UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Lucas May Petry

A Recurrent Neural Network Approach for Multiple-Aspect Trajectory Classification via Space, Time, and Semantic Embeddings

Florianópolis
2020

Lucas May Petry

A Recurrent Neural Network Approach for Multiple-Aspect Trajectory Classification via Space, Time, and Semantic Embeddings

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.
Orientadora: Prof[a]. Vania Bogorny, Dr.

Florianópolis
2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Petry, Lucas May
A Recurrent Neural Network Approach for Multiple-Aspect Trajectory Classification via Space, Time, and Semantic Embeddings / Lucas May Petry ; orientadora, Vania Bogorny, 2020.
71 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2020.

Inclui referências.

1. Ciência da Computação. 2. Classificação de trajetórias. 3. Trajetória multiaspecto. 4. Redes neurais recorrentes. 5. Codificação Geohash. I. Bogorny, Vania. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Lucas May Petry

A Recurrent Neural Network Approach for Multiple-Aspect Trajectory Classification via Space, Time, and Semantic Embeddings

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Andrea Esuli, Dr.
Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche

Prof. Mauro Roisenberg, Dr.
Universidade Federal de Santa Catarina

Prof. Ronaldo dos Santos Mello, Dr.
Universidade Federal de Santa Catarina

Prof[a]. Ticiana Linhares Coelho da Silva, Dr.
Universidade Federal do Ceará

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Ciência da Computação.

Documento assinado digitalmente
Vania Bogorny
Data: 15/06/2020 14:48:37-0300
CPF: 684.586.570-15

Prof[a]. Vania Bogorny, Dr.
Coordenadora do Programa

Documento assinado digitalmente
Vania Bogorny
Data: 15/06/2020 14:49:11-0300
CPF: 684.586.570-15

Prof[a]. Vania Bogorny, Dr.
Orientadora

Florianópolis, 2020.

To all the students and researchers out there that, like me, are often frustrated with their work, but carry on for the sake of learning.

To my parents and family.

ACKNOWLEDGEMENTS

I would like to thank my parents, Edemar and Rita, for supporting me and my studies since I started my bachelor's degree until now. I want to extend these acknowledgements to my sisters and my grandmothers, and to all my longtime friends.

Second, I would like to say how grateful I am for my supervisor, Dr. Vania Bogorny, and my colleagues at UFSC, André, Andrés, Camila, and Francisco. To my supervisor, thank you for the excellent supervision and for the academic and emotional support. Thank you for the support and for giving me the opportunity to visit Dalhousie University in Canada for a year. Thank you also for the opportunity to meet people from other countries via international collaborations – special thanks to Dr. Andrea Esuli and Dr. Chiara Renso, for your help and research collaboration. To my colleagues at UFSC, thank you for all the interesting and insightful discussions, either about our research projects or random things, and for the research collaboration.

I would also like to show my gratitude to my supervisor at the Institute for Big Data Analytics at Dalhousie University, Dr. Stan Matwin, and to all of my colleagues at the lab, specially Amilcar, Farshid, Fateha, Mahtab, Marzie, and Nader. I learned a lot with you and you made my time in Halifax more fun and enjoyable. Special thanks to Dr. Matwin and Farshid for providing me with a new workstation and a dedicated GPU, through which I ran some of the experiments in this thesis. I want to acknowledge my other friends and colleagues too, Alessandra, Balaji, Cyndi, Fatemeh, Fernando, Inês, Martha, Leonardo, Luiz, Mateus, Nuno, Samuel, Sima, and Vinicius, who I had the pleasure to meet at or through the lab.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo a Pesquisa e Inovação do Estado de Santa Catarina (FAPESC) - Project Match - co-financing of H2020 Projects - Grant 2018TR 1266, and the European Union's Horizon 2020 research and innovation programme under GA N. 777695 (MASTER[1]). A special acknowledgement to Global Affairs Canada (GAC) via the Emerging Leaders in the Americas Program (ELAP) and to the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding my visit to Dalhousie University for a year.

[1] http://www.master-project-h2020.eu/

RESUMO

Com a popularização de redes sociais baseadas em localização na última década e, consequentemente, o enriquecimento semântico de dados de mobilidade em diferentes contextos, um grande volume de dados de trajetórias é gerado diariamente. Diferentemente de dados de trajetória bruta obtidos de dispositivos GPS, trajetórias de redes sociais atreladas à múltiplos contextos são dados mais complexos, pois apresentam diversos atributos textuais com diferentes semânticas, além das dimensões espacial e temporal, o que pode revelar padrões de mobilidade interessantes. Por exemplo, pessoas podem visitar lugares diferentes ou realizar atividades distintas de acordo com as condições climáticas. De forma similar, animais talvez escolham seu habitat baseado nas características climáticas e cobertura de vegetação do lugar. À este novo dado espaçotemporal semanticamente rico se dá o nome de *trajetória multiaspecto*. Este novo tipo de dado complexo traz novos desafios para a classificação de trajetórias, que é o problema tratado nesta dissertação. O problema de classificação de trajetórias consiste na categorização de um objeto móvel de acordo com a sua trajetória. A maioria dos métodos existentes para classificação de trajetórias não suportam dimensões heterogêneas incluindo dados textuais semânticos, não consideram o aspecto sequencial do movimento, e/ou são muito custosos computacionalmente. Neste trabalho é proposto o método MARC, uma abordagem baseada em técnicas de *embedding* de atributos e redes neurais recorrentes, para a classificação de trajetórias multiaspecto, tratando todos aspectos possíveis de trajetórias como espaço, tempo, semântica e sequência. MARC é o primeiro método para classificação de trajetórias que suporta todas as dimensões, incluindo a dimensão espacial. MARC apresenta bom desempenho especialmente quando trajetórias possuem vários atributos textuais/categóricos. Resultados experimentais em quatro conjuntos de dados disponíveis no domínio público mostram que o método proposto obtém valores de acurácia, precisão, cobertura e F1-score superiores ao estado da arte.

Palavras-chave: Classificação de trajetórias. Trajetória multiaspecto. Classificação de trajetórias semânticas. Redes neurais recorrentes. Codificação Geohash.

RESUMO ESTENDIDO

Introdução

Com a popularização de redes sociais baseadas em localização na última década e, consequentemente, o enriquecimento semântico de dados de mobilidade em diferentes contextos, um grande volume de dados de trajetórias é gerado diariamente. Uma trajetória é uma sequência de posições geográficas registradas ao longo do tempo. Diferentemente de dados de trajetória bruta obtidos de dispositivos GPS, trajetórias de redes sociais atreladas à múltiplos contextos são dados mais complexos, pois possuem diversos atributos textuais semânticos além das dimensões espacial e temporal, o que pode revelar padrões de mobilidade interessantes. Por exemplo, pessoas podem visitar lugares diferentes ou realizar atividades distintas de acordo com as condições climáticas presenciadas. De forma similar, animais talvez escolham seu habitat baseado nas características climáticas e cobertura de vegetação do lugar. À este novo dado espaçotemporal semanticamente rico se dá o nome de *trajetória multiaspecto*, trazendo assim novos desafios para a classificação de trajetórias, que é o problema tratado nesta dissertação.

O problema de classificação de trajetórias consiste na categorização de um objeto móvel de acordo com a sua trajetória. Os primeiros métodos propostos na literatura para classificação de trajetórias consideravam apenas trajetórias brutas (LEE et al., 2008; DODGE; WEIBEL; FOROOTAN, 2009; PATEL et al., 2012), suportando somente a dimensão espacial e temporal de trajetórias. Em geral, estes trabalhos propõem a extração de atributos e estatísticas de trajetórias em uma etapa de pré-processamento, os quais são as *features* utilizadas para o treinamento de um modelo de classificação tradicional como árvores de decisão, *Support Vector Machine (SVM)*, entre outros. Outros métodos foram propostos considerando uma rede neural recorrente que utiliza apenas os identificadores dos POntos de Interesse (POIs) visitados pelo objeto móvel (usuário) no contexto de uma rede social (GAO et al., 2017; ZHOU et al., 2018). Além disso, estas abordagens extraem uma representação dos POIs de forma análoga aos *word embeddings* propostos por Mikolov et al. (2013a), considerando somente os identificadores. Além de não considerarem espaço, tempo e outras informações semânticas presentes em trajetórias multiaspecto, o real benefício da utilização de *embeddings* não foi experimentalmente verificado nestes trabalhos. Recentemente, Ferrero et al. (2018) desenvolveu um novo método de busca de padrões sequenciais para a classificação de trajetórias que suporta todas as dimensões de trajetórias multiaspecto, chamado Movelets. Contudo, Movelets é capaz apenas de encontrar padrões sequenciais de pontos consecutivos na trajetória e não consegue ponderar os atributos que melhor descrevem cada classe de trajetórias.

Neste contexto, a hipótese de pesquisa do presente trabalho é de que o processamento sequencial de trajetórias por meio da utilização de redes neurais recorrentes, juntamente com a codificação e utilização de todas as dimensões de trajetórias multiaspecto, pode beneficiar a classificação de trajetórias.

Objetivos

O objetivo principal desta dissertação é propor uma nova abordagem para a classificação de trajetórias multiaspecto. Neste contexto, os seguintes objetivos específicos foram definidos:

- Definir um método que suporte as dimensões espacial, temporal e semântica de trajetórias, comportando todos os atributos que estiverem disponíveis no conjunto de dados;
- Explorar redes neurais artificiais e o uso de técnicas de *embedding* no contexto de processamento de linguagem natural para codificar os diferentes atributos de trajetórias

multiaspecto para classificação.

Metodologia

Neste trabalho é proposto o método MARC, uma abordagem baseada em técnicas de *embedding* de atributos e redes neurais recorrentes, para a classificação de trajetórias multiaspecto, tratando todos aspectos possíveis de trajetórias como espaço, tempo, semântica e sequência. Tendo em vista os objetivos propostos, a metodologia adotada foi composta pelos seguintes passos:

1. Conduzir uma revisão da literatura para classificação de trajetórias, incluindo todos os tipos de trajetória analisados pelos trabalhos existentes;
2. Desenvolver um novo método para classificação de trajetórias multiaspecto com base nas limitações dos métodos existentes;
3. Coletar, organizar e pré-processar conjuntos reais de dados de trajetórias para avaliação experimental;
4. Avaliar experimentalmente a eficiência do método proposto nos conjuntos de dados coletados considerando métricas bem conhecidas, comparando se possível com os métodos existentes;
5. Selecionar e implementar métodos do estado da arte para comparação;
6. Escrever um artigo científico descrevendo o método proposto, a avaliação experimental e os resultados obtidos.

Resultados e Discussão

MARC foi avaliado experimentalmente em quatro conjuntos de dados disponíveis no domínio público, provenientes de *check-ins* de usuários nas redes sociais Brightkite, Gowalla e Foursquare. Na avaliação, o problema de classificação considerado foi o de determinar o usuário que realizou uma dada trajetória, como proposto em Gao et al. (2017). Os quatro conjuntos de dados avaliados possuem entre 193 e 498 usuários, com 3.079 trajetórias no menor conjunto e 20.911 no maior, sendo que cada usuário possui no mínimo 10 trajetórias e cada trajetória tem no mínimo 10 pontos. Três experimentos foram realizados: (1) uma comparação entre MARC e os métodos do estado da arte Bi-TULER (GAO et al., 2017), TULVAE (ZHOU et al., 2018) e Movelets (FERRERO et al., 2018), considerando os quatro conjuntos de dados; (2) uma comparação entre MARC e os métodos anteriores, considerando um problema mais difícil e próximo à realidade que leva em conta a privacidade dos usuários, utilizando os dados da rede social Foursquare; (3) uma análise experimental do impacto de diferentes técnicas de *embedding* de atributos na qualidade dos resultados obtidos pelo método proposto, considerando o problema do experimento anterior. Para os experimentos (1) e (2) uma avaliação estratificada por *holdout* foi realizada, com 1/3 dos dados para teste. Já no último experimento, utilizou-se uma validação cruzada aninhada com 5 partições. Os resultados obtidos mostram que MARC é capaz de reduzir em até 50% o erro de classificação em comparação ao estado da arte no experimento (1), e em até 47% no experimento (2). Na avaliação de diferentes técnicas de inicialização da camada de *embeddings* (3), os resultados obtidos não mostraram uma melhora significativa no desempenho do classificador proveniente da utilização de *embeddings* baseados em *autoencoders* e *word2vec* (MIKOLOV et al., 2013a). Na verdade, tais resultados questionam a utilidade da extração dos *embeddings* de POIs em Gao et al. (2017) e Zhou et al. (2018) para classificação de trajetórias. De modo geral, espera-se que MARC apresente bom desempenho especialmente quando trajetórias possuem vários atributos textuais/categóricos.

Considerações Finais

Neste trabalho foi proposto um novo método para classificação de trajetórias multiaspecto, chamado MARC. MARC suporta todas as dimensões de dados existentes em trajetórias multiaspecto, as quais são codificadas através de uma camada de *embedding*, posteriormente processada por uma rede neural recorrente. MARC utiliza o Geohash para codificar a dimensão espacial e suporta de forma robusta atributos não numéricos de trajetórias. Diferentemente dos trabalhos existentes, o método proposto não é baseado na extração manual de atributos, estatísticas e padrões específicos de trajetórias, sendo esta tarefa atribuída à rede neural artificial. Considerando uma avaliação experimental sobre quatro conjuntos de dados disponíveis no domínio público, MARC se mostrou mais robusto que o estado da arte em termos de acurácia, precisão, cobertura e F1-score.

Embora MARC tenha obtido bons resultados, as maiores limitações do método dizem respeito à interpretabilidade dos resultados e padrões encontrados, e à necessidade de um conjunto de dados de tamanho razoável para treinamento do modelo de classificação. Trabalhos futuros incluem a proposição de métodos que melhorem estes aspectos, além de considerar uma avaliação experimental completa comparando MARC com um método recentemente proposto, MasterMovelets. Ainda, a exploração e avaliação de outras técnicas de *embedding* deveriam ser consideradas em trabalhos futuros, de forma a determinar se estes métodos podem beneficiar de fato a classificação de trajetórias multiaspecto.

Palavras-chave: Classificação de trajetórias. Trajetória multiaspecto. Classificação de trajetórias semânticas. Codificação Geohash. Redes neurais recorrentes.

ABSTRACT

The increasing popularity of Location-Based Social Networks (LBSNs) and the semantic enrichment of mobility data in several contexts in the last few years has led to the generation of large volumes of trajectory data. In contrast to GPS-based trajectories, LBSN and context-aware trajectories are more complex data, having several semantic textual dimensions besides space and time, which may reveal interesting mobility patterns. For instance, people may visit different places or perform different activities depending on the weather conditions and their geographical location. Animals may choose their habitat based on climate and vegetation characteristics. These new semantically rich data, known as *multiple-aspect trajectories*, pose new challenges in trajectory classification, which is the problem that we address in this thesis. Existing methods for trajectory classification cannot deal with the complexity of heterogeneous data dimensions or the sequential aspect that characterizes movement. In this work we propose MARC, an approach based on attribute embedding and Recurrent Neural Networks (RNNs) for classifying multiple-aspect trajectories, that tackles all trajectory properties: space, time, semantics, and sequence. We highlight that MARC exhibits good performance especially when trajectories are described by several textual/categorical attributes. Experiments performed over four publicly available datasets considering the Trajectory-User Linking (TUL) problem show that MARC outperformed all competitors in all datasets, with respect to accuracy, precision, recall, and F1-score.

Keywords: Trajectory classification. Multiple-aspect trajectory. Semantic trajectory classification. Recurrent neural network. Geohash encoding.

LIST OF FIGURES

Figure 1 – Example of the multiple-aspect trajectory of a tourist in the city of Paris. . . 24
Figure 2 – Example of the raw trajectory (left) and equivalent POI sequence (right) of a tourist in the city of Paris. 28
Figure 3 – An RNN architecture for sequence classification. The input sequence is unrolled in the recurrent layer and, in this specific example, the final state of the recurrent layer is forwarded to the output layer. 30
Figure 4 – A CBOW model with only one context word. When more context words are used, an aggregation function such as sum is used for merging the words projections in the vector space. The word embeddings are the lines of the matrix $W_{V \times E}$ (where V is the size of the vocabulary and E is the embedding size) corresponding to the weights of the first layer of the neural network. . . 32
Figure 5 – Example of nested 5×4-fold cross-validation with validation and test sets. . 34
Figure 6 – Overview of the architecture of MARC. 41
Figure 7 – Trajectory point embedding example. 42
Figure 8 – Geohash encoding toy example. 43
Figure 9 – An autoencoder model for trajectory points described by j different attributes. The autoencoder is trained with an input trajectory point p_t, which is encoded into a hidden layer representation, and the goal is to predict the same point p_t. A separate probability distribution is estimated for every attribute in the output of the model. 45
Figure 10 – A Grouped CBOW (G-CBOW) model for trajectory points described by j different attributes, as proposed in this work, with a context window size of 1 (the context of point p_t is given by the previous point p_{t-1} and the following point p_{t+1}). After the concatenation of the embedded attributes of each point in the context, the embedded points are aggregated (via averaging in our work) and forwarded to the output layer. A separate probability distribution is estimated for every attribute of the target point p_t in the output of the model. Although multiple points can be embedded in the input, there is only one physical embedding layer which is used to embed as many points as there are in the input. 47
Figure 11 – Classification accuracy (ACC@1) over training time (Epoch) of compared approaches, considering the datasets with all attributes. 56
Figure 12 – Classification accuracy (ACC@1) over training time (Epoch) of compared approaches, considering the datasets without the POI identifiers. 57

LIST OF ABBREVIATIONS

AI Artificial Intelligence. 30

ANN Artificial Neural Network. 25, 27, 29, 30, 45, 48

Bi-TULER Bidirectional Trajectory-User Linking via Embedding and RNN. 37, 39, 40, 51, 54, 55

CBOW Continuous Bag of Words. 17, 31, 32, 44–46, 59

CNN Convolutional Neural Network. 36, 63

DT Decision Tree. 35, 36

GPS Global Positioning System. 23, 27, 36, 37

GPU Graphics Processing Unit. 49

GRU Gated Recurrent Unit. 30, 36

IJGIS International Journal of Geographical Information Science. 25

LBSN Location-Based Social Network. 23, 28, 29, 37, 39, 51

LSTM Long Short-Term Memory. 27, 30, 46, 48

MARC Multiple-Aspect tRajectory Classifier. 17, 25, 41, 48–51, 53, 55–61, 63

MLP Multilayer Perceptron. 31, 35, 54

NLP Natural Language Processing. 31, 44, 64

PCA Principal Component Analysis. 35, 45

POI Point of Interest. 17, 23, 27–29, 35, 37, 39, 40, 42–44, 46, 51–54

RNN Recurrent Neural Network. 17, 25, 30, 36, 37, 41, 46, 48, 54

SVM Support-Vector Machine. 35

TUL Trajectory-User Linking. 37, 47, 51

TULVAE Trajectory-User Linking via Variational AutoEncoder. 37, 39, 40, 51, 54, 55

VAE Variational Autoencoder. 37

CONTENTS

1	**INTRODUCTION**	**23**
1.1	OBJECTIVE AND CONTRIBUTIONS	25
1.2	METHODOLOGY	26
1.3	SCOPE AND OUTLINE	26
2	**BACKGROUND**	**27**
2.1	FROM RAW TO MULTIPLE-ASPECT TRAJECTORIES	27
2.2	THE PROBLEM OF TRAJECTORY CLASSIFICATION	29
2.3	ARTIFICIAL NEURAL NETWORKS	29
2.4	LONG SHORT-TERM MEMORY (LSTM) UNITS	30
2.5	WORD EMBEDDINGS	31
2.6	EVALUATION OF CLASSIFICATION: METRICS AND TECHNIQUES	32
2.6.1	**Metrics**	**32**
2.6.2	**$k \times (k-1)$-fold nested cross-validation with validation and test sets**	**33**
3	**RELATED WORK**	**35**
3.1	RAW TRAJECTORY CLASSIFICATION	35
3.2	SEMANTIC AND POI SEQUENCE CLASSIFICATION	37
3.3	MULTIPLE-ASPECT TRAJECTORY CLASSIFICATION	38
3.4	SUMMARY OF RELATED WORKS	39
4	**MARC: A METHOD FOR MULTIPLE-ASPECT TRAJECTORY CLASSIFICATION VIA SPACE, TIME, AND SEMANTIC EMBEDDINGS**	**41**
4.1	TRAJECTORY ENCODING	42
4.2	EMBEDDING LAYER INITIALIZATION	44
4.2.1	**Random Initialization**	**45**
4.2.2	**Autoencoder Initialization**	**45**
4.2.3	**Isolated CBOW (I-CBOW) Initialization**	**45**
4.2.4	**Grouped CBOW (G-CBOW) Initialization**	**46**
4.3	RECURRENT AND CLASSIFICATION COMPONENTS	46
4.4	TRAINING AND OPTIMIZATION	47
4.5	ASYMPTOTIC COMPLEXITY ANALYSIS	48
5	**EXPERIMENTAL EVALUATION**	**51**
5.1	DATASETS	51
5.2	EXPERIMENTAL SETUP	53
5.3	COMPARISON WITH THE STATE OF THE ART	54
5.4	RESULTS DISCUSSION	57

5.5 EVALUATION OF EMBEDDING SIZE AND INITIALIZATION TECHNIQUES 59

6 CONCLUSION 63

BIBLIOGRAPHY 65

1 INTRODUCTION

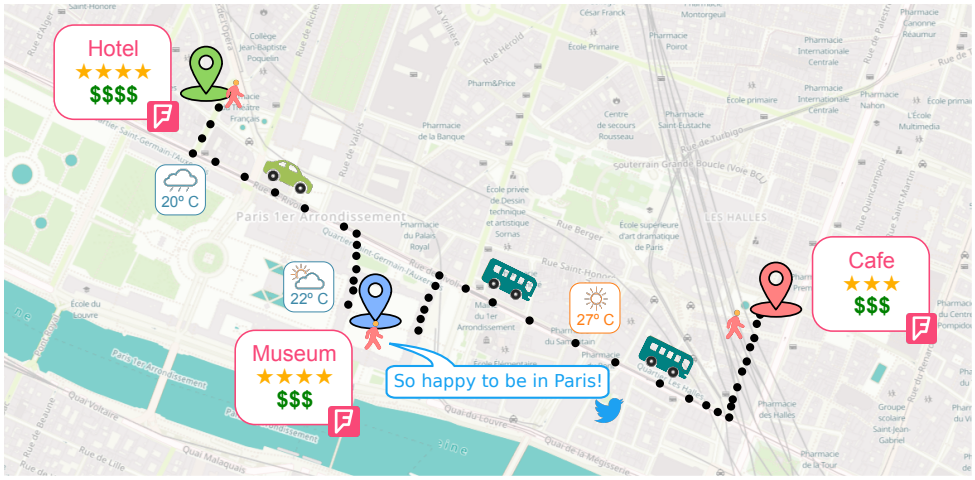
We are witnessing an explosion of big data generated on the internet and an increasing popularity of Location-Based Social Networks (LBSNs), such as Foursquare, Twitter, and Facebook. These networks collect not only information about visited places, feelings, and thoughts of their users, but their real movement as sequences of *check-ins* or geotagged posts, that represent the sample points of a *trajectory*. A trajectory is a sequence of locations recorded over time. We strongly believe that this new type of trajectory data is the challenge of the next-generation data mining methods to be explored by large companies, for obtaining one of the most valuable information: the daily routine patterns and behaviors of every human being.

Semantically rich movement data are important for analyzing mobility patterns in a vast range of applications, from Point of Interest (POI) recommendation (ZHOU et al., 2016; FENG et al., 2018), profiling taxi trip purposes (CHEN et al., 2018), inferring the effect of external factors as the weather conditions on human mobility (BRUM-BASTOS; LONG; DEMŠAR, 2018), to discovering animal behaviors and habitats (GROEVE et al., 2016; GROEVE et al., 2019; TOOR et al., 2016).

In this work, we focus on the trajectory classification problem, which consists of categorizing a moving object according to its trajectories (LEE et al., 2008). In other words, given a set of labels and a set of trajectories, we want to build a model for predicting and assigning such labels to every trajectory in the dataset. Examples of trajectory classification tasks are transportation mode inference (e.g. car, bus, taxi) (DABIRI; HEASLIP, 2018; ETEMAD; JÚNIOR; MATWIN, 2018), inferring the strength level of a hurricane (LEE et al., 2008; FERRERO et al., 2018), predicting the user/owner of a trajectory (GAO et al., 2017; ZHOU et al., 2018), among others.

Trajectory data can be very complex because of the nature of their multiple dimensions. A raw trajectory, for instance, that is generated from a Global Positioning System (GPS) device, and that is the simplest type of movement data, is a sequence of spatio-temporal points in the form of (x, y, t), where x and y represent the spatial position of the moving object at time instant t. Raw trajectories are more complex than time series, since they combine both time and space attributes, in which time is more than the temporal sequence. Trajectory data extracted from LBSNs (GAO et al., 2017; ZHOU et al., 2018) or context-aware trajectories (DODGE et al., 2013; ANDRIENKO; ANDRIENKO; HEURICH, 2011) pose new challenges when compared to raw trajectories. While in raw trajectories a point has basically the space and time dimensions, in trajectories extracted from LBSNs a spatio-temporal point is enriched with several levels of heterogeneous semantic dimensions as the text of the social-media posts, the reviews of a venue, the humor/opinion of the moving object, etc. This new type of trajectory data is called *multiple-aspect trajectory* (FERRERO; ALVARES; BOGORNY, 2016; MELLO et al., 2019; PETRY et al., 2019). Figure 1 shows an example of the multiple-aspect trajectory of a tourist in Paris. Besides the spatial and temporal dimensions, the trajectory is enriched with POI

Figure 1 – Example of the multiple-aspect trajectory of a tourist in the city of Paris.



information (category, rating, and price), the means of transportation, the weather conditions, and the social-media posts of the tourist.

Differently from classical data mining problems, trajectory data cannot be summarized or normalized without information loss, and require a special treatment because of the complexity associated to the data. For instance, the spatial dimension is composed of two attributes, latitude and longitude, and they should be considered as a whole. Earlier methods for trajectory classification (LEE et al., 2008; DODGE; WEIBEL; FOROOTAN, 2009; PATEL et al., 2012) have been developed for raw trajectories, where attributes are derived from the spatial and temporal dimensions in a preprocessing step, such as speed, acceleration, traveled distance, etc. In fact, existing works (LEE et al., 2008; PATEL et al., 2012; FERRERO et al., 2018) do not propose new classifiers, but propose different feature[1] extraction and/or trajectory partition methods that better capture the trajectory discriminant parts.

A recent method, called Movelets, was proposed by Ferrero et al. (2018) as a feature extraction technique for trajectories, supporting multiple attributes as space, time, and semantics. Movelets supports different types of data and heterogeneous dimensions, and it outperformed most previous works for raw trajectory classification. Although it is robust to several dimensions, it only explores patterns of consecutive trajectory points (i.e., subtrajectories) and does not leverage the aspects that better describe each class. In other words, the trajectory patterns found by the method always consider all attributes available. Indeed, it is a time consuming method because it explores all possible subtrajectories of different size.

The works of Gao et al. (2017), Zhou et al. (2018), and Ferrero et al. (2020) have been developed for LBSN trajectory data. Differently from all previous works, Gao et al. (2017) and Zhou et al. (2018) use the idea of word embeddings, and embed POIs based on the concept of the distributional hypothesis, similarly to the idea proposed in Mikolov et al. (2013a). An embedding is a numerical vector in an l-dimensional space $\mathbb{R}^l$, which can be a more meaningful representation of information for machine learning methods.

[1] We use the terms *attribute* and *feature* interchangeably.

In Gao et al. (2017) and Zhou et al. (2018) a given POI is embedded based on the previous and the next POI that the user visited. For instance, suppose a user has visited the POIs *Home, Restaurant, and Office*, while another user has visited the POIs *Home, Cafe, and Office*. The embeddings of *Restaurant* and *Cafe* will be similar, because they happened in the same context (after Home and before Office). However, in these works the embeddings are solely based on the POI sequence, so they neither explore the spatio-temporal dimensions nor the different semantic aspects that characterize multiple-aspect trajectories. We argue that the more trajectory aspects (or dimensions) the classifier is able to treat, the more robust is the method, and that different aspects can contribute for characterizing the behavior of a moving object. For instance, in Brum-Bastos, Long & Demšar (2018) the authors explored the weather effects on human mobility and were able to find different patterns in commuters behavior depending on the weather conditions. Therefore, we define our research hypothesis as follows:

Research Hypothesis: *Trajectory classification can benefit from multiple-aspect trajectories through the embedding and sequential processing of their multiple spatial, temporal, and semantic attributes, especially when multiple-aspect trajectories are described by many textual attributes*.

1.1 OBJECTIVE AND CONTRIBUTIONS

The main objective of this thesis is the proposal of a new approach for multiple-aspect trajectory classification. The specific objectives can be further detailed as:

- Design a method that supports as many trajectory attributes as there may be available, including spatial, temporal, and semantic attributes;
- Exploit Artificial Neural Networks (ANNs) and the use of embedding techniques for encoding the different attributes of multiple-aspect trajectories for classification.

In summary, we make the following contributions:

1. Introduce a new Recurrent Neural Network (RNN)-based classification method for trajectory data, namely Multiple-Aspect tRajectory Classifier (MARC). We model trajectory points via a multi-attribute embedding layer, specifically including an approach for embedding the spatial dimension of trajectories using Geohash encoding (NIEMEYER, 2008). MARC was published in *International Journal of Geographical Information Science (IJGIS)*;
2. Propose and experiment different embedding initialization techniques, in order to evaluate their effectiveness for multiple-aspect trajectory classification;
3. Evaluate MARC on four real-world LBSN datasets enriched with multiple semantic aspects. This allows us to show the robustness of our approach for classifying users based on

their semantically rich trajectories. We compare MARC with state-of-the-art approaches to show that it outperforms competitors in all datasets.

1.2 METHODOLOGY

The methodology used in this work is described as follows:

1. Conduct a review of the literature for trajectory classification, including all types of trajectory data addressed by existing works;
2. Design a new classification method for multiple-aspect trajectories based on the limitations of existing methods;
3. Collect, organize, and preprocess datasets of real trajectory data for evaluation;
4. Evaluate the efficiency of the proposed method through well-known metrics on the collected datasets, comparing if possible with existing methods;
5. Select and implement state-of-the-art methods for comparison;
6. Write an article describing the proposed method, experimental evaluation, and achieved results.

1.3 SCOPE AND OUTLINE

The present thesis is limited to the proposal of a new approach for classifying multiple-aspect trajectories. We assume that trajectories are tagged with a single class/label and, therefore, any segmentation process that may be necessary is outside of the scope of this work, which is done before the training and inference with the classification model. The remainder of this work is organized as follows: in Chapter 2 we introduce basic concepts that are necessary for understanding the subsequent chapters; Chapter 3 describes related works and highlights their limitations for multiple-aspect trajectory classification; Chapter 4 presents the proposed method; Chapter 5 presents the experimental evaluation of our method, including a comparison with the state of the art; and Chapter 6 concludes the present thesis and outlines future work.

2 BACKGROUND

In this chapter, we describe fundamental concepts that are used throughout this thesis. In Section 2.1, we introduce definitions for the different types of trajectory used or introduced in the literature. Afterwards, in Section 2.2 we define the problem of trajectory classification. In subsequent sections, we describe ANNs (Section 2.3), Long Short-Term Memory (LSTM) units (Section 2.4), and word embeddings (Section 2.5), which are important concepts for the understanding of this thesis. Finally, in Section 2.6 we present commonly used metrics and techniques for evaluating methods on the task of data classification.

2.1 FROM RAW TO MULTIPLE-ASPECT TRAJECTORIES

Trajectory data mining has been long studied in the literature, focusing on techniques for pattern discovery, similarity measuring, clustering, and classification. As technology evolved and matured, from GPS devices to smartphones to social networks, the definition of trajectory also changed in order to incorporate different characteristics of the data available. In this section, we describe and present definitions for the different types of trajectory data used or introduced in the literature. The primary type of trajectory studied in most of the previous works is *raw trajectory*, a sequence of points characterized by a pair of spatial coordinates and a timestamp (ALVARES et al., 2007). We define raw trajectory in Definition 2.1, based on the concept introduced in Alvares et al. (2007).

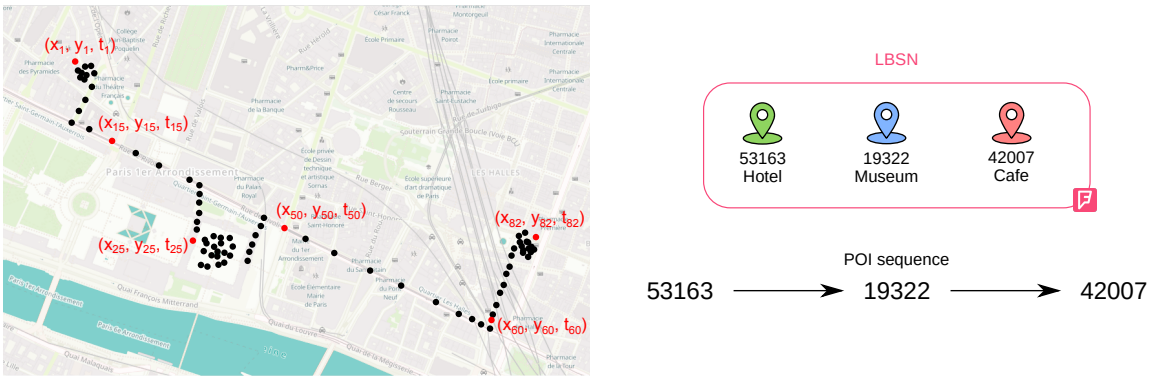
Definition 2.1. *(raw trajectory)* A raw trajectory T is a sequence of space-time points $\langle (x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_m, y_m, t_m) \rangle$, in which $(x_i, y_i) \in \mathbb{R}^2$ and $t_i \in \mathbb{N}$ for $i = \{1, \ldots, m\}$, and $t_1 < t_2 < \cdots < t_m$.

Raw trajectories can be obtained directly from GPS devices, such that the x and y coordinates correspond to longitude and latitude coordinates, respectively. While these data can be generated by any moving object equipped with a GPS device, such as a person, an animal, or a car, raw trajectories may also be mapped from satellite imagery, as in the case of hurricanes or other natural phenomena. Figure 2 (left) shows an example of the raw trajectory of a tourist in the city of Paris.

Spaccapietra et al. (2008) introduced a new model of trajectory named *semantic trajectory*, in which a trajectory is defined as a sequence of *stops* and *moves*. Stops are the parts of the trajectory where the moving object remained still for a period of time, i.e., collections of sample points (x, y, t) that are very close in space that denote a POI. On the other hand, moves correspond to the movement of the moving object between stops, and at the beginning or at the end of the trajectory. The first approach for enriching trajectories with stops and moves was introduced by Alvares et al. (2007).

Another data type that has been commonly used in recent works (GAO et al., 2017; ZHOU et al., 2018) is defined by the sequence of POIs visited by a moving object in the

Figure 2 – Example of the raw trajectory (left) and equivalent POI sequence (right) of a tourist in the city of Paris.



context of a LBSN, which we call *POI sequence*. Figure 2 (right) shows an example of the POI sequence of the same tourist previously mentioned in the city of Paris. Differently from semantic trajectories, POI sequences do not contain the spatial location of the moving object and no guarantee is given for the amount of time that a moving object stayed at a POI. In practice, these trajectories are derived from user check-ins in LBSNs, such as Foursquare[1], Twitter[2], and Facebook[3].

Bogorny et al. (2014) introduced a model for representing trajectories according to several semantic aspects related to movement, such as means of transportation, environmental conditions, places visited by the object (POIs), and events related to the trajectory goal. The challenges related to the analysis of trajectories regarding multiple aspects related to movement were recently discussed in Ferrero, Alvares & Bogorny (2016). More recently, Mello et al. (2019) presented a more general model for trajectory representation, namely *multiple-aspect trajectory*, which includes aspects related to the moving object, to the whole trajectory, to segments of the trajectory, or related to individual trajectory points. In this thesis, for the sake of simplicity, we define multiple-aspect trajectory constraining aspects and attributes to trajectory points only, similar to the definition presented in Petry et al. (2019), which is described in Definition 2.2.

Definition 2.2. *(multiple-aspect trajectory)* Given a set $\mathcal{A} = \{A_1, A_2, \ldots, A_l\}$ of l attribute domains, where $A_i = \{a_1, a_2, \ldots, a_{|A_i|}\}$ are the possible values that the attribute A_i may hold, a multiple-aspect trajectory is defined as a sequence of points $T = \langle p_1, p_2, \ldots, p_m \rangle$, with $p_i = (x_i, y_i, t_i, a_1, a_2, \ldots, a_j)$ being the i-th point of the trajectory composed of a spatial location $(x_i, y_i) \in \mathbb{R}^2$, a timestamp $t_i \in \mathbb{N}$ for $i = \{1, \ldots, m\}$, and j additional characterizing attributes in which $a_i \in A_i$. Additionally, $t_1 < t_2 < \cdots < t_m$.

From Definition 2.2, if $j = 0$ then a multiple-aspect trajectory becomes a raw trajectory. For the sake of simplicity, we abstract the concept of aspects in the definition of multiple-aspect

[1] https://foursquare.com/city-guide
[2] https://twitter.com
[3] https://facebook.com

trajectory, i.e., each of the j attributes are defined in the context of an aspect but such information is omitted. For instance, an aspect may be the means of transportation, the weather conditions, the POI and its characteristics, the person's humor or feelings, among others. The attributes of the aspect means of transportation could be the name (bus, car, foot) and the speed; for weather conditions there could be the temperature and humidity; and the POI could have its category, its rating in a social media network, the price tier, among others.

As already shown in Chapter 1, Figure 1 illustrates the multiple-aspect trajectory of the same tourist in Paris. Notice that some aspects may not exist in some parts of the trajectory. For example, when the user is moving from one place to another, there is no POI information in the trajectory. In our definition, this implies that the attributes corresponding to the aspect POI would receive *null* values while the person is not visiting a POI if data from both the stops and moves of the trajectory were available.

2.2 THE PROBLEM OF TRAJECTORY CLASSIFICATION

Trajectory classification is the problem of categorizing a moving object according to its trajectories (LEE et al., 2008). We define trajectory classification as follows.

Definition 2.3. *(trajectory classification)* Given a set of labels $\mathcal{L}$ and a trajectory set defined by a set of pairs $\mathcal{T} = \{(T_1, \text{label}(T_1)), (T_2, \text{label}(T_2)), \ldots, (T_{|\mathcal{T}|}, \text{label}(T_{|\mathcal{T}|}))\}$, where each pair contains a trajectory T_i and its class label $\text{label}(T_i) \in \mathcal{L}$, trajectory classification is the task of learning a prediction function (i.e., a model) f that maps each trajectory $T_i \in \mathcal{T}$ to one of the class labels in $\mathcal{L}$.

Typical examples of trajectory classification are transportation mode inference (e.g., $\mathcal{L} = \{car, bus, taxi, foot\}$) (DABIRI; HEASLIP, 2018; ETEMAD; JÚNIOR; MATWIN, 2018), determining the strength level of a hurricane (e.g., $\mathcal{L} = \{1, 2, 3, 4, 5\}$) (LEE et al., 2008; FERRERO et al., 2018), identifying the user/owner of the trajectory (GAO et al., 2017; ZHOU et al., 2018), among others.

The type of the trajectory classification problem is closely related to the type of trajectory used to address such problem. For instance, while a model for transportation mode inference is usually trained with raw trajectories, predicting the user of a trajectory in the context of an LBSN has been addressed with POI sequences since only sparse user check-ins are available.

2.3 ARTIFICIAL NEURAL NETWORKS

ANNs are biologically-inspired computational models that, although first studied decades ago (KLEENE, 1956), have only become practical in recent years with the ever-increasing computational power and the proposal of better neuron models and optimization techniques. An ANN can be seen essentially as a stack of layers, in which each transforms the input data with a linear transformation followed usually by a non-linear activation function (e.g.

sigmoid, ReLU). In recent years, much research attention has been given to ANNs with many layers or specifically more than one hidden layer, popularly known as *deep learning*.

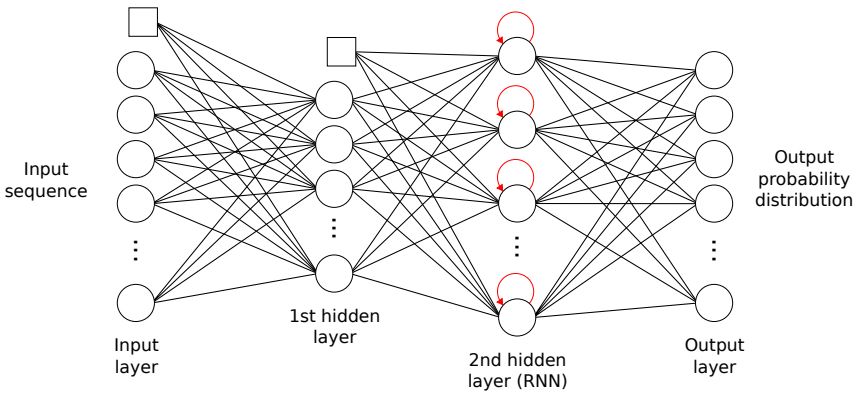
Deep learning has been shown to be a promising approach for many Artificial Intelligence (AI) tasks that are intuitive for humans to perform, but hard to formally describe, such as recognizing objects and understanding speech (GOODFELLOW; BENGIO; COURVILLE, 2016). Goodfellow, Bengio & Courville (2016) explain that the success of deep ANNs lies in the fact that knowledge is built as a hierarchy of concepts, in which the computer learns more complicated concepts out of simpler ones, layer by layer. As a consequence, humans do not have to manually engineer features that might help a computer to solve a specific problem.

A particular type of deep networks of interest is the RNN. RNNs allow the processing of sequential information via recurrent units. Figure 3 illustrates an RNN architecture for classifying an arbitrary input sequence (this is one example as different RNN models can be designed). Unlike regular fully-connected layers (e.g. the 1st hidden layer in Figure 3), recurrent layers process a sequence of inputs and each input affects the subsequent ones. RNNs are used nowadays in a variety of applications, from speech recognition (SAK; SENIOR; BEAUFAYS, 2014) to machine translation to trajectory classification (GAO et al., 2017). The state-of-the-art types of RNN cells for sequence processing are Gated Recurrent Units (GRUs) (BAHDANAU; CHO; BENGIO, 2014) and LSTM units (HOCHREITER; SCHMIDHUBER, 1997).

2.4 LONG SHORT-TERM MEMORY (LSTM) UNITS

LSTM units are a special type of ANN units, proposed by Hochreiter & Schmidhuber (1997), that are capable of processing and learning patterns in sequential inputs. The architecture of LSTM units allow the neural network to loop over sequences, where what is learned for each input in the sequence can be affected by the previous inputs. Such units not only apply a weight to their inputs, but also hold a memory cell and have gates that control how much information is learned and forgotten at a given point of a sequential input. The parameters of LSTM cells can

Figure 3 – An RNN architecture for sequence classification. The input sequence is unrolled in the recurrent layer and, in this specific example, the final state of the recurrent layer is forwarded to the output layer.



be learned in a similar fashion as Multilayer Perceptron (MLP) units are, using an optimization algorithm combined with backpropagation through time.

2.5 WORD EMBEDDINGS

Word embeddings are methods in the Natural Language Processing (NLP) field that map words from a vocabulary (of usually millions or billions of words) to vectors of real numbers in a continuous vector space with a much lower dimensionality. These low-dimensional representations of words have been used to boost the performance of other machine learning tasks, such as machine language translation (ZOU et al., 2013) and sentiment analysis (MAAS et al., 2011). Many word embedding techniques are based on the distributional hypothesis (FIRTH, 1957), which says that words that occur in the same context tend to have similar meanings. Here, the context is usually defined as a fixed-size window around a word, i.e., the words that come before and after a given word. These methods span from count-based approaches as co-occurrence matrices to neural-based models (MIKOLOV et al., 2013a; MIKOLOV et al., 2013b; LE; MIKOLOV, 2014). For instance, considering a corpus of d documents, in a count-based approach an embedding vector for a word w could be defined as a d-dimensional vector where each number in the vector corresponds to the number of occurrences of w in each document. On the other hand, in a neural-based embedding method word vectors are defined by the learned neural network weights.

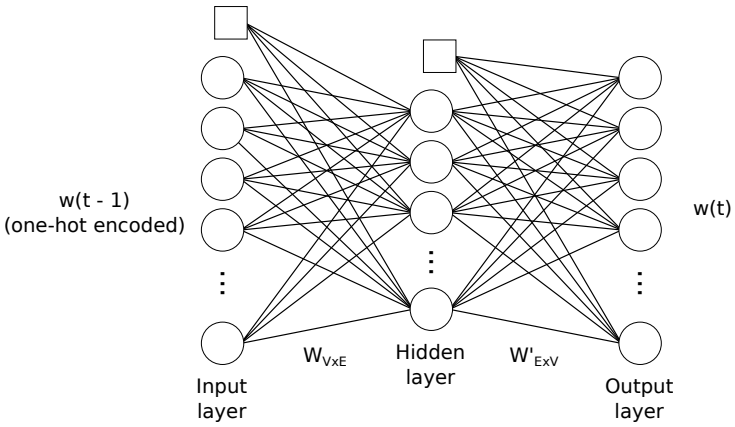
Mikolov et al. (2013a) proposed two neural network models for word embedding, the Continuous Bag of Words (CBOW) and Skip-Gram models (commonly referred to as *word2vec*), which have attracted significant attention from researchers in the NLP community and also from many other areas. Figure 4 illustrates a simple CBOW model. It is a shallow neural network trained to predict a target word given its neighboring words in the text (and vice-versa for the Skip-Gram model), able to capture many linguistic patterns and regularities. The network output is a probability distribution, normalized by a Softmax function (BRIDLE, 1990), as follows

$$\sigma(a)_i = \frac{e^{a_i}}{\sum_{j=1}^{V} e^{a_j}} \tag{2.1}$$

where V is the vocabulary size, $a \in \mathbb{R}^V$ are the outputs from the hidden to the last layer, and $\sigma(a)_i$ is the computed probability for the i-th word in the vocabulary. Such probability distribution is then used in the backpropagation process for updating the weights of the network during training.

Since the size of the vocabulary is usually extremely large, the computation of softmax becomes very costly, because each individual output depends on all the other ones (see Equation 2.1). Hence, Mikolov et al. (2013b) proposed a modification to word2vec named *negative sampling*, where only the expected word output and a small subset of "negative" sampled words (words whose expected probability is zero) are observed during every training iteration. With this approach the authors reported that a model could be trained on a single machine on more

Figure 4 – A CBOW model with only one context word. When more context words are used, an aggregation function such as sum is used for merging the words projections in the vector space. The word embeddings are the lines of the matrix $W_{V \times E}$ (where V is the size of the vocabulary and E is the embedding size) corresponding to the weights of the first layer of the neural network.



than 100 billion words in only one day (MIKOLOV et al., 2013b). The simplicity and flexibility of word2vec resulted in the development of many analogous works for embedding other data objects, from whole documents (LE; MIKOLOV, 2014) to user behavior (CHEN, 2018; ESULI et al., 2018) to genes (DU et al., 2019).

2.6 EVALUATION OF CLASSIFICATION: METRICS AND TECHNIQUES

In this section, we describe the metrics for evaluating classification models and a nested cross-validation technique used in this work.

2.6.1 Metrics

Commonly used metrics for classification evaluation and information retrieval are Accuracy at K (ACC@K), Macro Precision (Macro-P), Macro Recall (Macro-R), and Macro F1 score (Macro-F1) (MANNING; RAGHAVAN; SCHüTZE, 2008). ACC@K shows how well a method correctly estimates the probability of the correct labels for a given set of input trajectories among the K most probable labels. In other words, if the predicted probability for the true label of a trajectory is among the K most probable labels predicted by the model, it is counted as a correct prediction towards the accuracy score. We compute ACC@K as

$$\text{ACC@K} = \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} 1_{T \in \mathcal{L}_K(T)}, \tag{2.2}$$

where $\mathcal{T}$ is the set of trajectories being evaluated and $\mathcal{L}_K(T)$ is the set of K labels with the highest probabilities predicted for trajectory T.

Macro-P and Macro-R are the mean precision and recall among all classes, respectively, computed as follows

$$\text{Macro-P} = \frac{1}{|\mathcal{L}|} \sum_{L \in \mathcal{L}} \frac{TP_L}{TP_L + FP_L}, \tag{2.3}$$

$$\text{Macro-R} = \frac{1}{|\mathcal{L}|} \sum_{L \in \mathcal{L}} \frac{TP_L}{TP_L + FN_L}, \tag{2.4}$$

where TP_L, FP_L, and FN_L are the number of true positives, false positives, and false negatives for class L, respectively. While Macro-P shows the ability of the classifier not to give false positives for each class, Macro-R exhibits its ability to retrieve all relevant trajectories of each class. Macro-F1 is the harmonic mean of Macro-P and Macro-R, averaged across all classes, computed as

$$\text{Macro-F1} = \frac{2 \cdot \text{Macro-P} \cdot \text{Macro-R}}{\text{Macro-P} + \text{Macro-R}}. \tag{2.5}$$

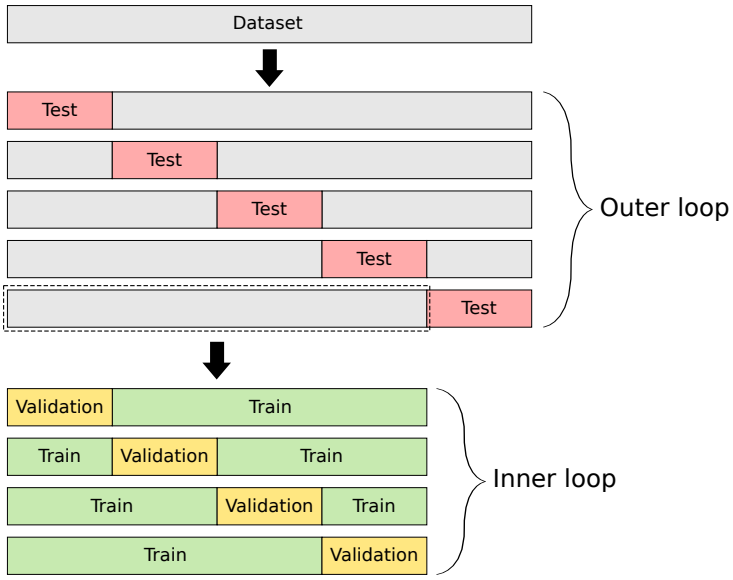
Macro-P, Macro-R, and Macro-F1 do not take the class imbalance into account, so they are useful for measuring the average performance of a classifier among all classes. In contrast to ACC@K scores, these macro scores will be more penalized if certain (even if small) classes are neglected by the classifier.

2.6.2 $k \times (k-1)$-fold nested cross-validation with validation and test sets

Cross-validation is a widely used technique for evaluating how well a model generalizes to previously unseen data. A popular type of cross-validation that is commonly applied is the *leave-one-out* cross-validation. In this setting, a dataset is split into k folds and for k iterations each fold is used as the test split for evaluating the model, while the remaining folders are used for training the model. However, in some cases it is also necessary to estimate which are the best parameters of a model, also known as *hyperparameter optimization*. For this particular case, we apply what is called $k \times (k-1)$-*fold nested cross-validation with validation and test sets*.

Figure 5 exemplifies a 5×4-fold nested cross-validation. Similarly, a dataset is divided into 5 folds and each fold will be the evaluated test set for each iteration of the outer loop. During one iteration of the outer loop, an inner loop iterates over the remaining 4 folds and one by one they are used as the validation set while the other ones are effectively used to train the model. For the specific case of training a neural network model, the validation set is used for selecting the best validated model during training and, afterwards, such model is evaluated on the test set. In the end, there are $5 \times 4 = 20$ test scores for a given evaluation metric.

Figure 5 – Example of nested 5 × 4-fold cross-validation with validation and test sets.



3 RELATED WORK

We describe the related works on trajectory classification according to the type of trajectory that they address: raw trajectories (Section 3.1), semantic or POI sequences (Section 3.2), and multiple-aspect trajectories (Section 3.3). In Section 3.4, we present a brief overview and discussion of related works.

3.1 RAW TRAJECTORY CLASSIFICATION

Most existing works in trajectory classification are limited to raw spatio-temporal data. Such works focus on the extraction and summarization of attributes from trajectories, which are then given as input to traditional classifiers, such as Decision Tree (DT), Random Forest, Support-Vector Machine (SVM), or MLP classifiers. The attributes are extracted from the spatio-temporal points of trajectories, such as the average speed, acceleration, bearing, etc. They can be related to the whole trajectory (global features), and to specific trajectory segments or individual trajectory points (local features).

One of the first works proposed for raw trajectory classification was TraClass (LEE et al., 2008). TraClass uses a grid algorithm to spatially cluster the trajectories of the same class, considering only the spatial dimension of trajectories. For spatial regions containing trajectories of different labels, trajectories are partitioned according to a turning angle threshold, and subsequently clustered using a density-based algorithm. The attributes of each trajectory are the clusters that the trajectory belongs to, which are the input features given to the classifier. Patel et al. (2012) extended the work of Lee et al. (2008) including the time dimension. Rather than just clustering trajectories based on spatial proximity, the authors create duration-based region rules to take into account how much time trajectories stayed within a region.

Dodge, Weibel & Forootan (2009) proposed a method for trajectory classification based on global characterization of trajectories as well as on time-varying profiles of trajectories. First, global descriptive statistics of speed, acceleration, bearing, and traveled distance are used for extracting global movement properties of moving objects. Subsequently, trajectories are decomposed into segments based on attribute deviation from the median and sinuosity values, which are used for attribute computation. As the number of computed attributes can be high and there may exist redundancies among attributes, Principal Component Analysis (PCA) (WOLD; ESBENSEN; GELADI, 1987) is employed for reducing the number of features that are actually given to the classifier. Recently, Xiao et al. (2017) extended the method proposed by Dodge, Weibel & Forootan (2009), extracting a larger number of statistics from global and local features of trajectories.

An approach for transportation mode inference was introduced in Zheng et al. (2010). Zheng et al. (2010) first segment trajectories based on upper bounds for velocity and acceleration, as well as segment length thresholds. Afterwards, attributes such as bearing change rate, stop

rate, and acceleration are extracted from the segments for trajectory classification with a DT.

Soleymani et al. (2014) proposed a cross-scale method for raw trajectory classification, in which trajectories are analyzed from different granularities in both space and time. Three different levels of granularity are considered for the spatial dimension, which in practice means the size of the cells of a defined spatial grid. Features such as time spent in different cells and descriptive statistics are extracted for each trajectory, so as to evaluate the contribution of different scaling levels for the classification task. Similarly, different temporal windows are taken into account for analyzing the temporal scaling of trajectories. Finally, a feature selection algorithm is used for selecting the best spatial and temporal features, which are then given as input to a classifier.

Jiang et al. (2017) proposed an RNN-based approach for point-based transportation mode inference from raw trajectory data, namely TrajectoryNet. Instead of classifying whole trajectory segments, TrajectoryNet assigns transportation modes to each individual GPS point of trajectories, which alleviates the problem of wrong segmentation of trajectories. The RNN architecture of TrajectoryNet consists of two bidirectional layers of GRU units for classifying trajectory points. Another neural-based technique was introduced in Dabiri & Heaslip (2018), using Convolutional Neural Networks (CNNs) for transportation mode classification. Dabiri & Heaslip (2018) transform trajectories into 4-dimensional time series, where the dimensions represent the speed, acceleration, jerk, and bearing rate along the trajectories. Each attribute dimension corresponds to a different input channel in the CNN. For all convolutional layers, a kernel size of 1×3 is used, which basically translates into a sliding window over trajectories.

The extraction of more attributes and descriptive statistics for transportation mode prediction was introduced in Etemad, Júnior & Matwin (2018). From trajectory points, the authors extracted attributes such as speed, acceleration, jerk, bearing, bearing rate, and rate of bearing rate. Afterwards, global trajectory statistics are derived from individual point attributes, including the minimum, maximum, mean, median, standard deviation, and distribution percentiles. Before the data is fed into a classifier, a noise removal step is applied using a median filter based on the average trajectory speed.

An active learning method was proposed by Júnior, Renso & Matwin (2017) for classifying raw trajectories, namely ANALYTiC. The goal of ANALYTiC is to find the minimum set of trajectories that should be labeled to improve the performance of a trajectory classifier, while simultaneously minimizing user annotation effort. In ANALYTiC, a classifier is trained in an iterative manner and, in each iteration, the user is asked to label a small set of trajectories, which are used to train the classifier. Similarly to other works, attributes are extracted from raw trajectories such as speed, bearing, and the traveled distance, as well as statistics such as average, minimum, and maximum values for a trajectory segment.

3.2 SEMANTIC AND POI SEQUENCE CLASSIFICATION

Lee et al. (2011) introduced one of the earliest works for trajectory classification with the use of additional semantic information not extracted from the trajectories alone. Lee et al. (2011) match trajectories against road networks, in order to represent trajectories as sequences of visited road segments rather than individual GPS points. Such representations are used for frequent sequential pattern mining, which can subsequently be given as input features to a classifier. Later, Tragopoulou, Varlamis & Eirinaki (2014) designed a mobile application for transportation mode inference from user GPS movement, also considering additional semantics. Besides the spatial and temporal attributes collected by mobile devices, they used subway stations locations to compute a new attribute representing the distance of the user location to the closest station. In Varlamis (2015), the mobile application was extended to support additional semantics, such as the proximity of trajectories to touristic places, as well as their coincidence with bus lines or train railways. Moreover, an evolutionary algorithm is used to generate new trajectory samples for training classifiers, in order to improve the classification performance of models trained on small datasets.

More recently, Gao et al. (2017) introduced the Trajectory-User Linking (TUL) problem, a trajectory classification task where the labels to be predicted are the users, i.e., the person who generated each trajectory. Gao et al. (2017) proposed Bidirectional Trajectory-User Linking via Embedding and RNN (Bi-TULER), a bidirectional RNN model for classifying LBSN trajectories. However, Gao et al. (2017) analyze only the POIs visited by the users, thus representing trajectories as the sequence of places visited by a user. Bi-TULER learns POI embeddings following the word2vec approach proposed by Mikolov et al. (2013a) (see Section 2.5), but considering trajectories as the sentences or phrases of a text and the POI identifiers as the words, ignoring the space and time dimensions, as well as additional semantics. The pretrained POI embeddings are used to train a bidirectional RNN model for classifying trajectories and, even though it is able to capture more complex patterns of mobility data than previous works, only the POI identifier is used for training the model.

Zhou et al. (2018) extended Bi-TULER and proposed Trajectory-User Linking via Variational AutoEncoder (TULVAE) to address the TUL problem. Similarly to Bi-TULER, embeddings are first learned for encoding the visited POIs using the word2vec approach. However, Zhou et al. (2018) employ a Variational Autoencoder (VAE) architecture to model trajectories in a semi-supervised way, being able to extract interpretable representations of POI dependencies present in trajectories. TULVAE allows the use of unlabeled information that might be available, aiming to improve the performance of the classifier. As Bi-TULER, TULVAE is solely based on the POIs visited by users, thus not supporting the spatial, temporal, and additional semantic attributes of trajectories. Furthermore, the model shows a relatively complex network architecture similar to sequence-to-sequence models and may not exhibit significant improvement compared to Bi-TULER if no additional unlabeled data is available. In fact, the results presented

in the respective paper exhibit improvements of only 1-4% in ACC@1 and F1 score compared to Bi-TULER.

3.3 MULTIPLE-ASPECT TRAJECTORY CLASSIFICATION

To the best of our knowledge, only one work has been proposed and experimentally tested for multiple-aspect trajectories (FERRERO et al., 2020). However, a method was also introduced by Ferrero et al. (2018), namely Movelets, and even though it was only evaluated on raw trajectory datasets, it supports the spatial, temporal, and semantic dimensions of trajectories, thus being able to process multiple-aspect trajectories. Ferrero et al. (2018) explore the training set looking for relevant subtrajectories, which are the trajectory segments that best characterize the movement patterns of a given class, called *movelets*. These patterns can be interpreted as frequent multidimensional sequential patterns of each label/class in the dataset. However, they may have continuous numeric trajectory attributes and are determined by their distance to other trajectories in the dataset, i.e., they are most frequent in trajectories of a class and less frequent in the rest of the dataset. The presence or absence of the discovered movelets in trajectories are then given as the input features to a traditional classifier. For classifying novel unseen trajectories, it is first necessary to compute their distance to all movelets found in the training step, and thereafter they are given to the classifier. Although Movelets supports multiple trajectory attributes, it cannot automatically find those that better discriminate each class, which means that all attributes are always taken into account and have the same weight. Furthermore, Movelets cannot ignore noise in trajectory points, because it only explores patterns of consecutive trajectory points.

The Movelets method was recently extended into MasterMovelets (FERRERO et al., 2020), a novel approach that overcomes the limitation present in Movelets of exploring the trajectory attributes that better discriminate each class. In contrast to the previous approach, MasterMovelets explores subtrajectories characterized by different attribute subsets, finding the best subtrajectory and attribute combination for classifying multiple-aspect trajectories. Besides improving the performance of trajectory classification, MasterMovelets also provides insight on the sequential patterns and specific attributes that best describe the behavior of each class. Similarly to Movelets, MasterMovelets is a parameter-free technique, simply requiring the user to define the distance functions that will be used for comparing each trajectory attribute in the movelet discovery process. Despite its significant improvements in trajectory classification accuracy, MasterMovelets is computationally costly and may become unfeasible depending on the size of the dataset and computational power available.

Recently, a survey and comparison of trajectory classification methods was introduced by Silva, Petry & Bogorny (2019). The authors compare existing methods regarding raw and multiple-aspect trajectories, showing that either Movelets or MasterMovelets outperforms all other works on multiple-aspect trajectory classification.

3.4 SUMMARY OF RELATED WORKS

Table 1 presents a comparison among previous works on trajectory classification. Existing methods are categorized according to the trajectory type for which they have been proposed, the attributes used or extracted by them from raw trajectories and external sources of information, as well as the target classification task.

Existing methods for raw trajectory classification were developed for dense GPS trajectories (e.g. points are regularly sampled every 30 seconds), and are mostly limited to the spatial dimension of trajectories and/or numerical attributes and statistics inferred from space and time. Consequently, these works cannot be easily extended to support any other attributes (mainly semantic attributes) that nowadays can be aggregated to trajectory data, in order to improve the performance of a classification task. Another limitation is that using the space dimension in its raw form can result in overfitting the method to the spatial region of the data used for training and testing. For instance, training a method for transportation mode inference in Paris using solely the latitude and longitude attributes will prevent its use in New York, even though the means of transportation in both cities might follow similar behavior patterns.

Most of existing works were developed for raw trajectory classification, with fewer works considering additional semantics (LEE et al., 2011; TRAGOPOULOU; VARLAMIS; EIRINAKI, 2014; VARLAMIS, 2015), but still to a limited extent. On the other hand, two recent methods were proposed considering a new trajectory representation consisting of the sequence of POIs visited by a person in the context of an LBSN (GAO et al., 2017; ZHOU et al., 2018). Both Bi-TULER and TULVAE embed POIs with the word2vec approach, aiming at alleviating data sparsity and providing meaningful representations of POIs for the classifier. However, they use only the POI identifiers assigned to places in an LBSN for classification. Besides being a limited description of the behavior of a moving object, using these identifiers can also overfit the classification model to the existing locations of an LBSN. For instance, if a new place opens in a city, their classification model will need to be extended and potentially trained further to support visits to that new place. Yet, this limitation could be removed altogether by analyzing characteristics of these places instead of their identifiers.

Only two methods have been proposed for trajectory classification supporting multiple-aspect trajectories, Movelets (FERRERO et al., 2018) and MasterMovelets (FERRERO et al., 2020). Both methods search for discriminative subtrajectories that best characterize each class, but only the latter is able to explore and determine the best trajectory attributes for discriminating each class. However, MasterMovelets is computationally much more expensive, exhibiting a spatial complexity of $O(m \times n^2 \times j)$ and time complexity of $O(m^3 \times n^3 log\ n \times 2^j)$ for training, where m is the number of trajectories, n is the length of the longest trajectory in the dataset, and j is the number of trajectory attributes in the dataset. Later in Section 4.5, we discuss and compare the complexity of MasterMovelets with MARC, which we introduce in the next chapter.

Table 1 – Comparison of the state of the art for trajectory classification according to the type of trajectory addressed, attributes used or extracted from raw trajectories and other sources, and the target prediction (type of labels) of the classification task.

	Work	Attributes used/extracted*	Underlying approach	Target prediction
Raw trajectory	TraClass (LEE et al., 2008)	Longitude, latitude	Feature extraction	Generic***
	Patel et al. (2012)	Longitude, latitude, time	Feature extraction	Generic***
	Dodge, Weibel & Forootan (2009)	Speed, acceleration, bearing, traveled distance	Feature engineering/extraction	Generic***
	Xiao et al. (2017)	Speed, acceleration, bearing, sinuosity	Feature engineering/extraction	Transportation mode
	Zheng et al. (2010)	Speed, acceleration, bearing rate, stop rate	Feature engineering/extraction	Transportation mode
	Soleymani et al. (2014)	Longitude, latitude, time	Feature engineering/extraction	Fish treatment
	TrajectoryNet (JIANG et al., 2017)	Longitude, latitude	RNN	Transportation mode
	Dabiri & Heaslip (2018)	Speed, acceleration, jerk, bearing rate	Feature extraction + CNN	Transportation mode
	Etemad, Júnior & Matwin (2018)	Speed, acceleration, jerk, bearing, bearing rate, rate of bearing rate	Feature engineering/extraction	Transportation mode
Semantic trajectory & POI sequence	Lee et al. (2011)	Longitude, latitude, road segment	Feature extraction	Transportation mode
	Tragopoulou, Varlamis & Eirinaki (2014)	Speed, proximity to subway station, altitude change, time zone, day of the week	Feature engineering/extraction	Transportation mode
	Varlamis (2015)	Speed, proximity to subway station, altitude change, time zone, day of the week, proximity to touristic place, on bus line, on train railway	Feature engineering/extraction	Transportation mode
	Bi-TULER (GAO et al., 2017)	POI identifier	RNN + word2vec	User
	TULVAE (ZHOU et al., 2018)	POI identifier	RNN + seq2seq + word2vec	User
Multiple-aspect trajectory	Movelets (FERRERO et al., 2018)**	Longitude, latitude, time	Feature extraction	Generic***
	MasterMovelets (FERRERO et al., 2020)	Longitude, latitude, time, day of the week, POI identifier, POI category, POI rating, POI price, weather condition	Feature extraction	Generic***
	MARC (this work)	Any	RNN	Generic***

* We do not include aggregation/descriptive statistics derived from segments or the entirety of trajectories.
** Although the technique was only evaluated on raw trajectory datasets, it supports any trajectory attributes regardless of trajectory type.
*** The method can be applied to classification problems in different domains.

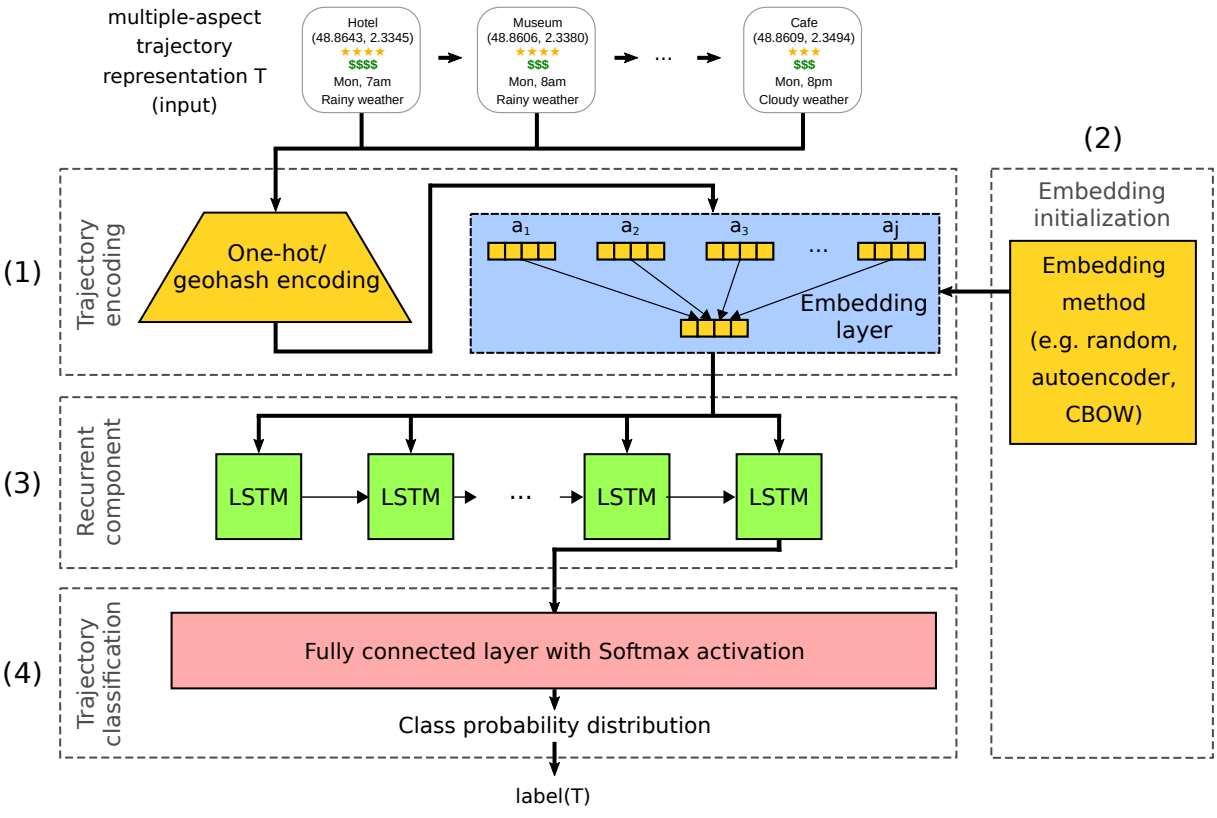
4 MARC: A METHOD FOR MULTIPLE-ASPECT TRAJECTORY CLASSIFICATION VIA SPACE, TIME, AND SEMANTIC EMBEDDINGS

In this chapter, we introduce MARC: a novel method for multiple-aspect trajectory classification. MARC is an RNN that takes a multiple-aspect trajectory as input and outputs the corresponding label (class). As we discussed in Chapter 2, RNNs are a special class of neural networks capable of processing sequences of inputs. As they demand higher computational power than normal feed-forward neural networks, their use in practical applications was only made possible in recent years.

As multiple-aspect trajectories may have several textual attributes, it may be harder for a classifier to measure attribute similarity because of the high number of dimensions in the feature space. Given their heterogeneity and sparsity, we use a multi-attribute embedding layer in order to encode these multiple attributes, which is detailed in the following section. Afterwards, we describe embedding initialization methods for pretraining the embedding layer (Section 4.2), we elaborate on the recurrent and classification components (Section 4.3), give details of the training and optimization of the method (Section 4.4), and discuss the asymptotic complexity of MARC (Section 4.5).

Figure 6 illustrates the components of MARC: (1) trajectory encoding via a multi-attribute embedding layer; (2) an embedding initialization method; (3) a recurrent component for modeling the sequential factor present in trajectories; and (4) the classification component, which uses information from the previous components for assigning labels to trajectories. Our

Figure 6 – Overview of the architecture of MARC.



architecture is more complex than the one proposed in Gao et al. (2017), but much simpler than the one proposed by Zhou et al. (2018).

4.1 TRAJECTORY ENCODING

The first component of our method is responsible for encoding trajectory attributes. Since trajectory attributes can have a variety of formats, we use an embedding layer for uniformly encoding them. Figure 7 illustrates the overall process of encoding a trajectory point. We consider an example of a single trajectory described by the attributes POI, hour, and spatial location, representing the semantics, temporal, and spatial dimensions, respectively. We encode the second trajectory point, which is a visit to a Park at 11am located at the spatial location (40.767667, -73.97334).

The attributes are first one-hot encoded, except for spatial attributes that are encoded with Geohash. The one-hot encoding of an attribute a is a d-dimensional vector of zeros with a 1 in the position corresponding to the value of the attribute, where d is the number of values that the attribute may take. In our example in Figure 7, the dataset has only 6 different POIs (as shown in the legend of the figure), so the one-hot encoded POI vector has 6 dimensions, and Park corresponds to the 5-th element in the vector. In a real trajectory dataset, however, the number of different POIs can be much larger and may be determined by the observations in the dataset or the source from where the data was extracted. Similarly, as there are 24 different hours, the encoded hour is a 24-dimensional vector and the 11-th element corresponds to the hour 11am (hours encoded from 1am to 12pm and 1pm to 12am). Even though the hour is numeric and its raw value could be used as input to our method, we choose to uniformly encode all attributes with one-hot encoding. We leave for future works the investigation of a hybrid approach for dealing with nominal and numeric attributes.

Figure 7 – Trajectory point embedding example.

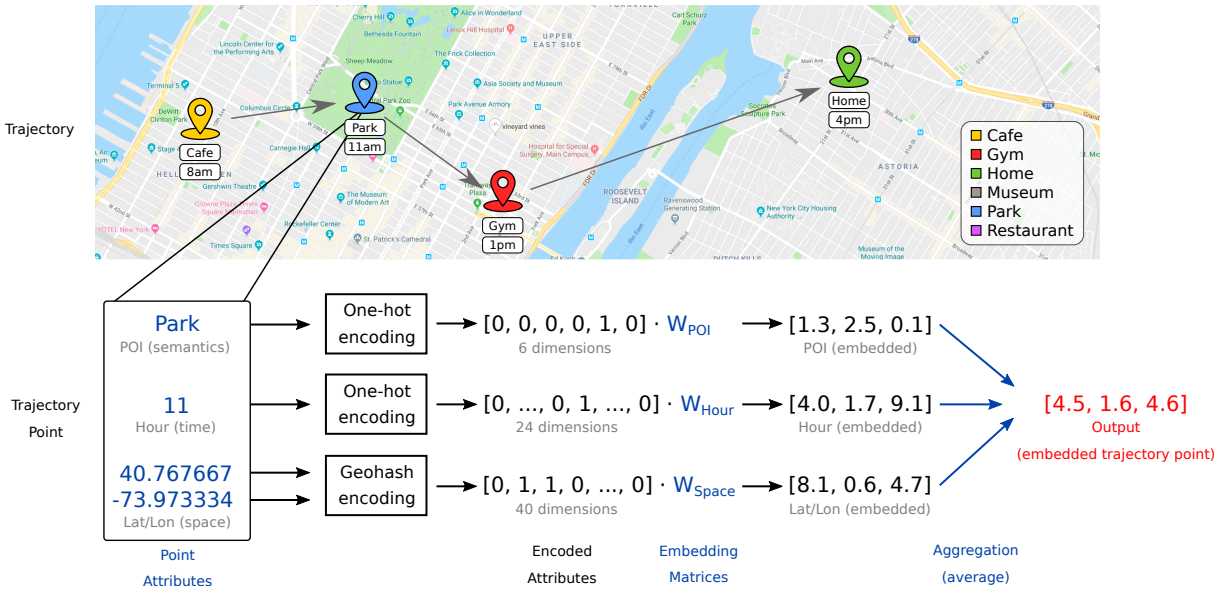
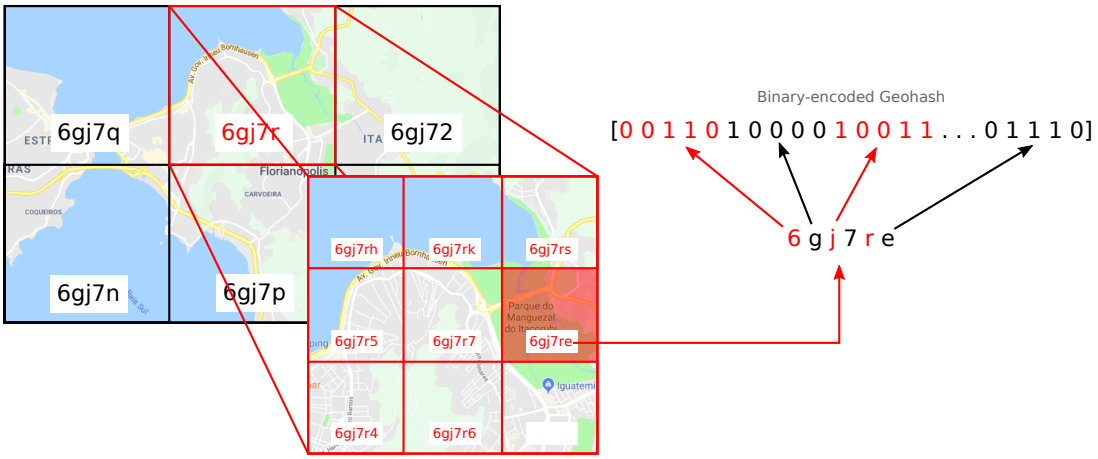


Figure 8 – Geohash encoding toy example.



The spatial dimension of trajectories is usually modeled with two attributes, latitude and longitude, and these attributes are meaningless if considered separately. Therefore, we encode them with the Geohash algorithm (NIEMEYER, 2008), as it is necessary to merge both latitude and longitude into a single meaningful attribute in order to further embed them. Figure 8 illustrates the Geohash algorithm. It successively divides the space into rectangular grid cells, encoding a spatial location (latitude and longitude coordinates) as a Base32 character string. Two locations with a common Geohash prefix are spatially close to each other. We further extract the binary representation of Geohash, and use it as the encoding of the spatial dimension of trajectory points. Since Base32 is used for building the Geohash encoding, each character is mapped to a 5-digit binary string ($2^5 = 32$). For instance, considering 32 characters from 0 to 9 and A to V, 0 is mapped to 00000, 1 is mapped to 00001, 2 to 00010, and so on up to V, which is mapped to 11111. The size of the encoded spatial location depends on the precision chosen for the Geohash algorithm, i.e., how many cells we have in the grid (see Figure 8).

The encoded attributes are multiplied by their respective embedding matrices (W_{POI}, W_{Hour}, and W_{Space} in Figure 7) to extract their corresponding embedded representations. In neural networks, embedding layers map attributes to an embedding space, so that they may be fed to the subsequent layers of the network. Embeddings are numerical vector representations that can be interpreted as points in a continuous l-dimensional space $\mathbb{R}^l$, created according to a model (e.g. vector space models for modeling textual data). Attributes are usually embedded in order to reduce the dimensionality of the underlying space, so that the similarity of attributes can be better measured; or simply to map discrete attributes into equivalent but meaningful representations for machine learning algorithms. For instance, let us consider the POIs Park, Restaurant, and Cafe. A naive way of measuring the similarity of POIs could be comparing their names with a string similarity function such as the Edit Distance (ED) (WAGNER; FISCHER, 1974). In that case, Cafe and Park would have a higher similarity than Cafe and Restaurant, which is not realistic considering the semantics of the POIs. Hence, embedding methods are used to create similarity-based representations for textual attributes. As the number of attributes can grow fast in multiple-aspect trajectories, it may be unfeasible and hard to define similarity

measures for every attribute, as proposed in Ferrero et al. (2018), Ferrero et al. (2020), Furtado et al. (2016), for instance.

Embedding matrices can be learned and initialized via different techniques, which we discuss in Section 4.2. We formally define the embedding of a single attribute as follows.

Definition 4.1. *Attribute embedding*. Given a trajectory $T = \langle p_1, p_2, \ldots, p_m \rangle$ and a set $\mathcal{A} = \{a_1, a_2, \ldots, a_j\}$ of attributes describing p_i, the embedding of attribute a_k is given by $e(a_k) = encoding(a_k)\mathbb{W}_{a_k}$, where $\mathbb{W}_{a_k}$ is the embedding matrix of attribute a_k.

$\mathbb{W}_{a_k}$ is a matrix with $|a_k| \times l$ dimensions, where $|a_k|$ is the number of values that attribute a_k may take. For instance, in the example in Figure 7 we have 6 different types of POIs and decided to embed them in the space $\mathbb{R}^3$, so $\mathbb{W}_{POI}$ has 6×3 dimensions. As we mentioned previously, we use one-hot encoding as the $encoding()$ of nominal and numeric attributes, so that embeddings are properly selected from the embedding matrix. In other words, given an ordering of the values of a_k (i.e. a fixed mapping of POIs to a position in the encoding vector), the $encoding(a_k)$ is built in a way that when multiplied by $\mathbb{W}_{a_k}$, the i-th row of $\mathbb{W}_{a_k}$ is the embedded representation of the i-th value of a_k.

In order to encode trajectory points with multiple attributes, we apply an aggregation function to the embedded point attributes. Attributes must be aggregated so that trajectory points can be fed to the recurrent component of the network. We may combine attributes similarly to how words are aggregated in Le & Mikolov (2014), via element-wise average or concatenation. In Figure 7, we aggregate the embedded attributes by averaging each dimension individually. This implies that if we aggregate attributes by sum or average, all embeddings must have the same size. If we opt for concatenation, then the embedded attributes may have different sizes, and the size of the final encoded trajectory point will be the sum of the embedding sizes. In Section 5.3 we present different results considering the sum, average, and concatenation of attributes.

4.2 EMBEDDING LAYER INITIALIZATION

Word embeddings such as the CBOW model (see Section 2.5) became very popular recently as they have proven to be beneficial to the performance of several NLP tasks. Similarly, Gao et al. (2017) and Zhou et al. (2018) use the CBOW approach proposed by Mikolov et al. (2013a) for obtaining pretrained POI embeddings, and use them as POI representations for training a trajectory classifier. Their approaches, however, were limited to using only POI identifiers and the real benefit of the use of pretrained embeddings was not clear. Given this limitation, in this work we assess different embedding initialization techniques based on random embedding initialization, autoencoders, and CBOW. After learning attribute embeddings through each technique, they are used for initializing the embedding layer of the classification model, i.e., the weights of the embedding layer are replaced by the pre-learned weights. We detail each initialization technique in the next subsections.

4.2.1 Random Initialization

Random initialization consists simply in the random initialization of the weights in the embedding layer of the trajectory classification model. Instead of learning embedding representations for the input attributes and initializing the embedding layer of the classifier with such representations, random numbers are drawn from a distribution and set as the initial weights of the embedding layer. This is analogous to the random initialization of ANN weights, the usual method commonly used for training a neural network from scratch.

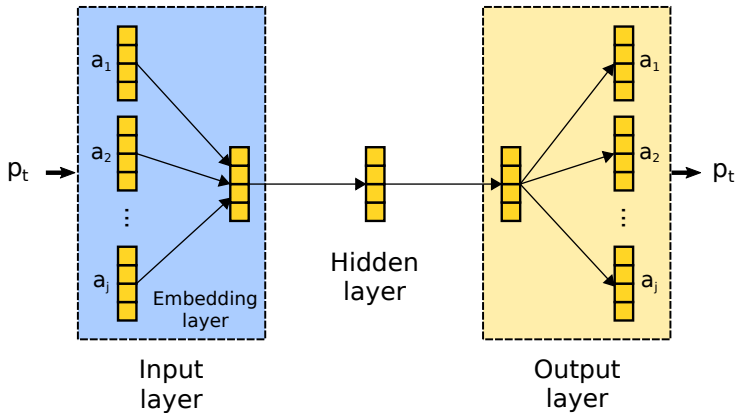
4.2.2 Autoencoder Initialization

Autoencoders are typically used as an unsupervised representation learning technique, usually aiming at dimensionality reduction. A simple autoencoder can be seen as similar to PCA dimensionality reduction, but instead of being a linear transformation like PCA, autoencoders can model non-linear relationships present in the data due to the use of non-linear activation functions. We model an autoencoder for embedding trajectory points, illustrated in Figure 9. Each attribute of trajectory points is embedded and a sigmoid activation function is applied. The concatenated embedded point attributes are then forwarded to the output, which aims at reconstructing the input point by estimating the probability distribution of each of the point attributes. Finally, the autoencoder is trained by minimizing the categorical cross-entropy loss.

4.2.3 Isolated CBOW (I-CBOW) Initialization

This embedding method, which we call Isolated CBOW (I-CBOW), is a simple application of the CBOW method described in Section 2.5 for embedding trajectory attributes individually. Thus, similarly to previous works as Gao et al. (2017) and Zhou et al. (2018),

Figure 9 – An autoencoder model for trajectory points described by j different attributes. The autoencoder is trained with an input trajectory point p_t, which is encoded into a hidden layer representation, and the goal is to predict the same point p_t. A separate probability distribution is estimated for every attribute in the output of the model.



where this embedding technique has been used for embedding POIs, we use it to embed each of the attributes of multiple-aspect trajectories. For I-CBOW, however, the embedding of each attribute is independent from all the other ones. In the next section, we propose a new extended version of CBOW that overcomes this limitation.

4.2.4 Grouped CBOW (G-CBOW) Initialization

We extend CBOW, described in Section 2.5, so that trajectory points described by multiple attributes can be embedded altogether instead of embedding each attribute individually. Figure 10 illustrates this embedding approach, namely Grouped CBOW (G-CBOW). Differently than I-CBOW, in G-CBOW points of multiple-aspect trajectories are embedded with all of their attributes, which means that in the embedding learning process relationships between different attributes may also be encoded.

Although the architecture of G-CBOW is similar to the autoencoder initialization aforementioned (Subsection 4.2.2), in G-CBOW the input is composed of trajectory points that represent the context of a target trajectory point (the output). In addition, linear activation functions are used instead of non-linear ones. Following the same idea of the context window in CBOW (see Section 2.5), trajectory points before and after a target point p_t are embedded and averaged, aiming to correctly predict p_t. Because all the attributes are used in this embedding process, the embeddings learned by G-CBOW may encode the relationships of different attributes across different points according to the size of the context window.

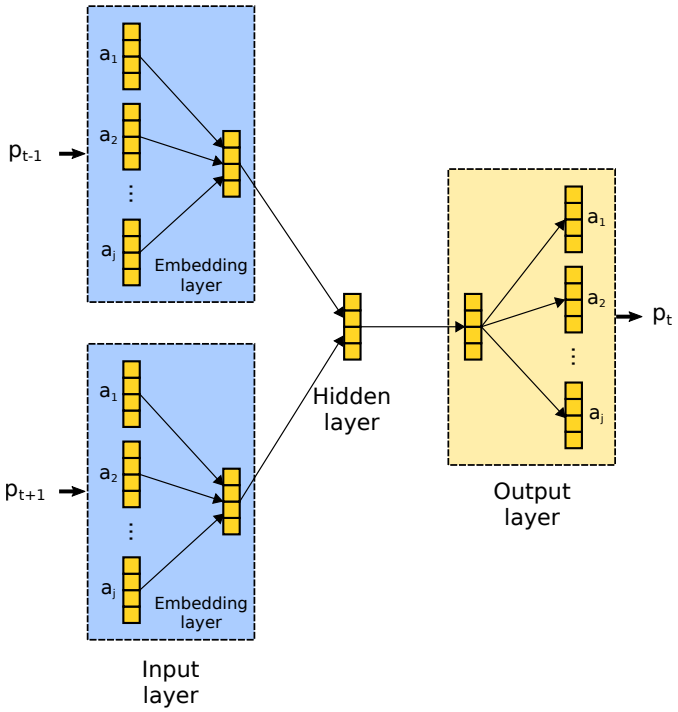
4.3 RECURRENT AND CLASSIFICATION COMPONENTS

To properly assess trajectory data, we use an RNN with LSTM cells (HOCHREITER; SCHMIDHUBER, 1997), which is the state of the art for sequence processing in neural networks. RNNs are able to represent more complex patterns than shallow networks, and can deal with variable-length sequences of information.

After encoding trajectory points, trajectories are fed to the recurrent module. LSTM cells capture patterns in sequences over variable-length time intervals, regulating how much information is remembered via their input, output, and forget gates. Compared to Movelets (FERRERO et al., 2018), for instance, this can be an advantage for modeling trajectory patterns. Movelets only captures sequential patterns of consecutive trajectory points, while LSTM cells may learn patterns that include arbitrary points of a trajectory. For instance, this means that LSTM units can model the relationships between different trajectory points and their attributes, even if they are far away and not necessarily a contiguous sequence.

The last component of our method uses the information provided by the recurrent component for inferring the trajectory label. The output of the recurrent module is fed into a fully-connected layer and subsequently a softmax function is applied. The goal of the last fully-connected layer is to map the learned knowledge to the corresponding label. Then, a

Figure 10 – A Grouped CBOW (G-CBOW) model for trajectory points described by j different attributes, as proposed in this work, with a context window size of 1 (the context of point p_t is given by the previous point p_{t-1} and the following point p_{t+1}). After the concatenation of the embedded attributes of each point in the context, the embedded points are aggregated (via averaging in our work) and forwarded to the output layer. A separate probability distribution is estimated for every attribute of the target point p_t in the output of the model. Although multiple points can be embedded in the input, there is only one physical embedding layer which is used to embed as many points as there are in the input.



softmax function is applied to emphasize the differences between labels, further outputting the probability distribution for all possible data labels. As for some classification problems the number of labels can be particularly high, such as the TUL problem introduced by Gao et al. (2017), negative sampling (MIKOLOV et al., 2013b) can be employed to alleviate the cost of Softmax computation (see Section 2.5).

4.4 TRAINING AND OPTIMIZATION

For training the proposed model, our goal is to minimize the categorical cross-entropy loss, as given by the following equation

$$-\frac{1}{|\mathcal{T}_{train}|}\sum_{T\in\mathcal{T}_{train}}\sum_{L\in\mathcal{L}} 1_{T\in L}\cdot log\ p[T\in L] \tag{4.1}$$

where $\mathcal{T}_{train}$ is the set of trajectories for training the model and $\mathcal{L}$ is the set of labels according to which trajectories are classified. In other words, we want to maximize the probability of our model to correctly predict the label of each trajectory T.

To avoid overfitting our model to the training data, which is a problem inherent to deep neural networks such as RNNs, we use dropout (SRIVASTAVA et al., 2014) and regularization techniques. Dropout layers are applied throughout the model, so units are randomly dropped during the training process. In addition, the weights and biases of the LSTM units are regularized using L1 regularization.

4.5 ASYMPTOTIC COMPLEXITY ANALYSIS

In this section, we briefly analyze the asymptotic complexity of MARC for both space and time, with respect to the number of trajectories (n), the length of the longest trajectory (m), the number of trajectory attributes (j), and the number of training iterations/updates (e). In reality, the complexity of training and inference with a neural network model depends on the number of network weights, which is given in the number of layers and units. However, these are hyperparameters that can be "guessed" or optimized separately, and the overall behavior of MARC changes only with changes on the size of the input trajectories (number, length, and number of attributes).

We first describe the overview of the classification process of MARC in Algorithm 1. Given a trajectory T of length m, MARC embeds the attributes of each trajectory point $T[i]$ (lines 3-5) and then aggregates the embedded attributes into a unified representation (line 6). Subsequently, the embedded point representation x is used to update the recurrent state h with the $lstm$ function (line 7). This process is repeated for all trajectory points (lines 2-9), to obtain the final recurrent state h. Finally, a linear transformation is applied to h followed by a softmax function, in order to obtain the most probable class c (line 10).

The run time complexity of $MARC$ for inference is determined by the two nested loops of Algorithm 1, thus it is O($m \times j$). In terms of space, Algorithm 1 stores a recurrent state h of constant size, as well as a set x of embedded attributes, which is asymptotically linear in the number of attributes j. Therefore, the spatial complexity of MARC for inference is $O(j)$. We encapsulate the parameters and multiplications of the neural network in the functions $embed$, $aggregate$, $lstm$, and $classify$, because such operations have constant time with respect to n, m, and j.

The training process of MARC follows the regular training procedure of an ANN. First, Algorithm 1 (forward pass) is run to obtain the class probabilities of the n trajectories in the training set, which is $O(n \times m \times j)$. Afterwards, the network weights are updated through backpropagation, which is asymptotically linear in the number of weights, even for the parameters of the LSTM cells (HOCHREITER; SCHMIDHUBER, 1997). As the number of weights of MARC grows linearly in the number of attributes, the backpropagation update process is $O(j)$. Given that these steps are repeated for e epochs/iterations, the run time complexity of training MARC is $O(e \times n \times m \times j)$. Similarly, the spatial complexity of training is $O(j)$, determined by the forward pass and the need to store errors/update values for all network weights (HOCHREITER; SCHMIDHUBER, 1997).

Table 2 shows a comparison of the complexity of MARC with the only two approaches in the literature that support multiple-aspect trajectories, Movelets and MasterMovelets. In terms of space complexity, MARC is much superior for training but more costly for inference, taking $O(j)$ in contrast to $O(1)$ of Movelets and MasterMovelets. For training, in comparison with the faster competitor, the difference in the terms for run time complexity of MARC and Movelets is $e \times j$ and $n \times m^2$, respectively. In the datasets evaluated in the experiments, we observe that the number of attributes j is much lower than n and m. However, we cannot draw exact boundaries for e as it depends on the optimization problem, i.e., the classification task and the dataset. Lastly, the run time for classifying a trajectory with MARC is only $O(m \times j)$, which is low compared to the complexities of the other approaches.

Since training a neural network is an optimization problem with no exact convergence guarantees, some aspects of the time taken for training MARC are not reflected in the asymptotic complexity. On the other hand, the constant research and improvements on optimization and training parallelization and acceleration via dedicated Graphics Processing Units (GPUs) make training MARC much faster. In the following chapter, we present an experimental evaluation, showing the robustness of our method in relation to state-of-the-art approaches.

Algorithm 1: MARC Inference

input : trajectory T // $m \times j$
output : class c

$h \leftarrow 0$; $x \leftarrow \emptyset$;
for *i* ***in*** *1 ... m* **do**
  for *k* ***in*** *1 ... j* **do**
    $x \leftarrow x \cup embed(T[i][k])$;
  end
  $x \leftarrow aggregate(x)$;
  $h \leftarrow lstm(x, h)$;
  $x \leftarrow \emptyset$;
end
$c \leftarrow classify(h)$;
return c;

Table 2 – Space and time complexity comparison of MARC and existing approaches for multiple-aspect trajectory classification, with respect to the number of trajectories for training (n), the length of the longest trajectory (m), the length of the longest movelet (m') for Movelets and MasterMovelets, the number of trajectory attributes (j), and the number of training epochs (e) for MARC. We give the complexity for inference considering a single input trajectory.

Method	**Asymptotic complexity for training**		**Asymptotic complexity for inference**	
	Space	Time	Space	Time
Movelets* (FERRERO et al., 2018)	$O(n \times m^2)$	$O(n^2 \times m^3)$	$O(1)$	$O(n \times m^2 \times m')$
MasterMovelets* (FERRERO et al., 2020)	$O(n \times m^2 \times j)$	$O(n^3 \times m^3 log\ m \times 2^j)$	$O(1)$	$O(n \times m^2 \times m' \times j)$
MARC	$O(j)$	$O(e \times n \times m \times j)$	$O(j)$	$O(m \times j)$

* We assume that the asymptotic complexity is determined by the movelet discovery process, which means that the training and inference steps of the classifier used afterwards grow at most in the same order of magnitude stated. We also consider that the number of movelets generated is $O(n \times m)$.

5 EXPERIMENTAL EVALUATION

We evaluate our approach over four real-world trajectory datasets extracted from the Foursquare, Brightkite, and Gowalla LBSNs. These datasets have been widely used in several works on both trajectory classification (GAO et al., 2017; ZHOU et al., 2018) and next POI prediction (ZHAO et al., 2017; FENG et al., 2018).

We compare our work to the state-of-the-art methods that can handle trajectories that have other dimensions than space and time. These works are Bi-TULER (GAO et al., 2017) and TULVAE (ZHOU et al., 2018), developed for POI sequences, and Movelets (FERRERO et al., 2018), that outperformed previous methods developed for raw trajectories. We evaluate existing works using a similar approach to the TUL problem described in Gao et al. (2017), in which the classification task is to predict the corresponding user who generated a given trajectory. MARC was implemented in Python using the Keras[1] and PyTorch[2] frameworks. For reproducibility purposes, we made the source code of MARC available on GitHub.[3]

In the next sections, we describe the datasets used (Section 5.1) and the general experimental setup of our experiments (Section 5.2). In Section 5.3, we present the evaluation results of MARC compared to the state of the art, discussing the achieved results in Section 5.4, and finally in Section 5.5 we evaluate the effect of different embedding sizes and initialization techniques on our method.

5.1 DATASETS

We run the experiments over four datasets extracted from the Foursquare, Brightkite, and Gowalla LBSNs : (i) a dataset of Foursquare check-ins in New York, USA, collected between April 2012 and February 2013 (YANG et al., 2015); (ii) a dataset of Foursquare check-ins in several cities around the world, collected between April 2012 and September 2013 (YANG; ZHANG; QU, 2016); (iii) a dataset of user check-ins around the world on Brightkite, collected between April 2008 and October 2010 (CHO; MYERS; LESKOVEC, 2011); and (iv) a dataset of user check-ins around the world on the Gowalla LBSN, collected between February 2009 and October 2010 (CHO; MYERS; LESKOVEC, 2011). We chose these datasets because they have been used for evaluation in previous works (GAO et al., 2017; ZHOU et al., 2018; FERRERO et al., 2020), and because they either contain or we were able to aggregate information of different aspects.

Tables 3 and 4 describe the attributes of trajectory points in each dataset. In order to show that our approach can handle many dimensions and that additional information contributes to the classification of trajectories, we enrich the original dataset with more information. For the Foursquare datasets, we enriched check-in data with venue information (e.g. price tier and

[1] https://keras.io/
[2] https://pytorch.org/
[3] http://github.com/lucaspetry/msc-thesis-marc

Table 3 – Attributes description of Foursquare NYC and Foursquare Global.

Attribute	Type	Range/example	Unique values
POI	Nominal	{15703, 21580, . . . }	{13,848; 131,155}
Space	Composite	{(38.7423, -90.3658), . . . }	-
Category	Nominal	{Arts & Entertainment, College & University, . . . }	10
Price tier	Numeric	{−999, 1, 2, 3, 4}	5
Rating	Numeric	{−999} ∪ [4.0, 10.0]	62
Hour	Numeric	[0, 23]	24
Day of the week	Nominal	{Monday, Tuesday, . . . , Sunday}	7
Weather*	Nominal	{Clear, Clouds, Rain, . . . }	6
User ID	Nominal	{1, 2, 3, . . . }	{193; 498}

* Available only for the Foursquare NYC dataset.

Table 4 – Attributes description of Brightkite and Gowalla.

Attribute	Type	Range/example	Unique values
POI	Nominal	{19959, 74884, . . . }	{4,913; 24,374}
Space	Composite	{(38.7423, -90.3658), . . . }	-
Hour	Numeric	[0, 23]	24
Day of the week	Nominal	{Monday, Tuesday, . . . , Sunday}	7
User ID	Nominal	{142, 143, . . . }	300

rating) collected from the Foursquare API.[4] In addition, weather information was collected from the Weather Wunderground API[5] and added to each check-in in the Foursquare NYC dataset. We enrich trajectories with information that may affect the movement behavior of a moving object. As we mentioned in Chapter 1, an example of such influence has been shown in Brum-Bastos, Long & Demšar (2018), where different commuter patterns were observed according to different weather conditions. Lastly, the attribute *User ID* is the label of each trajectory in the datasets.

In order to ensure variability and consistency in the evaluation, we applied a few transformations to the datasets. For the Foursquare datasets we removed noisy check-ins belonging to broad categories, such as roads and neighborhoods, because each venue has a unique geographic location. We also removed duplicated check-ins considering a 10-minute threshold. For all datasets, we created weekly trajectories from user check-ins, and we selected only trajectories with at least 10 check-ins, as well as users who have at least 10 weekly trajectories. For the Gowalla and Brightkite datasets we randomly selected 300 users. Table 5 shows the statistics of the curated datasets. As may be observed, the evaluated datasets are heterogeneous, with different sizes, number of classes, and trajectories.

[4] https://developer.foursquare.com/
[5] https://www.wunderground.com/weather/api/

Table 5 – Summary of the datasets (averages reported in the format AVG ± SD).

Dataset	# of points	# of trajectories	# of labels	Avg. traj. length	Avg. # of traj. per class/label
Brightkite	130,494	7,911	300	16.50 ± 7.07	26.37 ± 16.37
Foursquare Global	663,020	20,911	498	31.71 ± 22.05	41.99 ± 13.04
Foursquare NYC	66,962	3,079	193	21.75 ± 14.46	15.95 ± 6.33
Gowalla	98,158	5,329	300	18.42 ± 8.05	17.76 ± 8.20

5.2 EXPERIMENTAL SETUP

In this section, we describe the general setup used in the experimental evaluation of our method. We designed two different experimental scenarios:

1. Evaluation of classification on the datasets with all trajectory attributes available;
2. Evaluation on the datasets without the space dimension and with the POI identifiers generalized to the POI categories.

The goal of this second scenario is to show the robustness of our work, considering a more realistic scenario. This is motivated by the fact that human movement exhibits high spatio-temporal regularity, so users tend to visit a few POIs regularly (GONZALEZ; HIDALGO; BARABASI, 2008). Therefore, the POIs visited by a user, as well as the spatial dimension, are highly discriminative information, and because of privacy concerns, in some situations the exact locations visited by users may not be publicly available. Indeed, as we discussed in Section 3.4, using the specific POI identifiers and spatial locations can cause the overfitting of the classification model. Hence, we consider this second scenario to be representative of a realistic setting, which is harder than the one typically faced in previous literature.

For the two scenarios aforementioned, we run three different experiments:

1. Comparison of MARC with the state of the art on scenario 1 (Section 5.3);
2. Comparison of MARC with the state of the art on scenario 2 (Section 5.3);
3. Evaluation of MARC with different embedding initialization techniques on scenario 2 (Section 5.5).

For all datasets and all methods we report ACC@1, ACC@5, Macro-P, Macro-R, and Macro-F1, as described in Section 2.6. Table 6 describes the parameters and configuration of the data splits, model architecture and optimization of MARC for each of the three experiments. These settings were based on the settings of previous neural-based works (GAO et al., 2017; ZHOU et al., 2018) or defined by preliminary experiments. For the first experiment, we also run a variant of our method, named MARC (Geohash), with only the space dimension, so that we can validate the use of the Geohash representation in the model.

Table 6 – Description of the parameters and hyperparameters for training our model.

	Parameter	Experiment 1/2	Experiment 3
Data splits	Method	Holdout	Nested cross-validation
	Stratified	Yes	Yes
	Train/validation/test ratio	70/0/30	60/20/20
Architecture	Attribute embedding size	100	Rate of input
	Embedding aggregation	{Sum, Average, Concatenation}*	Concatenation
	LSTM layer size	100	100
	Embedding dropout rate	0.5	0.5
	LSTM dropout rate	0.5	0.5
	LSTM regularization	L1(0.02)	-
	Embedding layer initialization	Random	{Random, Autoencoder, I-CBOW, G-CBOW}
Optimization	Optmizer	Adam	Adam
	Learning rate	10^{-3}	10^{-3}
	Learning rate decay	-	-
	Loss function	Categorical cross-entropy	Categorical cross-entropy
	Batch size	64	128
	Number of epochs	1000	1000
	Early stopping patience	30	50
	Early stopping metric	Test loss	Validation loss

* For experiment 2 only concatenation was used.

For Bi-TULER and TULVAE we use the same settings reported in the respective papers. We embed POIs into 250-dimensional vectors and use 300 units for the classifier RNNs. We use 512 units for the encoder-decoder RNN and 100 units for the latent variable z in TULVAE. For Movelets, we experimented several attributes and their combinations and report the best results, achieved by using only the POI identifiers. For the experiments on scenario 2, where the POI identifier is generalized, we consider all trajectory attributes. Additionally, as we run Movelets for multiple-aspect trajectories, we use binary distance for nominal features and euclidean distance for numeric, temporal, and spatial dimensions. We evaluated Movelets with a single-layer MLP classifier with 100 units, Decision Trees and Random Forest, and only the best results (achieved with MLP) are reported. For all networks we use a dropout rate of 0.5, training batch size of 64, and we minimize the categorical cross entropy loss using the Adam optimizer with a fixed learning rate of 10^{-3}.

5.3 COMPARISON WITH THE STATE OF THE ART

Table 7 shows the classification results for the proposed method compared to existing techniques over the four datasets. For each metric, the best result is highlighted in bold and

the second best result is underlined. The results show that MARC systematically outperforms existing methods on all datasets. Movelets is the second best method on Foursquare NYC, Brightkite (tied with Bi-TULER), and Gowalla, which used only the POI identifiers to classify trajectories. Such results show that, indeed, the POIs visited by users are highly discriminative data for trajectory classification. Moreover, the high space complexity of Movelets is shown by the fact that we were not able to run Movelets without filtering the set of discovered patterns on Foursquare Global, the largest dataset evaluated.

Bi-TULER and TULVAE perform poorly on Foursquare NYC (accuracies of 48.20 and 54.33, respectively), but significantly better on Foursquare Global (accuracies of 80.58 and 80.67, respectively), Brightkite (accuracies of 90.64 and 88.41, respectively), and Gowalla (accuracies of 66.15 and 67.94, respectively). Considering the three variations of our method, we observe that aggregating attributes via concatenation (MARC-C) yielded the best results for the majority of the metrics. However, summing (MARC-S) and averaging (MARC-A) attributes also gave great results, on average no more than 1% above or below concatenation scores. As stated in Section 4.1, MARC-C uses embedded representations and weight matrices j times larger than MARC-S and MARC-A (considering the same embedding dimension for all attributes). Considering only the spatial dimension (MARC (Geohash)), we observe that MARC was able to achieve competitive results with the compared approaches.

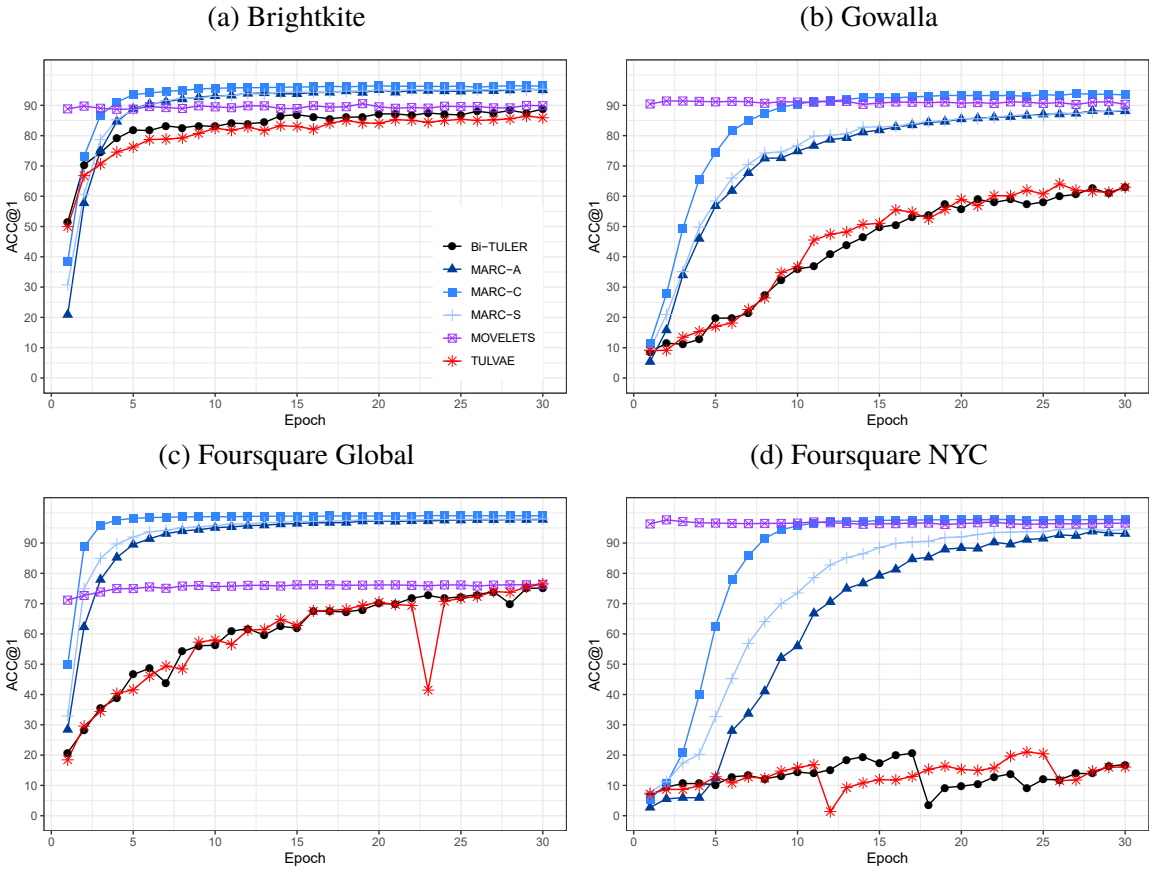
Figure 11 shows the convergence of ACC@1 for all methods on all datasets. All variants of MARC show fast convergence in comparison to existing deep learning methods, Bi-TULER and TULVAE. Both Bi-TULER and TULVAE pre-learn embeddings for POIs in an unsupervised manner, which we claim to be one of the underlying factors for their worse performance. Movelets exhibits fast convergence as well, but it is important to highlight that

Table 7 – Classification results on stratified holdout evaluation, considering all attributes available in the datasets.

Dataset	Metric	Bi-TULER (GAO et al., 2017)	Movelets (FERRERO et al., 2018)	TULVAE (ZHOU et al., 2018)	MARC-S	MARC-A	MARC-C	MARC (Geohash)
Foursquare NYC	ACC@1	48.20	97.66	54.33	97.57	98.05	**98.64**	93.77
	ACC@5	67.38	98.93	73.81	98.93	98.93	**99.03**	97.86
	Macro-P	43.48	96.85	48.66	97.94	97.77	**98.34**	94.33
	Macro-R	41.88	96.67	47.32	96.94	97.50	**98.24**	92.56
	Macro-F1	40.56	96.45	46.54	97.11	97.37	**98.20**	92.75
Foursquare Global	ACC@1	80.58	76.66*	80.67	98.78	98.65	**99.07**	96.67
	ACC@5	90.23	80.38*	90.48	**99.33**	99.20	99.31	98.87
	Macro-P	81.69	75.88*	81.01	98.78	98.64	**99.07**	96.67
	Macro-R	77.57	68.13*	77.60	98.63	98.53	**98.98**	96.24
	Macro-F1	78.31	70.27*	78.07	98.64	98.52	**98.97**	96.28
Brightkite	ACC@1	90.64	90.55	88.41	96.35	96.01	**96.85**	94.92
	ACC@5	95.55	93.79	92.15	98.62	98.62	**98.99**	97.61
	Macro-P	88.98	93.53	85.00	95.97	95.85	**96.76**	93.90
	Macro-R	88.23	87.92	83.86	95.43	95.00	**96.21**	93.19
	Macro-F1	87.92	89.41	83.63	95.15	94.86	**96.17**	93.04
Gowalla	ACC@1	66.15	91.44	67.94	92.36	93.46	**94.33**	88.31
	ACC@5	78.36	94.04	78.76	96.01	97.16	**97.57**	95.31
	Macro-P	67.44	93.24	69.19	92.72	94.11	**94.77**	88.93
	Macro-R	63.21	89.27	64.80	91.25	92.63	**93.28**	86.90
	Macro-F1	63.26	90.25	64.91	91.13	92.43	**93.28**	86.81

* We could not run Movelets without filtering the set of discovered Movelets, because too many are generated.

Figure 11 – Classification accuracy (ACC@1) over training time (Epoch) of compared approaches, considering the datasets with all attributes.



it uses a shallow neural network and there is an extensive feature extraction process before the classification task is performed.

Table 8 presents the results for our second experiment, where the POI identifier was generalized to POI category and the spatial dimension removed. We highlight that this experiment shows a more difficult yet realistic scenario, motivated by privacy concerns about the users information. Similarly to the previous experiment, for each metric the best result is highlighted in bold and the second best result is underlined. We observe that for all methods the accuracy decreased significantly in comparison to the previous experiment in which the detailed information of the POI identifier and the spatial dimension was considered. However, among all approaches, MARC still keeps a significantly higher accuracy when compared to state of the art.

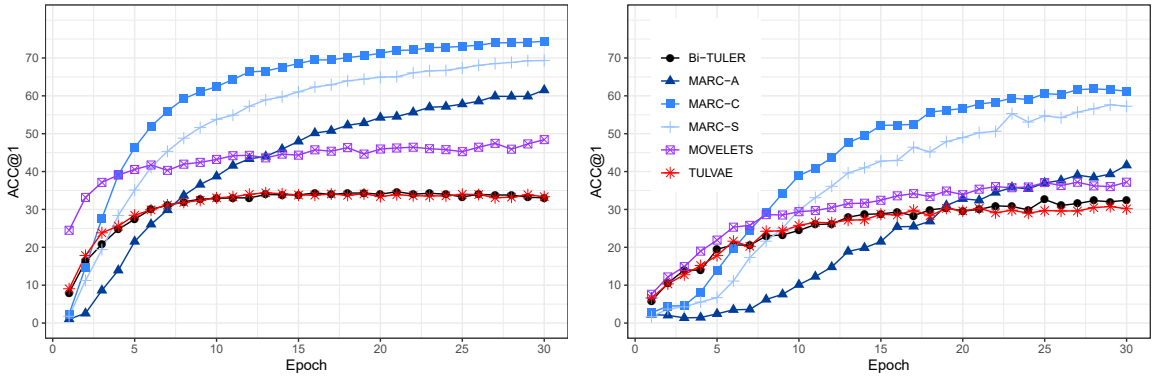
Figure 12 shows the convergence of ACC@1 for all methods in this new scenario. MARC achieves accuracies lower than the ones of Bi-TULER and TULVAE for the first few epochs, because while they use pre-trained embeddings, we train an end-to-end task. Afterwards, our models continue to learn whereas both Bi-TULER and TULVAE converge to an accuracy of about 33%. Movelets also exhibits much slower convergence compared to the previous experiment, which suggests that the discovered patterns are not as discriminating as they were before. In summary, the results of the experiments show that MARC is more robust than

Table 8 – Classification results on stratified holdout evaluation, considering the datasets without the POI identifiers.

Dataset	Metric	Bi-TULER (GAO et al., 2017)	Movelets (FERRERO et al., 2018)	TULVAE (ZHOU et al., 2018)	MARC-S	MARC-A	MARC-C
Foursquare NYC	ACC@1	33.50	41.58	32.81	74.10	70.59	**74.87**
	ACC@5	60.76	67.96	61.15	**88.32**	87.83	88.12
	Macro-P	29.58	38.76	29.55	73.16	70.83	**75.19**
	Macro-R	29.70	36.93	29.52	71.28	67.34	**72.23**
	Macro-F1	28.29	35.11	28.14	70.21	66.59	**71.22**
Foursquare Global	ACC@1	34.70	51.88	34.40	**81.97**	81.47	81.55
	ACC@5	60.46	73.74	62.01	92.73	**92.74**	92.28
	Macro-P	34.22	54.64	33.64	82.47	**82.55**	82.20
	Macro-R	33.43	49.96	32.96	**80.53**	80.16	80.34
	Macro-F1	32.41	49.19	31.58	**80.63**	80.40	80.46

Figure 12 – Classification accuracy (ACC@1) over training time (Epoch) of compared approaches, considering the datasets without the POI identifiers.

(a) Foursquare Global (b) Foursquare NYC



existing approaches, and good classification accuracies can be achieved after only a few epochs of training.

5.4 RESULTS DISCUSSION

In Section 5.3, we showed that the proposed classification method, MARC, outperforms existing approaches in all datasets. Due to the lack of multiple-aspect trajectory datasets from other domains, our evaluation was constrained to the TUL problem with LBSN data, as this problem has been consistently addressed by previous works in the literature (GAO et al., 2017; ZHOU et al., 2018; PETRY et al., 2019).

In the first experiment (Table 7), Bi-TULER and TULVAE performed poorly on the Foursquare NYC dataset, but significantly better on the other ones. We conjecture that these results are related to (1) the larger size of these datasets and (2) the geographic distribution of check-ins. Check-ins in the Foursquare Global, Brightkite, and Gowalla datasets are distributed around the globe, so the data is inherently more discriminative than in the Foursquare NYC dataset. Moreover, although MARC-C was the best performing variant of our method on all datasets, for applications with memory/storage constraints MARC-S and MARC-A are the best

alternatives, since the sum and average of attributes results in fewer parameters (weights) in the neural network. Additionaly, the high scores achieved by MARC (Geohash) show that the Geohash representation can successfully preserve the discriminative power of the spatial dimension in neural networks.

In the second experiment (Table 8), we presented results considering a harder and more realistic scenario in which very sensitive information of the datasets is not available (visited POIs and spatial locations). Instead of relying on the specific locations and POIs visited by users, MARC leveraged user preference information (rating and price tier of the visited POIs) and daily habits in order to discriminate between different users. In this scenario, the accuracy of the other methods decreased substantially in comparison with the previous experiment, while MARC was still able to perform classification with a high accuracy. We explain the poor performance of previous works with two major aspects of this experiment. First, as we observed in the previous experiment, the specific POI identifiers play a big role in discriminating user trajectories. Bi-TULER and TULVAE consider only the POI attribute, which was replaced by the POI category for this experiment. Since they do not consider multiple trajectory attributes, their performance decreased substantially, achieving 33.50 and 32.81 accuracy on Foursquare NYC, and 34.70 and 34.40 on Foursquare Global, respectively. Second, classifiers must be able to leverage between the remaining attributes in order to correctly classify users based on their trajectories.

Although the Movelets technique supports multiple attributes, it always considers all attributes when looking for trajectory patterns, therefore the less discriminant dimensions will then add noise to the pattern, as Movelets is not able to find the best dimensions for classification problems. Movelets achieved 41.58 and 51.88 accuracy scores on Foursquare NYC and Global, respectively, and performed better than Bi-TULER and TULVAE for all evaluated metrics on both datasets. The method should perform better by using only the spatial dimension as it is a more discriminant feature. However, by embedding multiple attributes and modeling complex sequential patterns with MARC, we were able to achieve much higher accuracy and F1 scores than existing approaches (between 70% and 92%).

MARC is generic enough to deal with different trajectory classification problems, such as transportation mode inference, predicting the profile of a person, etc, as highlighted in Chapter 1. However, we expect our method to perform better in problems with more textual/categorical attributes. This is because attributes are one-hot encoded (i.e. discretized), thus the precision of numerical attributes may be affected as different values of the attribute may have the same one-hot encoding. In that case, methods that explicitly define distance functions for numerical attributes (e.g. Ferrero et al. (2018)) will most likely perform better than MARC. For example, the proposed approach would probably perform well at predicting the profile of a person based on qualitative attributes of the person, but would not be as good at predicting the transportation mode of a trajectory based on speed and direction information.

From a high-level point of view, the results show that MARC can be a useful tool for other important tasks that affect our daily lives. Regarding social aspects, correlating trajectories

with their users allows for a better understanding of the movement patterns of users, as well as identifying user profiles for making more personalized recommendations. From a security point of view, identifying the user of a given trajectory may be helpful, for instance, in identifying criminals or terrorists (GAO et al., 2017). Even though in LBSNs the users may be already known, the classification model may assist in the detection of hacked user profiles by identifying abnormal behavior (through the analysis of the misclassifications made by the model).

Lastly, we did not experimentally compare MARC with MasterMovelets (FERRERO et al., 2020) because it was developed by our group simultaneously to this work. However, we conjecture, based on preliminary experiments, that MasterMovelets is able to achieve slightly better classification accuracies compared to our method. On the other hand, MARC is considerably faster and can better benefit from parallelism for training and inference.

5.5 EVALUATION OF EMBEDDING SIZE AND INITIALIZATION TECHNIQUES

We now evaluate the effect of different embedding sizes and initialization techniques on the performance of MARC. For this experiment we use only the generalized Foursquare datasets, as they represent a harder scenario and thus we expect to better visualize the effects of the embedding parameters. We perform a 5×4-fold nested cross-validation and the models were implemented using the framework PyTorch[6,7].

We set the embedding sizes as a compression rate of the input size, i.e., we set the embedding size of each attribute proportionally to its encoded input size. Thus, we experimented with embedding compression rates of 0.25, 0.5, 0.75, 1, 2, and 5. Table 9 shows the absolute embedding sizes and model sizes for each dataset. Attributes were aggregated through concatenation as it was the best performing aggregation method in previous experiments. We compare random initialization of embeddings, pretraining with an autoencoder, and pretraining with G-CBOW and I-CBOW, as described in Section 4.2. For autoencoder and CBOW, we show results with the pretrained embedding layer both frozen (i.e., it does not change in the classifier training) and trainable.

[6] https://pytorch.org/

[7] A different framework was used in this experiment for learning purposes and for validating the model implementation in a different framework.

Table 9 – Embedding sizes and model sizes in number of parameters, for each dataset and input compression factor.

	Dataset	**Input compression/expansion factor**					
		0.25	0.50	0.75	1.00	2.00	5.00
Embedding size	Foursquare NYC	27	54	81	108	216	540
	Foursquare Global	25	52	79	104	208	520
Model size	Foursquare NYC	72,072	83,855	95,636	107,415	154,537	295,903
	Foursquare Global	102,093	113,956	125,817	136,812	182,526	319,668

For pretraining the embeddings with autoencoder and CBOW, the models were trained for 500 epochs with an initial learning rate of 0.025 and a minimum learning rate of 0.0001, which was decayed by a factor of 0.1 every time the model reached a plateau for 10 consecutive epochs. A batch size of 1000 was used and training was early stopped if the model loss did not show improvement for 20 epochs. Both the train and validation splits were used for learning the embeddings, and for CBOW-based approaches a context window of size 3 was used, since it worked best in preliminary experiments. For training the trajectory classification model MARC with the pretrained embeddings, the same configurations described in Section 5.2 were used.

Table 10 and Table 11 present the ACC@1 and Macro-F1 scores, respectively, of all evaluated techniques and embedding sizes. The best and second best scores are bolded and underlined, respectively, for each dataset and input compression factor evaluated. Overall, the results are consistent across embedding techniques, with an upward trend up to a compression factor of 1.00, followed by a saturation of the scores for some of the techniques. While I-CBOW and Random were the best performing techniques for Foursquare Global, on Foursquare NYC the best results were achieved mostly with the use of an Autoencoder. Regardless of embedding size, in terms of ACC@1 the best result on Foursquare Global was achieved by I-CBOW (74.74), and on Foursquare NYC by Random (74.70).

It is interesting to highlight that while Autoencoder was the best technique on Foursquare NYC, it exhibited very poor performance on Foursquare Global. This significant difference may have been caused by an overfitting of the Autoencoder and thereafter poor generalization to the test data. In addition, its good performance on Foursquare NYC may be due to the small size of the dataset. Foursquare NYC is almost 10 times smaller than Foursquare Global in number of trajectory points and it is well known that deep learning models perform better when trained with

Table 10 – Average ACC@1 scores of 5 × 4-fold nested cross-validation of classification, reported in the format AVG ± SD. Embedding techniques were used to initialize the embedding layer of MARC and were either trainable or kept frozen during model training.

Dataset	Embedding technique	Embedding layer	Input compression/expansion factor (embedding size)					
			0.25	0.5	0.75	1.00	2.00	5.00
Foursquare Global	Autoencoder	Frozen	10.60 ± 3.15	4.84 ± 1.55	52.16 ± 7.87	65.11 ± 2.09	68.03 ± 1.29	50.94 ± 3.06
		Trainable	21.82 ± 6.52	41.82 ± 8.38	62.59 ± 4.08	69.42 ± 1.48	71.73 ± 1.04	59.45 ± 2.72
	G-CBOW	Frozen	56.85 ± 2.58	70.00 ± 0.72	71.62 ± 0.94	71.90 ± 0.76	72.32 ± 0.89	71.99 ± 0.97
		Trainable	**61.28** ± 2.29	70.49 ± 0.93	72.19 ± 0.92	72.80 ± 0.64	73.46 ± 0.85	73.85 ± 0.95
	I-CBOW	Frozen	39.61 ± 5.93	67.50 ± 2.10	71.30 ± 0.66	72.31 ± 0.99	73.51 ± 0.80	74.00 ± 0.82
		Trainable	58.56 ± 5.60	71.00 ± 0.90	**72.75** ± 0.98	73.24 ± 0.83	74.32 ± 0.65	**74.74** ± 0.82
	Random	Trainable	61.04 ± 3.10	**71.19** ± 0.88	72.70 ± 0.84	**73.27** ± 0.83	**74.34** ± 0.79	74.65 ± 0.83
Foursquare NYC	Autoencoder	Frozen	68.07 ± 3.25	72.70 ± 2.28	73.65 ± 1.99	73.83 ± 2.03	**74.44** ± 1.95	74.40 ± 1.77
		Trainable	69.58 ± 3.19	**73.35** ± 2.02	**74.22** ± 1.79	**73.96** ± 2.02	74.38 ± 2.06	74.65 ± 2.01
	G-CBOW	Frozen	69.07 ± 2.20	72.31 ± 2.50	72.80 ± 2.06	72.84 ± 2.11	72.55 ± 2.34	72.47 ± 1.99
		Trainable	**69.80** ± 2.20	72.31 ± 2.23	72.72 ± 1.97	72.77 ± 2.32	72.58 ± 1.73	72.56 ± 2.20
	I-CBOW	Frozen	63.87 ± 3.19	71.19 ± 2.22	72.46 ± 2.23	72.99 ± 2.22	73.82 ± 1.64	74.35 ± 1.82
		Trainable	69.64 ± 2.38	72.54 ± 2.13	73.20 ± 1.94	73.35 ± 1.98	74.03 ± 1.99	74.47 ± 1.82
	Random	Trainable	69.00 ± 2.78	72.67 ± 1.94	73.16 ± 1.89	73.49 ± 1.90	73.93 ± 2.12	**74.70** ± 1.98

Table 11 – Average Macro-F1 scores of 5 × 4-fold nested cross-validation of classification, reported in the format AVG ± SD. Embedding techniques were used to initialize the embedding layer of MARC and were either trainable or kept frozen during model training.

Dataset	Embedding technique	Embedding layer	Input compression factor (embedding size)					
			0.25	0.5	0.75	1.00	2.00	5.00
Foursquare Global	Autoencoder	Frozen	6.93 ± 2.63	2.34 ± 1.04	43.74 ± 7.17	55.31 ± 1.80	57.83 ± 1.17	42.32 ± 2.89
		Trainable	16.55 ± 5.87	34.21 ± 7.69	53.09 ± 3.63	59.07 ± 1.30	60.94 ± 1.01	50.04 ± 2.51
	G-CBOW	Frozen	47.85 ± 2.40	59.41 ± 0.80	60.85 ± 1.01	61.00 ± 0.85	61.36 ± 0.99	61.07 ± 1.00
		Trainable	51.66 ± 2.23	59.70 ± 0.99	61.18 ± 1.03	61.72 ± 0.78	62.28 ± 0.98	62.53 ± 1.05
	I-CBOW	Frozen	32.97 ± 5.26	57.43 ± 1.74	60.62 ± 0.80	61.45 ± 1.08	62.48 ± 0.90	62.82 ± 0.88
		Trainable	49.43 ± 5.06	60.25 ± 0.99	**61.77** ± 1.06	62.16 ± 0.91	63.02 ± 0.77	**63.42** ± 0.94
	Random	Trainable	**51.67** ± 2.75	**60.40** ± 0.91	61.66 ± 0.97	**62.17** ± 0.95	**63.06** ± 0.88	63.33 ± 0.91
Foursquare NYC	Autoencoder	Frozen	59.67 ± 3.39	64.15 ± 2.60	64.95 ± 2.24	65.11 ± 2.46	**65.71** ± 2.29	65.67 ± 2.15
		Trainable	60.96 ± 3.39	**64.70** ± 2.36	**65.48** ± 2.14	**65.31** ± 2.46	65.60 ± 2.51	**65.83** ± 2.39
	G-CBOW	Frozen	60.45 ± 2.62	63.63 ± 2.95	64.14 ± 2.33	64.22 ± 2.54	64.01 ± 2.56	63.71 ± 2.33
		Trainable	61.05 ± 2.46	63.54 ± 2.60	63.94 ± 2.47	63.98 ± 2.60	63.94 ± 2.17	63.93 ± 2.64
	I-CBOW	Frozen	55.65 ± 3.38	62.60 ± 2.71	63.69 ± 2.64	64.40 ± 2.60	65.06 ± 2.13	65.76 ± 2.28
		Trainable	**61.10** ± 2.74	63.88 ± 2.61	64.45 ± 2.29	64.70 ± 2.50	65.30 ± 2.38	65.71 ± 2.29
	Random	Trainable	60.43 ± 2.90	63.91 ± 2.40	64.57 ± 2.41	64.59 ± 2.31	65.09 ± 2.57	65.81 ± 2.43

large volumes of data (GOODFELLOW; BENGIO; COURVILLE, 2016). Therefore, the pretraining of the embeddings with Autoencoder contributed, even if only a little bit, to improving the performance of MARC.

Another point worth mentioning for G-CBOW and I-CBOW is that, for the majority of the results, allowing further training of the pretrained embeddings (i.e. embedding layer is trainable) yields better results compared to when they are kept frozen. Such results suggest that the previously proposed methods, Bi-TULER and TULVAE, could achieve better results if they allowed the pretrained POI embeddings to be further trained in the classifier. In fact, our results challenge the usefulness of pretrained embeddings for trajectory classification as proposed by Gao et al. (2017) and Zhou et al. (2018), since Random initialization exhibits similar or even better accuracy results.[8]

In summary, the results indicate that embedding initialization techniques other than Random may be useful for smaller datasets. However, random initialization performed consistently well across both datasets and is less costly than all other techniques, because it does not require the training of another machine learning model before the training of the classifier.

[8] No baseline was reported in their works (GAO et al., 2017; ZHOU et al., 2018) considering the random initialization of the classifier and no pretraining of embeddings.

6 CONCLUSION

In this work we presented a new method, named MARC, for classifying multiple-aspect trajectories. Our method focuses on the different spatial, temporal, and semantic features that characterize multiple-aspect trajectories, and a multi-attribute embedding layer is used to encode these heterogeneous dimensions. We leave to the neural network the task of learning abstract features and sequential patterns that are present in trajectory data. We use Geohash for encoding the spatial dimension and our approach is robust to non-numeric trajectory attributes. Moreover, we designed an architecture with similar or lower network complexity compared to existing works, yet it achieved significantly higher levels of accuracy compared to other state-of-the-art approaches. MARC was published in IJGIS (PETRY et al., 2020).

Although we outperformed state-of-the-art results with MARC, one of its major drawbacks is the interpretability of the results. In contrast to a recently proposed method by our research group, MasterMovelets (FERRERO et al., 2020), MARC cannot easily provide human-interpretable insights on the classification patterns. An interesting direction of future work is the proposition of a neural-based method analogous to MasterMovelets, using CNNs. Besides benefiting from high parallelism, CNN patterns can nowadays be identified and interpreted in a similar manner as movelets are (ZHOU et al., 2016), and an architecture can be designed to identify patterns of different sizes (e.g. defining different kernel sizes that could be interpreted as different subtrajectory sizes). In addition, a CNN-based approach would only require a proper encoding for categorical attributes rather than the definition of distance functions for all trajectory attributes.

As MARC is essentially a deep learning model, another limitation lies in the need for a reasonably large dataset in order to obtain a classification model that generalizes well to unseen trajectory data. A path of study to mitigate this limitation is the use of transfer learning techniques to assist the training of MARC on small datasets, which has proven to be helpful for improving the classification accuracy of deep learning models (CHEN et al., 2019). For instance, knowledge (the model parameters) from larger datasets as Foursquare Global used in this work could boost the performance of classification models trained on smaller datasets like Foursquare NYC and Gowalla.

Future work could also exploit changes to embedding initialization techniques that could more significantly benefit trajectory classification. For instance, including some sort of supervision for learning embeddings to obtain class-specific embeddings might yield embeddings with better discriminative power. Class-specific embeddings could then be used in a nearest-neighbor classification approach, similar to the work of Snell, Swersky & Zemel (2017). Lastly, a thorough experimental comparison of MARC with MasterMovelets should be included in future works.

During this thesis, other works have also been developed as:

1. A similarity measure for multiple-aspect trajectories, published in *Transactions in GIS*

(TGIS) (PETRY et al., 2019);

2. A survey on trajectory classification methods, published in the proceedings of the *8th Brazilian Conference on Intelligent Systems (BRACIS)* (SILVA; PETRY; BOGORNY, 2019);

3. Frequency-based approaches for trajectory classification, published in the proceedings of the *35th Annual ACM Symposium on Applied Computing* (VICENZI et al., 2020).

Other contributions of the author, that are not necessarily related to this work but that were developed or contributed to in the context of this thesis, are:

1. An extended abstract on challenges in machine learning for detecting different vessel behaviors and anomalies, presented in the *2019 Montreal AI Symposium* (PETRY et al., 2019);

2. A literature review and description of open challenges in the analysis and detection of different vessel behaviors, based on their trajectories, published by Springer in Advances in Artificial Intelligence and presented at the *33rd Canadian Conference on Artificial Intelligence* (PETRY et al., 2020);

3. A method for finding relevant subtrajectories (patterns) for multiple-aspect trajectory classification, published in *Data Mining and Knowledge Discovery (DMKD)* (FERRERO et al., 2020);

4. A pivot-based method for optimizing the processing time of Movelets, under review for publication in *DMKD* (SILVA; PETRY; BOGORNY, 2020);

5. An exploratory analysis of NLP techniques for measuring content replication, spreading, and categorization in news portals on the Internet, published in the proceedings of the *8th BRACIS* (SILVA et al., 2019);

6. A method for boosting the training of deep learning models for few-shot learning, to be submitted to the *Journal of Machine Learning Research*;

7. An analysis of the impact of external data sources in future crime prediction, accepted for publications in the proceedings of the *16th International Conference on Data Science* (BAPPEE et al., 2020).

BIBLIOGRAPHY

ALVARES, L. O. et al. A model for enriching trajectories with semantic geographical information. In: ACM. **Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems**. [S.l.], 2007. p. 22.

ANDRIENKO, G.; ANDRIENKO, N.; HEURICH, M. An event-based conceptual model for context-aware movement analysis. **International Journal of Geographical Information Science**, Taylor & Francis, v. 25, n. 9, p. 1347–1370, 2011.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

BAPPEE, F. K. et al. Analyzing the impact of foursquare and streetlight data with human demographics on future crime prediction. In: **16th International Conf. on Data Science (accepted)**. [S.l.: s.n.], 2020.

BOGORNY, V. et al. Constant–a conceptual data model for semantic trajectories of moving objects. **Transactions in GIS**, Wiley Online Library, v. 18, n. 1, p. 66–88, 2014.

BRIDLE, J. S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: **Neurocomputing**. [S.l.]: Springer, 1990. p. 227–236.

BRUM-BASTOS, V. S.; LONG, J. A.; DEMŠAR, U. Weather effects on human mobility: a study using multi-channel sequence analysis. **Computers, Environment and Urban Systems**, Elsevier, v. 71, p. 131–152, 2018.

CHEN, C. et al. Trip2vec: a deep embedding approach for clustering and profiling taxi trip purposes. **Personal and Ubiquitous Computing**, Springer, p. 1–14, 2018.

CHEN, H.-H. Behavior2vec: Generating distributed representations of users' behaviors on products for recommender systems. **ACM Transactions on Knowledge Discovery from Data (TKDD)**, ACM, v. 12, n. 4, p. 43, 2018.

CHEN, W.-Y. et al. A closer look at few-shot classification. In: **International Conference on Learning Representations**. [s.n.], 2019. Disponível em: https://openreview.net/forum?id=HkxLXnAcFQ.

CHO, E.; MYERS, S. A.; LESKOVEC, J. Friendship and mobility: user movement in location-based social networks. In: ACM. **Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2011. p. 1082–1090.

DABIRI, S.; HEASLIP, K. Inferring transportation modes from gps trajectories using a convolutional neural network. **Transportation research part C: emerging technologies**, Elsevier, v. 86, p. 360–371, 2018.

DODGE, S. et al. The environmental-data automated track annotation (env-data) system: linking animal tracks with environmental data. **Movement Ecology**, BioMed Central, v. 1, n. 1, p. 3, 2013.

DODGE, S.; WEIBEL, R.; FOROOTAN, E. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. **Computers, Environment and Urban Systems**, Elsevier, v. 33, n. 6, p. 419–434, 2009.

DU, J. et al. Gene2vec: distributed representation of genes based on co-expression. **BMC genomics**, BioMed Central, v. 20, n. 1, p. 82, 2019.

ESULI, A. et al. Traj2user: exploiting embeddings for computing similarity of users mobile behavior. **arXiv preprint arXiv:1808.00554**, 2018.

ETEMAD, M.; JÚNIOR, A. S.; MATWIN, S. Predicting transportation modes of gps trajectories using feature engineering and noise removal. In: SPRINGER. **Canadian Conference on Artificial Intelligence**. [S.l.], 2018. p. 259–264.

FENG, J. et al. Deepmove: Predicting human mobility with attentional recurrent networks. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. **Proceedings of the 2018 World Wide Web Conference on World Wide Web**. [S.l.], 2018. p. 1459–1468.

FERRERO, C. A.; ALVARES, L. O.; BOGORNY, V. Multiple aspect trajectory data analysis: research challenges and opportunities. In: **GeoInfo**. [S.l.: s.n.], 2016. p. 56–67.

FERRERO, C. A. et al. Movelets: Exploring relevant subtrajectories for robust trajectory classification. In: **Proceedings of the 33rd ACM/SIGAPP Symposium on Applied Computing, Pau, France**. [S.l.: s.n.], 2018. p. 9–13.

FERRERO, C. A. et al. Mastermovelets: discovering heterogeneous movelets for multiple aspect trajectory classification. **Data Mining and Knowledge Discovery**, Jan 2020. ISSN 1573-756X. Disponível em: https://doi.org/10.1007/s10618-020-00676-x.

FIRTH, J. R. A synopsis of linguistic theory, 1930-1955. **Studies in linguistic analysis**, Basil Blackwell, 1957.

FURTADO, A. S. et al. Multidimensional similarity measuring for semantic trajectories. **Transactions in GIS**, v. 20, n. 2, p. 280–298, 2016.

GAO, Q. et al. Identifying human mobility via trajectory embeddings. In: AAAI PRESS. **Proceedings of the 26th International Joint Conference on Artificial Intelligence**. [S.l.], 2017. p. 1689–1695.

GONZALEZ, M. C.; HIDALGO, C. A.; BARABASI, A.-L. Understanding individual human mobility patterns. **nature**, Nature publishing group, v. 453, n. 7196, p. 779, 2008.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

GROEVE, J. D. et al. Individual movement-sequence analysis method (im-sam): characterizing spatio-temporal patterns of animal habitat use across landscapes. **International Journal of Geographical Information Science**, Taylor & Francis, p. 1–22, 2019.

GROEVE, J. D. et al. Extracting spatio-temporal patterns in animal trajectories: An ecological application of sequence analysis methods. **Methods in Ecology and Evolution**, Wiley Online Library, v. 7, n. 3, p. 369–379, 2016.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

JIANG, X. et al. Trajectorynet: An embedded gps trajectory representation for point-based classification using recurrent neural networks. In: IBM CORP. **Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering**. [S.l.], 2017. p. 192–200.

JÚNIOR, A. S.; RENSO, C.; MATWIN, S. Analytic: An active learning system for trajectory classification. **IEEE computer graphics and applications**, IEEE, v. 37, n. 5, p. 28–39, 2017.

KLEENE, S. Representations of events in nerve nets and finite automata. **Automata Studies [Annals of Math. Studies 34]**, Princeton Univ. Press, 1956.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: **International conference on machine learning**. [S.l.: s.n.], 2014. p. 1188–1196.

LEE, J.-G. et al. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 1, n. 1, p. 1081–1094, 2008.

LEE, J.-G. et al. Mining discriminative patterns for classifying trajectories on road networks. **IEEE Transactions on Knowledge and Data Engineering**, IEEE Educational Activities Department, v. 23, n. 5, p. 713–726, 2011.

MAAS, A. L. et al. Learning word vectors for sentiment analysis. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. **Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1**. [S.l.], 2011. p. 142–150.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. [S.l.]: Cambridge University Press, 2008.

MELLO, R. d. S. et al. Master: A multiple aspect view on trajectories. **Transactions in GIS**, Wiley Online Library, v. 23, n. 4, p. 805–822, 2019.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 3111–3119.

NIEMEYER, G. **Geohash**. 2008. Disponível em: https://en.wikipedia.org/wiki/Geohash.

PATEL, D. et al. Incorporating duration information for trajectory classification. In: IEEE. **28th International Conference on Data Engineering (ICDE)**. [S.l.], 2012. p. 1132–1143.

PETRY, L. M. et al. Towards semantic-aware multiple-aspect trajectory similarity measuring. **Transactions in GIS**, v. 23, n. 5, p. 960–975, 2019. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12542.

PETRY, L. M. et al. Marc: a robust method for multiple-aspect trajectory classification via space, time, and semantic embeddings. **International Journal of Geographical Information Science**, Taylor & Francis, p. 1–23, 2020. Disponível em: https://doi.org/10.1080/13658816.2019.1707835.

PETRY, L. M. et al. Unsupervised behavior change detection in multidimensional data streams for maritime traffic monitoring. **arXiv preprint arXiv:1908.05103**, 2019.

PETRY, L. M. et al. Challenges in vessel behavior and anomaly detection: From classical machine learning to deep learning. In: GOUTTE, C.; ZHU, X. (Ed.). **Advances in Artificial Intelligence**. Cham: Springer International Publishing, 2020. p. 401–407. ISBN 978-3-030-47358-7.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: **Fifteenth annual conference of the international speech communication association**. [S.l.: s.n.], 2014.

SILVA, C. L. da; PETRY, L. M.; BOGORNY, V. A survey and comparison of trajectory classification methods. In: **2019 8th Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2019. p. 788–793. ISSN 2643-6256.

SILVA, C. L. da; PETRY, L. M.; BOGORNY, V. A pivot-based method for movelets search space reduction and a benchmark for trajectory classification. **Data Mining and Knowledge Discovery (under review)**, 2020.

SILVA, C. L. da et al. Mining journals to the ground: An exploratory analysis of newspaper articles. In: **2019 8th Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2019. p. 78–83. ISSN 2643-6256.

SNELL, J.; SWERSKY, K.; ZEMEL, R. Prototypical networks for few-shot learning. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017. p. 4077–4087.

SOLEYMANI, A. et al. Integrating cross-scale analysis in the spatial and temporal domains for classification of behavioral movement. **Journal of Spatial Information Science**, v. 2014, n. 8, p. 1–25, 2014.

SPACCAPIETRA, S. et al. A conceptual view on trajectories. **Data & knowledge engineering**, Elsevier, v. 65, n. 1, p. 126–146, 2008.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The Journal of Machine Learning Research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

TOOR, M. L. van et al. Temporal segmentation of animal trajectories informed by habitat use. **Ecosphere**, Wiley Online Library, v. 7, n. 10, p. e01498, 2016.

TRAGOPOULOU, S.; VARLAMIS, I.; EIRINAKI, M. Classification of movement data concerning user's activity recognition via mobile phones. In: ACM. **Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics (WIMS14)**. [S.l.], 2014. p. 42.

VARLAMIS, I. Evolutionary data sampling for user movement classification. In: IEEE. **2015 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2015. p. 730–737.

VICENZI, F. et al. Exploring frequency-based approaches for efficient trajectory classification. In: **Proceedings of the 35th Annual ACM Symposium on Applied Computing (A1)**. New York, NY, USA: Association for Computing Machinery, 2020. (SAC '20), p. 624–631. ISBN 9781450368667. Disponível em: https://doi.org/10.1145/3341105.3374045.

WAGNER, R. A.; FISCHER, M. J. The string-to-string correction problem. **Journal of the ACM (JACM)**, ACM, v. 21, n. 1, p. 168–173, 1974.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. **Chemometrics and intelligent laboratory systems**, Elsevier, v. 2, n. 1-3, p. 37–52, 1987.

XIAO, Z. et al. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. **ISPRS International Journal of Geo-Information**, Multidisciplinary Digital Publishing Institute, v. 6, n. 2, p. 57, 2017.

YANG, D.; ZHANG, D.; QU, B. Participatory cultural mapping based on collective behavior data in location-based social networks. **ACM Transactions on Intelligent Systems and Technology (TIST)**, ACM, v. 7, n. 3, p. 30, 2016.

YANG, D. et al. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, v. 45, n. 1, p. 129–142, 2015.

ZHAO, W. X. et al. A time-aware trajectory embedding model for next-location recommendation. **Knowledge and Information Systems**, Springer, p. 1–21, 2017.

ZHENG, Y. et al. Understanding transportation modes based on gps data for web applications. **ACM Transactions on the Web (TWEB)**, ACM, v. 4, n. 1, p. 1, 2010.

ZHOU, B. et al. Learning deep features for discriminative localization. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 2921–2929.

ZHOU, F. et al. Trajectory-user linking via variational autoencoder. In: **Proceedings of the 27th International Joint Conference on Artificial Intelligence**. [S.l.: s.n.], 2018. p. 3212–3218.

ZHOU, N. et al. A general multi-context embedding model for mining human trajectory data. **IEEE transactions on knowledge and data engineering**, IEEE, v. 28, n. 8, p. 1945–1958, 2016.

ZOU, W. Y. et al. Bilingual word embeddings for phrase-based machine translation. In: **Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2013. p. 1393–1398.