硕士专业学位论文

面向轨迹相似度计算的轨迹深度表达学习研究

Learning Trajectory Deep Representation for Trajectory Similarity Computation

作者：王亚珊

导师：郭宇春

北京交通大学

2020 年 5 月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

学校代码：10004 密级：公开

北京交通大学

硕士专业学位论文

面向轨迹相似度计算的轨迹深度表达学习研究

Learning Trajectory Deep Representation for Trajectory Similarity Computation

作者姓名：王亚珊 学 号：18125063

导师姓名：郭宇春 职 称：教授

专业学位类别：电子与通信工程 学位级别：硕士

北京交通大学

2020 年 5 月

致谢

本论文的工作是在我的导师郭宇春教授的悉心指导下完成的，郭宇春教授是在求学路上给予我强大力量的榜样，她严谨认真的治学态度引导我不断在学习的道路上努力钻研、反思进取，生活中积极向上的态度、开阔的视野和丰富的人生经验也带给我许多帮助，在今后的学习生活中我将继续秉承认真踏实的态度，做好分内的每件小事。衷心感谢郭老师对我的关心和指导。

感谢陈一帅老师对我一直以来的鼓励和帮助，在研究生学习期间，陈老师亲切地指导和帮助我们利用代码实现想法，包容我们不断试错，身体力行地指导我们如何成为一名合格的工程师。陈老师经常分享的新思路、新理念也帮助我们打开了视野，是我在学习和做人上的榜样。感谢赵永祥、李纯喜、郑宏云、张立军、孙强、张梅老师等在研究生期间对我的帮助。进入实验室、在实验室学习的时光能得到诸位老师的指导是一件十分幸运的事情。

此外，感谢刘翔师兄对我的耐心指导和有力帮助，以及陈滨师兄、魏中锐师兄、杨晶晶师姐、尹姜谊师姐、冯梦菲师姐、于兹灏师兄、高志朋师兄、赵红娜师姐、宋云鹏师兄等师兄师姐对我的支持和帮助，此外同样感谢曹中、韩致远、潘翰祺、孙欢、戚余航、李想、王珍珠、李小乐、崔子琦等同学、我的室友王玉环和王萱对我的帮助，能遇到良师和益友是人生一件幸事，他们的智慧和鼓励让我能顺利地完成本论文的研究工作。

感谢我的家人在精神上、生活上无微不至的关心与鼓励，令我在研究生学习期间努力进取、以至今日。今后我将继续努力，争取不负你们的嘱托，成为一个更好的人，实现自己的价值，为集体和社会做出贡献。

最后感谢北京交通大学及电子信息工程学院对我的栽培，感谢参与论文评审和答辩的各位老师的辛勤工作和付出，老师们的宝贵指导意见为我指明了不断改进的方向。

摘要

支持 GPS 的移动设备已经普及，大规模轨迹数据获取及挖掘分析随之成为可能。轨迹相似度计算是轨迹挖掘中的一项基本任务，为移动模式研究、出行行为分析、城市热门区域发现等应用奠定了基础，其研究具有重要的现实意义和实际价值。

轨迹是空间中一条连续的曲线，通常以由样本点组成的序列形式出现。现实中得到的轨迹通常存在采样频率不一致、采样频率低、有噪声干扰等问题，基于轨迹点对匹配的相似度计算方法难以准确发现相似轨迹。深度学习的方法可以在向量空间提取轨迹的深层特征，得到轨迹的定长、低维表达，在此基础上相似度的计算会更加准确。但目前的轨迹深度表达模型大多数是基于循环神经网络（RNN）的编码器-解码器结构，处理数据时的视野有限，无法解决长距离依赖问题，轨迹特征提取能力有待提升。

本文将自然语言处理领域的模型与轨迹表达问题结合，提出了两种基于多头注意力机制的轨迹深度表达模型 Transformer-traj 和 BERT-traj，为相似轨迹的发现提供有力的工具。本文的主要贡献如下：

（1）在轨迹深度表达模型中引入了多头注意力机制，代替了传统的以循环神经网络为基础的模型。多头注意力机制在提取特征时能够提取轨迹的长距离特征，并支持不同尺度特征的有效融合，具有更好的轨迹表达能力，同时支持并行计算。

（2）提出了轨迹深度表达模型 Transformer-traj 和 BERT-traj，在结构上前者是后者的基础，在性能上后者实现了较高的准确率和较强的特征提取能力。本文通过将经过下采样和加噪声处理的数据作为输入进行模型训练，使得模型在轨迹采样率低、采样率不均匀和有噪声干扰的情况下依然能够较为准确地获得轨迹的向量表达，有良好的健壮性。

（3）在真实数据集上的实验验证了 BERT-traj 获得准确轨迹表征的能力。本文采用了 3 种测度指标，即自身相似度、交叉相似度和轨迹 KNN 查询，评估模型的表达准确度，实验结果显示 BERT-traj 性能优于基线模型。其中交叉相似度既考虑到相似轨迹间的距离、也考虑到不同轨迹间的距离，是一个较为全面的衡量指标。在小型数据集上的结果显示，加噪声和下采样干扰情况下，BERT-traj 相比基线模型 t2v 在这个指标上分别高出 23.26%和 14.06%；在大型数据集上，BERT-traj 相比基线模型 t2v 分别高出 23.71%和 22.42%。

图 26 幅，表 12 个，参考文献 48 篇。

关键词：深度学习；轨迹相似度计算；注意力机制

ABSTRACT

Mobile devices equipped with GPS have been widely spread, so that obtaining and analysis of large-scale trajectory data have become available. Among all trajectory data mining tasks, trajectory similarity computation is fundamental, which can be further applied in mobility patten research, travel behavior analysis, popular area discovery in city and so on. Therefore the research on trajectory similarity computation is of great practical value.

Trajectory is a continuous curve in spatial domain, which is usually represented by sequences composed of sample points. In reality non-uniform or low sampling rates and data noise are common problems existing in trajectory data, traditional methods based on pairwise matching of sample points are not able to compute trajectory similarity accurately. Luckily, deep learning methods bring hope to the solving of the challenge, it can transform the input into a low-dimension vectors with fixed length, during which the features of the data can be learned, and trajectory similarity computation can be more accurate using the learned vectors. While existing methods tend to build encoder-decoder model based on Recurrent Neural Network (RNN), which has limited horizon while solving data, and it is not able to remember information far away from current moment. Therefore, a new model which can extract features in trajectories more accurately is needed to be researched.

We combine the models used in natural language processing area with the solving of trajectory representation problem by proposing trajectory deep representation learning models named Transformer-traj and BERT-traj, which are based on multi-head attention mechanism, and they can function as a powerful tool in trajectory similarity computation. The contribution of the dissertation can be summarized as follow.

(1) We introduce multi-head attention mechanism into the trajectory deep representation learning model, in substitute for models based on recurrent neural network. The attention mechanism can capture long-distance features in trajectories and integrate features of different scales, which ensures the model a more powerful trajectory representation capability, and in addition, the model based on attention mechanism can run in parallel.

(2) We propose the trajectory deep representation model Transformer-traj and BERT-traj. From the perspective of structure, the former serves as the foundation for the latter; and as for the performance the former achieves higher accuracy and more powerful

feature extraction capability. And models are trained by inputting data preprocessed by down sampling and noise adding, so that our models are robust enough to obtain accurate trajectory representation in the case of non-uniform and low sampling rates, as well as noise interference.

(3) We conduct extensive experiments on real-world dataset, which proves the capability of BERT-traj in obtaining accurate trajectory representation. There are three evaluation methods utilized to assess the models, which are self-similarity, cross-similarity, and KNN query of trajectory respectively, and the results show that BERT-traj achieves better performance than baseline model. Particularly, cross similarity considers both the distance between similar trajectories and distance between different trajectories, it can be considered a more comprehensive evaluation measure. Using the evaluation method on small dataset, the accuracy of BERT-traj is higher than that of baseline model by 23.26% and 14.06% when distorting and down sampling are implemented, respectively; while on a larger dataset the accuracy of BERT-traj is higher than that of baseline model by 23.71% and 22.42%, respectively.

26 figures, 12 tables, and 48 reference articles are contained in the dissertation.

KEYWORDS：Deep learning; Trajectory similarity computation; Attention mechanism

目录

1 引言

本章首先介绍本文的研究背景与意义，阐述基于深度学习的轨迹表达研究的重要性及目的。接着对于轨迹相似度计算及轨迹表达学习等方面的研究现状进行简单介绍，对前人工作和待发展方向进行表述。再针对研究现状提出本文主要研究内容。最后给出本文的组织结构。

1.1 研究背景与意义

随着各种基于 GPS 的定位设备、内置定位功能的智能手机的应用和发展，大量交通轨迹数据不断产生，这也使得获取城市交通轨迹数据并对其进行研究成为可能，智能交通系统（ITS）[1]的研究建立在对这些轨迹数据进行分析的基础上。由于数据产生及存储技术的发展，ITS 得以建立丰富的数据库资源，这些交通轨迹数据不仅有利于进行交通管理，减少拥堵等情况的发生，而且对轨迹数据进行挖掘可以辅助人类进行决策。例如对轨迹数据进行分析可以发现运动对象的移动规律和行为模式，比如轨迹聚类[2, 3]、司机驾驶行为分析[4]等研究，进而为路径推荐、路况预测、城市规划、出行服务提供有效支持。可以为城市智能交通系统的建设、公安部门的轨迹分析、企业的商业化活动提供强大帮助。因此，对于移动对象的轨迹挖掘分析具有重要的研究价值和现实意义[5]。

在所有轨迹分析任务中，轨迹相似发现是最基础也最重要的任务[4]。在交通导航系统中，基于轨迹相似性查询可以获得类似行为用户的历史轨迹，为司机的驾驶行为提供多样化、个性化的参考方案；在交通规划和管理中，相关部门可以根据相似轨迹的车辆的出行特点对道路流量进行预估，施行措施从而针对性地减少拥堵，如规划新道路等；商家可以利用相似轨迹数据分析某类用户特点，更好地开展营销活动，增加经济效益；公安部门可以利用相似轨迹在处理案件时筛选可疑团体，在案件发生后根据相似轨迹记录提高嫌疑人捕获的效率。

但海量的数据在给我们提供分析资源的同时也带来了挑战，在处理和分析海量数据从而进行数据挖掘时要求强大的计算机存储、处理能力。而根据目前的发展情况，直接处理、挖掘原始海量数据实现难度太高。因此我们需要研究高效完善的策略[6,7]对海量的交通数据进行去重、去噪、规约[8]等预处理。一条轨迹通常是物体移动路线的采样，表示为一串离散的地理位置序列或样本点，描述了一个移动物体随时间变化的潜在路线。现实采集到的轨迹数据的数据量庞大、稀疏，在时间上不同用户不同时间采集到的轨迹点分布不均匀，表现为采样间隔不等、采样点个数不

同等。直接用原始轨迹或使用 one-hot 编码进行相似度计算面临着维度过高、过于稀疏等挑战，基于 DTW[9], LCSS[10], ERP[11], EDR[12]等的传统方法将轨迹点对进行匹配的方法不适用于当前情况，尤其在轨迹采样频率低、采样频率不一致、样本点有噪声等情况时性能较差，且计算复杂度普遍较高。

针对上述问题，如果能够在深层次提取轨迹时空特征的情况下得到对轨迹的准确表达，也即对轨迹进行降维，进行数据简约，同时保留关键信息，相似轨迹发现算法的准确度和效率能够有效提升。这使得对轨迹的降维表达成为任务的核心。现有的轨迹降维方法包括在线和离线两种方式，其中后者较为常见，如主成分分析法（PCA）等。大部分方法通过提取部分轨迹数据形成轨迹的粗粒度表达，但上述方法可能会造成轨迹关键特征的丢失，影响后续任务的准确性，所以亟需一种新的轨迹降维方法，在对轨迹数据进行简约的同时对轨迹的特征进行表达。

近年来基于深度学习的表达学习算法逐渐得到了人们的关注，不同的深度学习模型如 RNN（Recurrent Neural Network）[13]，Seq2Seq（sequence to sequence）[13, 14]等都被发现具有强大的特征提取和表征能力，能够为我们获取低维空间的轨迹表示提供有力帮助，此外因为原始轨迹被转换为固定长度的向量表示进而学习特征，而非对轨迹点对进行匹配和比较，在应对异质采样频率、噪声、轨迹相似性时空迁移等情况下也会有良好的表现。但现有的基于深度学习的轨迹表达算法[13,14]使用循环神经网络将轨迹作为时序数据进行学习，视野较窄，牺牲了空间信息，且不能并行处理数据，当数据长度较长时无法利用距离较远处的信息。而 Transformer[15]，BERT（Bidirectional Encoder Representations from Transformers）[16]等模型的研究给轨迹的深度表达学习提供了新的思路。因此本文的工作主要是研究基于深度学习模型的轨迹表达算法，设计了基于 BERT 模型的轨迹表达学习方法，旨在提高轨迹表达的准确性，并利用轨迹在向量空间的表达进行轨迹聚类发现相似行为的个体，从而为城市交通规划和行为分析提供有效参考。通过在数据处理、模型设计等方面的措施提高了轨迹表达的准确性，对轨迹的关键特征进行降维表达，具有较好的轨迹特征提取能力。

1.2 国内外研究现状

本节对轨迹表达及相似度计算领域的研究现状进行阐述，其中对轨迹进行降维表达是进行相似度计算的基础。内容上本节首先介绍传统的轨迹降维表达、及相似度衡量方法；在此基础上阐述深度学习的发展为低维轨迹表达的获取提供的新的发展方向；最后，通过自然语言处理领域内表达学习算法、及注意力机制研究情况的分析，为本文提出的基于多头注意力机制的轨迹深度表达学习算法奠定基础。

（1）传统轨迹表达及相似度衡量方法

在传感器及无线通讯设备快速发展、轨迹数据的产生和获取越来越便捷的情况下，海量轨迹数据与有限的计算机存储、计算能力之间存在矛盾，轨迹的降维表示有着明确的现实必要性。传统的轨迹降维方法包括均匀采样法、Douglas-Prucker算法、主成分分析法[17-19]等。其中对于均匀采样法，研究者对这种方法进行了改进，使用了基于加权的K-SVD的非均匀的方法[20]达到数据采样的目的，改变了均匀采样的方式而通过计算权重来得到采样结果。Douglas-Peucker算法是一种在曲线上进行采样、得到折线数据从而实现降维的数据处理方法，它需要给定一个距离阈值，小于该值则接受当前的直线作为曲线段的近似。之后针对Douglas-Peucker算法的改进算法[21]被提出，这种方法对轨迹数据进行了压缩，在阈值的确定上更为科学合理。此外也有研究者利用主成分分析和傅里叶变换结合的方法[22]进行降维，首先用主成分分析的方法提取特征，再使用傅里叶变换减少信号滞后问题带来的影响。但上述降维方法的缺点在于降维过程中数据被不同程度损失，给轨迹表达的准确性带来直接且较大的不利影响，因此我们需要探究更为合理的轨迹表达方法。

在得到轨迹的降维表达后，我们需要对轨迹间的相似性进行衡量。在上世纪90年代，R. Agrawal和C. Faloutsos提出采用欧几里得距离比较两条轨迹序列[23-24]，类似地，H. Ding提出比较两个轨迹序列的第i个点的方法[25]（也被称为lock-step方法），但由于上述方法都基于轨迹点个数相同的假设，在实际中应用受到限制。

作为解决上述问题的方法，D. J. Berndt利用DTW使得时间序列能被扩展或压缩进行相互匹配[26]，同时允许一个点对应多个点，为不同长度的轨迹相似性比较提供了解决方法。DTW（Dynamic Time Warping）是动态时间规整算法，用动态规划的方法寻找两条序列的最短距离匹配。通过将一条轨迹中的每个点寻求与另一条轨迹的对齐，计算对齐点对之间的距离并累计，当两条轨迹的最后一个点都被匹配完成时获得的距离即为两轨迹的相似度。由此我们可以知道距离越近则相似度越高，在进行距离计算时一般采用欧氏距离。从上述方法可以看出虽然DTW解决了不同长度间轨迹相似度计算的问题，但是它对于每个轨迹点必须寻求匹配的要求使得其对个别点的差异性十分敏感，也就是无法在样本有噪声或者总体相似的情况下局部不相似的场景中依然保持良好的性能。同样对轨迹的空间距离进行比较的方法还有通过对轨迹形状进行轨迹相似计算的Hausdorff距离[27]和Fréchet距离[28,29]，对于前者，给定轨迹A和B，对于A中的每个点，该方法计算该点到B中最近点的距离，再在所有得到的距离值中取最大值，这样以A和B两条轨迹中距离最远的点作为衡量轨迹间相似度的标准。后者与这种方法的区别在于Fréchet距离把轨迹中点出现的顺序考虑在内。总体上可以看出，上述方法对于局部差异较为敏感，如果轨迹存在完全不相似区段的情况无法分辨是真的不相似还是样本噪声。

作为上述方法的改进，研究者们提出了基于编辑距离的一系列方法，如EDR[12]和ERP [11]等，这种方法对于两条轨迹大部分相似、局部有差异的情况具有较好的包容性。编辑距离是一种来源于文本处理的概念，它指的是将一个文本序列通过增、删、改从而变成另一个序列所需的最小操作数，该方法用于计算轨迹相似度的场景中就是将一条轨迹转换为另一条所需的编辑操作个数（如插入删除或替换）作为两轨迹之间的距离。同样基于编辑距离的空间相似性计算方法还有EDwP[30]，采用替换和插入操作计算轨迹相似性，可以应对轨迹采样率不同和异步采样的情况。然而不管是DTW还是基于编辑距离的方法都存在以下两个共同缺陷：1）只考虑了空间位置上的距离而推断相似性、缺乏时间特征，这与轨迹本身是时空数据的特点不匹配；2）除直接使用欧几里得距离计算相似性的时间复杂度为O(n)外，其余的相似度计算方法的时间复杂度都为$O(n^2)$，耗费的计算成本较高。

作为对DTW没能考虑时间信息的改进，研究者对DTW规整的时间窗口大小进行了限制[31]，同时可以提升计算的速度和准确性，因为避免了为达到匹配而导致的轨迹过度扩张或压缩，但仍未对轨迹的时间特征进行足够的挖掘。另外一种同时对时空距离的比较方法是最长公共子串(LCSS)，它也是基于编辑距离对轨迹的公共部分进行发现，通过分别设定来自两条轨迹的样本点间时间和空间距离阈值，小于该值则认为匹配，两条轨迹的相似度即为能够匹配的点的个数。而这种方法在应对低采样频率和异质采样频率时性能较差，无法判别两条反映同样路线、却有不同采样频率的相似轨迹。除利用轨迹点对以外，研究者们也提出了基于轨迹聚类进行相似轨迹发现的方法[32-34]。

可以看出，上述对轨迹进行相似距离衡量的方法大多只针对轨迹特征进行人为发现或构造进而进行比较，难免出现不够全面或准确的情况，而对获取到的轨迹进行时间、空间上的特征学习，即进行合理降维表达，则能更为直接和根本地提升轨迹相似发现的准确性和健壮性。

（2）深度学习方法

随着深度学习的发展，部分研究者开始利用深度学习模型对轨迹的潜在路线进行推断和表示，提出了t2vec[13]，对于低采样频率、不均匀采样频率和样本噪声表现出较强的健壮性。在具体模型设计上，作者先将区域进行划分得到每个轨迹点对应的索引，再使用前馈神经网络得到轨迹点的嵌入表达，输入经过下采样、加噪后的轨迹序列经过基于Seq2Seq结构的编码器-解码器结构学习完整轨迹的向量表达，利用这种得到的轨迹表达作者设计了三种衡量轨迹相似度的实验方法，结果显示相比于传统的LCSS, EDwP等方法，这种方法学到的轨迹表达具有良好的特征提取能力和表达能力。此外还有基于自编码器结构的轨迹聚类模型[35]，作者利用LSTM组成自编码器的基本单元对轨迹的序列特征进行学习。可以看出目前基于深

度表示学习的轨迹表达算法基本都利用了Seq2Seq结构将原始轨迹映射为低维向量空间的表达，在取得较好表现的同时我们也可以发现更多的自然语言处理领域的表征模型可以与轨迹表达任务相结合，实现准确的轨迹特征提取。

在此基础上其他学者对相似轨迹发现的速度也进行了研究。由于利用大量数据进行深度学习训练所需的计算时间成本较高，Yao等研究者提出了能够实现较为准确而快速进行轨迹相似度计算的模型NEUTRAJ[36]，这基于两个模块的设计，一是在RNN的基础上引入空间注意力机制，二是设计了根据距离进行权重分配的损失函数。这也给我们的研究提供了借鉴意义。此外还有学者在计算轨迹相似度时考虑将时间信息、空间信息分别进行嵌入表征[14]，此外加入了轨迹的语义信息。在具体结构方面，作者首先使用skip-gram方法分别对时间、空间位置、轨迹点邻近的兴趣点语义信息进行嵌入[37]，将得到的三个嵌入表达进行拼接，得到最终的轨迹嵌入表示，再送入基于Seq2Seq的编码器-解码器结构学习各维特征，输入是原始轨迹的下采样序列而输出的是原始轨迹，通过不断的迭代训练深度学习模型能够捕捉到轨迹的时间和空间及语义特征，在进行轨迹相似度计算时也有更好的表现。

可以看出目前在利用深度学习进行轨迹表达的领域，大部分都采用循环神经网络对轨迹序列进行学习，而循环神经网络在提取特征的过程中存在以下问题：一是无法解决数据的长距离依赖问题；二是受数据输入的顺序限制、只能学习历史数据间的特征；三是无法并行计算。基于上述问题，使用注意力机制的模型逐渐进入人们的视野。

（3）基于注意力机制的表达学习算法

注意力机制可以通过学习获得数据不同部分的权重，也即数据的相互关系，而不受数据顺序的影响，有助于模型能够更准确地捕捉数据特征。其最早被谷歌团队应用在图像分类任务上、完成了物体的较为准确识别[38]，使得结合注意力机制的神经网络被广泛关注。之后Bahdanau等首先将注意力机制引入机器翻译领域[39]，这是注意力机制在自然语言处理领域的首次应用。随后在此基础上，其在文本理解、翻译等任务上取得了较大的进展，2017年谷歌机器翻译团队提出包含多头注意力机制的Transformer[15]，摒弃了传统依靠复杂的RNN或CNN搭建的编码器-解码器结构，在WMT2014英文-德语翻译任务中超过最先进的模型2分，使得注意力机制成为自然语言处理领域的有力工具。2018年，从事自然语言处理相关研究的学者开始探索并使用预训练与微调的结合的模型完成文本处理任务，首先在大规模语料库上执行无监督预训练任务，再通过微调、有监督地进行特定的下游任务训练，其中预训练任务的设置能够准确提取输入数据的特征，且泛化能力强，其中具有代表性的模型包括BERT[16]，它利用Transformer中的自编码器、采用多头注意力机制进行数据的深度表征，取得了比以往模型都更好的效果。

而目前在轨迹深度表达领域，大多数模型都采用的是基于RNN结构的编码器-解码器模型，受BERT模型强大特征提取能力的启发，我们希望能够将这种基于多头注意力机制的表达模型用在轨迹数据的降维表达问题上，在轨迹采样频率不一致、采样率低、轨迹噪声等问题的挑战下获取较为准确的轨迹向量表征。

1.3 研究内容及主要贡献

根据上述研究现状的分析我们可以看出，现有的轨迹深度表达模型主要使用循环神经网络（RNN）提取轨迹的序列特征，搭建编码器-解码器结构获得轨迹的向量表征。但这种模型对数据输入的顺序依赖性较强，没办法捕捉远距离数据的信息，视野有限，也无法并行计算。因此，我们考虑在轨迹表达算法中引入多头注意力机制，获取多尺度的轨迹特征，并通过模型训练提高算法对轨迹采样率不均匀、轨迹噪声干扰等问题的健壮性。

本文主要研究基于深度学习的轨迹表达算法及相似轨迹发现问题，通过将Transformer[15]、BERT[16]等包含注意力机制的模型应用在轨迹表达任务上，获取具有健壮性的轨迹深度表达模型。目的在于设计能够将车辆轨迹进行准确降维表达的深度学习方法，从而为轨迹聚类、城市热门路线挖掘、移动群体发现等应用奠定基础。

本文主要研究工作包括以下 3 个方面。

（1）将自然语言处理领域语言表征模型与轨迹表达问题进行结合。轨迹数据通常以包含时间信息的经纬度点对的形式存在，我们需要将其转化为类似于文本中词汇的形式送入模型进行训练和评估。如何完成交通轨迹数据与自然语言处理问题中的文本数据之间的类比映射是我们需要考虑的问题。

（2）实现具有健壮性的轨迹表达算法。在真实的轨迹数据中常出现采样频率低、相似轨迹的采样频率不一致、轨迹包含噪声等情况，这给轨迹相似度的计算带来了巨大的挑战，我们需要在模型中针对这类问题对数据集进行处理，利用经下采样、加噪后的数据对模型进行训练，获得具有健壮性的准确轨迹表达，从而为相似轨迹的发现奠定基础。

（3）模型评估方法的设计。我们获取的轨迹向量表示用于相似轨迹发现任务，但由于轨迹是否相似缺乏公认的标准，因此在评估轨迹表达的准确度时，我们需要设计单独的实验进行评估。

本文针对上述研究内容开展算法研究和实验验证，主要贡献如下。

（1）将交通轨迹数据通过对轨迹落点区域的编码转换为能够进行表达学习的词汇数据形式，并使用注意力机制对轨迹序列进行特征提取，代替了以往使用循环

神经网络的深度学习模型，从而轨迹表达模型能够突破数据输入顺序对特征提取的限制，并且可以并行计算。

（2）提出了具有健壮性的轨迹深度表达模型 Transformer-traj 和 BERT-traj，将经过下采样和加噪声处理后的数据送入模型进行训练，模型能在轨迹采样频率不均匀、有噪声干扰的情况下获得较为准确的轨迹向量表达，并进一步为轨迹相似发现任务的完成提供有力的支持。

（3）利用真实数据集，在现有研究[47-48]的基础上实施并改进了轨迹表达模型的评估方法，采用自身相似度、交叉相似度、轨迹 KNN 查询三种方法对模型的准确度进行评估。实验结果显示，在小型数据集上加噪声和下采样干扰的情况下，BERT-traj 相比基线模型 t2v 在这个指标上分别高出 23.26%和 14.06%。在大型数据集上，BERT-traj 相比基线模型 t2v 分别高出 23.71%和 22.42%。

1.4 论文组织结构

本文的工作主要分为六章，内容安排如下。

第一章为引言，首先阐述了本文的研究背景及研究意义，对轨迹表达算法研究的必要性和应用领域进行介绍，再对目前的研究现状及需要改进的方向进行阐述，主要包括传统的轨迹降维与相似度计算方法，现有的基于深度学习的轨迹表达算法，以及表达学习方法与轨迹特征学习的结合。再对本文研究内容进行具体介绍，最后阐明论文的组织结构。

第二章为相关理论基础。主要介绍在本文涉及的研究过程中运用的理论知识，在介绍轨迹基本概念的基础上阐述基线模型中用到的循环神经网络及编码器-解码器结构，接着介绍了基于注意力机制的 Transformer 和 BERT 模型，最后是欧几里得距离和余弦距离的计算方法。后续的实验需要利用上述内容作为基础，在实践中有所运用，在最后对本章内容进行总结。

第三章对实验环境进行介绍并对数据集进行预处理，首先介绍了在研究工作中用到的开发平台和集成框架，在此基础上介绍了如何将轨迹数据转换为表达学习模型需要的形式，包括数据集特征分析、数据清洗、编码得到轨迹序列等过程。最后对本章进行总结。

第四章介绍了本文提出的轨迹表达方法 Transformer-traj 和 BERT-traj，首先介绍了两种模型的内部结构，包括多头注意力层、前馈神经网络层等，再对模型的训练方法进行了介绍，包括损失函数的设定，输入输出的处理，以及模型参数的选择。最后对本章内容进行总结。

第五章是对模型的评估实验与结果分析。利用自身相似轨迹距离、交叉轨迹相

似距离、KNN 相似轨迹查询等方法对评估轨迹表达准确度的方法进行介绍，并展示实验结果。接着对 BERT-traj 的表达能力进行验证，将本文提出方法与已有方法在轨迹聚类任务上可视化，通过将轨迹向量降维到二维空间、再使用 K-Means 方法进行聚类，验证了本文提出的方法能够获得更加准确的轨迹表达，类内间距小、类间间距大，聚类效果较为明显。此外对轨迹表达算法的实践应用场景进行分析。最后对本章内容进行总结。

第六章对本文的工作内容进行总结，阐述主要结论及下一步研究方向。主要列举了本文解决的挑战、做出的贡献，以及有待完善的地方，为下一步的研究提供思路。

2 论文相关知识

本章介绍轨迹深度表达学习及相似度计算的相关理论基础。现有的研究中将轨迹作为时序数据利用循环神经网络等深度学习模型进行降维表达，本文提出的轨迹表达算法中借鉴了自然语言处理领域中文本表示学习方法等相关思想，并以注意力机制代替循环神经网络学习轨迹特征。因此本章首先介绍循环神经网络模型，接着介绍基于注意力机制的轨迹表达学习模型，包括 Transformer[15]模型以及 BERT[16]模型。之后对轨迹相似性衡量方法进行介绍，最后对本章内容进行总结。

2.1 轨迹的基本概念

轨迹数据是当前大数据研究应用中重要的信息资源，通过无线信号或传感器等设备采样得到离散的轨迹点，这些轨迹点反映了移动物体的移动路线。而包含汽车行驶时间和位置信息的车辆轨迹即为轨迹数据中的一种，在获取到的轨迹数据中车辆的轨迹点按照时间顺序排列，因而轨迹数据也是时序数据的一种。轨迹的定义如下。

定义 2.1 轨迹：一条轨迹数据 $Traj=\{p_1,p_2,.....,p_n\}$，是从移动物体的路线采样得到的一串样本点，该样本点是二维时序数据，信息中包含了采样位置和采样时间，一条典型的轨迹如图 2-1 所示，可将其描述为

$$Traj=\{(l_1,t_1),(l_2,t_2),...,(l_i,t_i),...,(l_n,t_n)\} \tag{2-1}$$

其中 $l_i=(l_{ix},l_{iy})$ 表示在时刻 i 测量到的位置信息。

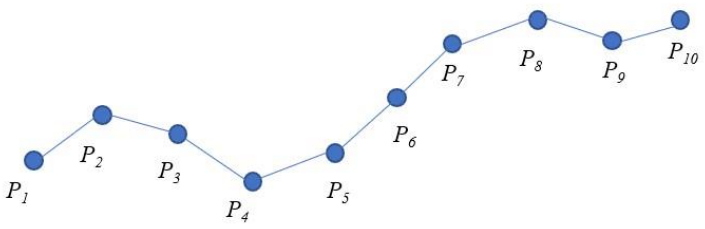

图 2-1 一条典型轨迹

Figure 2-1 An example of trajectory

2.2 深度学习算法

深度学习是机器学习领域的一个研究方向，最早由多伦多大学的 G.E.Hinton 于 2006 年提出[40-41]，近年来在诸如语音识别、计算机视觉多个应用领域有了获得了较为迅猛的发展。它的研究源自于建立模型模拟人类大脑的神经连接结构，当模型在处理图像、文本、声音等信号时通过多个变换阶段分层对数据特征进行描述[32, 33]，从而得到对数据的解释。

深度学习的“深度”是相对于支持向量机（support vector machine）、提升方法（boosting）等较为“浅层”的学习方法而言。浅层学习方法需要人工的经验获取样本特征，习得的是单层特征，而深度学习将样本在多个层之间变换、得到不同的特征表示，自动地学到不同层次的特征，更有助于复杂分类任务的完成。在每层神经网络中包含大量神经元，不同神经元之间相互连接，且获取信息的权重会随着训练不断调整更新，深度学习网络因而能够提取输入样本间的特征信息。经这样多层训练得到的深层网络与神经网络的工作机制类似，也即深度神经网络（Deep Neural Network, DNN）。

目前基于深度学习的轨迹表达方法大多将轨迹数据送入由循环神经网络及其变体组成的编码器-解码器模型进行训练，提取轨迹数据的时间特征。下面对该类方法设计到的模型进行基础知识介绍。

2.2.1 循环神经网络

循环神经网络（Recurrent Neural Network, RNN）被广泛用于处理时序数据相关问题，比如机器翻译。它刻画了一个序列当前的输出和之前信息的关系。在网络结构上，循环神经网络会记忆之前的信息，并利用之前的信息影响后面节点的输出。它可以被用于处理时序数据，提取数据的时间特征。

在传统的神经网络模型中，输入层、隐藏层和输出层之间全连接，而不同时序的网络间无连接，模型无法获得输入的数据之间的关系，这在处理序列数据相关问题（比如预测）时无法具有良好的表现，因为在对未来数据进行预测时需要利用历史信息，它们之间并不独立而是有一定联系的。

而 RNN 会对之前输入的信息进行记忆并用于当前的输出，即不同时间步的神经元之间不再无连接、而变成有连接的状态，当前时刻隐藏层的输入不仅包括输入层的数据、还包括上一时刻隐藏层的输出。基本的 RNN 是由神经元组成的多层神经网络，其隐藏层结构如图 2-2 所示。

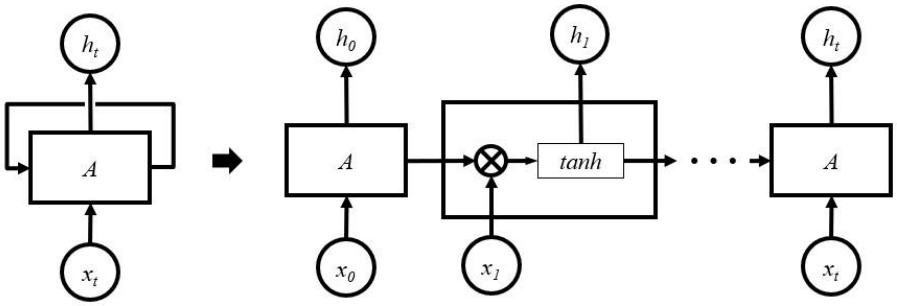

图 2-2 RNN 隐藏层结构示意图

Figure 2-2 An illustration of the hidden layer of RNN

其中 A 代表神经元，x_i 表示隐藏层的输入，h_i 是 RNN 的隐藏状态。图 2-2 中右边部分是左边部分的展开图。其中隐藏层中每个神经元都与下一时间步的神经元相连。隐藏层计算过程可以表示为

$$h_i = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2-2}$$

其中 h_t 表示 t 时刻 RNN 的隐藏状态，x_t 表示 t 时刻隐藏层的输入，W_{xh} 和 W_{hh} 分别是输入层和隐藏层之间、隐藏层之间的权重矩阵，由训练学习得到，b_n 是偏置。

理论上 RNN 可以处理任何长度的时序数据，但随着数据长度的增加梯度消失和梯度爆炸的问题，而长短期记忆网络（Long Short-Term Memory, LSTM）[42]可以很好地弥补这个问题。LSTM 将图 2-2 中隐藏层的 RNN 细胞替换为 LSTM 细胞，LSTM 细胞结构如图 2-3 所示。通过利用输入门、遗忘门、输出门分别控制哪些信息应被输入、哪些应被遗忘、以及哪些应被输出。它针对任务学习历史信息中应被保留部分的权重，具有长期记忆能力。

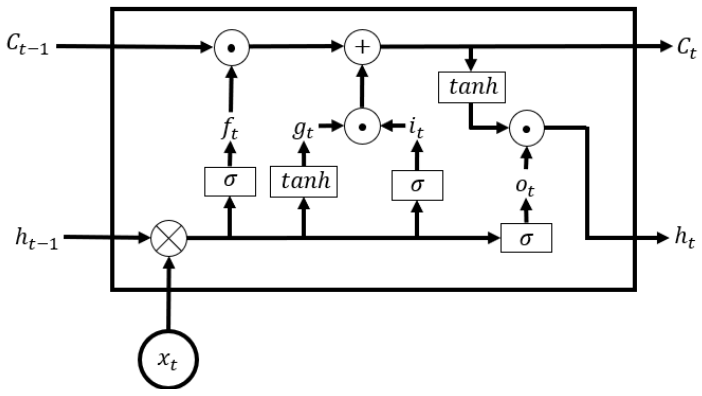

图 2-3 LSTM 细胞结构示意图

Figure 2-3 An illustration of the memory cell in LSTM

其前向计算方法可以表示为

$$f_t = \sigma(W_f \,|\, h_{t-1}, x_t \,|\, + b_f) \tag{2-3}$$

$$i_t = \sigma(W_i \,|\, h_{t-1}, x_t \,|\, + b_i) \tag{2-4}$$

$$g_t = \tanh(W_g \,|\, h_{t-1}, x_t \,|\, + b_g) \tag{2-5}$$

$$o_t = \sigma(W_o \,|\, h_{t-1}, x_t \,|\, + b_o) \tag{2-6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2-7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2-8}$$

式中 f、i、o、c 分别是遗忘门、输入门、输出门和细胞状态；W 和 b 分别是对应的权重系数矩阵和偏置；σ 和 tanh 分别是 sigmoid 和双曲正切激活函数。在计算过程中 LSTM 首先按照式（2-3）~（2-8）计算细胞的输出值，再反向计算得到每个细胞的误差，依据此计算每个权重矩阵的梯度，最后用基于梯度的优化算法更新权重矩阵。

在此基础上，LSTM 模型演化了一些变体，其中包括门限循环单元（Gated Recurrent Unit, GRU）[43]，其细胞结构如图 2-4 所示。相比 LSTM，GRU 有所简化但仍具有 LSTM 处理长序列数据的记忆能力，具体方法是将 LSTM 中的输入门、遗忘门、输出门改为更新门和重置门，将细胞状态和输出向量合在一起。

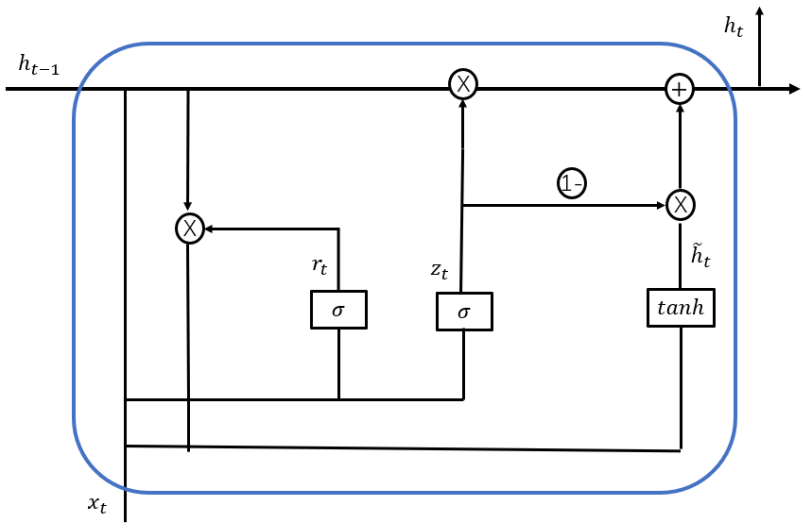

图 2-4 GRU 细胞结构示意图

Figure 2-4 An illustration of the cell structure of GRU

GRU 的前向计算过程可以表示为

$$z_t = \sigma(W_z \,|\, h_{t-1}, x_t \,|) \tag{2-9}$$

$$r_t = \sigma(W_r \,|\, h_{t-1}, x_t \,|) \tag{2-10}$$

$$\tilde{h} = \tanh(W \,|\, h_{t-1}, x_t) \tag{2-11}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{2-12}$$

其中 z_t 和 r_t 分别表示更新门和重置门，在 GRU 中使用一个门控 z_t 可以同时完成遗忘和选择记忆，相比 LSTM 减少了一个门但能够达到与 LSTM 相似的作用。在论文[13]中作者使用这种结构搭建了基于 GRU 的编码器-解码器结构进行轨迹表达。

2.2.2 编码器-解码器结构

为了提高深度学习网络对特征的表达能力，研究者们构建了由多层 RNN[13]、CNN 等模型组合形成的深层网络，称为编码器-解码器结构，如图 2-5 所示。这种网络可以完成输入序列和输出序列长度不相等、或数据是不定长序列等情况下的深度学习任务。编码器和解码器都由循环神经网络或卷积神经网络组成，编码器用来分析输入序列，解码器用来生成输出序列。

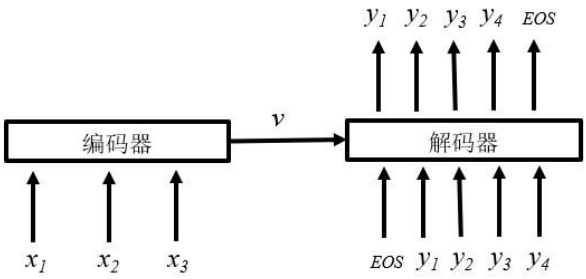

图 2-5 编码器-解码器结构示意图

Figure 2-5 An illustration of encoder-decoder

编码器对输入序列 $\{x_1, x_2, ..., x_T\}$ 进行特征提取，并将最后时刻 T 产生的隐藏层状态值作为序列的编码值，它包含了从时刻 1 到时刻 T 输入序列的所有信息，这里表示为 v，是一个固定长度的向量。作为解码器的网络每个时刻的输入值为 v 和 y_t，得到目标序列的条件概率。在测试阶段，解码器第一个时刻接收到的是序列开始的信号，解码器输出第一个预测值，再将输出的预测值和 v 作为第二次预测的输入，不断循环直到得到序列结束的信号。

在进行轨迹表达时，我们可以利用编码器-解码器结构对轨迹序列进行特征提取，通过训练模型对轨迹进行预测得到轨迹的向量表示，也即编码器、解码器中间的定长向量，它包含了整条轨迹的信息，相比于高维、稀疏的原始轨迹，经过深度学习模型表达后的轨迹向量更便于进行相似轨迹的发现。

2.3 基于注意力机制的深度表达学习算法

随着以深度学习为代表的表示学习模型的发展和研究，诸如 RNN 和 CNN 等模型被引入自然语言处理领域[44-45]，尤其是基于上述模型的编码器-解码器网络获得较为广泛的应用。但上述模型在应用过程中也存在一定问题，一是在处理较长序列时记忆能力有限，二是受顺序限制，模型只能按照顺序利用历史输入的信息预测输出。而注意力机制的引入[46]可以较好地解决上述问题，在不同时刻模型可以关注不同位置的信息。在此基础上，研究者们提出了 Transformer[15]和 BERT[16]模型在不同的表达学习任务上都有了较好的表现。

2.3.1 Transformer 算法

与大多数的序列到序列学习（Sequence to Sequence）模型类似，Transformer 由编码器和解码器构成，其模型结构如图 2-6 所示，其中 N 表示模块数量。但是与之前的模型不同的是 Transformer 只利用了注意力机制对数据进行处理，没有 RNN 或 CNN 的参与，且可以实现并行处理。

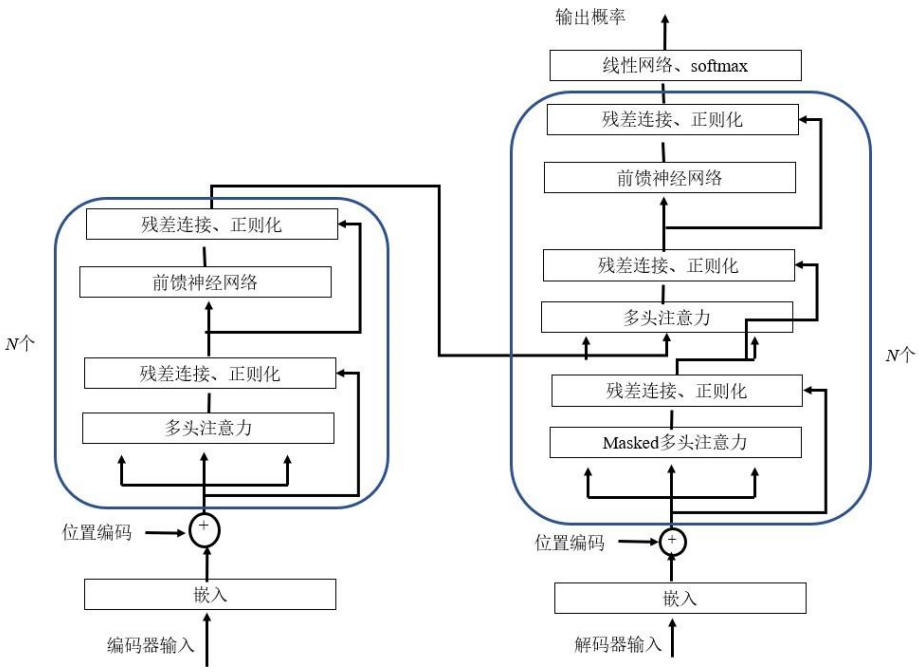

图 2-6 Transformer 结构示意图

Figure 2-6 An illustration of the structure of Transformer

图 2-6 中左边部分是编码器，右边是解码器。编码器由多个相同的神经层组成，也即边框内的部分，每个神经层包括两个子神经层，分别是多头注意力机制和全连接前馈网络，并且每个子神经层都添加了残差连接和正则化。因此设神经层的输入为 x，则每个子神经层的输出为

$$Sublayer\_output = LayerNorm(x + (Sublayer(x))) \tag{2-13}$$

注意力机制的核心思想为，许多个键值对组成了一个数据集 *Source*，此时对于给定的查询数据 *Query*，计算其与各个键 *Key* 的相似程度或相关性，由此作为与各个 *Key* 对应的 *Value* 的权重，再对所有的 *Value* 值加权求和得到最后的注意力值。可以看出注意力机制打破了之前神经网络中数据输入的顺序限制，视野扩大。将上述过程表示为数学表达式为

$$Attention(Query, Source) = \sum_{i=1}^{L_x} similarity(Query, Key_i) * Value_i \tag{2-14}$$

在 Bahdanau 等引入的注意力机制[39]中，模型结构如图 2-7 所示。

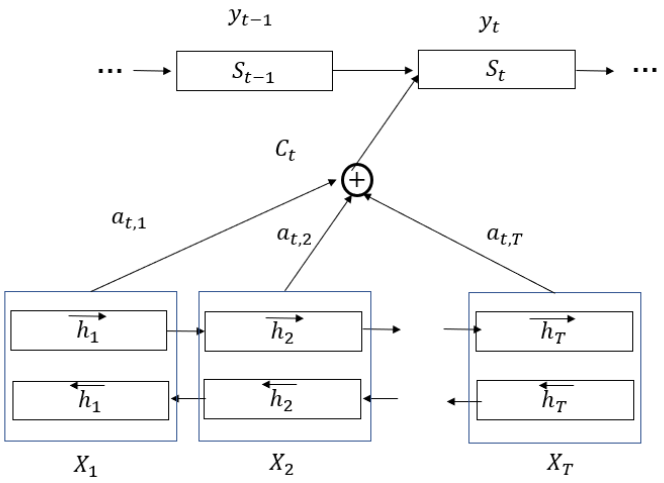

图 2-7 注意力机制结构示意图

Figure 2-7 An illustration of the attention mechanism

其中 S_{t-1} 是编码器在 $t-1$ 时刻的隐状态，y_t 是目标词，C_t 是上下文向量，h_j 表示编码器端第 j 个词的隐向量，$a_{t,j}$ 表示编码器段第 j 个词对解码器端第 t 个词的注意力权值，也即式（2-14）中的 *similarity* 值。由图 2-7 可知

$$S_t = f(S_{t-1}, y_{t-1}, C_t) \tag{2-15}$$

$$C_t = \sum_{j=1}^{T} a_{t,j} h_j \tag{2-16}$$

如果只关注自身内部的数据间特征，也即自注意力机制的情况，注意力的计算公式为

$$A=\text{softmax}(v_a \tanh(W_a H^T)) \tag{2-17}$$

其中W_a是训练得到的权重矩阵，v_a是参数向量，决定了注意力机制的视野大小，H是输入序列的隐向量矩阵，可以表示为

$$H=(h_1,h_2,...,h_T) \tag{2-18}$$

Transformer[15]模型中使用的是多头注意力机制，其结构示意图如图 2-8[15]所示。具体计算过程是通过不同的线性变换对输入的 *Key*、*Value*、*Query* 进行投影，再将不同变换的结果拼接起来。变换的个数就是头数，是我们可以设置的超参数h，这意味着同时有多个注意力机制关注数据中不同尺度的信息。其计算过程为

$$Multihead(Q,K,V)=Concat(head_1,...,head_h)W^o \tag{2-19}$$

$$head_i=Attention(QW_i^Q,KW_i^K,VW_i^V) \tag{2-20}$$

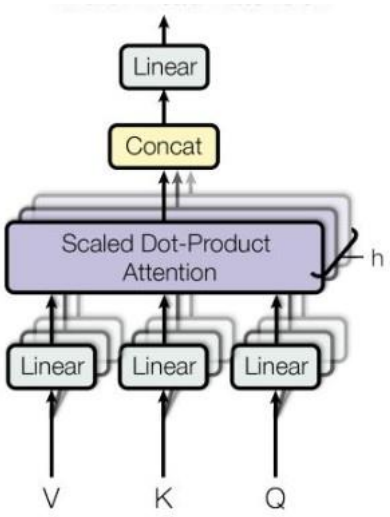

图 2-8 多头注意力机制示意图[15]

Figure 2-8 An illustration of multi-head attention mechanism[15]

值得关注的是，与式（2-17）中计算注意力的方法不同，在 Transformer 模型的设计中，计算注意力的方法如图 2-9[15]所示，模型通过参数d_k对计算结果的大小进行调整，经掩码和 softmax 层后再与V进行矩阵乘法得到最后的注意力值，该方法称为缩放点积。反映在数学公式上，其计算过程如式（2-21）所示。

$$Attention(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2-21}$$

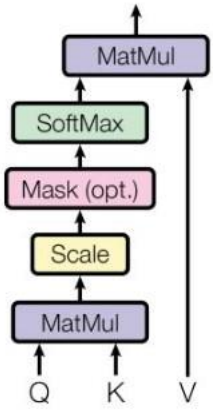

图 2-9 缩放点积计算示意图[15]
Figure 2-9 Calculation process of scaled dot-product[15]

Transformer[15]在编码器和解码器部分都使用了注意力机制，编码器部分的注意力计算模块是多头自注意力机制，即在数据内部进行注意力的计算，公式中的Q、K、V是相同的数据。解码器部分的 Masked 多头注意力层的计算中，K、V来自编码器，Q是上一时刻解码器的输出。

2.3.2 BERT 算法

BERT（Bidirectional Encoder Representations from Transformer）[16]是在 Transformer 基础上发展的预训练模型，其结构如图 2-10 所示，图中*Trm*表示 Transformer 编码器。该模型的预训练任务包括遮盖词语预测（完型填空）和下一句预测，两个任务分别关注相同、不同句子之间关系。

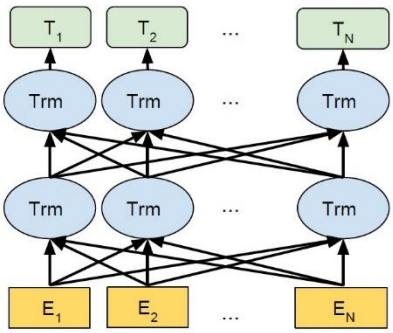

图 2-10 BERT 结构示意图[16]
Figure 2-10 Architecture of BERT[16]

在对输入数据进行嵌入时，BERT 采用如图 2-11 的结构。首先对句子中的词进行嵌入，获得单个词的向量表征；再将每个句子分为前后两部分，用 *SEP* 分隔符作为界限， *SEP* 之前的部分的词向量添加前半段标记， *SEP* 之后的词向量添加后半段标记；最后由于采用注意力机制的模型缺少数据的顺序信息，使用位置嵌入对数据的前后顺序进行嵌入表达。

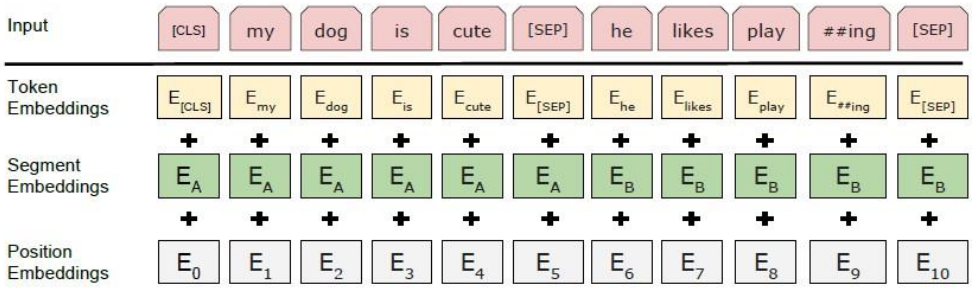

图 2-11 BERT 的输入向量结构[16]

Figure 2-11 The input representation of BERT[16]

如图 2-11 所示，第一行是对句子中每个词的嵌入，第二行是在把句子分为前后两个部分的基础上给前半段的词一个嵌入标记 *A*，后半段一个嵌入标记 *B*；第三行是对句子中每个词按照先后顺序进行嵌入标记。最终输入数据是每个词的上述三类嵌入向量的和。

而在特征提取阶段 BERT 使用双向的 Transformer 编码器，包含多头注意力机制和前馈神经网络。在多头注意力机制部分，首先将输入的嵌入向量进行维度扩展，得到查询向量 *Query* 、键 *Key* 、值 *Value* ； 再根据头的数量对数据进行划分，对划分后的词与其他词计算自注意力，得到新的向量表示；再将不同头划分后得到的向量拼接得到最终每个词的表示，经过 dropout 防止梯度爆炸、梯度消失，使用层正则化对数据的范围进行归一化。在前馈神经网络部分，输入数据经过线性变换后再通过 dropout、残差连接和层正则化最终得到输出向量。

2.4 轨迹相似度计算

两条轨迹间的相似性衡量是通过使用不同方法计算轨迹向量间距离判断相似程度，距离越小则表明轨迹间相似程度越高。由于在经过深度学习方法进行特征表达后的轨迹向量已经为低维、固定长度的向量表示，常见的较为简单的距离计算方

法就可以用于轨迹间相似性的度量。主要包括欧几里得距离、余弦距离等，下面详细介绍轨迹间距离度量方法。

2.4.1 欧几里得距离

欧几里得距离是一种常见的衡量对象间相似程度的方法，该方法要求被比较的对象具有相同的维度，对于空间中任意两个向量 $x=(x_1,x_2,...,x_n)$ 和 $y=(y_1,y_2,...,y_n)$，它们之间的欧几里得距离计算公式为

$$dist(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{2-22}$$

若是针对原始的轨迹间距离计算则先要计算每个点对间的欧几里得距离、再求和得到两条轨迹的距离。

欧几里得距离较为简洁，便于使用，但原始轨迹数据常出现采样率不一致、导致轨迹点数不同，进而无法使用欧几里得距离计算。且欧几里得距离对于噪声十分敏感，若轨迹包含噪声干扰，相似度的衡量也会出现偏差，影响准确率。

而本文提出的基于注意力机制的轨迹表达算法能够在加噪、下采样的情况下对轨迹进行较为准确的表达，得到固定长度、低维的轨迹向量表示，从而使用较为简单的欧几里得算法就可以得到轨迹间的距离。

2.4.2 余弦距离

余弦距离的计算建立在余弦相似度的计算基础上，对于欧氏空间的两个向量 $x=(x_1,x_2,...,x_n)$ 和 $y=(y_1,y_2,...,y_n)$，它们的余弦相似度的计算公式为

$$\cos(x,y)=\frac{\sum_{k=1}^{n}x_k y_k}{\sqrt{\sum_{k=1}^{n}x_k{}^2}\sqrt{\sum_{k=1}^{n}y_k{}^2}}=\frac{\sum_{k=1}^{n}x_k y_k}{\|x\|\|y\|} \tag{2-23}$$

余弦相似度的取值范围为-1 到 1，衡量的是两个向量间方向的相似度，两个向量的夹角越小、越趋近于 0，则余弦相似度越趋近于 1，表明方向较为一致。

与欧几里得距离相比，余弦距离关注的更多是方向上的相似程度，与向量的大小无关；而欧几里得距离体现的是向量大小的差异。可以结合不同情况下的需求使用这两种轨迹相似度衡量方法来对深度学习表达后的向量进行比较。

在本文的实验和研究工作中主要使用余弦距离作为衡量轨迹相似性的标准，包括在模型评估阶段的实验以及利用模型进行轨迹向量聚类的实验。

2.5 本章小结

本章首先介绍了轨迹的基本概念，在此基础上对利用深度学习方法进行轨迹表达的方法进行阐述，包括传统的深度学习方法以及引入注意力机制的深度学习方法。由于经过深度学习模型进行特征提取后的轨迹向量具有固定长度和低维特征，运用较为常见、简单的距离衡量方法就可以得到轨迹的相似度，接下来文章对欧几里得距离和余弦距离的计算进行阐述，最后是本章小结。

在传统的深度学习方法中，主要介绍了适用于处理序列数据的循环神经网络的结构与原理，介绍了基本的 RNN 结构及在此基础上发展的 LSTM 和 GRU 模型，之后介绍了广泛用于向量表达、序列预测等任务编码器-解码器结构。接下来对于引入多头注意力机制的 Transformer、BERT 模型进行介绍，相比 RNN 等已有深度学习模型，后者打破了数据的顺序限制，视野更宽，深层的网络也使得模型具有更强大的特征提取能力，且可以并行计算，此外利用位置编码可以保留位置信息，相比 RNN 等模型具有更好的表达能力。

轨迹的深度表达和距离计算，为后续的轨迹聚类、移动模式挖掘、城市热门区域发现、路径推荐等应用打下基础。

3 实验环境与数据预处理

本章主要阐述本文工作所需的实验环境，并详细介绍数据的预处理过程。首先介绍了在研究工作中用到的开发平台和集成框架，在此基础上介绍了如何将轨迹数据转换为表达学习模型需要的形式，包括数据集特征分析、数据清洗、编码得到轨迹序列等过程。最后对本章进行总结。

3.1 开发工具

3.1.1 Python 语言

本文的研究工作中使用的编程语言是 Python。Python 是近年来被广泛使用的编程语言之一，它是一种面向对象的解释型编程语言，它的语法较为简洁优雅、可读性强，需要注意的是它强制使用空白符作为语句缩进。在使用 Python 进行编程时，如果语句的缩进出现错误，那么会直接报错，并不会忽视空格。与 Java 或 C++ 相比，Python 中的语法可以帮助程序员用更少的步骤完成编程工作，同时能够把用其他语言（比如 C 或 C++）编写的模块较为轻松地联结起来，实现开发目标。

Python 在软件开发领域有广泛而多样的应用，比如游戏、Web 框架、原型设计、图形设计应用程序等，因而在使用范围上 Python 相比其他语言更广，具有更好的适应性。此外，Python 具有较多的支持库，能够帮助开发者借助 Python 获得更便捷的开发体验。

3.1.2 Anaconda 集成环境

Anaconda 是一个开源的可用于进行科学计算的集成环境，它支持 Python 语言实现数据科学、机器学习等领域的应用开发，可以进行大数据处理和预测分析工作，通过软件包管理系统 conda 简化软件包的安装和管理，在自带的终端输入 conda install 命令即可安装需要的库。此外 Anaconda 还可以通过创建和管理不同的虚拟环境，使得不同版本的语言共存，不同的虚拟环境的配置独立。

Anaconda 自带一些常用的用于数据处理的第三方软件包，包括 Numpy，Pandas，matplotlib，Scipy 等。此外在本文的研究工作中还用到了 pickle 和 haversine。它们的作用分别如下。

Numpy 主要用于大规模的矩阵和数组运算，同时包含了许多常见的数学函数库，帮助我们在高效进行矩阵和数组运算的基础上，方便快捷地完成所需的数据处理工作。且函数涉及的代码可读性很强，我们在编程时可以较快地掌握函数的使用。

Pandas 同样是用于进行数据分析的第三方软件包，能够给我们提供高性能、易于使用的数据分析工具。它可以将数据存储为 DataFrame 格式，较为便捷和高效地完成数据的读取、合并、函数运算，以较低的时间复杂度完成大规模数据的处理分析工作，是一个有力的数据科学软件包。

matplotlib 是一个用于进行数据可视化的软件包，在给定数据的基础上，使用诸如 matplotlib.pyplot.plot 等函数可以较为方便地将数据进行可视化，且可以设置横纵轴范围、横纵轴名称及图名、点和线的颜色粗细、图例的名称位置等等，而这也只是其强大可视化功能的一小部分。

Scipy 是一个可以完成插值积分等数学计算、信号处理、傅里叶变换等任务的软件包，可以与 Numpy 协同工作，完成 Numpy 矩阵计算等数据处理问题。Scipy 是由针对不同特定任务的子模块构成的，在本文的研究工作中主要利用了 scipy.spatial，它包含了用于空间计算的数据结构和算法，我们主要使用这个软件包计算向量间的欧几里得距离和余弦距离等。

pickle 模块在本文的研究工作中主要进行数据的保存和读取，它可以用二进制的形式将序列化后的数据保存在文件中，使用的函数为 pickle.dump，在进行文件的读取时，pickle 将保存好的数据进行反序列化再输出，使用的函数为 pickle.load。这个库可以帮助我们方便地将内存中的数据与本地数据进行传输，且所占用的计算资源较小。

haversine 在本文中主要用于经纬度和距离长度的转换，借助 haversine 函数，我们可以在输入两个地点的经纬度后，得到这两个位置之间的实际距离。是一个较为方便的距离转换工具。

本文的实验编写是在 jupyter notebook 中进行的，它是一种轻量级的基于网页的 Python 编写和运行工具，与其他工具相比，它更加灵活方便。

3.1.3 Pytorch 框架

Pytorch 是一个基于 Python、具有灵活性的深度学习开发框架。它的流程非常接近 Python 的数据科学计算库 Numpy，可以与 Python 数据科学栈顺利集成。除此之外，Pytorch 具有可以支持多 GPU，还可以自定义数据加载器，能够帮助我们方便快捷地搭建深度学习模型。

Pytorch 中的张量是一个重要的基础概念，它类似于 Numpy 的 ndarrays，同时

支持在 GPU 上使用。在定义张量的基础上 Pytorch 帮助我们执行各种矩阵运算，如相乘、转置等等。

在进行神经网络模型的搭建时，我们可以使用 torch.nn，它定义了一组模块，用于进行神经网络层的搭建、设定输入输出，从作用上来看 torch.nn 可以被认为是 Pytorch 的内核。

Pytorch 同样具有算法优化的模块 torch.optim 用于神经网络模型的优化，我们在进行神经网络模型搭建时可以不必自己撰写函数和功能。

基于 Pytorch 在与 Numpy 等数据科学栈的方便集成的特性、搭建神经网络框架时全面的功能，Pytorch 被越来越多的深度学习算法开发者使用。本文中我们也使用 Pytorch 框架进行轨迹表达算法的模型搭建。

3.2 数据预处理

数据预处理包括数据集的清洗筛选、轨迹区域编码、以及训练用数据集的生成。

首先通过对轨迹包含样本点的统计得出大部分轨迹的长度，筛选出合适长度、样本点在有效经纬度范围内、行程距离合理的有效数据。再通过对样本点所落区域的编码得到样本点的索引，转化为能够输入模型的轨迹点序列数据。

3.2.1 数据集描述

本文用到的数据集为真实的出租车 GPS 记录，收集了葡萄牙波尔图市内超过 19 个月共 170 多万条出租车出行记录，每条记录包含的信息如表 3-1 所示。出租车每间隔 15 秒汇报所在位置。数据集中包括了出租车行车的时间，轨迹点经纬度，出租车 ID，乘客上车方式等记录。

3.2.2 数据清洗

在进行数据的清洗筛选时，我们经过了以下步骤。

（1）对存在缺失值的轨迹的处理。在当前数据集中共包含轨迹 1710670 条，其中存在缺失的轨迹数为 10 条，相对全部数据来说占比较小，去除该数据对整体数据集的影响不大，因此我们去除有缺失的数据，获得 1710660 条数据。

（2）筛选合理长度范围的轨迹。将每条轨迹包含的经纬度点对数作为轨迹的长度，我们取 2000 条轨迹作为样本对轨迹的长度进行了统计，结果如图 3-1 所示。

表 3-1 出租车数据集字段描述

Table 3-1 Detailed description of the dataset

数据集字段	含义
TRIP_ID	行程标识
CALL_TYPE	乘客上车方式，包括电话、车站、随机等
ORIGIN_STAND	起点位置
TAXI_ID	车辆标识
TIMESTAMP	时间戳
DAY_TYPE	日期类型，包括节假日、非节假日等
MISSING_DATA	是否包含缺失值
POLYLINE	采样点经纬度记录

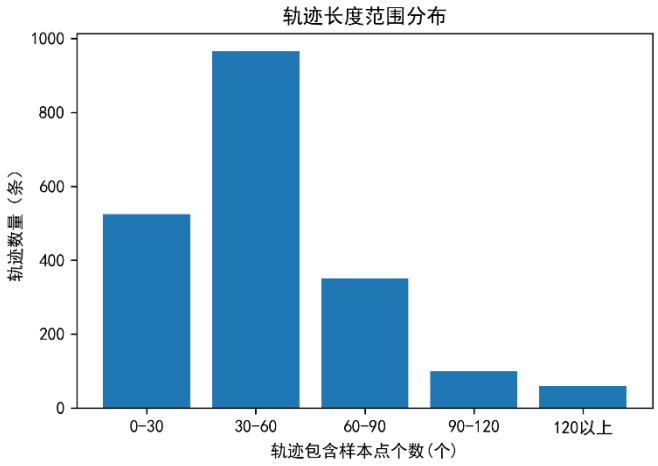

图 3-1 轨迹包含点数统计

Figure 3-1 Number of points contained in trajectories

从图 3-1 中可以看出，在 2000 个样本中，长度在 30 个点以上的轨迹较多，尤其是长度在 30-60 范围内的。计算前 2000 个样本中长度的均值，我们得到 48.5675。据此我们对数据进行筛选，选出包含长度在 30 至 120 之间的数据。

（3）对轨迹的经纬度位置进行筛选。根据波尔图市的经纬度起止范围，我们使用北纬[41.0 , 41.26]、西经[8.39, 8.73]分别作为纬度和经度的上下限，对轨迹点进行筛选，去除不在此范围内的 GPS 记录。

（4）对行程的跨度距离进行筛选。如果行程经过的距离过短则很有可能行程异常，或出租车没有在行驶状态。我们将出租车的行驶距离在 2.5 公里以内的行程

去除，将经过上述处理后的行程作为我们所用的数据集。

最终得到 783947 条行程，并按照 8：2 的比例划分训练集和测试集，其中训练集包含 627157 条数据，测试集包含 156790 条数据。

3.2.3 轨迹区域编码

在这一步我们将轨迹的经纬度二维数据转换为一维，在之后的模型训练中作为类似于文本数据中的词被输入。

根据之前确定的波尔图市的范围，我们计算被选定的范围在同一纬度下的东西距离，统计结果如表 3-2 所示，可知在该区域的最南和最北纬度上东西距离都接近 30 千米。

表 3-2 选定区域的东西距离

Table 3-2 Horizontal distance of the determined area

纬度（北纬）	东西距离（千米）
41.0	28.533
41.26	28.420

接下来需要将该区域划分成均等的方格，将落在该区域内的轨迹点编码为格子的索引。而方格长度的确定则基于上述统计结果，若过长，则轨迹点都会落在同一区域内，索引大量重复；若过短，则编码的维度过高，之后在送入模型时词汇数量过多，造成计算上的压力。在平衡划分的粒度后我们将该区域划分为 300*300 共 90000 个小方格，每个小方格的长度约为 100 米，对于轨迹点的编码来说较为适中。

在得到选定区域方格的索引后，我们使用 one-hot 编码将索引值变为轨迹词汇，并添加轨迹开始、结束等特殊词汇得到词汇表。

3.3 本章小结

本章首先介绍了在本文的研究工作中用到的实验环境，接着阐述了本文研究工作用到的数据集，并对数据集进行了预处理，从而原始的车辆经纬度点对数据被转化为可以用于轨迹表达模型的轨迹序列数据。上述工作为之后的模型搭建、算法实施等工作打下基础。

本文的实验编程语言为 Python，软件开发平台是 Anaconda 集成环境，主要在 jupyter notebook 中进行代码的编写，其中用到了 Numpy，Pandas，matplotlib，Scipy，

pickle 等软件包。模型的搭建使用 Pytorch 框架。

在进行数据预处理时首先根据数据集的描述进行数据的清洗和筛选，先通过轨迹长度的统计筛选出合理区间内的轨迹数据，再通过轨迹跨越距离、轨迹点的经纬度位置等筛除不满足要求的数据，这样通过长度、距离、地理位置等多种方式综合得出清洗后较为合理的轨迹数据；在得到上述数据的基础上，我们需要经纬度点对形式的轨迹序列转化为一维的轨迹序列、便于进行后续的模型训练。通过对轨迹落点区域进行编码，我们将原本的二维经纬度数据变为一维轨迹词汇数据，并将其作为之后模型的输入，数据的预处理部分完成，这为之后的模型训练、评估打好了数据基础。

4 基于多头注意力机制的轨迹表达算法

本章介绍引入多头注意力机制的轨迹表达模型 Transformer-traj 和 BERT-traj，相比现有的使用循环神经网络的模型，本文提出的模型在特征提取上使用注意力机制，解决了长距离依赖问题，能够在多尺度的基础上进行轨迹序列特征的表示，提高模型表达的准确率，从而在相似轨迹发现任务中具有更好的表现。

4.1 基于 Transformer 的轨迹表达算法

4.1.1 模型结构

现有研究大多基于循环神经网络（RNN）组成编码器-解码器结构进行轨迹数据的特征提取，而该类模型结构在捕捉远距离关系上表现较差，无法获取多尺度信息，作为改进，我们考虑使用注意力机制代替已有的 RNN 结构，解决表达模型的远距离依赖问题。

应用在轨迹深度表达问题上，本文提出 Transformer-traj 模型进行轨迹深度表达，其结构如图 4-1 所示。其中 CLS 为序列开始标记，END 为结束标记，V 为编码器输出的轨迹向量表达。模型主要包括嵌入和位置编码、编码器、解码器三个部分，我们将经过下采样和加噪声处理后的轨迹序列 src_data 作为输入，训练模型输出完整的原始序列 trg_data。

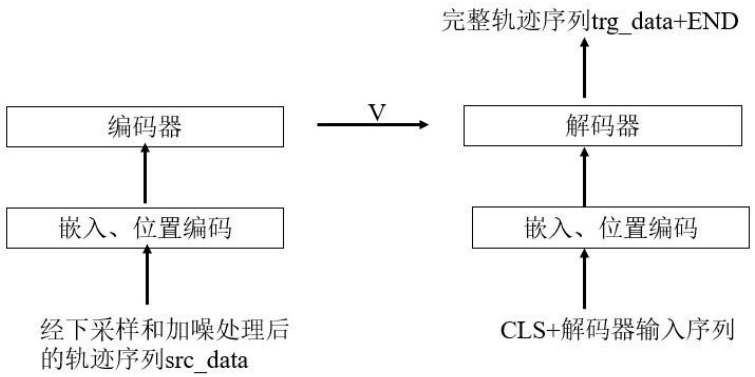

图 4-1 Transformer-traj 轨迹表达模型结构图

Figure 4-1 Structure of Transformer-traj for trajectory representation learning

数据嵌入和位置编码是经过预处理后的轨迹数据进入编码器前的准备。数据在嵌入层得到随机初始化的嵌入表达，而位置编码是给输入的序列数据保留位置信息，使用三角函数进行编码，轨迹序列点的向量表示和其位置编码相加得到编码器、解码器的输入。

第二部分是编码器，包含多头自注意力网络和前馈神经网络，作用是对输入的数据进行特征提取，得到轨迹向量表达。

第三部分是解码器，在输入编码器得到的轨迹向量和解码器自身输入的基础上，得到完整的原始轨迹序列输出。解码器的输入序列在训练阶段和测试阶段不同，训练阶段我们将真实的完整序列，也即目标序列，一个个送入解码器中，解码器对当前数据的下一时刻数据进行预测输出，通过与真实值的对比计算损失函数。而在测试阶段，我们将解码器上一时刻的输出作为下一时刻的输入，没有对结果的监督。

下面我们针对模型的三部分分别进行介绍。

（1）**数据嵌入和位置编码**。该模块结构示意图如图 4-2 所示。其中 $p_1,...,p_n$ 为输入数据，$x_1,...,x_n$ 为嵌入后的轨迹向量，$t_1,...,t_n$ 为位置编码向量，$vec_1,...,vec_n$ 为编码器的输入向量。这一部分主要是在输入编码器-解码器前的数据准备工作。我们首先使用神经网络框架 Pytorch 中的 torch.nn.Embedding 函数搭建嵌入神经层，将经预处理阶段编码处理后的轨迹序列变为嵌入向量表示。

与 RNN 不同，注意力机制在结构上无法保留数据位置信息，所以在将数据送入模型前我们需要对轨迹样本点的位置进行编码，与源数据求和作为最终输入。我们使用正弦函数和余弦函数分别对奇数和偶数位置的数据进行编码，分别使用

$$PE_{(pos,2i)} = \sin(pos / 10000^{2i/d_{\text{model}}})^{[15]} \tag{4-1}$$

$$PE_{(pos,2i+1)} = \cos(pos / 10000^{2i/d_{\text{model}}})^{[15]} \tag{4-2}$$

得到位置编码结果，其中 pos 是当前轨迹样本点在序列中的位置，i 是向量的维度数，可知在偶数位置使用正弦编码，在奇数位置使用余弦编码。

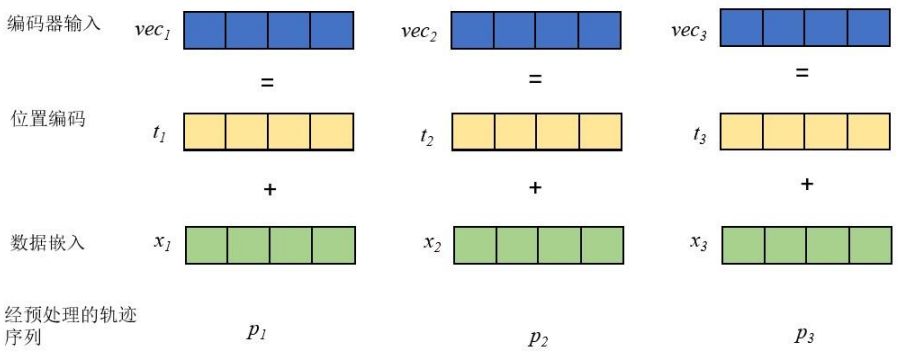

图 4-2 输入数据嵌入和位置编码

Figure 4-2 Embedding and positional encoding of the input data

（2）**编码器**。编码器结构如图 4-3 所示，它由多头自注意力网络和前馈神经网络组成。在得到的轨迹向量化表示中包含了轨迹序列中不同点之间的关系，提取了轨迹序列内部的时空特征，用于轨迹相似度计算。

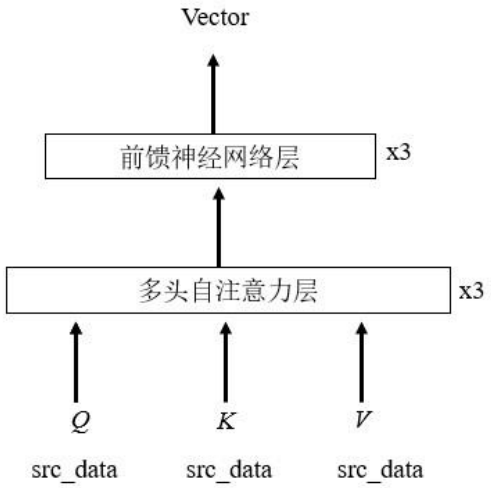

图 4-3 编码器结构示意图

Figure 4-3 Illustration of the structure of encoder

经过上一步的向量嵌入、位置信息编码后，我们将轨迹序列进行下采样和加噪处理，得到的数据称为 src_data，将其作为编码器的输入数据送入提取数据点之间的特征，输出即为我们需要的轨迹深度向量表达，该向量将在解码器中作为输入数据的一部分，训练模型输出目标数据。

编码器包括多头自注意力层和前馈神经网络层。在数据输入编码器后，首先通过自注意力层计算得到序列中不同轨迹点的注意力值，再通过前馈神经网络进行线性变换得到最终的轨迹向量表达。

多头自注意力网络的内部详细结构如图 4-4 所示。作为 Q 和 K 的轨迹序列 src_data 首先经过通过矩阵相乘计算输入序列中每个数据与其他数据的相似度，此即为注意力权重系数。softmax 层在上一层得到相似度的基础上使用函数进行归一化得到概率，经过 dropout 后再与 V 进行矩阵乘法得到注意力值，之后使用 torch.nn.Linear 函数将相乘后的结果送入线性网络。在此基础上再次经过 dropout 层、残差连接、和正则化层，得到新序列，此即为图中多头自注意力机制的输出 output_atten。

自注意力机制利用序列自身进行注意力的计算，输入数据相同，输出经过重新编码后的序列，经过自注意力机制后序列中的数据是其他数据的线性变换，表达了

当前数据和其他数据的关系，不受数据顺序限制。处理后的序列之间的整体性更强。

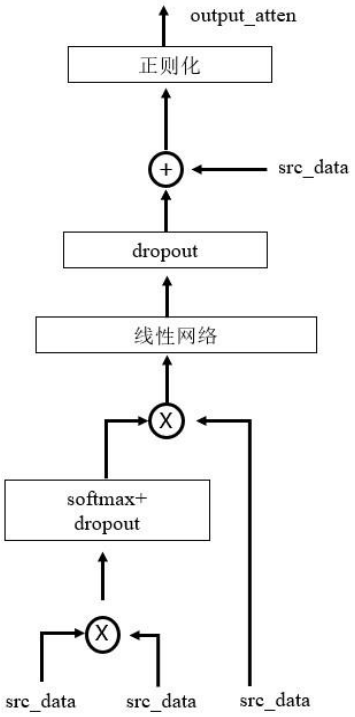

图 4-4 编码器中的多头自注意力机制

Figure 4-4 Illustration of the multi-head self-attention mechanism in encoder

而多头自注意力中的“多头”指的是对 Q、K、V 不同的线性变换，再计算相似度，通过这个过程的 h 次重复得到不同的注意力计算结果，再将 h 次不同的结果拼接起来、经过线性变换从而得到输出结果。在这个过程中会训练得到不同的权重矩阵 W，且 Q、K、V 对应不同的权重矩阵。这样设计的好处是通过多个头同时对序列的特征进行学习，得到不同的向量表示空间中的信息。此外我们将三个多头自注意力网络连接在一起形成深层网络，有助于轨迹特征的深层提取。

多头注意力机制通过学习轨迹序列内部的样本点之间的依赖关系，提取轨迹的时空特征。因为直接将序列中的元素两两进行比较和计算注意力，能进一步学习到不同尺度的信息，解决了现有序列模型中的长距离依赖问题。基于此机制的模型可以并行计算，计算的效率也有所提高。

前馈神经网络的内部结构如图 4-5 所示，它的作用是将上一环节多头注意力模块输出的加权后的轨迹序列 output_atten 转换为轨迹表达向量。主要包括全连接神经层、dropout 层、残差网络、和正则化层。

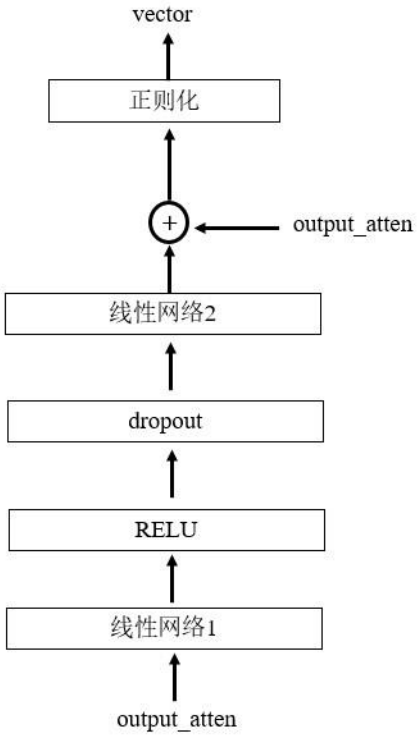

图 4-5 编码器中的前馈神经网络

Figure 4-5 Structure of feed forward neural network in encoder

其中线性网络 1 和线性网络 2 均通过 Pytorch 框架中的 torch.nn.Linear 搭建，RELU 为激活函数，输入序列经过线性变换、激活、dropout、残差连接和正则化之后输出，此即我们训练得到的轨迹深度表达 vector，将作为后续解码器输入的一部分用于解码得到目标序列。

（3）**解码器**。它的作用是利用编码器得到的轨迹向量和解码器的输入序列，最终输出目标序列。

解码器模块结构示意图如图 4-6 所示，它包含三个部分，多头自注意力网络、多头注意力网络、和前馈神经网络。解码器将轨迹序列真值 trg_data 和编码器输出的轨迹向量 vector 送入多头注意力机制，提取 trg_data 和 vector 之间的关系，训练模型根据输入序列和编码器得到的轨迹向量表达得到轨迹序列的真值。

下面分模块介绍解码器中的三个部分。首先是多头自注意力网络，解码器的中的多头自注意力模块结构如图 4-7 所示，该模块提取 trg_data 数据内部的特征，首先利用缩放点积计算注意力值，接着在获取 trg_data 数据内部的关系后经过残差连接和正则化层得到新的向量。

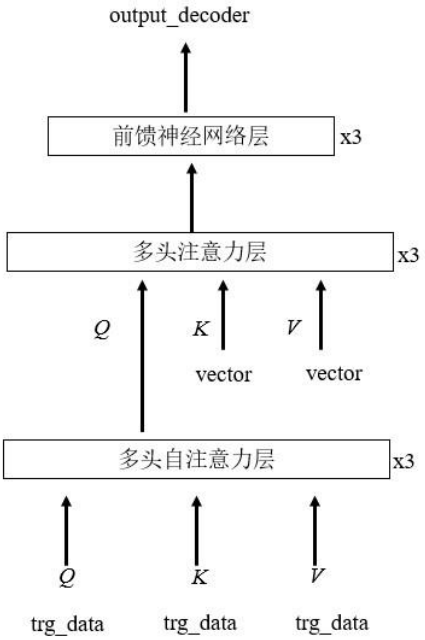

图 4-6 解码器结构示意图

Figure 4-6 Illustration of the structure of decoder

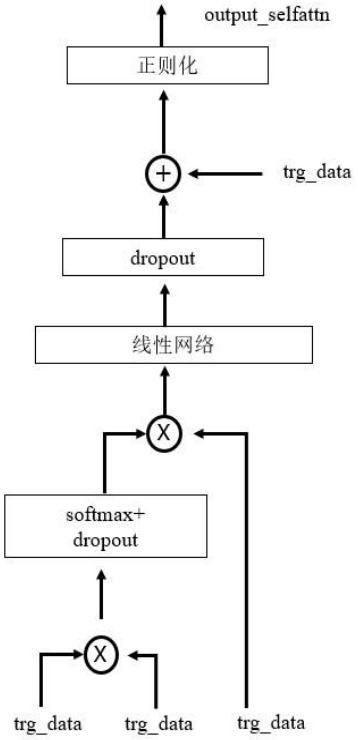

图 4-7 解码器多头自注意力层示意图

Figure 4-7 Illustration of the multi-head self-attention layer in decoder

与编码器中的注意力模块相同，解码器中的多头自注意力提取同一序列自身数据间的依赖关系。但与编码器部分的多头自注意力不同的是解码器只能利用当前时刻数据之前的信息和编码器得到的序列信息进行解码输出，相当于预测模块，而编码器中的输入数据是整条轨迹序列。在训练阶段，解码器的多头自注意力的输入序列是轨迹序列真值。在测试阶段，解码器的输入是之前时刻自身的输出，随着时间不断延伸，解码器的输入序列也越来越长，得以根据之前的输出对下一时刻的输出进行预测。相比基于 RNN 的编码器-解码器结构，多头注意力机制的应用具有解决了长距离依赖关系的捕捉能力较差和无法并行处理的问题。

解码器中多头注意力网络的结构如图 4-8 所示。与前述中的多头自注意力机制不同的是，解码器的这一模块捕捉的是不同序列间的依赖关系，将上一多头自注意力模块的输出 output_selfattn 和编码器的输出 vector 送入解码器，提取两者之间的关系用于轨迹的预测。

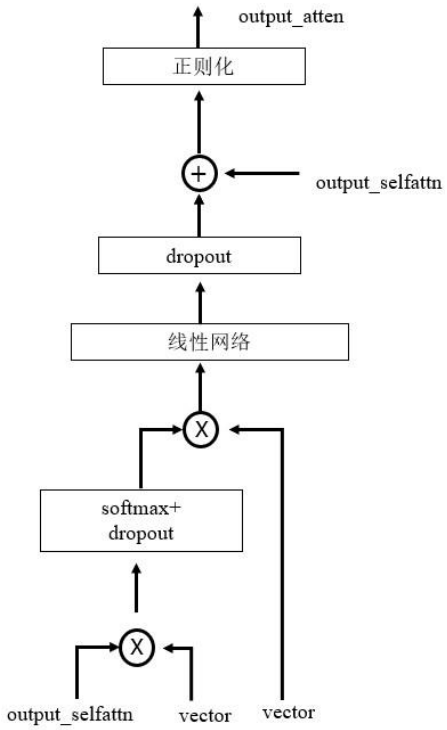

图 4-8 解码器的多头注意力机制

Figure 4-8 Illustration of the multi-head attention mechanism in decoder

解码器中前馈神经网络的结构与编码器中的对应模块结构相同。它的作用是，将上一步骤中多头注意力网络输出的序列 output_atten 进行线性变换、非线性激活、转换为最终的目标序列 trg_data。

4.1.2 模型训练

为了使轨迹表达模型对不均匀轨迹采样频率、轨迹噪声具有良好的健壮性，我们设计的训练任务是，对经过预处理的轨迹点序列进行下采样和加噪声处理，训练模型输出未经处理的轨迹序列。在进行上述处理时我们将设定下采样率α和加噪率β，若原序列长度为 L，则下采样操作后原序列中 L*α个元素被去掉，元素位置随机；加噪操作后有 L*β个元素会在原有经、纬度点对基础上经纬度分别增加随机大小的漂移成为噪声轨迹数据点，通过 numpy.random.normal(0,1,2)* numpy.array([0.00035189 ,0.0002698])实现，其中第一部分表示标准正态分布，第二部分表示 30 米对应的经纬度变化量，也即数据至多在经度和纬度上增加 30 米，反映在轨迹区域编码环节，轨迹点所在的格子在水平和垂直方向上将各移动一个单位。该噪声参数的设置对应我们在前述预处理部分将轨迹落点区域划分为长度为 30 米的格子。

如果以 0.1、0.2、0.3、0.4、0.5 共五种采样率对源数据进行下采样，接着再分别对上述结果以与上面相同的五种加噪率对源数据进行加噪处理，组合在一起可以得到 25 种处理方式。具体来说，原本经过预处理后的训练数据集的规模为 62 万余条，我们将每 50000 条划分为一个子数据集，取前 12 个子数据集中的数据进行下采样和加噪声共 25 种变换，这样每个子数据集中包含 1250000 条轨迹的数据，我们将其作为 Transformer-traj 的输入数据 src_data（后文称为源数据），训练输出没有经过干扰和轨迹数据。

在模型进行训练时我们将交叉熵损失函数作为模型训练的损失函数，设我们期望输出的轨迹序列真值为 y，实际解码器的输出序列为 $pred$，则损失函数为

$$Loss(pred, y) = -\sum_i y_i \log(\text{softmax}(pred_i)) \quad (4\text{-}3)$$

该交叉熵损失函数是 softmax 函数和 NLLLoss 的结合，在实验中我们使用 torch.CrossEntropyLoss 函数计算损失。

此外在超参数的设置方面，我们设定的 dropout 率为 0.1，头的个数为 8，编码器和解码器中的注意力层数和前馈神经网络层数都为 3，注意力层的神经元个数为 256，前馈神经网络的神经元个数为 512，训练轮数 epoch 为 2。

4.2 基于 BERT 的轨迹表达算法

虽然 Transformer-traj 相比循环神经网络在远距离特征提取上有了改进和弥补，但其语言模型为单向，在多头自注意力层中模型只考虑了当前时刻轨迹点与之前数据间的关系，而实际上某轨迹点与其前后的轨迹数据都有关系。因此作为改进，

我们考虑使用双向编码器进行轨迹点前后的时空特征提取，也即把在Transformer[15]基础上发展的BERT[16]模型用于轨迹序列的表征，BERT是使用双向Transformer 编码器的预训练模型，我们设计通过遮蔽轨迹预测的预训练任务训练模型得到轨迹的准确表达。现实中获得的轨迹数据常因人为或其他因素导致轨迹采样点遗漏问题，因而产生轨迹采样频率低或不一致的情况，本文的预训练任务通过训练模型对遮蔽轨迹预测使得模型能在上述干扰下获得较为准确的向量表示，具有健壮性。

下面就本文设计的基于BERT的轨迹表达模型BERT-traj进行介绍。

4.2.1 模型结构

本文提出的BERT-traj模型结构如图4-9所示。其中emb_i表示经过嵌入和位置编码处理后的轨迹嵌入向量，将经过预处理的源数据src_data进行遮盖，替换成为一个固定的轨迹表示vec_i，再将遮盖后的数据送入模型、训练模型对被遮盖部分进行预测。在结构上在已有的BERT模型基础上增加了一层线性网络，将获得的轨迹向量表达转换为序列数据。

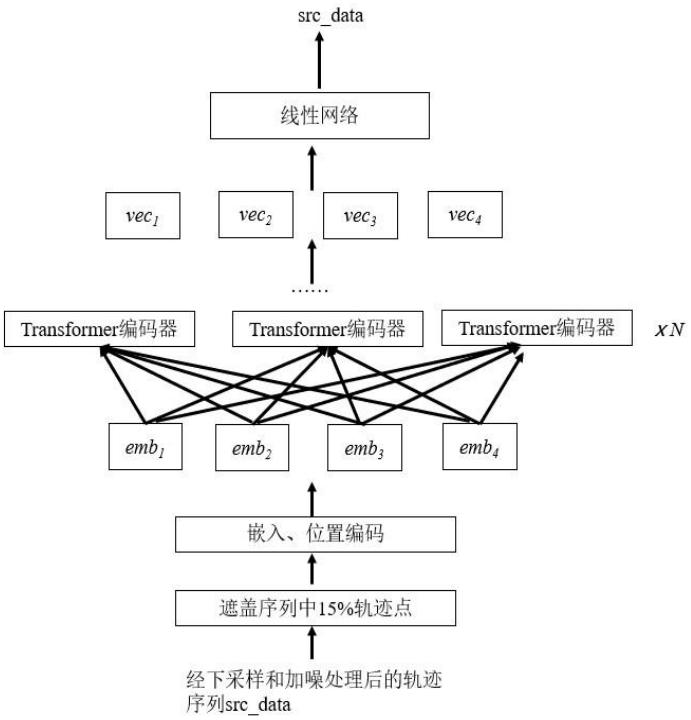

图4-9 BERT-traj轨迹深度表达模型

Figure 4-9 The structure of BERT-traj for trajectory representation

在预训练任务的选择上，如图 4-10 所示，模型 BERT-traj 采用轨迹遮蔽点预测的方法，捕捉当前数据与序列中其他数据的关系，利用整条轨迹序列作为上下文提取轨迹序列间的特征。

我们首先对输入的源数据 src_data 进行遮蔽处理，选择遮盖源数据 src_data 中 15%的数据，使用 random.sample 函数随机选择序列中的轨迹点的索引，遮盖后的轨迹点都替换为特殊表示。

再将上述数据进行嵌入和位置编码，保留数据的时序信息。通过使用 torch.nn.Embedding 得到输入数据的初始化向量表示，再通过正弦、余弦函数分别对偶数和奇数位置上的序列进行编码，最后将轨迹向量与位置编码求和作为编码器的输入数据。

之后在双向编码器中使用多头自注意力和前馈神经网络计算注意力值并输出新的序列，再通过线性网络输出预测结果。在双向编码器网络中，模型不仅关注当前数据与之前数据的关系，同样也关注与之后数据的关系，视野更宽。在结构上编码器包括多头自注意力模块和前馈神经网络，对输入数据自身提取时空特征进行向量表达。

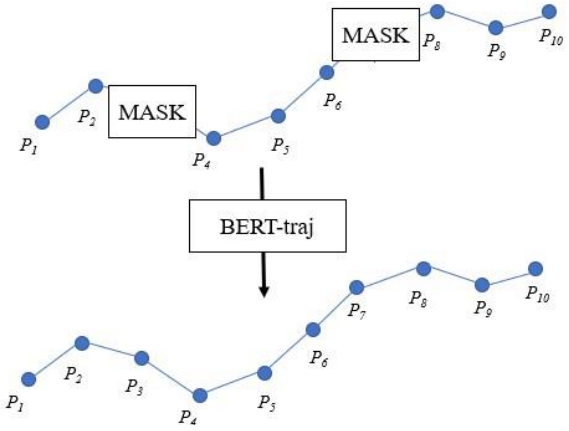

图 4-10 轨迹遮蔽点预测

Figure 4-10 Masked point prediction in trajectories

最后一部分线性网络模块的作用是利用编码器输出的轨迹表达向量输出完整的源序列 src_data。我们采用 Pytorch 中的 torch.nn.Linear 搭建线性网络，将线性网络的结果作为输出的预测结果，与真实值进行对比，依据此设计损失函数对模型进行训练。

4.2.2 模型训练

我们采用的预训练任务是轨迹点遮蔽预测，通过输入经遮蔽处理后的轨迹序列，训练模型对被遮蔽轨迹点的预测。采用的损失函数为交叉熵损失函数，计算公式如公式（4-3）所示。

BERT-traj 良好的特征提取能力建立在大规模数据和大量参数的基础上，我们对原始的数据集进行了扩充，将经过下采样和加噪声处理的数据和未经处理的数据都送入模型进行训练。此外在超参数的设置方面，我们设定的 dropout 率为 0.1，头的个数为 8，编码器和解码器中的注意力层数和前馈神经网络层数都为 3，注意力层的神经元个数为 256，前馈神经网络的神经元个数为 2048，训练轮数 epoch 为 500。

4.3 本章小结

本章主要介绍了基于多头注意力机制的轨迹表达模型 Transformer-traj 和 BERT-traj。在内容上首先分别介绍了两个模型的结构，接下来阐述了如何进行模型训练。

Transformer-traj 是使用多头注意力机制和前馈神经网络搭建的编码器-解码器结构，相比现有的基于循环神经网络的模型，这种结构能克服 RNN 的远距离特征提取困难的问题。此外在训练方式上我们通过将经过下采样和加噪声处理后的数据送入模型，训练输出完整的原始轨迹，使模型具有抵抗不均匀采样率和噪声的能力。

BERT-traj 是在上述模型基础上的改进模型，它使用 Transformer-traj 中的编码器部分进行轨迹特征提取，相比 Transformer-traj，它的编码器在进行轨迹序列的注意力计算时不仅关注轨迹点与之前轨迹的关系、也关注与之后轨迹点之间的关系，是由单向到双向的转变。此外轨迹遮蔽点预测的方式进行模型训练也使轨迹表达模型面对轨迹低采样率问题上具有健壮性。

在此章的基础上，下一章将针对模型的表达准确率进行实验评估和验证。

5 模型评估与结果分析

轨迹表达准确度是以轨迹相似度计算任务为基础进行衡量的。而轨迹相似度的计算和相似轨迹发现对于轨迹数据挖掘、城市热门区域发现、路径推荐、公安部门的可疑人员排查等任务都有重要的意义。因而获得准确的轨迹表达对建设智慧交通、智慧城市具有基础性作用。

但目前在轨迹相似度计算方面没有公认的真实值来衡量什么样的轨迹是相似的、什么样的轨迹不是，根据现有的研究者的研究方法[47-48]，我们设计了三种实验来衡量轨迹表达的准确度。它们设计的基础思想是一条轨迹和它的下采样轨迹、同一条轨迹的不同下采样轨迹间应是相似的基础上，下面我们首先介绍进行评估实验前的数据准备工作，再对三种评估方法进行详细的介绍，并对不同模型的评估结果进行展示和分析。

5.1 实验设置

在第三章数据预处理部分我们已经按照 8：2 的比例对数据集进行切割，共获得 156790 条数据作为测试数据集。接下来我们对测试数据集进行处理，主要包括以下几个步骤。

（1）查询数据集和数据库数据集的建立。由于轨迹相似度的计算需要建立在查询轨迹 query 和轨迹数据库 pool 的基础上，也即查询轨迹 query 与数据库 pool 中的轨迹一一对比、计算得到轨迹相似度。因此我们首先对查询数据集和轨迹数据库数据集进行构建，我们构建了两个规模不同的数据集，一个是取测试集的前 1000 条作为查询数据，前 11000 条作为轨迹数据库，称其为数据集_small；另一个是取测试集的前 5000 条数据作为查询数据，前 25000 条作为轨迹数据库，称为数据集_large。其规模统计如表 5-1 所示。

表 5-1 测试用数据集规模统计表

Table 5-1 Scale of different datasets for testing

数据集名称	查询轨迹 query 数量（条）	轨迹数据库 pool 数量（条）
数据集_small	1000	11000
数据集_large	5000	25000

（2）构建 downsample_distort 数据集。接下来我们分别对上述数据集_small 和数据集_large 进行下采样和加噪声，用于之后验证模型对于不均匀采样率和轨迹噪声的健壮性。

由于当前数据集中的记录为经纬度点对形式，为了将上述数据在测试阶段送入模型训练，我们将数据集中的经纬度点对通过第三章数据预处理的方式进行编码，得到轨迹词汇数据集。

再分别对轨迹词汇形式的查询 query 数据集和数据库 pool 数据集进行下采样和加噪声，采样率和加噪率分别有 0.1、0.2、0.3、0.4、0.5，共五种，最终得到两种规模、分别针对查询轨迹和数据库、包含 10 种下采样和加噪操作的共 2（规模数）*2（query 和 pool 两个数据集）*10（采样和加噪共 10 种操作）=40 个处理结果。

（3）构建 alternate 数据集。构建这个数据集的原因在于我们认为一条轨迹取奇数索引得到的子序列和偶数索引得到的子序列是相似的。可以利用模型对一条子轨迹序列的表达与另一条双胞胎子序列的表达进行距离计算，判断表达的准确度。

因此我们对数据集_small 的查询轨迹数据集 query 从第一个点开始、每间隔一个点取样，得到每条轨迹的第 1、3、5、7、9……个点组成的轨迹，再对轨迹数据库 pool 从第二个点开始、每间隔一个点取样，得到每条轨迹的第 2、4、6、8……个点组成的轨迹。以上述两个数据集作为新的查询数据集和轨迹数据库。

接着我们如上一步那样进行原始经纬度点对到词汇数据的转换，再进行下采样和加噪处理，同样得到 2*2*10=40 个处理结果。

在进行模型评估前的准备到此完毕，下面介绍评估方法的实验设计。

5.2 实验设计

本文共设计实施了三种针对轨迹表达准确性的评估实验，分别是相似轨迹间距离计算、交叉相似性计算、轨迹 KNN 查询。下面分别对三种评估方法进行介绍。

（1）自身相似度。该方法的实施过程如图 5-1 所示，我们将原始轨迹序列进行下采样得到两条子轨迹，也即 5.1 中 alternate 数据集的构建中所叙述的方法，再对子轨迹序列 1 和子轨迹序列 2 进行向量距离的计算。研究者们[13,47-48]认为同一条轨迹的两条下采样轨迹之间应是相似的，相应地它们的轨迹表达也应在向量空间中距离较近。因此通过计算这两个向量之间的距离我们可以得出轨迹表达的准确度。在向量空间中这两个子轨迹的距离越小，则越说明轨迹表达的准确性。

我们利用这个方法在数据集_small 和数据集_large 上分别使用欧几里得距离和

余弦距离作为距离衡量方法，对基于 RNN 编码器-解码器结构、基于 Transformer 的轨迹表达模型和基于 BERT 的轨迹表达模型进行对比，此外还衡量了在添加下采样和噪声干扰情况下的结果。

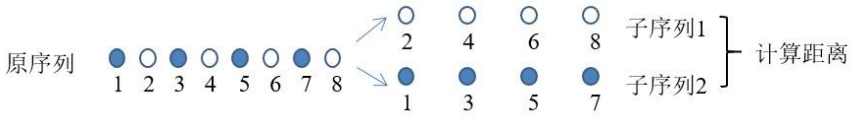

图 5-1 自身相似度评估

Figure 5-1 Evaluation of self-similarity

（2）交叉相似度。这一方法的设计是基于，好的轨迹表达算法不仅应该能够让轨迹与自身的子轨迹在向量空间上相距较近，还应让不同的轨迹在向量空间上相距较远。比如T_a和T_b是两条不同轨迹，$T_a^{'}$和$T_b^{'}$分别是前面两条轨迹进行下采样或加噪声之后的轨迹，则

$$\frac{|d(T_a^{'},T_b^{'})-d(T_a,T_b)|}{d(T_a,T_b)} \tag{5-1}$$

越小越好。

在数据准备方面我们采用上一节 5.1 中 downsample_distort 数据集，分别在两种规模数据集上、用不同距离衡量方法、在采用不同下采样和加噪声率处理情况下对不同模型表达结果进行对比。

（3）轨迹 KNN 查询。这个方法的设计出于与（1）中相同的自身相似性衡量的考虑，即一条轨迹的 k 个最近邻和其变体轨迹（下采样和加噪声）是相似的，那么它们的 k 个最近邻的重合度应该较高。所以得到的结果越大、则说明轨迹表达的准确度越高。

这一方法使用的数据集是 5.1 中介绍的 downsample_distort 数据集，我们在两种规模数据集上、采用不同距离衡量方法、采用不同采样率和加噪率对不同模型表达结果进行对比。

5.3 实验结果与分析

本文对基于 RNN 的编码器-解码器模型（t2v[13]）、与本文设计的基于 Transformer 的轨迹表达模型（Transformer-traj）、和基于 BERT 的轨迹表达模型（BERT-traj）的实验结果进行对比。下面就三种模型在自身相似度、交叉相似度、轨迹 KNN 查询 3 种评估测度下的实验结果进行展示与分析。

5.3.1 实验结果

（1）自身相似度评估的是同一轨迹的两条子轨迹向量之间的距离，越小则说明轨迹表达的准确度越高。在这里我们分别对数据以不做处理、下采样、加噪声三种方式进行评估，并得到结果如表 5-2、表 5-3、表 5-4 所示。

1）在不加采样和干扰的情况下，使用余弦距离衡量三种模型在数据集_small 和数据集_large 上的结果如表 5-2 所示。

表 5-2 使用余弦距离的自身相似度比较

Table 5-2 Evaluation of self-similarity using cosine similarity

数据集	模型名称	相似轨迹距离
数据集_small	t2v	0.977
	Transformer-traj	**0.780**
	BERT-traj	**0.770**
数据集_large	t2v	0.960
	Transformer-traj	**0.786**
	BERT-traj	**0.761**

从表 5-2 实验结果可以看出，在不同数据集上基于多头注意力机制的模型都取得了比 t2v 更好的效果。其中在数据集_small 上，Transformer-traj 相比 t2v 的表达准确度提高了 20.16%；BERT-traj 相比 t2v 的表达准确度提高了 21.19%。在数据集_large 上，Transformer-traj 相比 t2v 的表达准确度提高了 18.13%；BERT-traj 相比 t2v 的表达准确度提高了 20.73%。且 BERT-traj 相比 Transformer-traj 表现更好。

而当使用欧几里得距离作为标准评估模型表达准确度时，我们得到的结果如表 5-3 所示。

表 5-3 使用欧几里得距离的自身相似度比较

Table 5-3 Evaluation of self-similarity using Euclidean distance

数据集	模型名称	相似轨迹距离
数据集_small	t2v	2.257
	Transformer-traj	**0.082**
数据集_large	t2v	2.291
	Transformer-traj	**0.081**

使用欧几里得距离作为相似度计算方法时，本文的基于多头自注意力的模型依然相比基线模型 t2v 有更好的结果，且效果较为显著。其中在数据集_small 上，Transformer-traj 相比 t2v 的表达准确度提高了 96.4%。在数据集_large 上，Transformer-traj 相比 t2v 的表达准确度提高了 96.5%。

欧几里得距离关注的是被比较向量的绝对大小上的差距，余弦距离比较的是向量间夹角的大小，说明基于多头注意力机制的模型在向量方向、大小上均比基线模型有更好的表达效果。

2）在对数据进行下采样处理的情况下，使用余弦距离衡量三种模型在数据集_small 和数据集_large 上的结果如表 5-4 所示。

表 5-4 使用余弦距离的自身相似度比较

Table 5-4 Evaluation of self-similarity using cosine similarity

数据集	模型	采样率 0.1	采样率 0.2	采样率 0.3	采样率 0.4	采样率 0.5
数据集_small	t2v	0.981	0.974	0.967	0.958	0.947
	Transformer-traj	**0.877**	**0.868**	**0.855**	**0.838**	**0.816**
	BERT-traj	**0.873**	**0.832**	**0.847**	**0.837**	**0.810**
数据集_large	t2v	0.980	0.974	0.966	0.958	0.947
	Transformer-traj	**0.877**	**0.868**	**0.855**	**0.838**	**0.815**
	BERT-traj	**0.882**	**0.861**	**0.854**	**0.835**	**0.806**

由表 5-4 可以看出，本文采用的模型在应对现实中可能存在的轨迹采样率不均匀、采样率低的情况下依然有良好的表现。其中在数据集_small 上，Transformer-traj 相比 t2v 的表达准确度平均提高了 11.89%；BERT-traj 相比 t2v 的表达准确度平均提高了 13.02%。在数据集_large 上，Transformer-traj 相比 t2v 的表达准确度平均提高了 11.87%；BERT-traj 相比 t2v 的表达准确度平均提高了 12.18%。

3）当对数据进行加噪声处理时，使用余弦距离作为轨迹相似度衡量标准、评估三种模型在不同数据集上的表达效果时，评估结果如表 5-5 所示。该实验验证了本文的模型在轨迹噪声情况下依然有良好的表现，现实情况中由于人为或意外等因素，轨迹记录可能存在噪声干扰，从而同一路线的不同轨迹可能被现有方法认定为两条路线下的轨迹，造成准确度上的损失。本文提出的 Transformer-traj 和 BERT-traj 相比基线模型都有较好的表现。具体来说，在数据集_small 上，Transformer-traj 相比 t2v 的表达准确度平均提高了 8.96%；BERT-traj 相比 t2v 的表达准确度平均提高了 10.61%。在数据集_large 上，Transformer-traj 相比 t2v 的表达准确度平均提高

了 8.46%；BERT-traj 相比 t2v 的表达准确度平均提高了 8.91%。

表 5-5 使用余弦距离的自身相似度比较

Table 5-5 Evaluation of self-similarity using cosine similarity

数据集	模型	加噪率 0.1	加噪率 0.2	加噪率 0.3	加噪率 0.4	加噪率 0.5
数据集_small	t2v	0.974	0.973	0.971	0.969	0.967
	Transformer-traj	**0.885**	**0.885**	**0.884**	**0.882**	**0.883**
	BERT-traj	**0.870**	**0.868**	**0.867**	**0.866**	**0.868**
数据集_large	t2v	0.974	0.972	0.970	0.968	0.966
	Transformer-traj	**0.970**	**0.869**	**0.868**	**0.867**	**0.866**
	BERT-traj	**0.885**	**0.884**	**0.884**	**0.883**	**0.882**

（2）交叉相似度是综合考虑相似轨迹间距离应较近、不同轨迹间距离应较远的评估方法，数值越小则说明模型的表达准确度更好。其中需要比较不同轨迹、及相似轨迹（原轨迹与其加噪声和下采样后的轨迹）之间的距离，因此我们分别对原始数据进行了加噪声和下采样处理，下面分别进行阐述。

1）加噪声情况。我们仍然选择余弦距离作为轨迹相似度计算方法，得出结果如表 5-6 所示。

表 5-6 使用余弦距离的交叉相似度比较

Table 5-6 Evaluation of cross-similarity using cosine similarity

数据集	模型	加噪率 0.1	加噪率 0.2	加噪率 0.3	加噪率 0.4	加噪率 0.5
数据集_small	t2v	0.0093	0.016	0.022	0.026	0.030
	Transformer-traj	0.023	0.033	0.041	0.050	0.054
	BERT-traj	**0.0063**	**0.013**	**0.017**	**0.021**	**0.023**
数据集_large	t2v	0.010	0.017	0.022	0.026	0.030
	Transformer-traj	0.022	0.034	0.041	0.048	0.055
	BERT-traj	**0.008**	**0.012**	**0.017**	**0.020**	**0.023**

由表 5-6 可以看出，BERT-traj 的表现最好，t2v 的表达准确度次之，Transformer-traj 的效果最差。究其原因，可能是 Transformer 在进行训练时，解码器只能根据当

前时刻之前的数据进行预测输出，这样训练学到的特征具有局限性。相比之下BERT 的遮蔽词预测预测任务可以看到整条轨迹序列作为上下文，不仅能关注当前序列之前的数据，也能关注到当前时刻之后的数据。在双向自编码器的机制下学习到了更为丰富和全面的特征，具有更好的向量表征能力。

具体来说，在数据集_small 上，BERT-traj 相比 t2v 的表达准确度平均提高了23.26%。在数据集_large 上，BERT-traj 相比 t2v 的表达准确度平均提高了 23.71%。

2）下采样情况。我们选择余弦距离作为轨迹相似度计算方法，得出结果如表5-7 所示。

表 5-7 使用余弦距离的交叉相似度比较

Table 5-7 Evaluation of cross-similarity using cosine similarity

数据集	模型	采样率 0.1	采样率 0.2	采样率 0.3	采样率 0.4	采样率 0.5
数据集_small	t2v	0.035	0.049	0.064	0.081	0.103
	Transformer-traj	0.055	0.094	0.146	0.202	0.273
	BERT-traj	**0.029**	**0.038**	**0.057**	**0.076**	**0.089**
数据集_large	t2v	0.035	0.050	0.066	0.083	0.103
	Transformer-traj	0.056	0.097	0.147	0.206	0.276
	BERT-traj	**0.028**	**0.039**	**0.048**	**0.066**	**0.080**

从表 5-7 的结果可以看出，本文提出的 BERT-traj 模型在轨迹采样频率不理想的情况下依然有较好的表现，具体来说，在数据集_small 上，BERT-traj 相比 t2v 的表达准确度平均提高了 14.06%。在数据集_large 上，BERT-traj 相比 t2v 的表达准确度平均提高了 22.42%。

（3）轨迹 KNN 查询评估方法是建立在轨迹自身相似性的基础上的，比较一条轨迹的 k 个最近邻、和原轨迹的相似轨迹（加噪声和下采样后的轨迹）的 k 个最近邻的重合度，数值越高则说明表达效果越好。所以我们通过加噪声和下采样产生相似轨迹，评估结果见表 5-8 和表 5-9。

1）加噪声情况。使用余弦距离衡量轨迹相似度，得到结果如表 5-8 所示。小型数据集中查询轨迹为 1000 条，大型数据集中查询轨迹为 5000 条，我们这里设定的邻居数为 30，则结果中越接近 30 则说明模型效果越准确。由表 5-8 结果可以看出，BERT-traj 的准确度是最高的。具体来说，在数据集_small 上，BERT-traj 相比 t2v 的表达准确度平均提高了 9.36%。在数据集_large 上，BERT-traj 相比 t2v 的表达准确度平均提高了 10.23%。

表 5-8 使用余弦距离的 KNN 查询（k=30）

Table 5-8 Evaluation of k-nearest neighbor query using cosine similarity(k=30)

数据集	模型	加噪率 0.1	加噪率 0.2	加噪率 0.3	加噪率 0.4	加噪率 0.5
数据集_small	t2v	27.687	26.189	25.135	24.278	23.498
	Transformer-traj	26.392	24.781	23.568	22.546	21.823
	BERT-traj	**28.748**	**28.306**	**27.918**	**27.538**	**27.266**
数据集_large	t2v	27.365	25.884	24.660	23.655	22.872
	Transformer-traj	26.038	24.302	23.089	22.020	21.045
	BERT-traj	**28.655**	**28.043**	**27.638**	**27.229**	**26.929**

2）下采样情况。同样使用余弦距离衡量轨迹相似度，得到结果如表 5-9 所示。

表 5-9 使用余弦距离的 KNN 查询（k=30）

Table 5-9 Evaluation of k-nearest neighbor query using cosine similarity(k=30)

数据集	模型	采样率 0.1	采样率 0.2	采样率 0.3	采样率 0.4	采样率 0.5
数据集_small	t2v	25.387	23.433	21.857	20.194	17.985
	Transformer-traj	23.132	19.843	16.972	14.499	12.013
	BERT-traj	**26.080**	**24.013**	**22.235**	**21.593**	**18.842**
数据集_large	t2v	24.686	22.572	20.661	18.693	16.594
	Transformer-traj	22.398	18.878	15.994	13.363	10.821
	BERT-traj	**25.631**	**23.414**	**21.468**	**19.673**	**17.693**

对比表 5-8 与表 5-9 可知，采样干扰对模型的表达效果影响较大，三种模型的准确率普遍有所降低。但仍可以看出 BERT-traj 的表达效果相对来说是最好的。具体来说。在数据集_small 上，BERT-traj 相比 t2v 的表达准确度平均提高了 3.6%。在数据集_large 上，BERT-traj 相比 t2v 的表达准确度平均提高了 4.45%。

综上所述，在不同的评估方法下，本文设计的基于 BERT 的轨迹表达模型 BERT-traj 相比基线模型有更好的效果。其中第一个实验和第三个实验从轨迹和其相似轨迹间距离的角度设计了评估实验，而第二个实验综合考虑了相似轨迹距离、和不同轨迹间距离，是更为客观和全面的衡量方法，在这个评估模式下，不管是下采样干扰还是加噪声干扰的情况，BERT-traj 相比基于 RNN 的 t2v 基本都有大于

20%的准确度提升，证明了本文设计的模型的表征能力。

5.3.2 模型对比分析

（1）使用循环神经网络的模型 t2v 与使用注意力机制的模型 Transformer-traj 和 BERT-traj 相比，后者相比前者在轨迹表达上有更好的表现，尤其是 BERT-traj，在三种评估测度下，准确率相比循环神经网络（RNN）的基线模型 t2v 都有一定提升。其原因在于，注意力机制在特征提取过程中会综合整条轨迹的信息进行注意力值的计算，得出序列中每个点与其他点之间的时空关系。而循环神经网络的视野较小，在进行远距离数据的关系提取时性能较差。

在两类模型的编码器对输入数据进行特征提取时，基线模型 t2v 按照输入数据的顺序一个个将轨迹点送入进行编码，随着序列的不断延伸，模型对距离较远的数据间的关系记忆变得越来越少，只能依靠当前数据与周围邻近的数据间的时序关系对当前的轨迹样本点进行预测。而使用注意力机制的模型在开始就获取了全部序列，并针对每个轨迹样本点和其他点之间的关系进行学习，在更宽的视野、更多尺度的帮助下自然有更好的表现。

（2）Transformer-traj 和 BERT-traj 相比，Transformer-traj 虽然在编码器的部分获取了序列中不同尺度的信息，但解码器在进行轨迹点的预测时只能利用当前时刻之前的数据，但 BERT-traj 在进行遮蔽轨迹点预测时可以利用当前时刻之前和之后的轨迹样本点，相当于利用前后时刻的环境信息进行预测，相比 Transformer-traj 能够有更多的信息，提高了轨迹表达的准确度。

BERT-traj 模型能够在轨迹表达准确度上有更好的表现的原因包括以下几个方面。一是它能够关注当前轨迹样本点和下一个轨迹样本点之间的关系，这种机制主要作用于轨迹序列的内部，一定程度上具有与循环神经网络类似的作用；二是能够关注自身和上一个样本点之间的关系，因为 BERT 所具有的双向编码器，模型能够在两个方向提取自身与相邻数据之间的特征，类似于反向的神经网络，但与双向循环神经网络的区别是能够同时而非先后进行双向的信息提取；三是模型能够关注相同或相关的样本点，包括样本点自身，这种机制使得模型的注意力能够较为分散地分布在各个样本点上，而不是过于集中在上一个点或下一个点，对于长距离关系的捕捉具有较好的能力。

5.4 轨迹表达算法的验证与应用

本节是对本文设计的 BERT-traj 模型在轨迹聚类任务上的验证和应用讨论，通

过 BERT-traj 对数据集中的轨迹进行二维空间的向量表征和聚类，可以看到与其他模型相比，该模型的聚类效果更为明显，为下一步轨迹挖掘的应用打下了基础。

5.4.1 模型表达能力验证

本节通过使用不同模型对轨迹进行向量表征，接着对其进行聚类可视化，通过聚类后形成的轨迹簇验证模型的轨迹特征提取能力。使用轨迹聚类可视化的方法可以进行模型能力验证的原因在于，若轨迹表达模型能够对轨迹的特征进行有效、准确的提取，则模型能对不同轨迹数据的特征进行提取、并依次得到不同的类，在聚类效果上应形成类内间距小、类间间距大的几个轨迹簇，代表不同特性的几类轨迹。相反如果模型的特征提取能力较差，则模型无法找到能对轨迹进行分类的特征并进行表达，反映在聚类效果上，则类内间距大、类间间距小，不同簇之间边界不明显。

本文所用的聚类方法是 K 均值（K-Means）聚类算法。它是一种迭代求解的聚类分析算法。它的步骤是首先将数据随机分为 K 组，随机选取 K 个对象作为初始的聚类中心，然后对每个对象计算与种子聚类中心的距离，并把该对象分配给最近的聚类中心。此时，聚类中心以及分配给它们的对象就为一个类。值得注意的是，每当我们分配一个对象，聚类的中心会根据目前各个类内对象重新计算得到，过程不断重复直到满足终止条件。

针对前文提出的 BERT-traj 模型，我们设计了轨迹聚类任务对模型的表达能力进行验证。实验主要包括以下几个步骤。

（1）数据准备。在这一实验中我们选择了北京市浮动车 GPS 数据集，它以每 30 到 60 秒上传一条记录的形式汇总了北京市内 3 万多辆出租车及其他部分类型车辆的时间、位置等信息。经过与第三章所述方法类似的预处理步骤，我们将得到的轨迹序列送入训练好的 BERT-traj 和 t2v 中得到对应的轨迹向量表达，该数据的格式为 1000*256 维，其中第一维表示轨迹条数，第二维表示每条轨迹被表示为一个 256 维的向量，这是由实验设置的神经元个数决定的。

（2）为了将轨迹向量在二维空间可视化，我们需要将得到的数据通过主成分分析法（PCA）进行降维，得到二维空间的表达向量。我们使用 sklearn 库中的 manifold.TSNE 函数，设定向量空间的维度、也即参数 n_components 为 2，将原有的高维向量嵌入在 2 维空间中。

（3）设定类个数，在针对 1000 条轨迹进行处理的情况下，我们设定的类的个数为 30，使用 sklearn 库中的 KMeans 函数设定参数 n_clusters 为 30，观察不同模型将 1000 条轨迹聚成 30 个类之后，是否能得到类内间距小、类间间距大的不同

轨迹簇。

利用 BERT-traj 和 t2v 分别进行轨迹特征提取、再聚类可视化后得到的结果如图 5-2、图 5-3 所示。其中各个类簇上的黑色数字表示具有不同特征的轨迹。由图 5-2 可以看出，经 BERT-traj 表达形成的类簇类内间距小、类间距离大，能够较为清晰地将获得的轨迹向量在二维空间中进行聚类，形成了多个具有不同特征的向量簇。而如图 5-3 所示，经 t2v 表达得到的轨迹特征点是较为分散的，相比图 5-2 中的聚类结果，不同类的类内间距大、类间间距小，不同类的边界不明显，t2v 相比 BERT-traj 对轨迹特征的提取能力较弱。

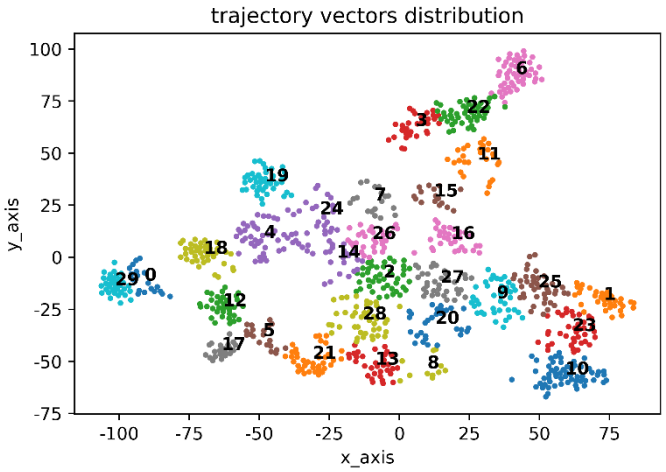

图 5-2 BERT-traj 聚类结果示意图

Figure 5-2 Clustering result using BERT-traj

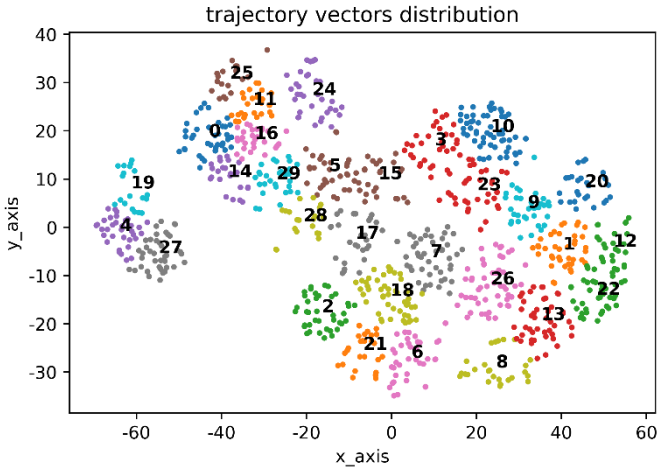

图 5-3 t2v 聚类结果示意图

Figure 5-3 Clustering result using t2v

我们计算图 5-2 和图 5-3 中的类中心之间的欧几里得距离再求均值，可以得到如表 5-10 的结果。可以看出，相比基线模型 t2v，利用 BERT-traj 得到的聚类后的轨迹向量簇的类中心间距离更大，得到了相比基线模型更好的聚类效果。

表 5-10 轨迹向量簇的类中心距离

Table 5-10 Distance between centers of trajectory clusters

模型	类中心距离
t2v	65.850
BERT-traj	**70.874**

进一步地，我们将图 5-2 其中一个轨迹簇 6 映射在地图上，结果如图 5-4 所示。由图 5-4 可以看出，该类内的轨迹呈现了较为相似的路线特征，集中在北京三环区域与东北方向的机场方向之间，是一条市区和机场之间的交通线路，证明了 BERT-traj 的表征能力。

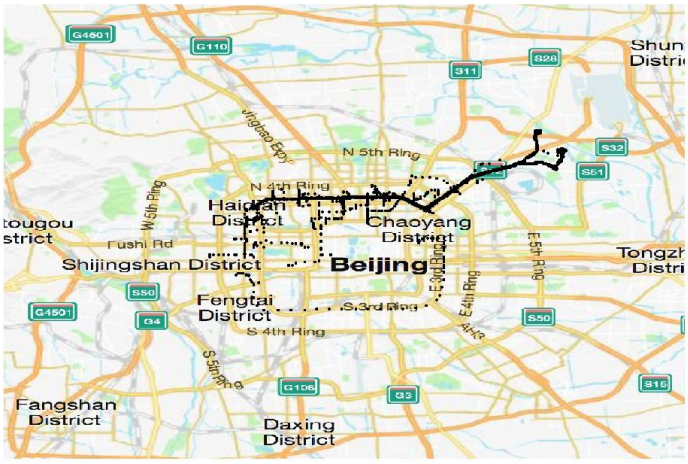

图 5-4 BERT-traj 得到的某个向量簇在地图上显示的结果

Figure 5-4 One cluster of trajectories obtained by using BERT-traj on map

5.4.2 轨迹表达算法的应用

本文的轨迹表达算法面向的是相似轨迹发现任务，在实验中所用的数据集为

车辆数据集，但其不仅可用于交通轨迹的相似发现，同样可根据人群轨迹数据计算轨迹相似性，从而达到预测筛选相似人群的作用，此外也可用于出行时间预估。

（1）预测筛选相似人群

收集人们交通轨迹数据的企业（如滴滴出行）可以利用轨迹表达算法进行路径推荐、拼车用户推荐。通过使用轨迹表达算法，我们可以得到与当前轨迹最相似的N条轨迹，在路线、关键特征上具有较高的一致性，可以为用户提供满足需求的个性化的服务和多样选择。同时用户可以了解与自己具有相似轨迹和出行习惯的其他用户，作为选择拼车服务时候的参考。

公安部门在处理案件时可以轨迹表达算法进行相似轨迹搜索，得到与目标轨迹最相近的 N 条轨迹。结合轨迹的时间、位置信息，轨迹表达算法可以得到与目标轨迹在时间和空间上最相似的轨迹，为案件排查提供有效的帮助。

（2）出行时间预估

通过轨迹中包含的时间信息，交通出行软件可以根据相似轨迹预估出行花费时间、到达时间等。在用户输入起点和终点后，根据数据库中与此路线最相似路线的搜索，我们可以得到与查询轨迹在时间、空间上都相似的历史轨迹的出行时间，从而得到对行程时间的预估。

上述场景为轨迹表达算法模型应用方向的基本概括，随着轨迹挖掘领域的不断发展，能够较为准确、高效提取轨迹特征的深度表达算法将会发挥更大的作用。

5.5 本章小结

本章主要对轨迹表达模型的准确性进行了评估，我们设计了自身相似度、交叉相似度、和轨迹 KNN 查询 3 种方法对模型进行相似轨迹发现的能力进行评估，我们获取相似轨迹的方式是将一条轨迹进行采样和加噪处理，由此处理后的轨迹与原轨迹可被认为是相似轨迹。

实验结果显示，在综合考虑相似轨迹距离和不同轨迹距离的情况下，BERT-traj 的表达效果最好。在对轨迹进行加噪变换情况下，在小型数据集上，BERT-traj 相比 t2v 的表达准确度平均提高了 23.26%；在大型数据集上，BERT-traj 相比 t2v 的表达准确度平均提高了 23.71%。当对轨迹进行采样变换时，在小型数据集上，BERT-traj 相比 t2v 的表达准确度平均提高了 14.06%；在大型数据集上，BERT-traj 相比 t2v 的表达准确度平均提高了 22.42%。

在得到实验结果的基础上，我们对不同模型的实验结果及原因进行分析。比较不同模型在特征提取上的特点，分析 BERT-traj 取得较好效果的原因。最后我们通过聚类对模型的表达效果进行了实验验证，并给出一些算法应用前景和方向。

6 总结及展望

本章主要对本文所做的工作进行总结归纳，并对与本文相关的未来的方向进行展望。首先是本文的结论部分，较为详细和系统地说明了本文的工作与关键性成果，对贡献进行了总结。之后对本文在理论和实验中存在的问题进行了分析，主要包括尚待改进的地方和尚未解决的问题，并依据此对未来的发展方向进行展望，阐述了轨迹深度表达模型的未来研究可能的方向。

6.1 论文工作总结

本节主要对论文主要的工作内容和贡献进行总结。

随着各种基于 GPS 的定位设备、内置定位功能的智能手机的应用和发展，大量交通轨迹数据不断产生，这也使得获取城市交通轨迹数据并对其进行研究成为可能。轨迹数据的挖掘和提取对于智慧城市的建设具有重要的作用，而在各项轨迹挖掘任务中，轨迹相似度计算是一项基础任务。如果能在向量空间中对轨迹的特征进行提取，那么轨迹相似度计算任务的完成就有了有力的工具。这使得轨迹的表达成为任务的核心。

本文主要针对基于多头注意力机制的轨迹表达模型进行研究，借鉴自然语言处理领域的 Transformer[15]、BERT[16]模型，并将其运用在轨迹数据的表达问题上，通过将数据送入深度学习网络进行特征提取。在内容上，首先对本文的研究背景与意义进行介绍，并阐述了轨迹表达和相似度计算问题的研究现状，在此基础上，提出了利用多头注意力机制进行轨迹深度表达。之后介绍了本文涉及到的基本理论知识，包括轨迹基本概念、深度学习算法、和基于多头注意力机制的深度表达学习算法，对不同算法的结构和原理进行阐述和比较，以及对轨迹相似度衡量方法进行介绍。在此基础上介绍了本文的研究工作用到的开发工具和平台，以及数据预处理方法。接下来对本文提出的 Transformer-traj 和 BERT-traj 进行详细阐述和评估，在有相关实验结果验证的基础上证明了本文提出模型相比基线模型的优越性。最后对本文的工作进行了总结。

本文的贡献主要包括以下几个方面：

（1）在轨迹深度表达模型中引入多头注意力机制，代替了以往的循环神经网络为基础的模型。多头注意力机制在提取特征时能够克服以往基线模型存在的长距离依赖问题，能够更好地提取序列内部的特征，同时可以并行计算。

（2）提出了轨迹深度表达模型 Transformer-traj 和 BERT-traj，在结构上前者是

后者的基础，在性能上后者实现了较高的准确率和较强的特征提取能力。本文将自然语言处理领域的模型应用在交通车辆轨迹挖掘领域，将经过下采样和加噪声处理的数据作为模型的输入，训练模型得到轨迹的深度表达，使得模型在轨迹采样率低、采样率不均匀和有噪声干扰的情况下依然能够较为准确地获得轨迹的向量表达，有良好的健壮性。

（3）通过在真实数据集上的多个实验验证，BERT-traj 具有较好的特征提取能力，能够对轨迹进行较为准确的向量表达。本文设计实施了三种评估模型表达准确度的实验，分别是自身相似度、交叉相似度和轨迹 KNN 查询。其中交叉相似度既考虑到同一轨迹间的距离、也考虑到不同轨迹间的距离，是一个较为全面的衡量指标，在小型数据集上的结果显示，加噪声和下采样干扰情况下，BERT-traj 相比基线模型 t2v 在这个指标上分别高出 23.26%和 14.06%。在大型数据集上，BERT-traj 相比基线模型 t2v 分别高出 23.71%和 22.42%。

6.2 未来工作展望

本文主要提出了基于多头注意力机制的轨迹深度表达模型 Transformer-traj 和 BERT-traj。目前将自然语言处理领域的模型与轨迹挖掘相结合的研究较少，本文仅仅针对这个领域进行了一点小小的探究，仍有很多需要解决的问题需要在之后继续探索发展，主要包括以下 3 个方面。

（1）BERT 模型强大的表征能力建立在大量的数据支持和参数设置上，而本文采用的数据集经过预处理后数量仍十分有限，可能在模型的效果优化、参数设置、神经层的搭建等方面仍有较大的局限性，在未来的工作中如果能够在更多更大数据集上进行训练和改进，相信会比现在的结果有更全面和丰富的评估结果。

（2）本文主要研究的方向是轨迹深度表达学习算法，是诸如城市热门区域发现、路径推荐、可疑人员搜索等现实应用的基础研究，基于此算法的相似轨迹查找等下游任务具有广阔的应用领域，需要我们接下来进一步发掘和研究。

（3）本文未能对更多的自然语言处理领域的模型进行更广泛的利用和分析对比，同时在使用 BERT 进行轨迹表达过程中我们没有使用微调，仅使用预训练任务对模型进行轨迹表达的获取训练，如何设计轨迹表达更为丰富的预训练和微调任务，是我们需要在未来思考的问题。

参考文献

[1] 史新宏,蔡伯根, 穆建成. 智能交通系统的发展[J]. 北京交通大学学报, 2016, 26(1):29-34.
[2] Hung C C, Peng W C, Lee W C. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes[J]. The VLDB Journal, 2015,24(2):169-192.
[3] Di Y, Chao Z, Zhu Z, et al. Trajectory clustering via deep representation learning[C]. // International Joint Conference on Neural Networks 2017, Anchorage, USA, May 14-19, 2017. 3880-3887.
[4] 郭岩, 罗珞珈, 汪洋等. 一种基于 DTW 改进的轨迹相似度算法[J]. 国外电子测量技术, 2016, (09):66-71.
[5] Lin M, Hsu W J. Mining GPS data for mobility patterns: a survey[J]. Pervasive & Mobile Computing, 2014, 12(11):1-16.
[6] 孟小峰, 慈祥. 大数据管理：概念、技术与挑战[J]. 计算机研究与发展, 2013, (1):146-169.
[7] 王元卓, 靳小龙, 程学旗. 网络大数据:现状与展望[J]. 计算机学报, 2013, 36(6):1125-1138.
[8] 章志刚, 金澈清, 王晓玲等. 面向海量低质手机轨迹数据的重要位置发现[J]. 软件学报, 2016, 27(7):1700-1714.
[9] Yi B K, Jagadish H V, Faloutsos C. Efficient retrieval of similar time sequences under time warping[C]. // Proceedings14th International Conference on Data Engineering, Orlando, FL, USA, Feb. 23-27, 1998. IEEE, 2002. 201-208.
[10] Vlachos M, Kollios G, Gunopulos D. Discovering similar multidimensional trajectories[C]. // Proceedings 18th International Conference on Data Engineering, San Jose, CA, USA, Feb. 26-March 1, 2002. IEEE, 2002. 673-684.
[11] Chen L, Ng R. On the marriage of lp-norms and edit distance[C]. // Proceedings of the Thirtieth International Conference on Very Large Data Bases, 2004. 792-803.
[12] Chen L, Özsu M T, Oria V. Robust and fast similarity search for moving object trajectories[C]. // Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, 2005. 491-502.
[13] Li X, Zhao K, Cong G, et al. Deep representation learning for trajectory similarity computation[C]. // 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018. IEEE. 2018. 617-628.
[14] Zhang Y, Liu A, Liu G, et al. Deep representation learning of activity trajectory similarity computation[C]. // 2019 IEEE International Conference on Web Services(ICWS), 2019. IEEE, 2019. 312-319.
[15] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. // Advances in Neural Information Processing Systems, 2017. 5998-6008.
[16] Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
[17] 李正欣, 张凤鸣, 张晓丰等. 多元时间序列特征降维方法研究[J]. 小型微型计算机系统, 2013, 34(2): 148-154.
[18] Yu Y, Li J, Guan H, et al. Learning hierarchical features for automated extractions of road

markings from 3-D mobile LiDAR point clouds[J]. IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, 2015, 8(2): 709-726.
[19] Gomez C I, Olivieri D N. A KPCA spatio-temporal different geometric trajectory cloud classifier for recognizing human actions in a CBVR system[J]. Expert Systems with Applications, 2015, 42(13): 5472-5490.
[20] Gobbetti E, Guitian J A I, Marton F C. COVRA: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks[J]. Computer Graphics Forum, 2012, 31(3pt4): 1315-1324.
[21] Zhang S. AIS trajectories simplification and threshold determination[J]. Journal of Navigation, 2016, 69(4): 729-744.
[22] Nuha H H, Suwastika N A. Fractional fourier transform for decreasing seismic data lossy compression distortion[C]. // 2015 3rd International Conference on Information and Communication Technology(ICoICT). IEEE, 2015. 590-593.
[23] Agrawal R, Faloutsos C, Swami A. Efficient similarity search in sequence databases[C]. // International Conference on Foundations of Data Organization and Algorithms. Berlin, Heidelberg: Springer-Verlag, 1993. 69-84.
[24] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases[J]. ACM SIGMOD Record, 1994, 23(2):419-429.
[25] Ding H, Trajcevski G, Scheuermann P, et al. Querying and mining of time series data: experimental comparison of representations and distance measures[J]. Proceedings of the VLDB Endowment, 2008,1(2):1542-1552.
[26] Berndt D J, Clifford J. Using dynamic time warping to find patterns in time series[C]. // KDD workshop, 1994, 10(16):359-370.
[27] Magdy N, Sakr M A, Mostafa T, et al. Review on trajectory similarity measures[C]// 2015 IEEE Seventh International Conference on Intelligent Computing & Information Systems(ICICIS). IEEE, 2015. 613-619.
[28] Buchin K, Buchin M, Wenk C. Computing the fréchet distance between simple polygons[J]. Computational Geometry: Theory and Applications, 2008,41(1-2):2-20.
[29] Tang B, Yiu M L, Mouratidis K, et al. Efficient motif discovery in spatial trajectories using discrete fréchet distance[C]. // EDBT 2017, Venice,Italy, 2017.
[30] Ranu S, Deepak P, Telang A D, et al. Indexing and matching trajectories under inconsistent sampling rates[C]. // 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015. 999-1010.
[31] Esling P, Agon C. Time-series data mining[J]. ACM Computing Surveys (CSUR), 2012,45(1):1-34.
[32] Lee J G, Han J, Whang K Y. Trajectory clustering: A partition-and-group framework[C]// Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007. 593-604.
[33] Besse P, Guillouet B, Loubes J M, et al. Review and perspective for distance based trajectory clustering[J]. arXiv preprint arXiv: 1508.04904, 2015.
[34] Tang W, Dechang P, He Y. A density-based clustering algorithm with sampling for travel behavior analysis[M]. Lecture Notes in Computer Science, 2016: 231-239.

[35] Di Y, Chao Z, Zhu Z, et al. Trajectory clustering via deep representation learning[C]. // 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017. 3880-3887.
[36] Yao D, Cong G, Zhang C, et al. Computing trajectory similarity in linear time: a generic seed-guided neural metric learning approach[C]. // 2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, 2019. 1358-1369.
[37] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
[38] Ba J, Mnih V, Kavukcuoglu K. Multiple object recognition with visual attention[J]. arXiv preprint arXiv: 1412.7755, 2014.
[39] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv: 1409-0473, 2014.
[40] Bengio Y. Learning deep architectures for AI[J]. Foundations and Trends in Machine Learning, 2009, 2(1):1-127.
[41] Serre T, Keriman G, Kouh M, et al. A Quantitative theory of immediate visual recognition[J]. Progress in Brain Search, 2007, 165:33-56.
[42] Graves A． Long short-term memory[M]. Berlin: Springer, 2012: 1735-1780.
[43] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[J]. arXiv preprint arXiv:1412.3555, 2014.
[44] Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv: 1408.5882, 2014.
[45] Sutskever I, Martens J, Hinton G E. Generating text with recurrent neural networks[C]. // Proceedings of the 28th International Conference on Machine Learning (ICML-11). Bellevue, Washington: DBLP, 2016. 1017-1024.
[46] Mnih V, Heess N, Graves A. Recurrent models of visual attention[C]. // Advances in Neural Information Processing Systems. 2014. 2204-2212.
[47] Ranu S, Deepak P, Telang A D, et al. Indexing and matching trajectories under inconsistent sampling rates[C]. // 2015 IEEE 31st International Conference on Data Engineering. IEEE, 2015. 999-1010.
[48] Su K, Zheng K, Wang H, et al. Calibrating trajectory data for similarity-based analysis[C]. // Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. 2013. 833-844.

作者简历及攻读硕士学位期间取得的研究成果

一、作者简历

王亚珊，女，1996年7月生。2013年9月至2017年7月就读于长沙理工大学电气与信息工程学院电子信息工程专业，取得工学学士学位。2018年9月至2020年7月就读于北京交通大学电子信息工程学院电子与通信工程专业，研究方向是信息网络，取得工程硕士学位。在攻读硕士学位期间，主要从事针对交通轨迹数据的深度学习算法相关的研究工作。

二、发表论文

[1] Wang Y S, Guo Y C, Wei Z R, et al. Traffic flow prediction based on deep neural networks[C]. // 2019 International Conference on Data Mining Workshops (ICDMW), Beijing, China, November 8-11, 2019. 210-215.

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。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。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文数据集

表 1.1： 数据集页

关键词*	密级*	中图分类号	UDC	论文资助
	公开			
学位授予单位名称*		学位授予单位代码*	学位类别*	学位级别*
北京交通大学		10004	工程	硕士
论文题名*		并列题名*		论文语种*
面向轨迹相似度计算的轨迹深度表达学习研究				中文
作者姓名*	王亚珊		学号*	18125063
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西直门外上园村 3 号	100044
专业学位*		研究方向*	学制*	学位授予年*
电子与通信工程		信息网络	2 年	2020 年
论文提交日期*				
导师姓名*	郭宇春		职称*	教授
评阅人	答辩委员会主席*		答辩委员会成员	
电子版论文提交格式 文本（ ） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） 推荐格式：application/msword；application/pdf				
电子版论文出版（发布）者		电子版论文出版（发布）地		权限声明
论文总页数*				
共 33 项，其中带*为必填数据，为 21 项。				