基于构建用户与物品的交互矩阵，这样的算法空间复杂度过高，已逐渐被深度学习方法替代。并且在现代推荐算法中，大多数方法都忽略了用户所消费物品之间的时序关系，基于深度学习的模型还需要构造大量的用户隐私相关的特征，本文讨论的序列感知推荐方法则避免了这些问题。

近年来，深度学习在计算机视觉和自然语言处理等许多研究领域引起了相当大的兴趣，这不仅源于其出色的大数据计算能力，还归功于其能从原始数据学习特征表达的迷人特性，这一影响早已扩散到了信息检索和推荐系统的研究领域。因此本文引入在自然语言处理领域表现出色的序列建模算法，创新地引入推荐系统领域，提出了基于双向长短期记忆网络的新颖序列感知推荐算法 BiLSTM4Rec，考虑到循环神经网络的并行效率问题，又引入自注意力机制提出了 Transformer4Rec 序列感知算法。两种算法都利用神经网络技术挖掘用户行为时序特征进行序列预测，并从推荐结果精确性和计算效率方面进行了分析，为现有推荐算法遇到的问题提出了可行地解决思路。

为了验证本文提出模型的性能，分别在真实的用户行为数据集 Movielens，Yoochoose 上进行了实验，实验结果表明，BiLSTM4Rec 和 Transformer4Rec 的序

列感知推荐模型，面对稀疏、大规模数据时，也能较好地反映用户的短期兴趣，并且 Transformer4Rec 在大数据面临效率更胜一筹。

关键词：推荐系统；序列建模；循环神经网络；注意力机制；数据挖掘

Research on Sequence-aware Recommendation Algorithm

ABSTRACT

With the continuous growth of Internet information, the recommendation system has become an effective strategy to overcome information overload. The recommendation system, as an important tool to help people find interest in the vast amount of information, has been widely integrated into many network applications, which not only helping people make choices but also helping service providers to increase the volume of transactions. The effectiveness of the recommendation system should not be underestimated. The recommendation system has been an important part of the daily life of the information society. The recommendation algorithm serves as the core force behind the recommendation system, it supports the performance of the recommendation system. Therefore, it is always a hotspot in the field of data mining to research the recommendation algorithm with strong generalization ability and strong robustness. Research in this field is often based on building interaction matrices between users and items, such algorithms are too spatially complex and have been gradually replaced by deep learning methods. However, in the modern recommendation algorithm, most methods ignore the time-series relationship between the items consumed by users, and the deep learning-based model also needs to construct a large number of user-private features. The sequence-aware recommendation method discussed in this paper avoids these problems.

In recent years, deep learning has attracted considerable interest in many fields of research, such as computer vision and natural language processing, not only because of its excellent big data computing power but also because of its fascinating properties of learning from the original data. This influence has long spread to the research field of

information retrieval and recommendation systems. Therefore, this paper introduces the excellent sequence modeling algorithms in the field of natural language processing into the recommendation system field. A novel sequence-aware recommendation algorithm BiLSTM4Rec based on bidirectional LSTM networks is proposed. Considering the parallel efficiency problem of the cyclic neural network, a self-attention mechanism, Transformer4Rec, is introduced to propose the sequence perceptual problem. Both algorithms use neural network technology to mine user behavior time series features for sequence prediction. We analyze the accuracy and computational efficiency of recommendation results, and propose feasible solutions for the problems encountered by existing recommendation algorithms.

In order to verify the performance of the proposed model, experiments were carried out on the real user behavior dataset Movielens and Yoochoose. The experimental results show that the sequence-aware recommendation models, BiLSTM4Rec and Transformer4Rec, are also acceptable in the situation of sparse and large-scale data. The location reflects the user's short-term interest, and Transformer4Rec is more efficient when facing big data.

KEY WORDS: Recommendation system; Sequence modeling; Recurrent neural network; Attention mechanism; Data mining

目 录

第 1 章　绪论

1.1　研究背景与意义

我国经过二十余年的信息化建设，互联网用户仍在持续增长，根据中国互联网信息中心发布的第 43 次《中国互联网络发展状况统计报告》[1] 显示，截止 2018 年 12 月，我国网民规模达 8.29 亿，较 2017 年末仍增加了 3.8%，互联网普及率达 59.6%，其中移动互联网用户比例更是高达 98.6%。浩如烟海的数据填充着互联网，人们要从海量数据中找到自己关心的部分变得越来越难。推荐系统是帮助用户从海量产品集合里面找到感兴趣目标的软件应用，具有千人千面的特点，也是数据挖掘和机器学习相关科技在实践领域最成功的应用之一。在许多网站和应用程序中，例如电子商务、新闻和视频网站、音乐和广播电台等，他们都需要为用户推荐可能喜欢物品的杰出服务，如今接收不同形式的自动推荐已经成为我们日常在线用户体验的一部分。在典型的在线网站上面，可以收集用户各种类型的相关动作，例如：用户点击、浏览、收藏、购买了某个商品。推荐系统（RS）已发展成为帮助用户做出明智决策和选择的基本工具，尤其是在大数据时代，客户必须从大量产品和服务中做出选择。因此现代推荐系统是在当前大数据环境下应运而生的，现代推荐系统的架构如图1-1所示，本文讨论的问题主要出于推荐算法层面。

推荐算法作为推荐系统的灵魂，承担着推荐高品质项目和高效反馈的重任，因此设计鲁棒性好的推荐算法有着重要的研究意义。现有的推荐策略都主要关注如何为用户或项目找到临近集，或者利用其它的显式或隐式信息（如标签、评论、项目属性和用户个人信息）来提升近邻感知能力。然而，这些静态算法都没有考虑用户兴趣变化的实时动态性。时序信息就是反应用户兴趣实时变化的一个重要特征，在许多任务比如用户下一行为预测中，用户下一首会听什么歌与用户的喜好、当前所处环境的上下文都高度相关。

推荐系统面对的任务主要有两部分：评分预测和产品推荐。所以根据用户的历史记录预测他下一次会选择什么也是推荐领域一个严峻的挑战。

因此本文提出序列建模的推荐思路，本文的研究内容是结合神经网络的序列

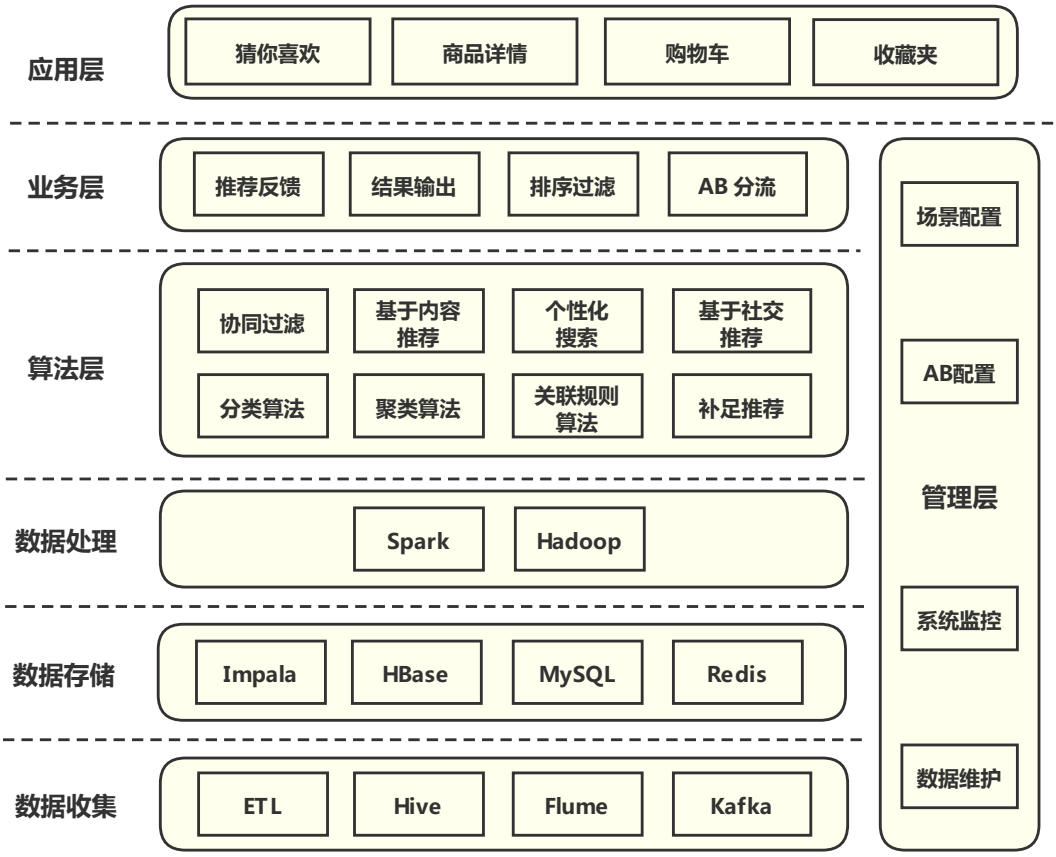

图 1-1　现代推荐系统架构图

建模技术和推荐系统相关理论知识，实现快速向用户推荐可能感兴趣的物品。充分利用深度神经网络强大的建模能力，提取用户行为的重要时序特征，构建推荐对象的兴趣模型，构建相关模型来识别用户的兴趣意图，综合考虑推荐对象和用户两者之间的特征信息挖掘两者之间的隐式联系，从而发掘出用户感兴趣的物品，以此来帮助用户快速的找到有感兴趣的物品，提升网络应用的流量及用户的黏性。

本章节主要从传统机器学习和深度学习两个方面介绍现代推荐系统中常用的各种推荐算法，从它们各自的特点分析现如今存在的问题。

1.2　国内外研究现状

为了分析和理解国内外现有推荐算法的各自特点，本章节根据其使用的技术不同分为两个大类分别进行描述，同时给出了如图1-2的思维导图。

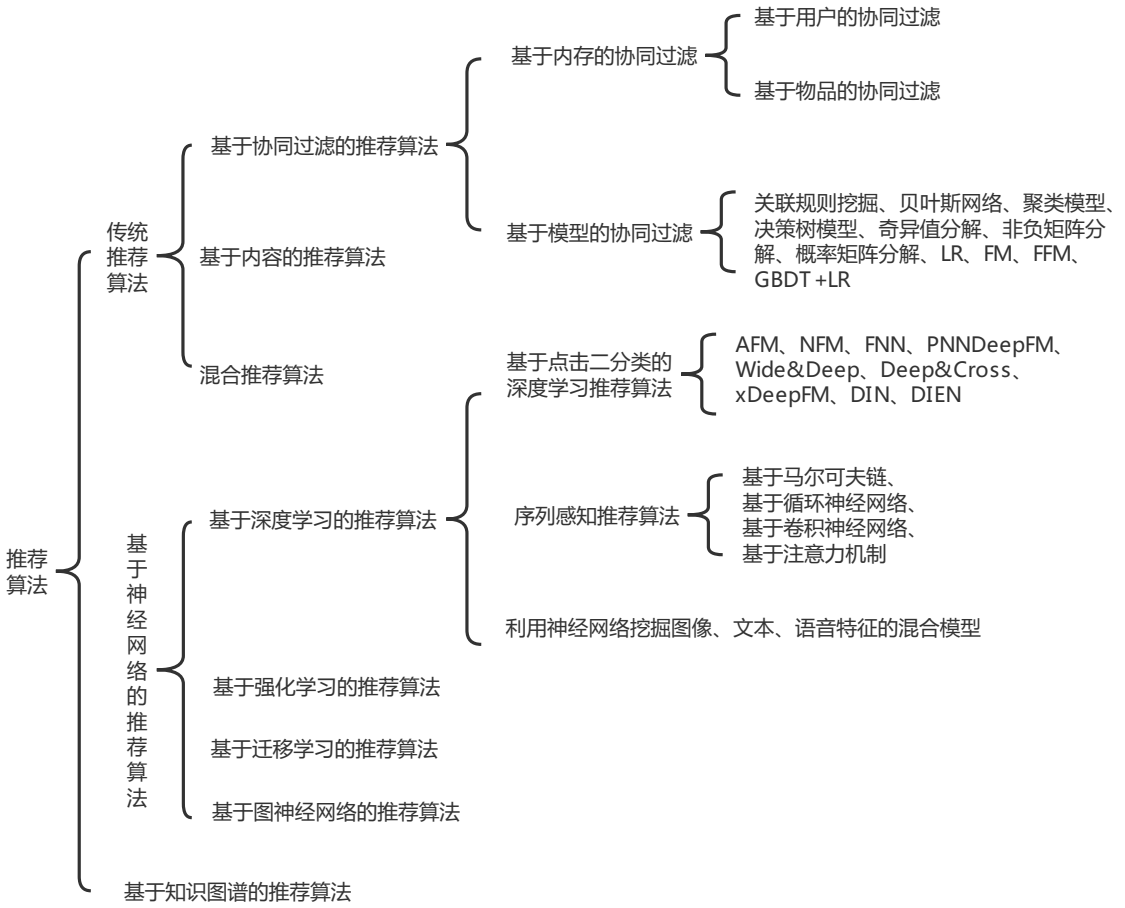

图 1-2　推荐算法分类思维导图

1.2.1　传统推荐算法

传统的经典推荐算法主要可分为三大类，它们分别是基于协同过滤的推荐算法、基于内容的推荐算法和混合推荐算法。协同过滤算法大体上可以分为基于内存的协同过滤 (Memory-based CF) 和基于模型的协同过滤 (Model-based CF)。基于内存的协同过滤通过为用户寻找具有相似行为的用户或有相似特点的物品集合做推荐，所以可细分为基于用户的协同过滤 (User-based Collaborative Filtering，UBCF) 和基于物品的协同过滤 (Item-based Collaborative Filtering，IBCF)。而基于模型的协同过滤则主要利用评分信息训练相应的模型，然后使用这个模型对未知数据进行预测，这类算法有贝叶斯网络[2]、聚类模型[3]、概率矩阵分解[4] 等等。在基于内容的推荐算法中仅利用了单个用户的行为和数据给出对用户的推荐，特别是物品的描述和用户的属性描述在内容推荐中起到了关键作用。基于内容的推荐

精确度往往有限，还面临着严重的冷启动问题。混合推荐系统则通过将上述两大类算法中的一个或多个结合起来以避免和克服某个单一算法带来的缺点。混合推荐算法中比较常见的方式是将基于内容的推荐方法与其他方法进行融合以避免冷启动、数据稀疏和扩展性等问题。基于传统推荐算法的特点，基于协同过滤的推荐算法中大多需要构造一个用户与物品的交互矩阵，随着大数据的极速发展，物品与用户的数量往往能达到上亿规模，使得交互矩阵构造的空间复杂度过高，在大数据时代传统推荐算法已慢慢无法解决当前所面对的问题。

1.2.2 基于深度学习的推荐算法

得益于神经网络的反向传播理论，批量梯度下降的网络权重优化方式使得机器学习特别是深度学习能处理的数据规模没有理论上的数量限制，现代推荐算法也借助神经网络的技术蓬勃发展。神经网络领域下面子领域里的许多技术都已经被应用于推荐系统当中，包括深度学习、强化学习、迁移学习和图神经网络。本文仅讨论基于深度学习的推荐算法。

学术界推荐算法的发展离不开工业界的实践，而推荐算法在工业界的实践应用也为学术界的发展指引着方向。推荐算法在工业界的一个重要实践就是进行点击率预估。随着深度学习技术被引入点击率预估方面，开始有大量的成果诞生。点击率预估的通常做法是将推荐问题转化为二分类的有监督学习问题，算法建模的样本对象是物品，将用户的特征与物品特征拼接在一起构造高维样本，预测对象用户是否可能会点击目标物品。在深度学习技术没有被引入点击率预估之前，传统做法通常使用逻辑回归[5] 进行点击二分类，但这需要构造大量手工特征，FM[6, 7]、FFM[8] 和 GBDT+LR[9] 的出现让这类点击二分类模型能够自动学习到一些二维组合特征或者一些高维特征。随着深度学习的引入，特征的挖掘变得更加充分，组合特征的学习变得更加高效。

深层神经网络模型引入到点击二分类算法当中大多是通过神经网络与其他基础算法结合的方式构建混合模型，这些方法根据其网络融合的方式大致可以分为**串行融合**和**并行融合**两大部分。**串行融合**的方式主要将 FM 等模型的输出作为 MLP 的输入，将 FM 视为交叉特征提取器，这类模型有：Weinan Zhang 等在 2016

年提出的因子分解机神经网络（Factorisation Machine supported Neural Network，FNN）[10]，其将考 FM 与 MLP 进行了串联结合，利用 FM 为多层感知机 MLP 提取二维交叉特征进行点击率预估推荐；Xiangnan He 等在 2017 年提出的神经网络因子分解机（Neural Factorization Machines，NFM）[11] 加入了池化层，将 FM 生成的二维交叉特征进行了元素级别的两两向量内积，引入了更多的非线性关系；和 Jun Xiao 等人在 2017 年提出了注意力因子分解模型（Attentional Factorization Machine，AFM）[12]，用注意力网络替换了上述两个模型中 MLP，让 Attention 网络更好利用 FM 的二阶特征；Yanru Qu 等人在 2016 年提出了基于向量积的神经网络（Product-based Neural Networks，PNN）[13]，与 NFM 不同的是使用了 FM 生成的二维交叉特征进行了元素级别的两两向量内积进行计算。

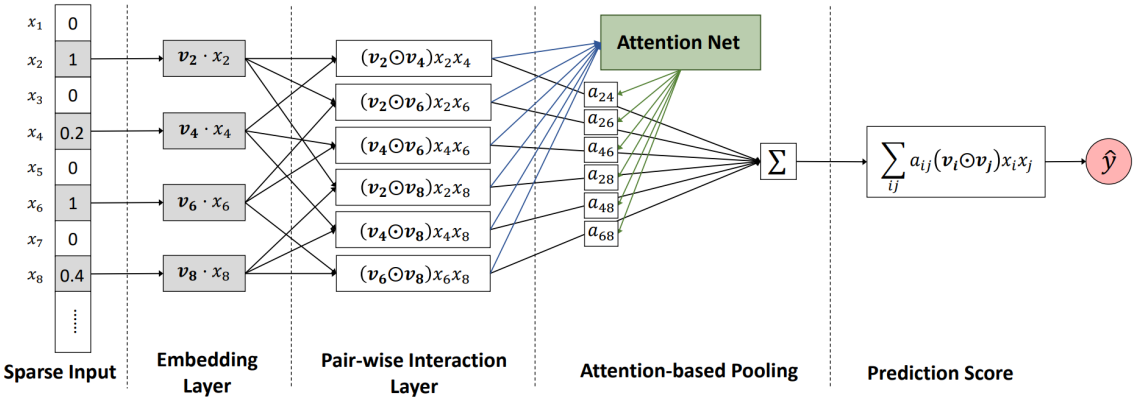

图 1-3　串行深度网络推荐模型之——AFM 网络结构图[12]

而**并行融合**的方式则是利用两个或多个网络模块，每个子模块负责提取不同维度的特征为主要思路，构建点击二分类算法。这类模型主要有：

2017 年 Huifeng Guo 等人将 FM 与 MLP 并行连接，提出的 DeepFM 模型[14]，两个组件网络共享同一个输入并行训练，能同时提取特征的低维和高维特征；2016 年 Heng-Tze Cheng 等人将一个 Wide 网络模型与深层网络 Deep 模型并行连接，利用 Wide 模块提升模型的记忆性，利用 Deep 模块提升模型的泛化性，构造的 Wide&Deep 模型[15]; 2017 年 Ruoxi Wang 等人将深层网络 Deep 与一个交叉网络 Cross 并行连接，利用不同层数的 Cross 网络构造出任意维度的交叉特征，提出了 Deep&Cross 模型[16]；2018 年 Jianxun Lian 等人提出了一个新颖的压缩交互网络（Compressed Interaction Network，CIN）的神经模型用于自动学习显式的高阶特征

交互，然后将其与 DNN 模块并行集成，构造了一个名为极深因子分解机的模型（xDeepFM）[17] 用于同时以显式和隐式的方式自动学习高阶的特征交互。

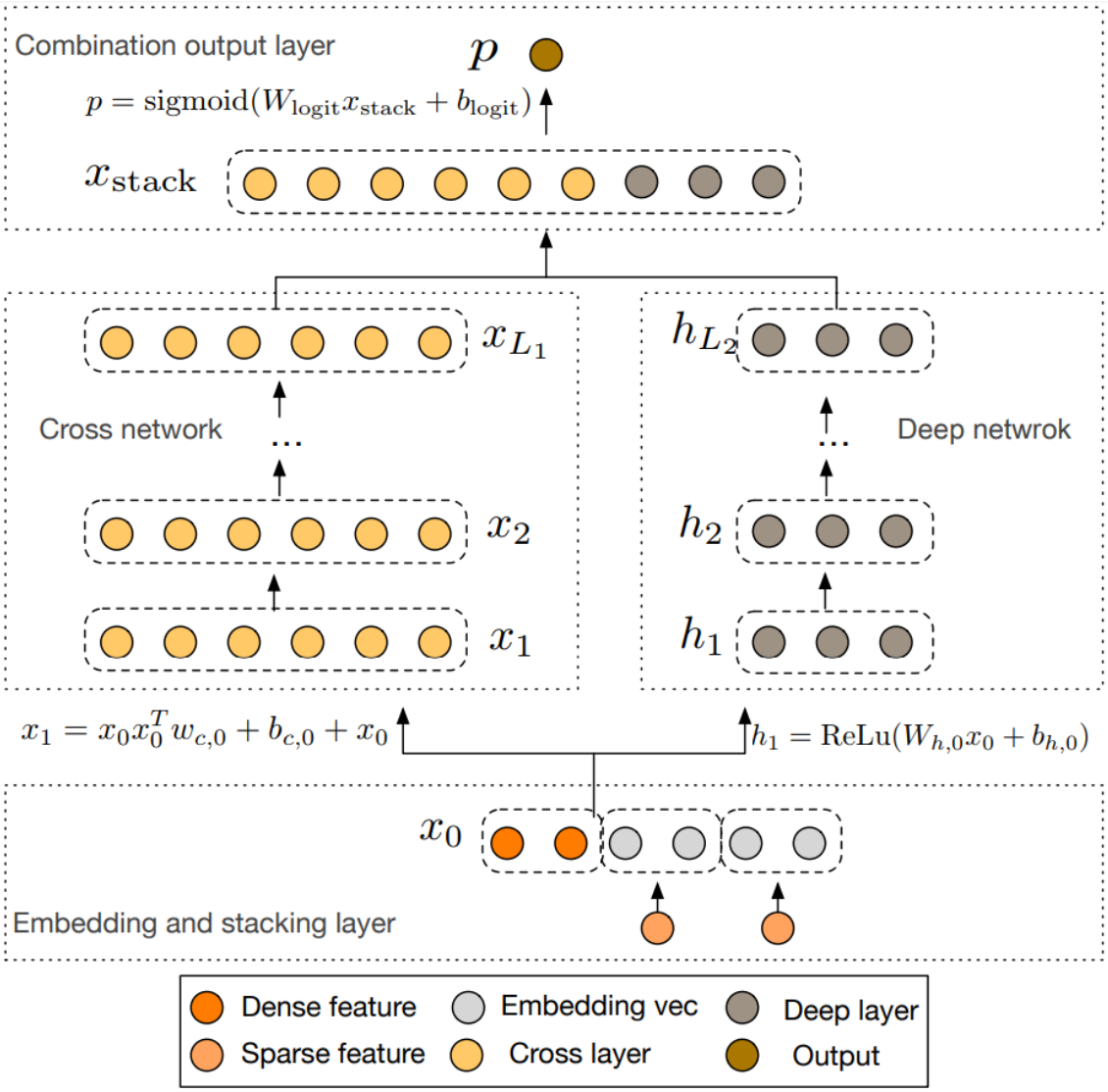

图 1-4　并行深度网络推荐模型之——Deep&Cross 网络结构图[16]

在串行模型与并行模型之外，又有一类**综合并行结构与串行的结构**，如 2018 年 Guorui Zhou 等人先后提出了深度兴趣网络（Deep Interest Network，DIN）[18] 和深度兴趣演化网络（Deep Interest Evolution Network，DIEN）[19]，DIN 的并行结构体现在对用户侧历史行为特征域和物品侧特征域并行嵌入，堆叠组合成的向量再输入到全连接神经网络中去。DIEN 对 DIN 的改进在于并行结构的特征嵌入部分考虑到了用户兴趣的演化进程，其使用双层 GRU 对用户历史行为的序列进

行建模，将建模后的向量一起并入堆叠向量当中。

1.2.3 序列感知推荐研究现状

现有的推荐算法几乎很少有考虑到隐藏在用户历史行为序列中的用户兴趣动态变化情况，而且以上提到的基于深度学习的推荐算法都是以构造大量特征为基础进行建模的，这些算法都需要收集大量用户隐私数据，况且在某些特殊应用场景下，例如用户在使用在线服务但并未登陆或匿名的情况下，以构造大量特征为前提的算法将全部失效。考虑到这三类情况，本文提出了序列感知推荐算法，以用户历史行为组成的序列为数据进行建模推荐，无需使用大量用户隐私相关数据，在用户匿名状态下也可以根据当前服务的会话构成行为序列。序列感知推荐算法许多学者称为基于会话的推荐算法，其中使用的序列建模算法基本可以分为 RNN 和 CNN 两大方向。在使用 RNN 进行序列建模推荐算法中，其中最广为人知的工作为 2016 年 Balazs Hidasi 等于提出的使用循环神经网络的基于会话的推荐算法 GRU4Rec[20]，首次通过使用循环神经网络利用会话的序列进行建模推荐，这类使用循环神经网络对序列进行建模进行推荐的还有 Devooght 等人于 2017 年提出的[21, 22]，2017 年 Yu Zhu 等人在使用 LSTM 对序列建模的基础上加上了对时间间隔的考虑，提出了 Time-LSTM 模型[23]。基于 RNN 的序列建模方法有较好的性能，

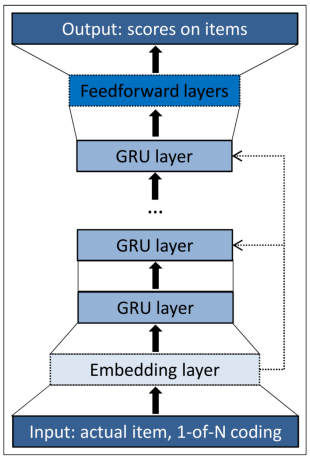

图 1-5　使用循环神经网络进行序列推荐算法之——GRU4Rec 网络结构图[20]

但受固于 RNN 的递归结构，算法难以并行实现。

另外一类序列感知推荐算法的方向是使用 CNN 对用户行为历史记录进行序

列建模，CNN 常用于计算机视觉领域图像特征提取，用于序列建模需要对基本的卷积结构进行大量的修改，这其中知名的工作有 2018 年 Jiaxi Tang 等人将序列嵌入矩阵使用 CNN 提取特征，提出的卷积序列嵌入推荐模型 Caser[24]。基于 CNN

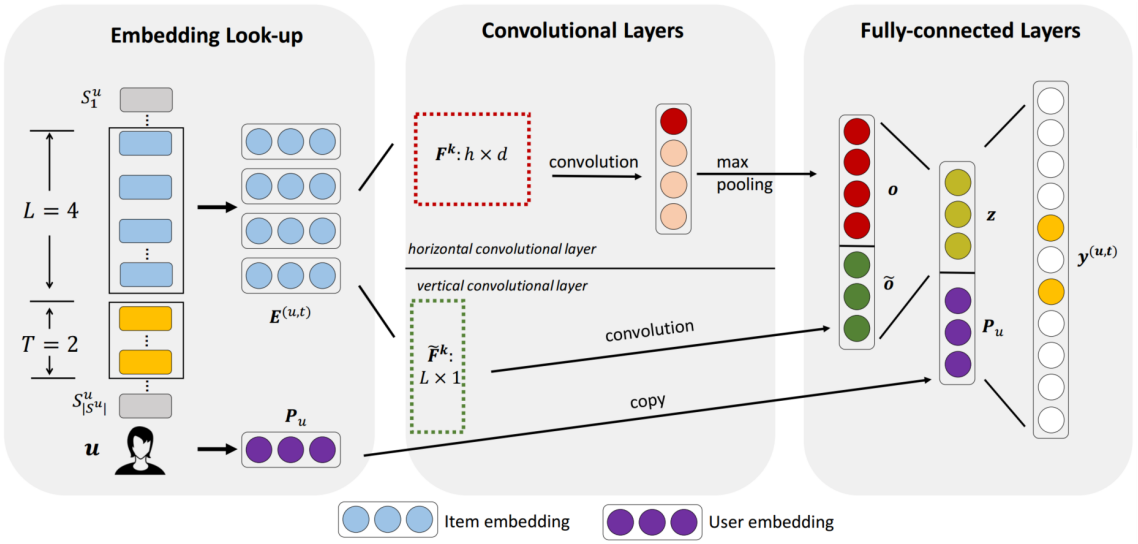

图 1-6　使用卷积神经网络进行序列推荐算法之——Caser 网络结构图[24]

的序列建模方法记忆的序列长度严重依赖卷积层的深度。

1.3　研究内容和方法

1.3.1　研究内容

本文通过调研分析国内外推荐算法的发展状况，特别是深度学习在推荐系统领域的发展情况，分析了目前深度学习在解决推荐系统实际应用中遇到的挑战，本文提出了一种新的思路——序列建模，去解决推荐系统现有的问题，详细阐述了该序列感知推荐模型的工作步骤和方式。受神经网络在自然语言处理方面强大的建模能力，本文还提出了两种序列感知推荐模型。最后本文通过对现实世界中两个数据集上进行的实验，对提出的模型进行了有效性验证。

1.3.2　研究方法

本文主要采用的研究方法有如下几点：第一，文献调研法。通过互联网技术访问线上各个数据库中检索了大量研究领域的相关书籍、论文等学术成果，经过

对国内外的相关研究文献与资料的全方位收集和分析，并结合工业界应用领域的最新解决方案，确立本文研究方向和主题，设计本文推荐模型框架及各模块之间的耦合。第二，迁移法，本文是建立在自然语言处理、文本挖掘、推荐系统等相关技术的研究基础之上，通过综合探索以上技术理论，将其自然语言处理中的序列建模技术迁移到推荐系统的算法设计上来。第三，实验仿真与分析法，本文通过调研和分析大量文献，提出本文的研究内容，为了验证本文提出的模型的有效性，本文在多个真实的数据上进行了实验，从而检验模型的可靠性。

1.4 本文主要贡献

本文通过前期大量文献调研，对比国内外深度学习推荐技术上进行的研究，通过深入探索后提出了本文的研究问题，并针对提出的问题进行了大规模的对比实验，直到得出最后的结论。整个过程中本文的主要贡献体现出如下几点：

1. 本文调研了深度学习在推荐系统应用领域的发展情况，针对现有推荐算法都需要使用大量用户隐私数据构造特征的情况下，提出了只利用用户历史行为序列的序列感知推荐算法。

2. 本文设计了基于双向长短期记忆网络的序列感知推荐模 BiLSTM4Rec，对用户历史行为序列进行嵌入式表达，生成稠密矩阵，利用双向长短期记忆网络对嵌入矩阵进行序列建模，预测下一个物品，将推荐问题转变成一个多分类问题。

3. 考虑到循环神经网络在训练方面的效率问题，本文从 Transformer[25] 结构中抽取 Self-Attention 模块，将序列感知推荐模型中的循环神经网络序列特征抽取算法替换为 Self-Attention，提出 Transformer4Rec 模型，该模型在生成较大推荐精度的情况下还不损失性能。

1.5 论文组织架构

本文的组织结构分为六章，如下所示：

第一章，绪论，首先介绍当前研究课题的背景并讨论其研究的意义，接着介

绍本课题最近的相关研究与进展，主要是从两个方面进行讨论：1）基于深度学习的推荐算法研究与相关进展；2）基于神经网络的序列感知推荐算法及其研究情况。紧接着阐述本文研究的主要内容和贡献。最后介绍本文的组织结构。

第二章，本章主要介绍序列建模领域涉及的相关知识与理论，本文的工作大部分基于这些理论知识上进行研究。包括经典的循环神经网络、常见的物品的离散表示方式、基础的卷积神经网络、以及目前用于自然语言处理的注意力机制。

第三章，本章主要介绍本文的主要工作：提出一个用于序列感知的推荐模型 BiLSTM4Rec，介绍该模型的整体情况，包括该模型的核心部分：双向长短期记忆网络。着重描述如何将该模型用于构建基于序列感知的推荐框架，包括加入 Embedding 的物品离散化稠密嵌入式表达。

第四章，本章介绍本文另外的工作，将在自然语言处理领域最好的语言翻译框架 Transformer 中提取 Self-Attention 模块，将其迁移至序列感知推荐算法当中，提出 Transformer4Rec 模型，克服现有基于循环神经网络的序列感知算法普遍面临的训练效率问题。

第五章，本章节主要描述本文的实验情况，在 MovieLens 数据集上和 Yoochoose 数据集上，将本文提出的两个算法与其他几个基准算法分别作对比评估，另外，本文为了衡量推荐算法的性能问题设置另外一组模型查询效率实验，衡量不同结构对推荐模型的性能影响。

第六章，本章节主要内容是对本文的工作进行总结，并且阐述未来的工作任务。

第 2 章　相关理论与技术分析

2.1　序列感知模型

序列感知模型是把数据根据时间日期排序之后，按照数据之间的时间先后顺序，发现时间上近邻的数据之间的隐藏关系或数据的周期性变化规律等与时间有关系的一类数据挖掘模型，数据挖掘领域又常称这类模型为时序模型。因此序列感知模型面对的数据必须包含时间戳或者数据的存储形式能够不丢失数据的诞生先后顺序。时序模型数据分析的目的就是为了挖掘出数据之间的内在时间规律，找到这种时间规律之后利用其归纳、类推、演绎未来的数据变化趋势，从而进行建模样本之外的数据预测。2018 年的 ACM RecSys 中还专门设立了关于序列感知推荐的课程并发布了一篇关于序列感知推荐的研究综述[26] 。

当待分析的数据具有固有的顺序性质，序列学习方法就会在这些应用领域中有用，比较常见的应用有如自然语言处理、语音识别、时间序列预测、DNA 建模，以及作为本文工作的核心内容，序列感知推荐。

2.2　序列建模技术分类

2.2.1　循环神经网络

因为传统前馈深度神经网络 (FNN) 无法了解给定输入的上下文环境关系，循环神经网络 (RNN)[27] 被发明的目的就是用来进行对可变长度的序列数据进行建模。循环神经网络与传统的 FNN 模型之间的主要区别在于组成网络的单元中存在内部隐藏状态，在一个序列建模步骤中的每个内部隐藏状态节点都接收来自上一个节点的输入，因此这可以用一个循环来表示，其结构如图2-1所示，隐藏状态层保留了过去序列编码的摘要，每当 RNN 呈现新的输入时，就会更新该隐藏层的状态。对于一个最简单的标准循环神经网络其通过以下形式来更新隐藏单元的状态 h：

$$h_n = f\left(Ux_n + Wh_{n-1} + b\right) \tag{2.1}$$

其中 h_{n-1} 是第 $n-1$ 层神经网络的向量化表示，x_n 是传递给第 n 层的输入序列编码，U, W 是该层包含的权重矩阵，b 是该层向量的偏置。函数 $f(\cdot)$ 为非线性转换函数，也称激活函数，常用的激活函数有 Logistic Sigmoid 函数 $\sigma(\cdot)$，$tanh(\cdot)$，线性整流单元 $ReLU(\cdot)$ 和一些它们的变体。经过以上隐藏层状态的更新之后，输出层的计算公式如下：

$$\hat{y}_n = g(Vh_n) \tag{2.2}$$

其中 $\hat{y}_n$ 是 n 时刻的输出值，V 是输出层的权重矩阵，$g(\cdot)$ 是输出层的激活函数。

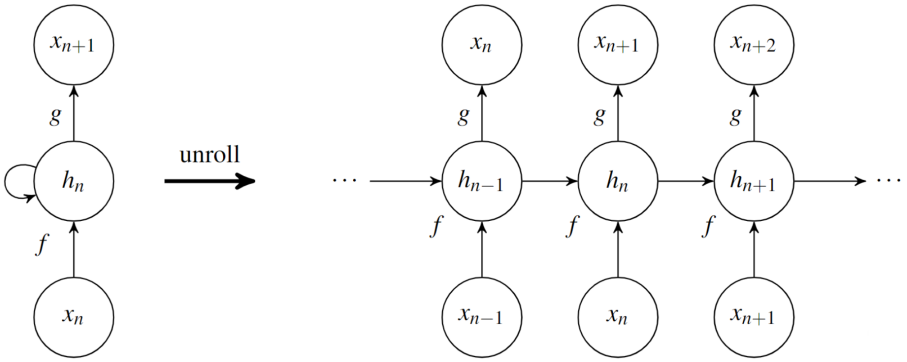

图 2-1　简单循环神经网络结构图

通过公式2.1,2.2可以明白，RNN 的每一个时间步骤都会有一个新的输入，并且特定时间步骤的输出依赖于之前所有步骤的输入，这意味着时间步骤 N 时刻的损失函数的计算要回溯到时间步骤 1，这一过程也称为基于时间的反向传播算法 *(backpropagation through time, BPTT)*[28]。但是如果要处理的序列很长的话，经过多层的反向传播，BPTT 会产生梯度消息或者梯度爆炸的问题，以至于无法从差的很远的时间步骤中感知上下文环境，使得 RNN 的训练变得非常麻烦，RNN 这一明显的缺点也称为长期依赖问题。

2.2.2　长短期记忆神经网络

为了解决 RNN 的长期依赖问题，一些基于公式2.1的变形工作诞生，其中最广为人知的是长短期记忆网络 (long short-term memory, LSTM)[29]。LSTM 通过精巧设计的记忆单元更换了 RNN 中的隐藏单元，其核心计算单元如图2-2所示。

LSTM 神经元内部通过精心设计的分别称为遗忘门、输　门、输出门的三个门结构来决定哪些信息更新到内部或者从内部去除。在遗忘门(2.3)当中，前一时间步的掩藏状态和当前时间步的输入经过 Sigmoid 函数非线性转换之后得到一个 $[0, 1]$ 之间的值，以表示需要遗忘信息的概率。在输入门(2.4),(2.5)中，将当前时间步骤中的输入和前一时间步骤学习到的隐藏状态经过 tanh 激活函数的计算生成一些候选值，并通过 Sigmoid 函数的传递从候选值中选出一些进行更新。而输出门(2.6)就决定了当前单元要输出哪些部分。LSTM 通过如下的组合函数来更新隐藏单元的状态：

$$f_t = \sigma(U_f x_t + W_f[h_{t-1} + c_{t-1}] + b_f) \tag{2.3}$$

$$i_t = \sigma(U_i x_t + W_i[h_{t-1} + c_{t-1}] + b_i) \tag{2.4}$$

$$c_t = f_t c_{t-1} + i_t \tanh(U_c x_t + W_c h_{t-1} + b_c) \tag{2.5}$$

$$o_t = \sigma(U_o x_t + W_o[h_{t-1} + c_t] + b_o) \tag{2.6}$$

$$h_t = o_t \tanh(c_t) \tag{2.7}$$

其中 $\sigma(\cdot)$ 是 Logistic Sigmoid 激活函数；而 i、f、o 和 c 分别是输入门、遗忘门、输出门和隐藏层的激活向量；变量 b 表示偏置向量，例如 b_f 表示为遗忘门的偏置向量。U、W 分别代表输入向量的权重和上一层输出的权重，例如在遗忘门中，U_f 代表遗忘门输入向量 x_t 的权重，W_f 代表上一个 LSTM 神经元输出 h_{t-1} 的权重。

2.2.3 门控循环单元

由于 LSTM 在循环神经单元中增加了三个门结构，与 RNN 相比，在一个神经元当中要完成更多的复杂计算。当使用更大的网络的时候，训练时间相比 RNN 也将显著增加。为了减少训练的时间复杂度并同时保留 LSTM 对长期依赖关系的记忆能力，2014 年 Cho 等人提出门控循环单元 (Gated Recurrent Unit, GRU)[30]。与 LSTM 相似，GRU 使用门结构建模单元内部信息的流动，不同的是，GRU 将 LSTM 三个门减少为两个。GRU 使用更新门来决定是否遗忘上时刻的信息或者

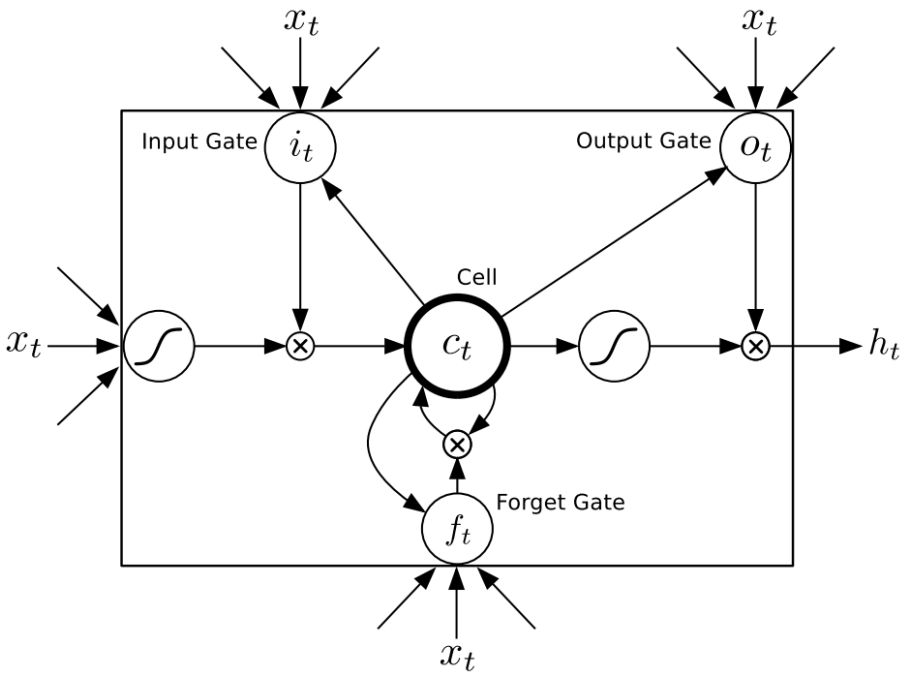

图 2-2　长短期记忆网络神经元结构图

记忆此时刻新的外部输入信息，其功能相当于组合了 LSTM 中的输入门与遗忘门。使用重置门来决定如何将新的输入信息与内部已有放入记忆相结合。在 t 时刻 GRU 单元中更新门的状态表达式为:

$$z_t^j = \sigma(U_z \mathbf{x}_t + W_z \mathbf{h}_{t-1})^j \tag{2.8}$$

其中，$\mathbf{x}_t$ 为第 t 个时间步骤的输入向量，$\mathbf{h}_{t-1}$ 中保存的是上一个时间步骤的信息。U，W 是更新门当中输入向量与上一时间步信息的权重矩阵。t 时刻 GRU 单元中重置门为：

$$r_t^j = \sigma(U_r \mathbf{x}_t + W_r \mathbf{h}_{t-1})^j \tag{2.9}$$

结合更新门与重置门，整个 GRU 神经元在 t 时间步的内部状态更新为：

$$z_t^j = \sigma(U_z \mathbf{x}_t + U_z \mathbf{h}_{t-1})^j \tag{2.10}$$

其中 $\tilde{h}_t^j$ 为候选信息，表示当前记忆的内容，其计算表达式为：

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \tilde{h}_t^j \tag{2.11}$$

单个 GRU 神经元的总体结构如图2-3所示。

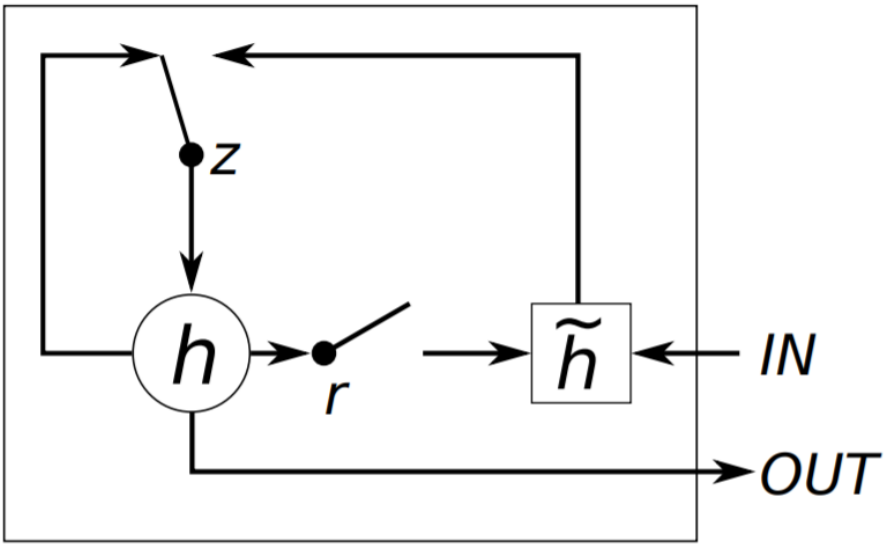

图 2-3　门控循环单元神经元结构图

2.2.4　双向循环神经网络

在自然语言处理的实体识别技术中，双向循环神经网络 (Bidirectional recurrent neural networks, BRNN)[31] 弥补了单项循环神经网络对于上下文感知能力的不足，因为单向 RNN 预测下一个单词时使用的只是此单次出现之前的信息，而 BRNN 则从两个方向获取信息，上下文感知能力也就更强了。BRNN 将隐藏层分为两个部分，前向状态层 $\vec{h}$ 和反向状态层 $\overleftarrow{h}$，其输出层的输入由 $\vec{h}$ 和 $\overleftarrow{h}$ 堆叠而成，其迭代公式如下：

$$\vec{h}_t = \sigma(U_{\vec{h}} x_t + W_{\vec{h}\vec{h}} \vec{h}_{t-1} + b_{\vec{h}}) \tag{2.12}$$

$$\overleftarrow{h}_t = \sigma(U_{\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \tag{2.13}$$

$$y_t = W_{\vec{h}y} \vec{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \tag{2.14}$$

BRNN 的隐藏层结构如图2-4所示。

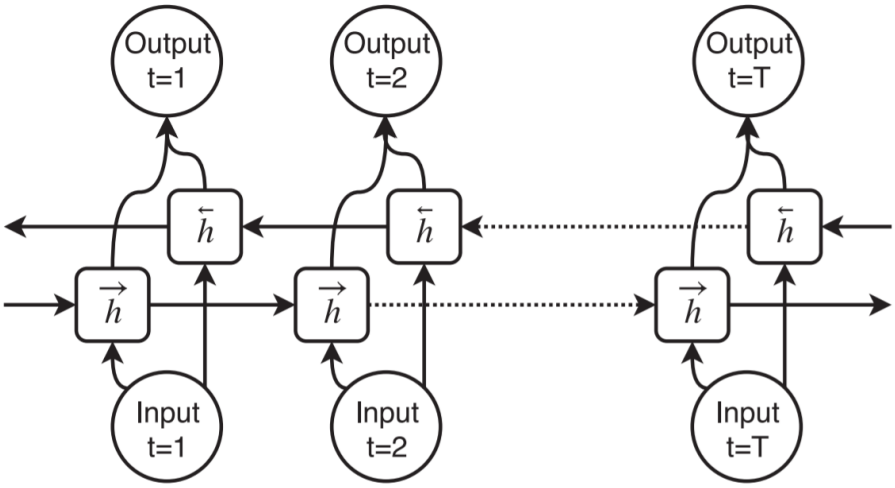

图 2-4　双向循环网络神经结构图

2.2.5　时间卷积网络

卷积神经网络通常用于计算机视觉领域的图像特征提取，将其应用于序列建模需要对卷积神经网络的结构做出修改，目前将卷积神经网络运用于序列建模的最良好算法为 2018 年 Shaojie Bai 等人提出的时间卷积网络（Temporal Convolutional Network，TCN）[32]，本文在此把 TCN 作为 CNN 在序列建模领域的代表。

TCN 由因果卷积（Causal Convolution）、扩张卷积（Dilated Convolution）和残差链接（Residual Connection）三部分构成。

因果卷积在序列建模任务中，首先得保证输入序列与输出序列的长度一直，因此在 TCN 中使用的卷积结构是 1 维全卷机网络，其中每一个隐藏层的长度都与输入层长度相同，对于经过卷积过滤器计算的中间隐藏层，长度会减小，通过 0 填充的方式使其扩充到长度相等。如果直接对输入序列进行卷积操作的话，卷积操作将会是对序列的全局进行计算，这会导致信息泄露的问题，即在训练时让现在时刻知道了未来时刻的信息，而这种情况面对真实世界时是不可能发生的。为了防止训练时将未来时刻信息泄露到了现在时刻，因果卷积被引入。所谓因果卷积, 就是计算 t 时刻的输出时, 仅对前一层 t 时刻及之前的状态进行卷积。因果卷积等价于掩码卷积（masked convolution）[33]，在输入序列上加上掩码把未来时刻的

信息对现在时刻进行屏蔽。

扩张卷积对于因果卷积，如果需要获得更大的感受野来捕获更长范围的序列的话需要很多层或者很大的过滤器，为了更加因果卷积的感受野，扩张卷积被引入与其结合。扩张卷积通过跳过部分输入来使过滤器能应用于比过滤器本身大小更大的范围，等同于通过 0 填充来扩大原先的过滤器。Dilation convolution 的运算如下:

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i} \tag{2.15}$$

其中 $\mathbf{x}$ 表示输入序列, f 表示过滤器 filter, d 是扩张因子,k 是过滤器大小, $s - d \cdot i$ 意味着只对过去的状态作卷积。图2-5为结合了因果操作和扩张操作的卷积示意图。

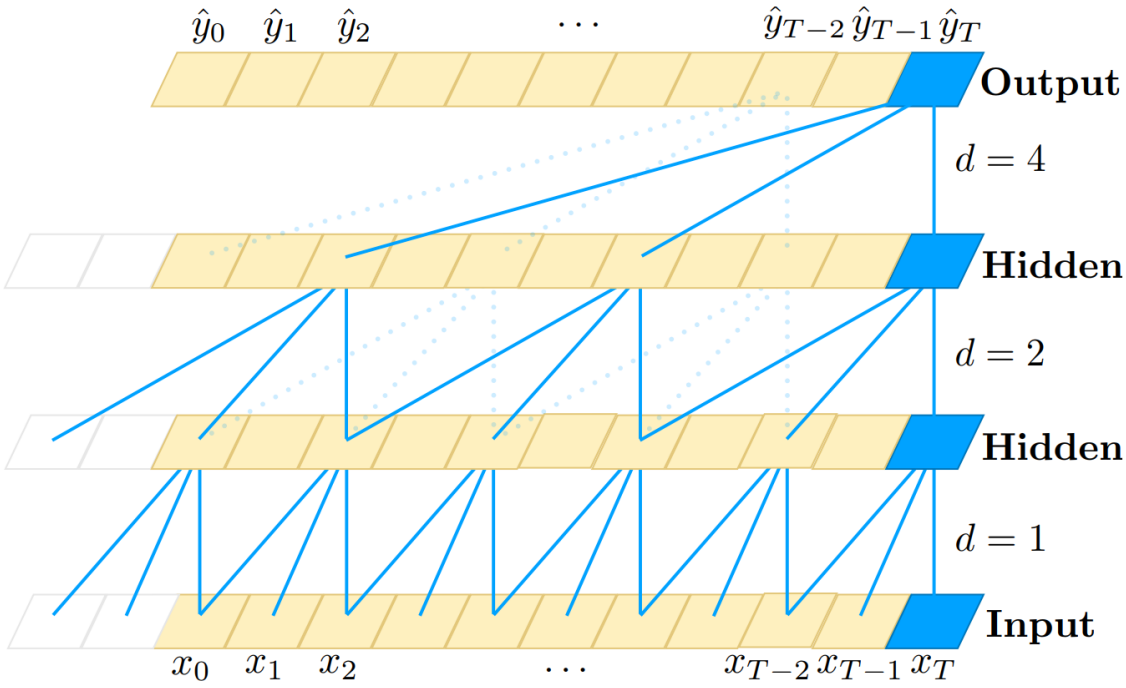

图 2-5　扩张因子分别为 1,2,4 和过滤器大小为 3 的扩张因果卷积网络结构图

残差链接由于卷积神经网络对序列建模能记忆的时间步长度依赖于网络的深度，TCN 为了获得更大的感受野，因此不得不增加网络的深度。为了使深层网络的训练不至于丢失浅层特征，残差连接是用来解决这个问题的常用办法，因此

TCN 结构中加入了残差连接以增加网络深度。

$$o = \text{Activation}(\mathbf{x} + \mathcal{F}(\mathbf{x})) \tag{2.16}$$

其中 $\mathcal{F}(\cdot)$ 为残差结构略过的深层隐藏网络。残差网络通过建立输入与输出之间的捷径网络，使有效的浅层特征能够直接传递给输出层，避免了所有特征向量都必须经过深层隐藏网络的计算，保留了浅层有效特征，使得训练深层网络变得更加容易。

2.2.6 注意力机制

在自言语言处理的机器翻译算法中，有一类端到端的模型有着出色的性能。而为了说明深度学习中的注意力模型，就不得不先谈这一类端到端框架，因为目前大多数注意力模型都作为端到端框架下的一部分使用。其中代表工作为 2014 年 Ilya Sutskever 等人提出的 Seq2Seq[34] 模型，Seq2Seq 有一个编码器（Encoder）和一个解码器（Decoder）构成。

Seq2Seq 模型中把序列建模问题当做一个条件概率问题：

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i) \tag{2.17}$$

其中 s_i 为 Decoder 中 i 时刻的状态，其计算公式为：$s_i = f(s_{i-1}, y_{i-1}, c_i)$，$c_i$ 为该时刻状态的权重，其计算公式为 $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$，其中 i 表示 Encoder 端的第 i 时刻输入。h_j 表示 Encoder 端的第 j 个输入的隐向量，α_{ij} 表示 Encoder 端的第 j 个输入与 Decoder 端的第 i 个输入之间的权值，表示源端第 j 个输入对目标端第 i 个输入的影响程度，α_{ij} 的计算公式如公式(2.18)所示：

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{2.18}$$

其中 $e_{ij} = a\left(s_{i-1}, h_j\right)$。

在公式(2.18)中，α_{ij} 是一个 softmax 模型输出，概率值的和为 1。e_{ij} 表示一个对齐模型，用于衡量 Encoder 端的位置 j 个词，对于 Decoder 端的位置 i 个词的

对齐程度（影响程度），换句话说：Decoder 端生成位置 i 的词时，有多少程度受 Encoder 端的位置 j 的词影响。对齐模型 e_{ij} 的计算方式有很多种，不同的计算方式，代表不同的 Attention 模型，最简单且最常用的的对齐模型是 dot product 乘积矩阵，即把 target 端的输出隐状态 ht 与 source 端的输出隐状态进行矩阵乘。常见的对齐计算方式如下：

$$\text{score}\left(\boldsymbol{h}_t, \overline{\boldsymbol{h}}_s\right) = \begin{cases} \boldsymbol{h}_t^{\top}\overline{\boldsymbol{h}}_s & \text{dot} \\ \boldsymbol{h}_t^{\top}\boldsymbol{W}_a\overline{\boldsymbol{h}}_s & \text{general} \\ \boldsymbol{v}_a^{\top}\tanh\left(\boldsymbol{W}_{\boldsymbol{a}}\left[\boldsymbol{h}_t;\overline{\boldsymbol{h}}_s\right]\right) & \text{concat} \end{cases} \tag{2.19}$$

2.3 本章小结

本章主要围绕序列感知推荐算法中关键的序列建模技术进行论述，先简要介绍了序列建模技术广泛的应用领域，对序列建模理论引入推荐算法领域的可能性进行了铺垫，然后对后文中涉及到的关键技术和理论进行简要概述，从 RNN、CNN、Attention 三个大的层次上对当前领域内使用比较多的序列建模基础技术 RNN、LSTM、GRU、BRNN、TCN、Attention Mechanism 等原理进行了详细介绍，通过完整的理论支撑充分证明了本文的可行性。

第 3 章　基于双向长短期记忆网络的序列感知推荐算法

3.1　问题的提出

推荐系统面临的问题主要有两大类：评分预测和项目推荐。所以在推荐领域根据用户的历史活动记录预测用户下一次行为可能会选择什么项目也是一个重要的问题。在许多在线网站和应用程序当中，如在线电子商务、新闻或视频推荐网站、音乐或广播电台，它们都需要为用户提供一个杰出服务来推荐用户在未来可能会喜欢的东西。现有的推荐系统主要关注于找出用户或项目的近邻集，或者利用隐式或显式信息 (如标签、评论、物品内容、用户属性) 来提升近邻感知能力。然而，却少有工作利用数据当中的时序属性来直接构建推荐系统。在本篇论文中，我们发现数据的序列中其实包含着许多有价值的且激动人心的信息，以视频网站为例，一个用户看了纪录片《河西走廊》第一集《使者》之后，接下来看的另一个节目很有可能会是《河西走廊》第二集《通道》。甚至早在 2011 年举办的 Recsys 推荐系统大会上，来自音乐应用 Pandora[1]的研究人员给出的演讲上都提到了许多用户听音乐具有时序特点。

在某些特别的应用场景下，常规的推荐系统甚至无法起作用。现有的推荐系统都需要分析用户的数据，因此每个网站和应用的使用到需要让用户完成注册以及登录，然而用户每次使用网站或者应用的服务时都不一定会愿意登录，这种场景下对匿名用户的推荐显然挑战更大，常规的推荐策略显然无法起作用，基于匿名用户本地浏览器和缓存的会话所蕴含的序列进行推荐则显现出很重要的实践意义与价值。

3.2　算法框架描述

为了挖掘许多现有算法忽视掉的用户行为序列特征，本文提出了使用神经网络来对用户的序列进行建模的思路。我们的模型（*BiLSTM4Rec*）主要由五个部分组成，分别以此是嵌入层、循环结构、全连接层、池化层和输出预测层。其整体

[1]www.pandora.com

框架结构如图3-1所示。

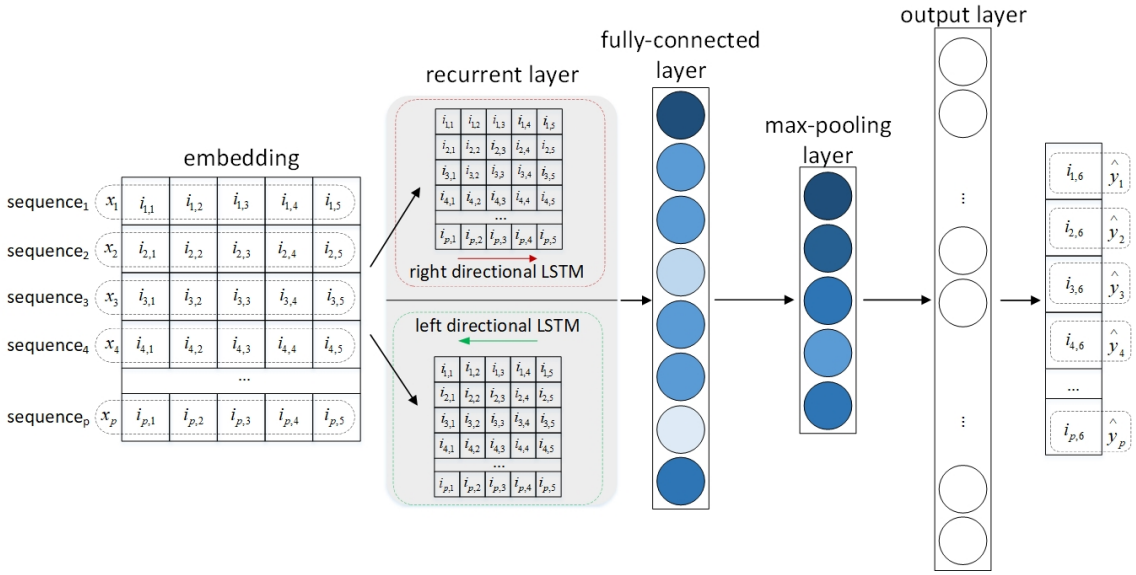

图 3-1　基于双向循环神经网络的序列推荐算法 BiLSTM4Rec 框架图

3.2.1　目标问题定义

序列感知推荐与传统的单类协同过滤推荐是有很大不同的，序列感知推荐的主要目标是预测用户下一步将会点击什么，而且利用数据仅仅包含用户当前历史行为的序列集合，而不接触为用户设置的长期偏好属性。接下来我们将定义序列感知这一问题的形式。

在序列感知推荐当中，我们定义 $\mathbb{U} = \{u_1, u_2, ..., u_N\}$ 代表不同的用户集合，定义 $\mathbb{I} = \{i_1, i_2, ..., i_M\}$ 代表在所有序列中出现过的不同物品集合，$s_u^t \in \mathbb{I}$ 表示用户 u 在时间刻 t 点击某一个物品的记录，该记录对应的物品包含在物品集合 $\mathbb{I}$ 当中。对于每一个用户 u，都记录一个按照数据诞生时间戳顺序排列的用户点击记录序列 $\mathbb{S}_u = \left\{s_u^1, s_u^2, ..., s_u^{t-1}, s_u^t\right\}$ 序列感知推荐的目标是预测下一次点击行为，也就是做出 $t+1$ 时刻的推荐 s_u^{t+1}。在序列感知推荐模型当中，对于序列 s，模型的输出是所有候选物品对象可能被点击的概率 $\hat{y}$，而概率最大的 K 个输出所对应的候选物品将作为推荐项目给到用户。

3.2.2 嵌入层

我们使用用户消费历史的最近几个序列当做特征，用户消费的最后一个物品当做标签，来构建一个超多分类的有监督学习模型。因此，在特征工程阶段，我们需要将原始序列特征数据转换为计算机容易处理的向量形式并且与标签映射。One_hot 编码是用来表达离散特征的最常用的向量表达形式，然而 One_hot 编码的向量会遇到高维和稀疏的问题。如果我们使用 One_hot 编码来处理具有 1000 个类别的特征，那么每个特征会被一个拥有 1000 个数字的向量来表示，但其中 999 个数字会是 0。在一个大规模数据集中，就计算效率而言这种方式是不可取的。

例如对序列 s_1, s_2, s_3, s_4, s_1 进行 One_hot 编码的话，将会得到如图3-2的形式

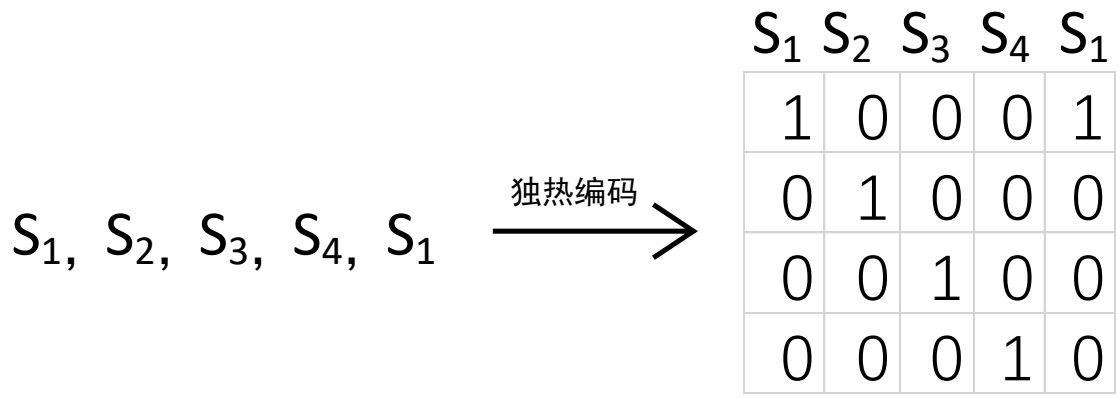

图 3-2　序列独热编码示意图

而词嵌入技术在自然语言处理领域大放异彩，我们可以借用词嵌入技术[35]，也构造一个嵌入矩阵来得到比 One_hot 形式小得多的向量结构：

$$e(I_i) = EI_i \tag{3.1}$$

其中 $E \in \mathbb{R}^{|e|\times|M|}$, $|e|$ 是嵌入层的大小，$|M|$ 是训练集中不同项目的数量。所以 $e(I_i)$ 是 I_i 的嵌入表达，其是一个具有 e 个实数的稠密矩阵。与一个 $|M| \times |M|$ 大小的 One_hot 编码形式相比，我们的序列嵌入矩阵的大小为 $|e| \times |M|$，当处理大数据集时，这显著减小了内存消耗。Embedding 嵌入编码中并不是每一个物品 ID 都会被一个向量来代替，而是被替换为用于查找嵌入矩阵中向量的索引。其次这种方法面对大数据时也可有效计算。由于在深度神经网络的训练过程中嵌入向量也会被

索引	嵌入因子					
1	.32	.02	.48	.21	.56	.15
2	. 65	.23	.41	.57	.03	.92
3	.45	.87	.89	.45	.12	.01
4	.65	.21	.25	.45	.78	.82

图 3-3　序列嵌入编码示意图

更新，还可以探索在高维空间中哪些物品之间具有彼此相似性。

3.2.3　用户短期兴趣学习

我们通过结合用户 u 在 t 时刻消费的项目 I_u^t 和其先后的项目来表达用户在此时刻的兴趣。行为序列帮助我们更精确地揭示了用户的短期兴趣。在这个推荐系统中，我们使用了一个双向的长短期记忆神经网络构建的循环结构来捕捉用户短期的兴趣变化。

我们定义 $h_b(I_i)$ 为用户他在消费物品 I_i 之前的兴趣，$h_a(I_i)$ 为用户消费物品 I_i 之后的兴趣。$h_b(I_i)$ 和 $h_a(I_i)$ 都是具有 $|h|$ 个实数的稠密向量。$W^{(b)}$ 是隐藏层的权重矩阵，用来继承用户之前的兴趣状态，矩阵 $W^{(cb)}$ 用来结合前一个物品的嵌入表达，σ 是一个非线性激活函数，因此通过学习表达式 $h_b(I_i)$ 来学习用户消费物品 I_i 之前的兴趣。同理，用户消费物品 I_i 之后的兴趣通过学习表达式 $h_a(I_i)$ 来学习。所有用户的初始兴趣使用同样的参数 $h_b(I_1)$，用户消费历史中的最后的兴趣则共享参数 $h_a(I_n)$。

$$h_b(I_i) = \sigma(W^{(b)}h_b(I_{i-1}) + W^{(cb)}e(I_{i-1})) \tag{3.2}$$

$$h_a(I_i) = \sigma(W^{(a)}h_a(I_{i+1}) + W^{(ca)}e(I_{i+1})) \tag{3.3}$$

通过以上公式，学习用户某个时刻之前和之后的兴趣，将它们和用户当前消费物品的嵌入矩阵集合来表达此时刻用户的临时兴趣状态，其结合形式如下：

$$x_i = [h_b(I_i); e(I_i); h_a(I_i)] \tag{3.4}$$

所以通过使用大量用户的历史行为序列 $\{i_1, i_2, ..., i_{n-1}, i_n\}$, 如果我们的模型学习到了

某个用户消费物品 i_{n-1} 时的临时兴趣 x_{n-1}，他将更有可能得到一个物品推荐 i_n. 双向循环结构能够对序列的前向扫描中捕获所有的 h_b，反向扫描则捕获了所有的 h_a。当训练集中所有的临时兴趣状态 x_i 都被捕获之后，运用一个线性转换与 tanh 激活函数将结果送到下一层。

$$y_i^{(2)} = tanh(W^{(2)}x_i + b^{(2)}) \tag{3.5}$$

$y_i^{(2)}$ 是潜在的兴趣向量，其中的每一个兴趣向量将通过上面权重和参数的更新来决定影响用户消费序列中最重要的因素。

3.2.4 流行趋势学习

当用户消费项目的所有序列被计算完之后，接下来应用一个最大池化层：

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)} \tag{3.6}$$

最大池化通过应用一个最大过滤器到上层代表的非重叠子区域，有了池化层，模型的参数或权重迅速减小了，这样也能减小上层输入的空间维度，减小计算消耗。通过对全局序列属性的捕获最大输出层能在用户的这个历史记录里找到那些最流行的序列组合。模型的最后一层就是常规的输出层了：

$$y^{(4)} = W^{(4)}y^{(3)} + b^{(4)} \tag{3.7}$$

输出层通过应用一个 softmax 激活函数到 $y^{(4)}$ 来转换成下一个类别的输出概率：

$$p_i = \frac{e^{y_i^{(4)}}}{\sum_{k=1}^{n} e^{y_k^{(4)}}} \tag{3.8}$$

3.2.5 模型的训练

将模型训练过程中所有需要更新的参数定义为 θ。

$$\theta = \left\{E, b^{(2)}, b^{(4)}, h_b(B_1), h_a(B_n), W^{(2)}, W^{(4)}, \ W^{(b)}, W^{(a)}, W^{(b)}, W^{(cb)}, W^{(ca)}\right\} \tag{3.9}$$

模型训练的优化目标是最小化交叉熵损失函数：

$$\mathcal{L}(y, S, \theta) = -\sum_{u \in \mathbb{U}} [y_u \log p(y_u|S_u, \theta) + (1 - y_u) \log(1 - p(y_u|S_u, \theta))] \tag{3.10}$$

3.3 复杂度分析

在嵌入层当中，主要包含一个矩阵的向量想乘操作，这一部分的时间复杂度是 $O(n)$。在双向长短期记忆网络中，网络的主体是 LSTM，其时间复杂度为 $O(n \cdot d^2)$，n 表示序列的长度，d 表示嵌入层因子的维度。而双向 LSTM 的时间复杂度我们可以知道为 $2 \cdot O(n \cdot d^2)$，最大池化层的时间复杂度为 $O(n)$，全连接层的时间复杂度为 $O(n)$，整个模型是上述这些结构的串联，所以整个基于双向循环神经网络的序列推荐算法的时间复杂度为 $O(n \cdot d^2)$。相比于基于用户的协同过滤算法 (UBCF)[36] 的时间复杂度 $O(n_u^2)$ 和基于物品的协同过滤 (IBCF)[37] 时间复杂度 $O(n_i^2)$，其中 n_u 表示用户数量，n_i 表示物品数量。与其相比，序列长度 n 和嵌入维度 d 存在这样的关系：$n_u > n_i \gg d > n$，所以基于双向循环神经网络的序列推荐算法在选择合适的序列长度和嵌入维度大小时面对大数据处理的压力时还是可以接受的。

3.4 算法实现

3.5 本章小结

本章节主要介绍本文提出的一个针对用户下一个点击推荐的新颖神经网络推荐模型 BiLSTM4Rec，其利用双向长短期记忆网络来对用户按照时间先后点击物品的历史记录进行序列建模，挖掘隐藏在序列当中的用户短期兴趣变化，来精确地对用户下一个可能感兴趣的物品做出推荐。通过对大规模物品 ID 进行矩阵嵌入压缩处理，借助该时序推荐算法，能够取得比传统协同过滤更高效更精确地短期推荐策略。

算法 3.1 基于双向长短期记忆网络的序列感知推荐算法

已知: 用户序列数据：S；序列长度：$maxLen$；最大物品数：$maxNum$
求: 推荐结果：$Items$

$sequence \leftarrow S$, 其中 $sequence = \{seq_1, seq_2 ... seq_n\}$
if $len(sequence) >= maxLen$ **then**
    选取序列最后的 N 物品：$sentences_{lastn} \subseteqq sequence$
    $items \leftarrow sequence_{topn}$，其中 $items = \{word_1, word_2 ... word_m\}$
    for each $i \in [1, maxLen]$ **do**
        将每个序列的物品 ID 编码到嵌入矩阵：$E = Embedding_{seq_i}$
    end for
else
    continue
end if
for each $i \in [1, maxLen]$ **do**
    $output_{right}$ = LSTM(E)
end for
for each $i \in [maxLen, 1]$ **do**
    $output_{left}$ = LSTM(E)
end for
$output = [output_{right}; output_{left}]$
$output = maxpolling(output)$
$NextItem = SoftMax(output)$
return $NextItem$

第 4 章　基于自注意力机制的序列感知推荐

4.1　问题的提出

在多个不同的机器学习领域，例如图像描述、机器翻译、阅读理解、摘要生成还有一些其他的应用上，已经证明注意力机制是有明显效果的这一机制背后的重要思想是影响序列输出的因素只与序列中某些重要信息相关，而无需过多关注其他无用的信息。这一机制的命名受到人类视觉系统中视觉焦点的启示，所以称为注意力机制。

2017 年 NIPS 大会上谷歌发表了一个完全用注意力机制构建的机器翻译框架 *Transfomer*[25]，到达了当时最好的效果。

4.2　算法框架描述

在基于循环神经网络的序列感知推荐系统方面，由于循环神经网络本身的特性也带来了性能上面的一些缺陷。由于循环神经网络的结构，当前神经元的输入有上一个神经元的输出，反向传播梯度更新时，下一个神经元的梯度更新也需要上一个神经元计算完再开始，这一串行的性质限制了循环神经网络的训练速度，特别是当需要使用更深的循环神经网络的时候。而由于自注意力机制完全抛弃了循环神经网络的串行结构而依然能捕获数据的序列关系，近来已成为序列建模领域的研究热点。使用注意力机制的另一好处是基于注意力的方法在推荐可解释性方面也更好说明。因此在这一部分，我们为了进一步的提升基于双向循环神经网络的序列感知推荐算法，本章基于编码器解码器框架，选用自注意力机制作为编码器和解码器的特征提取器，来建模用户短期行为模式的依赖因素，本文中将此模型称为 *Transformer4Rec*，其整体框架结构如图4-1所示。

4.2.1　目标问题定义

与第三章中基于双向长短期记忆网络的序列感知推荐模型问题一样，我们定义 $\mathbb{U} = \{u_1, u_2, ..., u_N\}$ 代表不同的用户集合，定义 $\mathbb{I} = \{i_1, i_2, ..., i_M\}$ 代表在所有序列

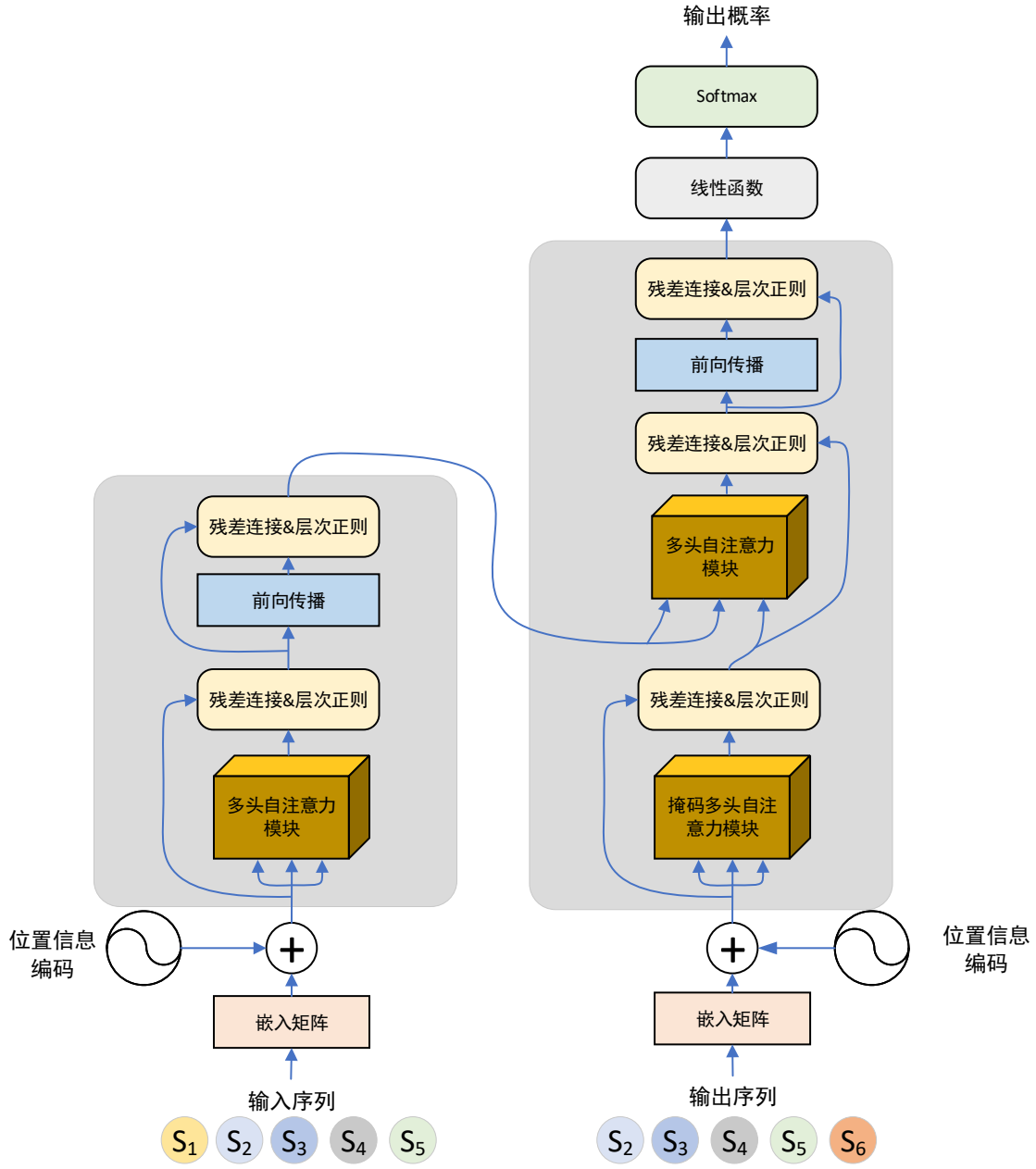

图 4-1　基于自注意力机制的序列推荐算法 Transformer4Rec 框架图

中出现过的不同物品集合，$s_u^t \in \mathbb{I}$ 表示用户 u 在时间刻 t 点击某一个物品的记录，该记录对应的物品包含在物品集合 $\mathbb{I}$ 当中。对于每一个用户 u，都记录一个按照数据诞生时间戳顺序排列的用户点击记录序列 $\mathbb{S}_u = \left\{s_u^1, s_u^2, ..., s_u^{t-1}, s_u^t\right\}$ 序列感知推荐的目标是预测下一次点击行为，也就是做出 $t+1$ 时刻的推荐 s_u^{t+1}。在序列感知推荐模型当中，对于序列 s，模型的输出是所有候选物品对象可能被点击的概率 $\hat{y}$，而概率最大的 K 个输出所对应的候选物品将作为推荐项目给到用户。

4.2.2 嵌入层

序列中的物品 ID 不具有实际数学意义，因此不能把 ID 编码直接传输给计算机进行计算，但为了使计算机能够以较少的资源消耗而感知不同的物品身份，这里同样不使用 One_hot 编码而使用与基于双向长短期记忆网络的序列感知算法一样的嵌入矩阵[35] 来对物品 ID 进行处理：

$$e(I_i) = EI_i$$

其中 $E \in \mathbb{R}^{|e|\times|M|}$, $|e|$ 是嵌入层的大小，$|M|$ 是训练集中不同项目的数量。所以 $e(I_i)$ 是 I_i 的嵌入表达，其是一个具有 e 个实数的稠密矩阵。

位置信息嵌入层

由于在利用自注意力方法构建的模型当中，没有了循环和卷积结构，因此抛弃了项目之间的序列信息，但用户的短期兴趣关注点隐藏在序列之中，为了使我们的模型能够捕捉到序列变化信息，必须将序列中物品的相对或者绝对位置信息加入到模型当中去。位置编码有可学习形式 (learned) 和固定参数形式 (fixed)[34]。这里我们使用了与 *Transfomer*[25] 一样的方式，在模型输入的嵌入层中加入物品序列的固定参数形式“位置编码”，即位置信息嵌入层。我们使用正弦和余弦函数并结合不同频率将物品的固定位置编码传递给模型，位置编码的一个维度对应于正弦曲线上的一个值，从而形成从 2π 到 $10000 \cdot 2\pi$ 的几何级数波长，这些信号作为额外的信息加入到输入（或输出）中以表达时间的流逝，使模型能够感知到当前正在处理的是输入（或输出）序列的哪个部分。：

$$PE(t, 2e) = \sin(t/10000^{2e/d_{model}}) \tag{4.1}$$

$$PE(t, 2e+1) = \cos(t/10000^{2e/d_{model}}) \tag{4.2}$$

其中 t 就是位置信息，表示该物品出现在序列中的第 t 个时间步骤，e 是位置嵌入层的维度。图4.3显示，不同的位置嵌入层维度大小，位置编码波形图具有不同的波长。由于位置信息嵌入层与嵌入层有相同的维度 $|e|$，因此可以把两者相

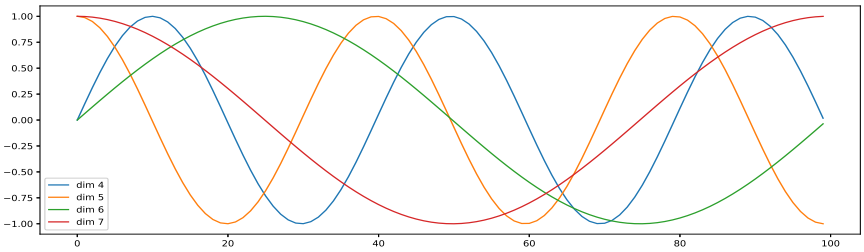

图 4-2　使用不同维度大小的位置编码波形图

加起来构成我们最终的输入嵌入层 $\widehat{\mathbf{E}}$：

$$\widehat{\mathbf{E}} = \begin{bmatrix} \mathbf{E}_{s_1} + \mathbf{PE}_1 \\ \mathbf{E}_{s_2} + \mathbf{PE}_2 \\ \cdots \\ \mathbf{E}_{s_n} + \mathbf{PE}_n \end{bmatrix} \tag{4.3}$$

4.2.3　编码器解码器框架

编码器解码器（Encoder-Decoder）框架[34, 38] 是自然语言处理中常用的一种序列生成框架，其常用来构建机器翻译模型。Encoder-Decoder 框架由一个解码器（Encoder）和一个编码器（Decoder）构成，Encoder 用来处理输入序列 $x_1, x_2, x_3, \cdots, x_{t-1}, x_t$，将输入序列处理成语义向量，Decoder 用来生成输出序列 $y_1, y_2, y_3, \cdots, y_{t-1}, y_t$，将处理过的语义向量转换成输出序列。Encoder 与 Decoder 的构建可以根据任务的不同选取具有不同特性的 CNN/RNN/BRNN/GRU/LSTM 算法。

但在我们的序列推荐框架中，考虑序列中的每一个输入节点，其期望的输出应为其序列中直接后继节点，因此我们可以根据这一性质构建一个 Encoder-Decoder 框架，如图所示：

输入节点 s_1，其对应的期望后继节点应为 s_2，输入节点 s_5，其期望的后继节点应为 s_6。由于循环神经网络自身结构导致的难以并行训练问题，Encoder 和 Decoder 的特征提取下我们准备选用自注意模型，下面对自注意模型的构建方式进行了介绍。

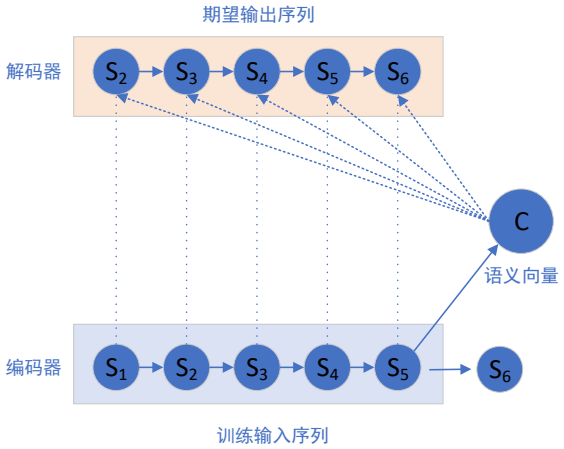

图 4-3　Encoder-Decoder 框架结构示意图

4.2.4　自注意力模块

自注意力方法是一种特殊的注意力机制，已有在各种任务上的成功应用。与最简单的注意力方法通过对整个语境学习得到有限的知识表示不同，自注意力方法即使面对相隔距离较远的两个元素，其依然能保存上下文中的序列信息并且捕获这两个元素在序列上的关系。因此我们这里使用自注意力方法来捕获用户历史行为中的序列特征。

构建自注意力模块的基本单元由缩放点乘注意力 *Scaled Dot-product Attention* 构成，一个自注意力模块的输入由查询 *(query)*，键 *(key)* 和值 *(value)* 三部分组成。自注意力模块的输出是模块中值 *(value)* 的加权和得来，而这里的权重矩阵则有查询 *(query)* 和其对应的键 *(key)* 决定。

我们用 Q 表示查询 *(query)* 向量，用 K 表示键 *(key)* 向量，用 V 表示值 *(value)* 向量，向量中的每一行代表一个物品。首先，我们通过使用共享参数的非线性变换将查询和键投影到同一空间：

$$Q' = \mathrm{ReLU}(W_Q\widehat{\mathbf{E}}) \tag{4.4}$$

$$K' = \mathrm{ReLU}(W_K\widehat{\mathbf{E}}) \tag{4.5}$$

其中 W_Q，W_K 分别是查询和键向量的权重矩阵，得到 Q' 和 K' 之后我们就

可以通过如下形式得到值 *(value)* 的权重矩阵：

$$V' = \text{softmax}(\frac{Q'K'^T}{\sqrt{d}}) \tag{4.6}$$

其中 $\sqrt{d}$ 为缩放因子，为了避免点乘之后出现过大的值导致 softmax 的输出被推到梯度消失的位置，因此加上缩放因子来优化矩阵点乘。于是 Self-Attention 模块的输出就可以计算出来了：

$$\mathbf{S} = \text{SA}(\widehat{\mathbf{E}}) = \text{Attention}(W_Q\widehat{\mathbf{E}}, W_K\widehat{\mathbf{E}}, W_V\widehat{\mathbf{E}}) = V'\widehat{\mathbf{E}} \tag{4.7}$$

多头注意力机制 Multi-Head Attention 与其只让单个 Self-Attention 模块来学习 *d* 维的查询 *(query)*，键 *(key)* 和值 *(value)*，可以通过集成学习的方法，让多个 Self-Attention 模块分别学习不同位置子空间的表达，由于随机初始化的不同，每个 Self-Attention 模块的关注点将会有所差异，所以每个 Self-Attention 模块可以分别学习到三个不同的 d_k 维的查询 *(query)*、d_k 维的键 *(key)* 和 d_v 维的值 *(value)* 向量。每个 Self-Attention 模块有了查询 *(query)*、键 *(key)* 和值 *(value)* 之后，就能得到 d_v 维的输出。而 Multi-Head Attention 是多个 Scaled Dot-product Attention 的并行叠加，因此 Multi-Head Attention 的计算复杂度与 Scaled Dot-product Attention 相当，但是却获得了更好的泛化性能。因此这部分 Multi-Head Attention 的计算方式如下：

$$\text{MultiHead}(\widehat{\mathbf{E}}) = \text{Concat}[\text{SA}_1(\widehat{\mathbf{E}}), \ldots, \text{SA}_h(\widehat{\mathbf{E}})]W^O \tag{4.8}$$

其中 *h* 表示我们将 *h* 个 Self-Attention 模块并联堆叠在一起得到 Multi-Head Attention 模块。

结构中逐项的 feed-forward 网络作用

考虑到 Self-Attention 模块在学习物品嵌入向量的过程中仍然是一个广义线性模型，为了增强整个 Transformer 模型的非线性拟合能力，Multi-Head Attention 子层后面跟了一个前馈神经网络 FFN 层，为了保证模型计算的效率，它仅由两个线性变换组成，中间嵌入一个 Relu 激活函数来提升非线性表达能力，因此这一部

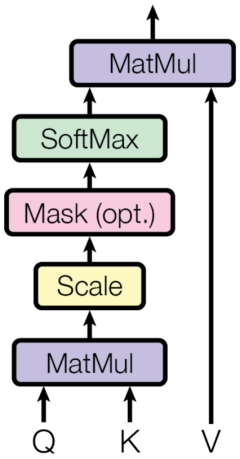

图 4-4　Scaled Dot-Product Attention 缩放点乘注意力结构图

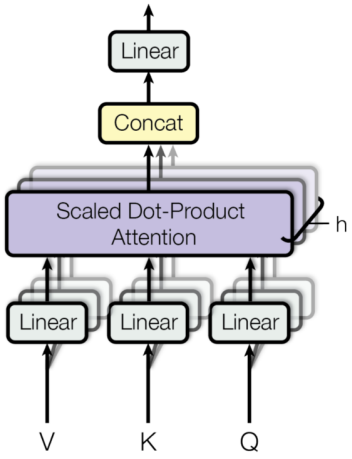

图 4-5　Multi-Head Attention 多头注意力模块结构图

分的形式如下：

$$\mathbf{F}_i = \mathrm{FFN}(\mathbf{S}_i) = \mathrm{ReLU}\left(\mathbf{S}_i\mathbf{W}^{(1)} + \mathbf{b}^{(1)}\right)\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \tag{4.9}$$

其中 $\mathbf{S}_i$ 表示嵌入层经过第 i 个 Self-Attention 模块计算后的输出

Residual Connections 残差连接

在序列建模的问题当中，用户最后一个点击浏览的物品通常会与下一个物品的联系较大，所以最后一个物品的特征权重应该会更大。由于 Self-Attention 模块在处理最后一个物品的嵌入向量的时候会将其与之前所有的其他物品嵌入向量做交互，因此最后一个物品的特征并没有特别关注。为了提升对这种浅层特征的提取能力，可以为模型增加残差连接操作，其主要思想是如果浅层特征对于目标学习有效的话，就直接将低层网络学习的浅层特征直接传递给输出层，而不用再经过深层网络提取复杂特征，残差连接的做法就是将浅层网络的输出建立旁路与输出层相连，跳过了部分中间复杂计算。

层次归一化 Layer normalization

为了使得模型训练的时候更加容易收敛，减少训练时间，在深度神经网络的模型中加入归一化操作是一个比较常见的做法。在训练卷积神经网络时候，常见的操作是加入批量归一化 (Batch Normalization)[39]，批量归一化通过计算均值和方差将一个批量的输入数据装换到 [0, 1] 之间，加速梯度下降，但其高度依赖批量的大小，同时也不适用于输入序列长度不固定的循环神经网络等模型。层次归一化

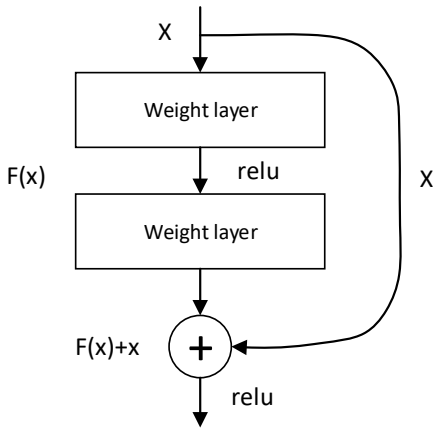

图 4-6　残差连接结构图

(Layer Normalization)[40] 则不是在样本层面，而是将该层神经元的输入通过计算均值和方差转换到在 [0, 1] 之间，这样的操作是跨特征计算的，与其他样本无关，因此可以使用任意大小的批量大小。

$$LN(x_i) = \alpha \times \frac{x_i - u_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{4.10}$$

Dropout

使用神经网络时，随着模型变得更大和更深，也带了过拟合的潜在风险，对于神经网络来说，除了在损失函数处加上 L1 和 L2 正则项之外，在模型的结构当中加入 Dropout 也是防止过拟合的常见操作之一，为了防止我们的模型过拟合、同时为了让训练更稳定，减少部分模型参数加速训练过程，在 Self-Attention 模块的输出之后加上 Dropout 操作：

$$\hat{y} = x + \text{ Dropout (SA(LN }(x))) \tag{4.11}$$

模型训练

最终模型根据输出序列是否为用户实际点击行为经过 $Sigmoid(\cdot)$ 函数映射到 [0, 1] 之间的概率，因此模型的优化目标为最小化如下的交叉熵损失函数：

$$-\sum_{u\in\mathbb{U}}\sum_{t\in[1,2,\ldots,n]}\left[\log(\hat{y}) + \sum_{\hat{s}_u^t \neq s_u^t}\log(1-\hat{y})\right] \tag{4.12}$$

4.3 复杂度分析

模型的时间复杂度只要由 Self-Attention 模块主导，Self-Attention 的时间复杂度为 $O(n^2d)$，其中 n 为序列的长度，d 为嵌入层维度，而 Multi-Head Attention 由多个 Self-Attention 并行叠加，假设有 h 个 Self-Attention，则 Multi-Head Attention 的时间复杂度仍为 $O(n^2d)$，

4.4 算法实现

算法 4.1 基于自注意力机制的序列感知推荐算法

已知: 用户序列数据：S；最大序列长度：$maxLen$；最大字数：$maxNum$
求: 推荐结果列表：$Items$

$sequences \leftarrow S$, 其中 $sequences = \{sen_1, sen_2 ... sen_n\}$
for $sequence$ in $input\_sequences$ **do**
    $embedding\_sequence()$
    $pos\_embedding\_sequence()$
    for $Self\_Attention$ in $Multi\_Head\_Attention$ **do**
        do $Self\_Attention()$
        do $Layer\_Normalize()$
        do $FeedForward()$
        $input\_sequence = sequence$*Mask
    end for
end for
for $sequence$ in $output\_sequences$ **do**
    $embedding\_sequence()$
    $pos\_embedding\_sequence()$
    for $Self\_Attention$ in $Multi\_Head\_Attention$ **do**
        do $Self\_Attention()$
        do $Layer\_Normalize()$
        do $FeedForward()$
        $output\_sequence = sequence$*Mask
    end for
end for
$sequence = output_sequence + input_sequence$
do $Layer\_Normalize()$
for $Self\_Attention$ in $Multi\_Head\_Attention$ **do**
    do $Self\_Attention(sequence)$
    do $Layer\_Normalize(sequence)$
    do $FeedForward(sequence)$
end for
$output = Sigmoid(sequence)$
$NextItem = Max(output)$
return $NextItem$

4.5 本章小结

本章节主要针对基于（双向）循环神经网络的序列感知推荐算法难以并行训练的缺点，采用自注意力机制，更进一步提升序列感知推荐算法的性能介绍本文提出的另外一个基于自注意力机制的序列感知推荐算法，其虽然抛弃了循环神经网络的递归结构，但引入位置信息编码的方式仍然能够捕获到序列位置的变化情况，没有了递归的结构依赖，也使得该模型的训练更容易并行化实现，在不损失推荐结果排序准确性的情况下还能获得较高的性能。

第 5 章　实验结果与分析

5.1　数据集介绍

为了比较上文提出的两个算法模型，分别从推荐结果的准确性、和算法训练与结果查询返回效率两个方面分别进行了两组实验。因此需要设计两组实验分别从算法推荐准确性和性能两个方面进行验证。本文评估上述两个算法在两个来自真实应用上的数据集，这些数据集都包涵用户与物品产生交互的时间戳：

MovieLens：MovieLens[1]是用来对推荐模型进行评估的最流行的基准数据集，它是由 GroupLens 研究组织从 MovieLens 网站收集的关于用户对电影评分的数据集，其中主要的评分文件以“用户 ID | 电影 ID | 评分 | 时间戳”为一行的格式保存了某个用户在某时刻对某个电影做出的评分标准，按照数据量的大小不同，本文选用的 MovieLens 数据集是 Movielens-1M。

Yoochoose：Yoochoose[2]数据集是 2015 年推荐系统顶级会议 RecSys 举办的挑战赛 RecSys Challenge 2015 上公开使用的目标数据集。Yoochoose 包含了一个在线电子商务网站在 6 个月时间内用户所有的点击会话流和购买会话流。这里我只使用了点击会话流数据集“yoochoose-clicks.dat”，其中每一行包含了一个点击交互行为，总共有 33003944 行，这里我们抽取了其中的前 100 万行进行建模实验。对于每一种数据集，在特征工程处理部分，需要将其处理成按时间排列的序列，因此需要将交互行为按照时间戳进行预排序，将排序好的交互行为按照其所属用户或会话流分组构成序列集合。图5-3为以 Movielens-1M 数据集为例，特征工程前后数据结构示意图，Yoochoose 数据集特征工程与此相同。不同的算法使用

[1]https://grouplens.org/datasets/movielens/
[2]http://2015.recsyschallenge.com

表 5-1　序列数据集对比

数据集名称	物品数目	序列数目	交互行为数目	平均序列长度
MovieLens-1M	3706	6040	1000000	165.56
Yoochoose	20843	254619	1000000	3.93

的数据集划分策略不同，下文针对每种算法将详细描述数据集划分策略。具体实验数据明细如表格5-1所示。

	UserID	MovieID	Rating	Timestamp
1000138	6040	858	4	956703932
1000153	6040	2384	4	956703954
999873	6040	593	5	956703954
1000007	6040	1961	4	956703977
1000192	6040	2019	5	956703977

图 5-1　原始数据

	UserID	sequence	Timestamp
0	1	[1193, 661, 914, 3408, 2355, 1197, 1287, 2804,...	978300019
1	2	[1357, 3068, 1537, 647, 2194, 648, 2268, 2628,...	978298124
2	3	[3421, 1641, 648, 1394, 3534, 104, 2735, 1210,...	978297018
3	4	[3468, 1210, 2951, 1214, 1036, 260, 2028, 480,...	978293924
4	5	[2987, 2333, 1175, 39, 288, 2268, 1466, 866, 2...	978241072

图 5-2　特征工程处理后数据

图 5-3　以 Movielens-1M 数据集为例，将原始交互数据按时间戳排序后分组构成行为序列

5.2　评价指标

作为一个推荐模型，模型会对查询目标给出点击可能性最高的 N 个物品，对于本文提出的序列推荐形式，模型的输入是 t 时刻及其以前的用户点击物品序列，需要的模型输出目标是 $t+1$ 时刻用户可能点击的物品，以 $\hat{I}_u^{t+1}$ 表示，所以本文采用了 $Precision@N, Recall@N$ 指标来评估我们的模型，其计算形式如下：

$$Precision@N = \frac{\sum_u |\hat{I}_u^{t+1} \cap I_u^{t+1}|}{|\mathbb{U}| * N} \tag{5.1}$$

$$Recall@N = \frac{\sum_u |\hat{I}_u^{t+1} \cap I_u^{t+1}|}{\left|\sum_u |I_u^{t+1}|\right|} \tag{5.2}$$

由于 $Precision@N$ 和 $Recall@N$ 值可以通过阈值调整来相互权衡，为了得到一个更加容易量化评估的指标，本文引入了 $F1@N$ 指标，如果 $F1@N$ 分数越大，可以认为模型的效果更好。而当一次为用户推荐多个结果时，推荐结果在屏幕上展示的位置也会影响物品被点击的概率，越靠前的物品越可能被点击，约强相关的推荐物品应该出现在结果列表的越前面，而引入 $NDCG@N$[41] 指标的原因就是为了感知这种位置的影响，让点击率越高的物品位置越靠前，$NDCG@N$ 值越接近 1，得到的相关推荐结果中前 N 个物品的排序越准确，所以 $F1@N$ 和 $NDCG@N$ 的计算形式如下：

$$F1@N = \frac{2 \times Precision@N \times Recall@N}{Precision@N + Recall@N} \tag{5.3}$$

$$
\begin{aligned}
DCG@N &= \sum_{i=1}^{N} \frac{2^{rel_i}-1}{\log_2(i+1)} \\
IDCG@N &= \sum_{i=1}^{|REL|} \frac{2^{rel_i}-1}{\log_2(i+1)} \\
NDCG@N &= \frac{DCG@N}{IDCG@N}
\end{aligned}
\tag{5.4}
$$

当推荐的物品是用户点击的内容时 $DCG@N$ 中 rel 取值为 1，否则 rel 取值为 0。而由于每条样本仅有一条为相关，其余均为不相关，所以 $IDCG@N$ 对于每条样本都是一样的，即 $\frac{1}{\log 2}$。

5.3 实验环境

表 5-2 实验软硬件环境配置表

实验环境	环境配置
CPU	Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz *2
GPU	Tesla K80 *4
内存	64126MB
OS	CentOS Linux 7 (Core)
编程语言	Python 3.6.8
开发工具	TensorFlow-GPU 1.12.0

5.4 基于双向 LSTM 的序列感知算法实验过程

本节将通过实验来验证第三章中提出的基于双向 LSTM 的序列感知算法的有效性，主要介绍针对该实验的数据集处理策略、实验主要研究的问题、影响结果的主要因素。基于双向 LSTM 的序列感知算法的特征工程部分，首先将每个交互样本按照时间戳从先到后排列，然后根据交互样本所属用户聚类构成行为序列，在序列感知推荐算法当中不需要用户的评分数据因此可以去除。对于行为序列长度小于 5 的序列进行了过滤，剩余序列集合前 80% 划分为训练集，后 20% 划分为测试集。将序列中最后一个交互物品 ID 作为模型学习的标签，截取序列中最近的 N 个物品作为 BiLSTM 的输入序列，其中 $N \geq 1$，当 $N > 1$ 而存在序列的长度不足

时，在长度不足序列的前部进行特殊值填充以表示缺失。当时间窗口为 1 时，意味着我们将序列的最后一个物品作为标签，最近的一个物品作为序列的特征输入模型就行学习。

BiLSTM 模型的实现基于开源深度学习框架 Keras，以 Tensorflow 作为其后端。使用批量梯度下降法 Adam 作为优化器，并设置初始学习率 α 为 0.01，学习率衰减权重因子 β 为 0.9。其他超参数本实验首先使用常见值作为初始值，为了得到更好的性能，使用网格搜索的形式自动寻找最优参数组合，但这会带来极大的计算量。经过多次尝试，本实验最终选定了以下超参数组合：嵌入层因子 $|e| = 100$，循环层神经元数目 $H = 200$，全连接层神经元数目 $O = 100$，梯度下降批量大小为 32。在训练阶段为避免过拟合，当验证数据上经过 3 个 batch 损失函数仍没有下降则 EarlyStopping。

为了探求基于双向 LSTM 的序列感知算法需要的序列长度，将超参数固定之后本文也设置了对比实验探求不同序列长度对模型的影响，图5-4显示了在 MovieLens-1M 数据集上输入序列长度分别从 1Ĩ0 时，训练阶段 EarlyStopping 时最优的损失函数值。通过图5-4中可以看到，在不同的输入序列长度下，BiLSTM

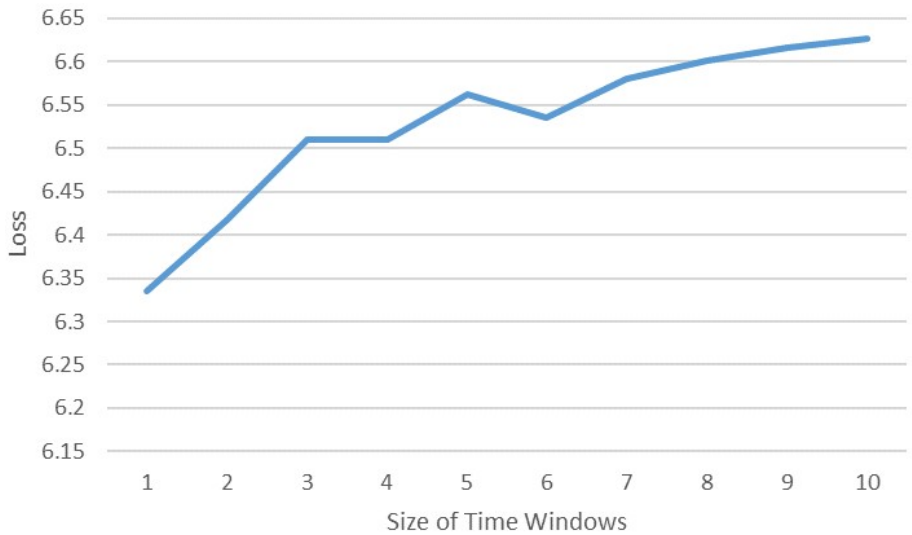

图 5-4　BiLSTM 在 MovieLens-1M 数据上不同序列长度的最小训练 loss

的性能有稍微为降低，为了模型的稳定性选取了中位值 5 作为最终序列长度。图5-5显示了固定时间窗口为 5 及其他超参数的情况下，训练集和测试集上的损失函数下降图。

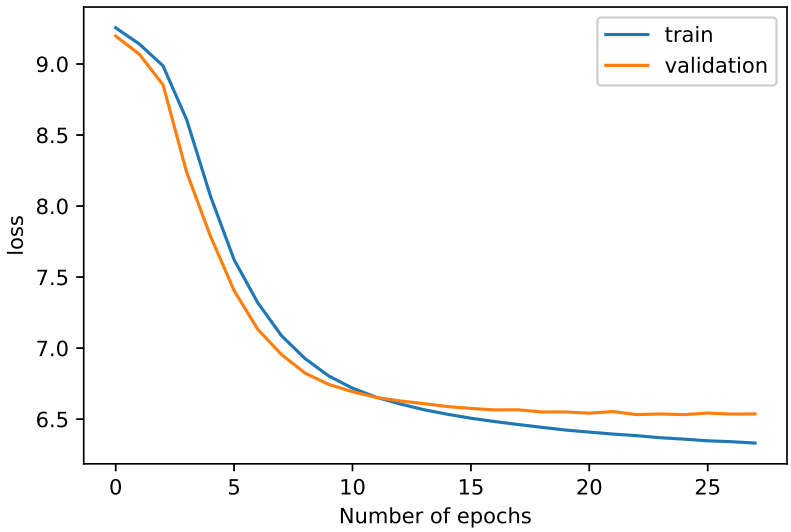

图 5-5　BiLSTM 在 MovieLens-1M 数据上不同序列长度的最小训练 loss

5.5　基于自注意力机制的序列感知算法实验过程

本节将通过实验来验证第四章中提出的基于自注意力机制的序列感知算法的有效性。与基于双向 LSTM 的序列感知算法的特征工程部分一样，首先将每个交互样本按照时间戳从先到后排列，然后根据交互样本所属用户聚类构成行为序列。在数据集划分部分，由于基于自注意力机制的序列感知算法包含 Encoder-Decoder 结构，Decoder 对应的输出序列由输入序列往后滑动一个窗口构成，在数据集划分部分也使用了同样的划分策略，将所有序列截止倒数第三个位置之前部分划分为训练集，截止倒数第二个之前部分划分为验证集，包含最后一个物品的序列作为测试集，超过最大长度 N 部分做截取处理。

图5-6为学习率分别设置为 0.001、0.003、0.01 时训练集上的损失函数的下降曲线，从图中可以看到，Transformer4Rec 模型在 MovieLens-1M 上使用合适的学习率损失函数可以更快收敛，节省训练时间，因此后文实验以学习率 0.01 为基础进行。

模型输入序列的长度也是影响模型性能的关键因素之一，在固定学习率及其他超参数之后本节试验了不同长度序列对模型性能的影响，如图 5-7所示，从图中可以看到不同序列长度对模型性能的影响不是非常明显，但越长的序列它的迭

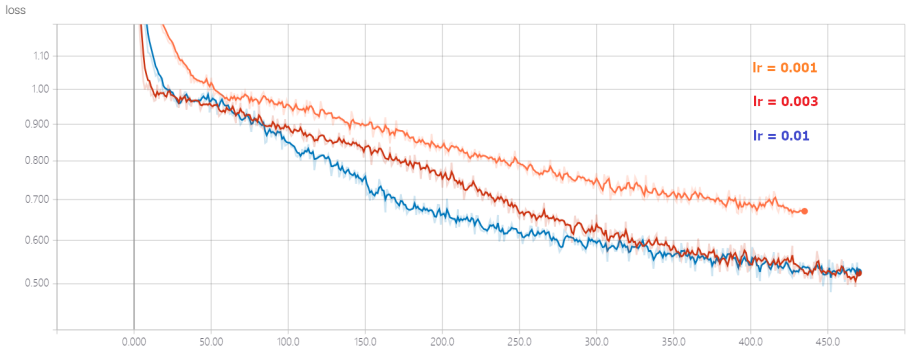

图 5-6　Transformer4Rec 在 MovieLens-1M 数据上学习率分别设置为 0.001、0.003、0.01 时训练集上的损失函数的下降曲线

代过程越慢，综合考虑性能与效率的原因，在本节 MovieLens-1M 数据集上最大序列长度设为 150，在 Yoochoose 数据集上最大长度设为 50。其他重要超参数如

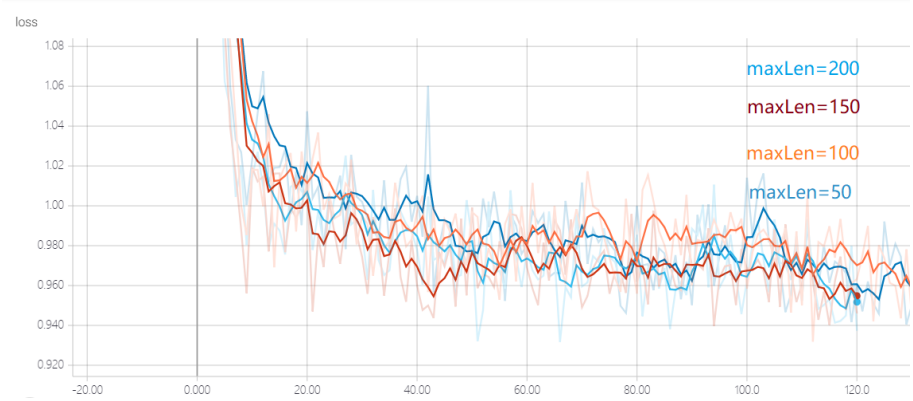

图 5-7　Transformer4Rec 在 MovieLens-1M 数据上最大序列长度分别设置为 200、150、100、50 时训练集上的损失函数的下降曲线

Dropout 比率、Multi-Head Attention 中 Self-Attention 个数、批量大小、迭代次数、隐藏层神经元数目等，经过自动网格搜索后分别确定为：0.2、2、128、10、50。

5.6　对比实验

5.6.1　算法测评指标对比

POP：最流行物品推荐法 POP 作为本文选择的最简单的基准线，通过统计训练数据中出现次数最多的物品，推荐策略为为用户推荐那些他还没有看过的最热门的物品。

GRU4Rec[20, 42]：GRU4Rec 是序列感知推荐领域最重要的算法之一，2016 年 Balazs Hidasi 等人首次将循环神经网络的方法应用到基于会话的个性化推荐领域，

表 5-3　各算法评价指标结果

数据集名称	测评指标	POP	GRU4Rec	**BiLSTM**	Caser	**Self-Attention**
MovieLens-1M	Recall@10	0.041390	0.109286	0.125679	0.1344	0.7055
	Precision@10	0.004139	0.010928	0.012567	0.0112	0.0705
	F1@10	0.007526	0.019870	0.023052	0.0244	0.1279
	NDCG@10	0.019398	0.056077	0.064489	0.4538	0.4398
Yoochoose	Recall@10	0.041493	0.105735	0.121595	0.1374	0.8134
	Precision@10	0.004149	0.010573	0.012159	0.0137	0.0813
	F1@10	0.007544	0.019224	0.022107	0.0249	0.1483
	NDCG@10	0.019779	0.055170	0.063445	0.4621	0.5066

并取得了当时的最好效果，因此本文将其作为基于循环神经网络的序列感知推荐算法中的代表，其代码基于 Keras[3]提供的 GRU 高层 API 实现。

Caser[24]：2018 年 Jiaxi Tang 等人将序列嵌入矩阵使用 CNN 提取特征，提出了卷积序列嵌入推荐模型 Caser，因此本文将其作为基于卷积神经网络的序列感知推荐算法中的代表，其代码实现基于原作者开源的 PyTorch 实现版本[43]。

本文讨论的两个模型与三个对比模型的分别在 MovieLens-1M 与 Yoochoose 数据集上的实验结果如表5-3所示。

使用双向长短期记忆网络的 BiLSTM4Rec 模型相比 GRU4Rec，有约 15% 的推荐准确性提升，这说明双向结构对序列建模问题能起到比简单单向结构更好的效果。而基于 Transformer 结构的 Transformer4Rec 达到了对比实验中最好的效果，说明引入 Attention 机制能够更好地突出重点信息，提取到的序列特征表达更加准确。

5.6.2　算法运行时间比较

在 3.4 节和 4.3 节，分别分析了第三章基于双向 LSTM 的序列感知推荐方法和第四章基于自注意力机制的序列感知推荐方法的时间复杂度，在推荐领域，面对大数据的压力时，模型的返回时延与效率更是选择模型应用的重要考虑因素，表5-4给出了本实验中各个算法在测试集上预测的运行时间，其中 MovieLens-1M

[3]https://keras.io

表 5-4　各算法测试集预测时间比较

数据集	POP	GRU4Rec	Caser	BiLSTM4Rec	Transformer4Rec
Movielens-1M	1.99ms	18.35s	19.1s	21.1s	1.7s
Yoochoose	17.5ms	30.7s	31.98s	46.9s	11.3s

测试集中包含 1208 条平均序列长度为 165.56 的样本，由于对序列长度小于 5 的用户数据进行了过滤，Yoochoose 测试集中包含 43298 条平均长度为 10.09 的样本，对于每个样本，每种算法都给出可能性最高的 10 个推荐结果，测试集算法运行时间不包含评估阶段。从表中可以看出，具有最简单结构的 POP 算法具有最高效的查询返回时延，除此之外，Transformer4Rec 模型在具有复杂深度网络结构的模型中取得了最好的查询返回时延，而基于双向循环神经网络的 BiLSTM4Rec 方法虽然正向与反向可以并行，但其 LSTM 本身结构并为做多大改进，双向结构反而增加了一定时间复杂度。

5.7　本章小结

本章首先主要介绍了实验中采用了两大公开数据集，MovieLens 与 Yoochoose，然后介绍了推荐排序领域广泛采用的测评标准。对于本文提出的两个基于双向循环神经网络的 BiLSTM4Rec 模型和基于 Self-Attention 的 Transformer4Rec 模型对其实验过程分别进行了描述。为了对比本文模型的性能，引入热门推荐 POP 和基于循环结构的序列感知推荐代表 GRU4Rec 和基于卷积结构的序列感知推荐代表 Caser。考虑到模型过于复杂时可能引入的大数据压力问题，并从查询返回时延的角度分析了各个对比算法的效率，最后根据实验结果从多个角度分析了本文设计的模型的可行性和有效性。

第 6 章　总结与展望

6.1　论文总结

推荐系统是数据挖掘与机器学习领域中最成功的应用之一，推荐算法作为推荐系统的灵魂，支撑着系统的表现。经典推荐算法面对大数据的压力时往往显得力不从心，而基于梯度下降优化的深度学习推荐算法没有理论上大数据上限。现有的推荐算法又往往忽视了隐藏在用户行为之中的时序关系，目前大多数深度学习推荐算法又需要大量的特征，当面对缺少用户隐私数据的场景这些算法都将无能为力。为了应对当前推荐算法这几个方面面临的问题，本文完成的工作如下：

1. 本文调研了深度学习在推荐系统应用领域的发展情况，针对现有推荐算法都需要使用大量用户隐私数据构造特征的情况下，提出了只利用用户历史行为序列的序列感知推荐算法。

2. 本文设计了基于双向长短期记忆网络的序列感知推荐模型 BiLSTM4Rec，对用户历史行为序列进行嵌入式表达，生成稠密矩阵，利用双向长短期记忆网络对嵌入矩阵进行序列建模，预测下一个物品，将推荐问题转变成一个多分类问题。

3. 考虑到循环神经网络难以并行训练的效率问题，本文抛弃神经网络循环结构，从 Transformer[25] 模型中抽取 Self-Attention 模块，提出 Transformer4Rec 模型，该模型在获得较大推荐精度的情况下还不损失性能。

6.2　论文展望

本文主要针对序列感知推荐领域进行了研究，利用神经网络序列建模技术，有效的提高了推荐结果的精确性，取得了一定的工作成果。但是序列感知推荐领域还存在许多的难题，这也是我们未来工作中可以持续改进的部分，主要包括以下几点：

1. 本文提出的两个序列感知推荐算法主要利用了深度学习中的序列建模技术，但深度学习模型的可解释性很差，而推荐系统中又存在大量的需要可解释性场景，在后期的研究中，应使用具有可解释性的序列建模技术，例如图神经网络。

2. 本文提出的序列感知推荐方法不需要大量的用户隐私数据与物品属性特征，但为了虽然满足某些特定的应用场景，但为了获得具有更加鲁棒性的推荐模型，应考虑将序列感知模型与现有的推荐模型结合，让不同特点的模型发挥各自的优点。

3. 本文对基于神经网络的序列感知模型进行了粗略的设计与实现，但由于时间与条件的限制，还要许多细节没有完善，与现有的许多成熟推荐模型相比还有相当大的差距，在后期的工作中，将对本模型进行持续完善，使其能成为当今领域推荐算法工具箱中的一员。

参考文献

[1] 中国互联网络信息中心. 第 43 次《中国互联网络发展状况统计报告》 [R]. 2019.

[2] hsin Chen Y, George E I. A Bayesian Model for Collaborative Filtering. 2002.

[3] Ungar L H, Foster D P. Clustering methods for collaborative filtering [C]. In AAAI workshop on recommendation systems. 1998: 114–129.

[4] Salakhutdinov R, Mnih A. Probabilistic Matrix Factorization [C]. In Proceedings of the 20th International Conference on Neural Information Processing Systems. USA, 2007: 1257–1264.

[5] Kondakindi G. A Logistic Regression Approach to Ad Click Prediction [C]. 2014.

[6] Rendle S. Factorization Machines [C]. In Proceedings of the 2010 IEEE International Conference on Data Mining. Washington, DC, USA, 2010: 995–1000.

[7] Rendle S. Factorization Machines with libFM [J]. ACM Trans. Intell. Syst. Technol. 2012, 3 (3): 57:1–57:22.

[8] Juan Y, Zhuang Y, Chin W-S, et al. Field-aware Factorization Machines for CTR Prediction [C]. In Proceedings of the 10th ACM Conference on Recommender Systems. New York, NY, USA, 2016: 43–50.

[9] He X, Pan J, Jin O, et al. Practical Lessons from Predicting Clicks on Ads at Facebook [C]. In Proceedings of the Eighth International Workshop on Data Mining for Online Advertising. New York, NY, USA, 2014: 5:1–5:9.

[10] Zhang W, Du T, Wang J. Deep Learning over Multi-field Categorical Data [J]. Advances in Information Retrieval. 2016: 45‑57.

[11] He X, Chua T-S. Neural Factorization Machines for Sparse Predictive Analytics [J]. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR ’17. 2017.

[12] Xiao J, Ye H, He X, et al. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks [J]. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017.

[13] Qu Y, Cai H, Ren K, et al. Product-Based Neural Networks for User Response Prediction [J]. 2016 IEEE 16th International Conference on Data Mining (ICDM). 2016.

[14] Guo H, TANG R, Ye Y, et al. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction [C]. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. 2017: 1725–1731.

[15] Cheng H-T, Ispir M, Anil R, et al. Wide & Deep Learning for Recommender Systems [J]. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems - DLRS 2016. 2016.

[16] Wang R, Fu B, Fu G, et al. Deep & Cross Network for Ad Click Predictions [C]. In Proceedings of the ADKDD'17. New York, NY, USA, 2017: 12:1–12:7.

[17] Lian J, Zhou X, Zhang F, et al. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems [C]. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA, 2018: 1754–1763.

[18] Zhou G, Zhu X, Song C, et al. Deep Interest Network for Click-Through Rate Prediction [C]. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA, 2018: 1059–1068.

[19] Zhou G, Mou N, Fan Y, et al. Deep Interest Evolution Network for Click-Through Rate Prediction [J]. arXiv preprint arXiv:1809.03672. 2018.

[20] Hidasi B, Karatzoglou A, Baltrunas L, et al. Session-based Recommendations with Recurrent Neural Networks [J]. CoRR. 2015, abs/1511.06939.

[21] Devooght R, Bersini H. Collaborative filtering with recurrent neural networks [J]. arXiv preprint arXiv:1608.07400. 2016.

[22] Devooght R, Bersini H. Long and Short-Term Recommendations with Recurrent Neural Networks [C]. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization. New York, NY, USA, 2017: 13–21.

[23] Zhu Y, Li H, Liao Y, et al. What to Do Next: Modeling User Behaviors by Time-LSTM [C]. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. 2017: 3602–3608.

[24] Tang J, Wang K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding [C]. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. New York, NY, USA, 2018: 565–573.

[25] Vaswani A, Shazeer N, Parmar N, et al. Attention is All you Need [M] // Guyon I, Luxburg U V, Bengio S, et al. Advances in Neural Information Processing Systems 30. Curran Associates, Inc., 2017: 2017: 5998–6008.

[26] Quadrana M, Cremonesi P, Jannach D. Sequence-aware Recommender Systems [C]. In Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization. New York, NY, USA, 2018: 373–374.

[27] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult [J]. IEEE Transactions on Neural Networks. 1994, 5 (2): 157–166.

[28] Sutskever I. Training Recurrent Neural Networks [D]. Toronto, Ont., Canada, Canada: University of Toronto, 2013. AAINS22066.

[29] Hochreiter S, Schmidhuber J. Long Short-Term Memory [J]. Neural Computation. 1997, 9 (8): 1735–1780.

[30] Cho K, van Merrienboer B, Bahdanau D, et al. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches [J]. CoRR. 2014, abs/1409.1259.

[31] Schuster M, Paliwal K K. Bidirectional recurrent neural networks [J]. IEEE Trans. Signal Processing. 1997, 45: 2673–2681.

[32] Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling [J]. arXiv preprint arXiv:1803.01271. 2018.

[33] van den Oord A, Kalchbrenner N, Kavukcuoglu K. Pixel Recurrent Neural Networks [J]. CoRR. 2016, abs/1601.06759.

[34] Gehring J, Auli M, Grangier D, et al. Convolutional Sequence to Sequence Learning [J]. CoRR. 2017, abs/1705.03122.

[35] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and Their Compositionality [C]. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. USA, 2013: 3111–3119.

[36] Resnick P, Iacovou N, Suchak M, et al. GroupLens: An Open Architecture for Collaborative Filtering of Netnews [C]. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work. New York, NY, USA, 1994: 175–186.

[37] Sarwar B, Karypis G, Konstan J, et al. Item-based Collaborative Filtering Recommendation Algorithms [C]. In Proceedings of the 10th International Conference on World Wide Web. New York, NY, USA, 2001: 285–295.

[38] Cho K, van Merrienboer B, Gulcehre C, et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation [C]. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014: 1724–1734.

[39] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [C]. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. 2015: 448–456.

[40] Ba J L, Kiros J R, Hinton G E. Layer normalization [J]. arXiv preprint arXiv:1607.06450. 2016.

[41] Valizadegan H, Jin R, Zhang R, et al. Learning to Rank by Optimizing NDCG Measure [M] // Bengio Y, Schuurmans D, Lafferty J D, et al. Advances in Neural Information Processing Systems 22. Curran Associates, Inc., 2009: 2009: 1883–1891.

[42] Hidasi B, Karatzoglou A. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations [C]. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management. New York, NY, USA, 2018: 843–852.

[43] Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch [C]. In NIPS-W. 2017.

符号列表

CF	协同过滤（Collaborative Filtering）
UBCF	基于用户的协同过滤 (User-based Collaborative Filtering，UBCF)
IBCF	基于物品的协同过滤 (Item-based Collaborative Filtering，IBCF)
RNN	循环神经网络（Recurrent Neural Network）
LSTM	长短期记忆神经网络（Long Short-Term Memory）
ANN	人工神经网络（Artificial Neural Network）
FNN	前馈神经网络（Feedforward Neural Network）
MLP	多层感知机（Multi-Layer Perceptron）
BRNN	双向循环神经网络 (Bi-directional Recurrent Neural Network)
GRU	门控循环单元 (Gated Recurrent Unit)
TCN	时间卷积网络（Temporal Convolutional Network)

表 目 录

图 目 录

致　谢

作者攻读学位期间发表的学术论文目录