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

Tarlis Tortelli Portela

Towards Optimization Methods for Movelets Extraction in Multiple Aspect Trajectory Classification

Florianópolis – Pisa
2023

Tarlis Tortelli Portela

Towards Optimization Methods for Movelets Extraction in Multiple Aspect Trajectory Classification

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina e pela Universidade de Pisa em regime de cotutela para a obtenção do título de Doutor em Ciência da Computação.
Orientadoras: Vania Bogorny (UFSC), Anna Bernasconi (UniPI)
Coorientadora: Chiara Renso

Florianópolis – Pisa
2023

Tarlis Tortelli Portela

Towards Optimization Methods for Movelets Extraction in Multiple Aspect Trajectory Classification

Thesis submitted to the Post-Graduate Program in Computer Science of the Federal University of Santa Catarina and University of Pisa under cotutela regime for the degree of Doctor of Philosophy.
Advisors: Vania Bogorny (UFSC), Anna Bernasconi (UniPI)
Co-advisor: Chiara Renso

Florianópolis – Pisa
2023

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

Portela, Tarlis Tortelli
Towards Optimization Methods for Movelets Extraction in Multiple Aspect Trajectory Classification / Tarlis Tortelli Portela ; orientadora, Vania Bogorny, orientadora, Anna Bernasconi, coorientadora, Chiara Renso, 2023.
115 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2023.

Inclui referências.

Trabalho elaborado em regime de co-tutela.

1. Ciência da Computação. 2. Mineração de dados. 3. Trajetórias Multi-Aspecto. 4. Classificação de Trajetórias. 5. Subtrajetórias relevantes. I. Bogorny, Vania. II. Bernasconi, Anna. III. Renso, Chiara. IV. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. V. Título.

Tarlis Tortelli Portela

Towards Optimization Methods for Movelets Extraction in Multiple Aspect Trajectory Classification

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

Prof. Mateus Grellert, Dr.
Universidade Federal de Santa Catarina, Florianópolis, Brasil

Prof.ª Anna Bernasconi, Dra.
Universidade de Pisa (UniPI), Pisa, Itália

Prof.ª Karine Zeitouni, Dra.
Universidade de Versailles Saint-Quentin-en-Yvelines (UVSQ), Versalhes, França

Chiara Renso, Dra.
Conselho Nacional de Pesquisa, Pisa, Itália

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

Documento assinado digitalmente
PATRICIA DELLA MEA PLENTZ
Data: 24/04/2023 08:28:38-0300
CPF: ***.243.700-**
Verifique as assinaturas em https://v.ufsc.br

Prof.ª Patricia Della Méa Plentz, Dra.
Coordenador do Programa

Documento assinado digitalmente
VANIA BOGORNY
Data: 20/04/2023 16:16:16-0300
CPF: ***.586.570-**
Verifique as assinaturas em https://v.ufsc.br

Prof.ª Vania Bogorny, Dra.
Orientadora

Florianópolis – Pisa, 2023.

Aos meus pais, Valério e Antonieta. À minha irmã, minha amiga
toda a vida.

ACKNOWLEDGEMENTS

I am incredibly grateful to my supervisors, Anna Bernasconi, Chiara Renso, and Vania Bogorny, without whom many of the results presented here would not have been possible. I am profoundly thankful to these women for their expertise, brilliant insights, personal advice, and continuous and inspiring guidance throughout this Ph.D. Thanks to Anna and Chiara for the fantastic reception in Pisa, patience, and kindness in the most difficult moments. I am grateful to the people I met in Pisa, the professors, the colleagues at CNR, and the friends I made. I am profoundly grateful to my supervisor, Vania, who found me in the hallway four years ago. Thanks for teaching me, trusting me, pushing my limits, and believing in me.

Special thanks to the internal committee members, Anna Monreale and Giovanni Manzini, for the suggestions presented in the yearly evaluations. I am also grateful to the external referees, Karine Zeitouni and Panagiotis Tampakis, for their expert feedback and helpful comments. I am grateful to Mateus Grellert for all the support given.

Thanks to my lab friends in Brasil and Italy, Vanessa, Yuri, Camila, Lucas, Guido, and Chiara, for the knowledge and funny moments we spent together. Vanessa, looking back on the countless hours we spent drinking coffee, discussing ideas, and our friendship, I can say how much it helped during these years.

Thanks to the University of Pisa, the National Research Council (CNR), The Federal University of Santa Catarina (UFSC), FAPESC, and CAPES for making this cotutela possible. I also thank my work institution, IFPR, for the license granted to develop this research and for the support of my coworkers, especially to Lilian who has been with me from the beginning.

My most special thanks to my family, they are the reason I got here. To my sister, Saura, that is the person I love most in the world. To my father and mother, that always gave everything to make us into the best people. I know they are somewhere beyond and proud. Thanks to my grandfather for the gesture that inspired me to study constantly.

Thanks to Gi, to whom I owe much of what I have become, the support, the crying shoulder, the ten years of friendship, and my person, Bródi. To my friends Vis, Matheus e Cristiane, that always asked how the thesis was, and Dani for the home medical care. To Can, Fran, and Gra, that pushed me to start a Ph.D. To a lot of friends, lots of thanks.

Un ringraziamento speciale ad Anna e Chiara, per la fantastica accoglienza a Pisa, per la loro pazienza e gentilezza nei momenti più difficili. Sono grato alle persone che ho incontrato a Pisa, ai professori, ai colleghi del CNR e alle nuove amicizie che ho stretto, specialmente a Daniele. Grazie per avermi insegnato la cultura italiana, per i pasticcini e le serate in città.

Meu maior agradecimento à minha família, eles são a razão de eu ter chegado até aqui. À minha irmã Saura, que é a pessoa que mais amo no mundo. Ao meu pai e à minha mãe que sempre deram tudo para nos tornar melhores pessoas. Eu sei que eles estão em algum lugar além, orgulhosos. Agradeço ao meu avô pelo gesto que me inspirou a estudar sempre.

Agradeço a Gi, a quem devo muito do que me tornei, o apoio, o ombro amigo, aos dez anos de amizade, minha pessoa, Bródi. Aos meus amigos Vis, Matheus e Cristiane, que sempre perguntavam como estava a tese, e a Dani pelos cuidados médicos domiciliares. A Can, Fran e Gra, que me incentivaram a iniciar um doutorado. A muitos amigos, muito obrigado.

"We consider the greatest end of science is the classification of past data. It is important, but is there no further work to be done?"
— Isaac Asimov (Foundation)

ABSTRACT

In the last few years there has been a significant increase in the collection of mobility data. By mobility data we refer to the collection of positioning data, called trajectories, of tracked moving objects. These objects could be humans, animals, vehicles or other devices like Internet of Things (IoT). The analysis of such data has been proved to be useful in several application domains from a urban scenario for traffic prediction or transportation means optimization, to maritime domain analysing vessels paths or environmental domain with the study of hurricanes evolution or animal behavior. One of the most typical and used analysis task on mobility data is classification, where trajectory data is automatically assigned a label or class. The explosion of social media data, sensors, IoT, and Internet-enabled sources allowed the semantic enrichment of such mobility data, which evolved from raw spatio-temporal data to high dimensional data. Mobility analysis, and specifically classification task, on such high dimensional data becomes therefore more challenging. In fact, existing trajectory classification methods have mainly considered space, time, and numerical data, ignoring the large number of semantic dimensions. Only recently research community proposed classification methods based on the concept of *movelets* that are the parts of a trajectory that better discriminate a class and that can therefore improve classification accuracy. State of the art methods in movelets extraction are computationally inefficient, which makes them unfeasible to be used for real large high dimensional datasets. The objective of this thesis is therefore to develop new algorithms for discovering movelets that are faster than state of the art while maintaining or improving classification accuracy. Our main contribution is a new high performance method for extracting movelets and classifying trajectories, called HiPerMovelets (High-performance Movelets). Experimental results show that HiPerMovelets is 10 times faster than the best state of the art method, reduces the high dimensionality problem, is more scalable, and presents a high classification accuracy in all evaluated datasets. A secondary contribution are the algorithms RandomMovelets and UltraMovelets. RandomMovelets reduces the search space by randomly extracting subtrajectories and evaluating their relevance for classification without exploring the entire dataset. UltraMovelets reduces the combinatorial explosion when exploring subtrajectories. Preliminary results suggest that these methods can reduce the search space, use less computational resources, and are at least 6 times faster than baselines.

Keywords: Data Mining. Multiple Aspect Trajectories. Trajectory Classification. Relevant Subtrajectories. Movelets

RESUMO

Nos últimos anos, houve um aumento significativo na coleta de dados de mobilidade. Dados de mobilidade referem-se ao conjunto de dados de posicionamento geográfico, chamados de trajetórias de objetos móveis. Esses objetos podem ser pessoas, animais, veículos ou outros dispositivos como a Internet das Coisas (IoT). A análise deste tipo de dados se revela útil em vários domínios de aplicação, desde um cenário urbano para previsão de tráfego ou otimização de meios de transporte, no domínio marítimo analisando trajetos de embarcações, no domínio ambiental com o estudo da evolução de furacões ou comportamento animal. Uma das tarefas de análise mais comuns e usadas em dados de mobilidade é a classificação, onde os dados de trajetória recebem automaticamente um rótulo ou classe. A explosão de dados de mídia social, sensores, IoT e outras fontes da Internet permitiram o enriquecimento semântico desses dados de mobilidade, que evoluíram de dados espaço-temporais brutos para dados de alta dimensionalidade. A análise de mobilidade, e especificamente a tarefa de classificação, em tais dados de alta dimensionalidade tem se tornado mais desafiadora. De fato, os métodos de classificação de trajetória existentes consideram principalmente espaço, tempo e dados numéricos, ignorando o grande número de dimensões semânticas. Apenas recentemente a comunidade de pesquisa propôs métodos de classificação baseados no conceito de movelets que são as partes de uma trajetória que melhor discriminam uma classe e que podem, portanto, melhorar a precisão da classificação. Métodos de última geração na extração de movelets são computacionalmente ineficientes, o que os torna inviáveis para serem usados em grandes conjuntos de dados de alta dimensão. O objetivo desta tese é, portanto, desenvolver novos algoritmos para descobrir movelets que sejam mais rápidos do que o estado da arte, mantendo ou melhorando a precisão da classificação. Nossa principal contribuição é um novo método de alto desempenho para extração de movelets e classificação de trajetórias, denominado HiPerMovelets (Movelets de alto desempenho). Os resultados experimentais mostram que o HiPerMovelets é 10 vezes mais rápido que o melhor método do estado da arte, reduz o problema de alta dimensionalidade, é mais escalável e apresenta uma alta precisão de classificação em todos os conjuntos de dados avaliados. Uma contribuição secundária são os algoritmos RandomMovelets e UltraMovelets. RandomMovelets reduz o espaço de busca extraindo subtrajetórias aleatoriamente e avaliando sua relevância para classificação sem explorar todo o conjunto de dados. UltraMovelets reduz a explosão combinatória ao explorar subtrajetórias. Os resultados preliminares sugerem que esses métodos podem reduzir o espaço de busca, usar menos recursos computacionais e são pelo menos 6 vezes mais rápidos que a linha de base.

Palavras-chave: Mineração de dados. Trajetórias Multi-Aspecto. Classificação de Trajetórias. Subtrajetórias relevantes. Movelets

RESUMO EXPANDIDO

Introdução

Há um amplo reconhecimento do valor dos dados e produtos obtidos através da análise de grandes volumes de dados (*Big Data*). No entanto, o desafio não se restringe a localizar, identificar, entender esses dados. A capacidade de analisar grandes conjuntos de dados tem valor limitado se os usuários não puderem entender a análise. O resultado precisa ser interpretável, o que envolve examinar as suposições feitas e verificar os resultados produzidos pelo computador.

Com a alta disponibilidade de informações, a popularização de dispositivos móveis, redes sociais e IoT, grandes volumes de dados de mobilidade estão sendo coletados sobre a rotina das pessoas. Com essa explosão de dados, são necessárias novas tecnologias e métodos para viabilizar sua mineração, categorização e processamento. Esses dados são foco de pesquisas em algoritmos e sistemas de uso intenso de dados que tratem de tais problemas computacionais.

Os dados de mobilidade são chamados *trajetórias de objetos móveis* que podem representar o movimento de pessoas, veículos, navios, furacões, etc. Essas trajetórias são sequências de pontos localizados no espaço e no tempo e elas evoluíram ao longo dos anos. A Figura 2 exemplifica como esses dados evoluíram ao longo dos anos, tornando-se mais significativos e revelando mais padrões sobre o objeto móvel. A medida que os dados se tornam mais complexos, as tarefas de mineração de dados também são mais desafiadoras.

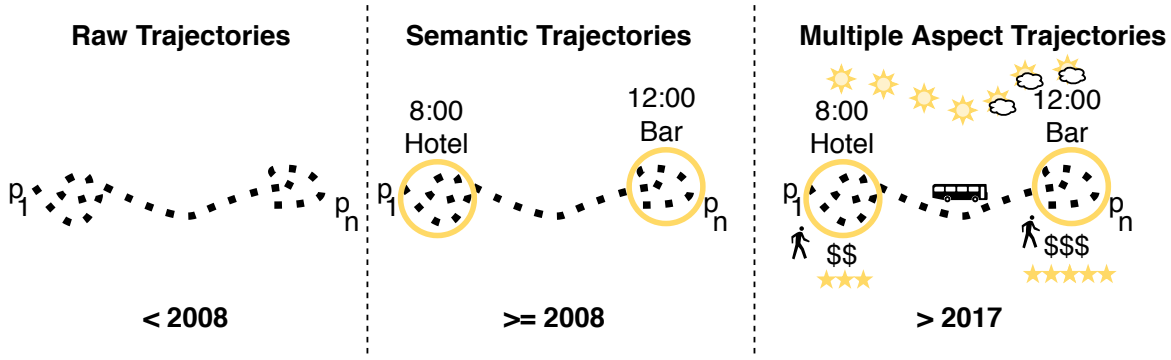


Figura 1 – Representação de trajetórias ao longo dos anos.

Antes de 2007, como pode ser visto na Figura 2, as trajetórias eram chamadas *trajetórias brutas (raw trajectories)* e representadas pelas dimensões espaço (e.g. latitude e longitude) e tempo. O conceito de *trajetórias semânticas (semantic trajectories)* surgiu em 2007, modelando *stops* e *moves*, onde os *stops* são as partes importantes e semanticamente ricas das trajetórias (SPACCAPIETRA et al., 2008).

Modelos e tentativas anteriores de enriquecer trajetórias com mais informações semânticas levaram a uma trajetória muito mais detalhada. Desde 2016, surgiu o conceito de *trajetórias multi-aspecto (Multiple Aspect Trajectories, MAT)* (FERRERO et al., 2018; MELLO et al., 2019). MATs representam um conceito mais amplo do que *stops* e *moves*, onde pontos espaço-temporais podem ser enriquecidos com qualquer tipo de informação semântica. Informações semânticas significam qualquer tipo de dado que não seja espacial nem temporal que representa qualquer aspecto dos dados como: nome do local visitado ou ponto de interesse (POI), avaliação ou preço do local, meio de transporte, condição climática, o humor de uma pessoa, etc. Antes de 2016, os dados de trajetória eram analisados, geralmente, sob um único ponto de vista e enriquecidos com uma única dimensão semântica. Depois disso, o uso das mídias sociais disseminou disponibilizando grandes quantidades de dados na Web de modo que os dados de trajetória começaram a ser enriquecidos e analisados em várias perspectivas. Essa nova

representação de trajetória é um desafio para a mineração de dados de trajetórias, particularmente na classificação que é o problema abordado neste trabalho.
A classificação de trajetórias pode ser definida como a técnica de descobrir o rótulo da classe de um objeto móvel com base em suas trajetórias (LEE et al., 2008). Classificação de trajetórias é importante para vários domínios de aplicação como, por exemplo, identificar o meio de transporte de um objeto móvel (ETEMAD; Soares Júnior; MATWIN, 2018), o nível de força de um furacão (LEE et al., 2008), o tipo de uma embarcação (carga, pesca, turismo, etc.) (LEE et al., 2008), o usuário proprietário de uma trajetória (FERRERO et al., 2020), etc. O principal desafio na classificação de trajetórias multi-aspecto é o grande número e a heterogeneidade das dimensões associadas a cada ponto da trajetória. Por exemplo, um POI é uma representação semântica da dimensão espacial e está relacionado ao seu preço e avaliação. Além disso, a dimensão espacial é composta por dois valores (latitude e longitude) que precisam ser considerados juntos para representar uma posição real no espaço. As trajetórias podem ser segmentadas para se encontrar padrões, pois considerar uma trajetória inteira pode não caracterizar o comportamento que discrimina as classes (LEE et al., 2008). Por exemplo, uma trajetória de $<$ *Casa*, *Escola*, *Parque* $>$ pode não caracterizar o movimento geral dos alunos, mas talvez uma subtrajetória $<$ *Casa*, *Escola* $>$ sim. O principal desafio é encontrar os melhores atributos (*features*) de trajetórias ou subtrajetorias para usar como entrada de um classificador. Subtrajetórias relevantes são importantes *features* para representar padrões em trajetórias. Espera-se que elas apareçam com frequência em uma determinada classe e menos frequentemente nas trajetórias de outras classes. Novos métodos de classificação de trajetórias se baseiam na extração dessas subtrajetórias relevantes, em vez de novos modelos ou algoritmos de classificação.
A análise de mobilidade, e especificamente a tarefa de classificação, em dados de alta dimensionalidade tem se tornado mais desafiadora. De fato, os métodos de classificação de trajetória existentes consideram principalmente espaço, tempo e dados numéricos, ignorando o grande número de dimensões semânticas. Apenas recentemente a comunidade de pesquisa propôs métodos de classificação baseados no conceito de *movelets* que são as partes de uma trajetória (subtrajetórias) que melhor discriminam uma classe e que podem, portanto, melhorar a precisão da classificação. Os métodos propostos por Ferrero em (FERRERO et al., 2018) e (FERRERO et al., 2020) superaram a maioria dos métodos existentes para classificação de trajetória em termos de acurácia, mostrando que as melhores *features* de trajetórias para problemas de classificação são subtrajetórias de tamanhos diferentes (por exemplo, um ponto, dois pontos, etc) e algumas de suas dimensões ou combinações de dimensões. MASTERMovelets (FERRERO et al., 2020) explora automaticamente todas as possíveis subtrajetórias de qualquer tamanho (por exemplo, um ponto, dois pontos, três pontos, etc.) e explora todas as combinações de dimensão (por exemplo, apenas espaço; espaço e tempo; espaço, tempo e categoria POI, etc), enquanto procura as melhores *movelets* para cada classe. No entanto, MASTERMovelets é computacionalmente caro porque gera um grande número de subtrajetórias e todas as combinações de dimensões. Portanto, sofre da curso da dimensionalidade devido ao grande número de candidatos a movelets. Este problema leva à nossa questão de pesquisa: **Podemos desenvolver novos algoritmos para descobrir subtrajetórias relevantes que se tornam movelets mais rapidamente que MASTERMovelets, mantendo ou melhorando a acurácia da classificação?** Nossa hipótese é que podemos reduzir a explosão de combinações de dimensões e acelerar o processo de segmentação para extração de movelets.

Objetivos

O principal objetivo desta tese é propor novos métodos para classificação de trajetórias de multi-aspecto que possam lidar com o alto custo computacional e o grande número de movelets do método do estado da arte MASTERMovelets, mantendo a precisão e reduzindo o custo compu-

tacional. Nosso objetivo é identificar quais são as *subtrajetórias* mais relevantes que provavelmente se tornarão movelets sem explorar todo o conjunto de dados, reduzindo assim o espaço de busca para descoberta de movelets. Como objetivos específicos iremos:

- Propor e implementar técnicas para extração eficiente de *features* de trajetórias de multi-aspecto para classificação reduzindo o espaço de busca e dimensionalidade;
- Avaliar as técnicas propostas em diversos conjuntos de dados e compará-las com os métodos do estado da arte;
- Propor uma ferramenta para apoiar a tarefa de classificação de trajetórias de multi-aspecto, especificamente fornecendo formas de visualização das *movelets* e dados de trajetória.

Metodologia
Para cumprir os objetivos propostos nesta tese adotamos a seguinte metodologia:

1. Realizar revisão de literatura em classificação de trajetórias, com foco em trabalhos que realizam classificação em dados de trajetórias multi-aspecto;
2. Propor e implementar algoritmos para redução do espaço de busca e tempo de execução em relação ao MASTERMovelets;
3. Avaliar o comportamento dos métodos propostos usando conjuntos de dados de trajetórias reais e sintéticas incluindo a acurácia de classificação, escalabilidade, tempo computacional, tempo de classificação e o número total de candidatos a movelet geradas e movelets avaliadas;
4. Definir e preparar os conjuntos de dados para avaliar os métodos e compará-los com o estado da arte;
5. Implementar ou adaptar os métodos do estado-da-arte que devam ser comparados à nossa proposta;
6. Propor ferramentas de visualização de trajetórias e subtrajetórias com suporte à classificação e análise de trajetórias multi-aspecto;
7. Escrever artigos descrevendo novos métodos para reduzir o espaço de busca para extrair as subtrajetórias relevantes e experimentos de *benchmark*;
8. Redação da tese descrevendo os principais conceitos necessários de dados de trajetória, o problema de classificação de trajetória, o estado da arte, a descrição das soluções propostas, avaliações experimentais e as conclusões obtidas.

O escopo desta tese está limitado à definição de novas técnicas de otimização para exploração de subtrajetórias relevantes, a fim de melhorar o desempenho da extração de características para classificação de trajetórias em relação aos métodos do estado da arte.

Resultados e Discussão

Nossa principal contribuição é um novo método de alto desempenho para extração de movelets e classificação de trajetórias, denominado HiPerMovelets (*High Performance Movelets*). Os resultados obtidos neste trabalho mostram que o HiPerMovelets é 10 vezes mais rápido que o melhor método do estado da arte, reduz o problema de alta dimensionalidade, é mais escalável e apresenta uma alta precisão de classificação em todos os conjuntos de dados avaliados. Uma contribuição secundária são os algoritmos RandomMovelets e UltraMovelets. RandomMovelets reduz o espaço de busca extraindo subtrajetórias aleatoriamente e avaliando sua relevância para classificação sem explorar todo o conjunto de dados. UltraMovelets reduz a explosão combinatória ao explorar subtrajetórias. Os resultados preliminares sugerem que esses métodos podem reduzir o espaço de busca, usar menos recursos computacionais e são pelo menos 6 vezes mais rápidos que a linha de base.

A fim de superar algumas limitações dos métodos apresentados, são propostos os seguintes trabalhos futuros: (i) nova estratégia para encontrar o split point e o melhor alinhamento que evita calcular distâncias para cada posição em todas as trajetórias; (ii) investigar uma nova medida de qualidade para avaliação das movelets; (iii) propor um método para processamento distribuído; (iv) investigar um novo classificador para melhores resultados; (v) experimentação em outras áreas; (vi) extensa experimentação de escalabilidade; e, (vii) investigação do uso de movelets em outros domínios de aplicação.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{T}$	Trajectory dataset of pairs, $\{(T_1, class_{T_1}), ..., (T_n, class_{T_n})\}$
T'	Subset of trajectories from one class
T_i	The ith trajectory in $\mathbf{T}$
$class_{T_i}$	The class label of T_i
A	Set of an element aspects, $A = \{a_1, a_2, ..., a_l\}$, interchangeably called dimensions
A'	A subset of aspects A
$n, \lvert\mathbf{T}\rvert$	number of trajectories
$m, \lvert T\rvert$	Length of a trajectory
w	Length of a subtrajectory
$l, \lvert A\rvert$	Number of aspects
s, u	Subtrajectory, also $s_{a,b}$
$\mathbf{M}$	Set of movelets from $\mathbf{T}$
$\mathbf{M}^{T_i}$	Set of movelets from trajectory T_i
$\mathcal{M}_i$	Movelet candidate
C	Movelet candidate aspects, $C \subseteq A$
V	Distance vector between two subtrajectories
$\mathbb{W}$	Set of pairs $(W_{T_i}^s, class_{T_i})$
$W_{T_i}^s$	Distance vector of the subtrajectory s to trajectory T_i
$\mathbf{S}_{T_i}^w$	Distance set of all subtrajectories of length w from T_i
sp	A set of split point distances for each dimension
$\mathbf{T}^{\mathcal{M}}$	Subset of a class *covered trajectories* by a movelet candidate $\mathcal{M}$, $\mathbf{T}^{\mathcal{M}} \subseteq$ **T'**
λ	Learned limit to dimension combinations
τ	Parameter for minimum proportion of selected candidates
α	Parameter for percentual selection of movelet candidates

CONTENTS

1 INTRODUCTION

There is broad recognition of the value of data and products obtained through the analysis of large volumes of data (*Big Data*). The ability to analyze large data sets is limited in value if users cannot understand the analysis. The result must be interpretable, which involves examining the assumptions made and verifying the results produced by the computer. Methods capable of revealing patterns in the data can help in the most diverse applications such as urban planning, smart cities, understanding natural phenomena, vaccine development, public safety, urban transport, etc. For instance, vehicle traces collected from traffic on city streets can be used to predict and avoid traffic jams. Taxi traces can be used to predict new bus lines. In public transport, analysis of bus traces can be used to early detect (as mobile sensors) events in regions that affect traffic to propose new routes for public transport. In ecology, tracking animals can help to understand the human impact on animal behavior, as well as their movement and migration patterns. In understanding natural phenomena, analysis techniques can be used to classify the intensity of hurricanes. In vaccine development, data from proteins with the potential to generate an immune response can be used to choose test targets for new vaccines. In digital security, trajectory similarity analysis techniques are used to classify scanned signatures as legitimate or fraudulent (VLACHOS; KOLLIOS; GUNOPULOS, 2002). In maritime monitoring, boat trajectories can, in real-time, be used to detect normal and anomalous behavior, for instance, to prevent boat accidents with refugees and illegal immigration, thus helping with humanitarian issues.

With the popularization of mobile devices, social networks, and IoT, large volumes of mobility data are being collected about our daily routines. Some companies, such as Google, Amazon, and Apple use their products to collect details about our movement, including the places we visit and the time we stay there. Facebook, for instance, captures our location, and stores our friendship relationships, as well as our thoughts and opinions about things and people. Pokémon GO emerged to capture our movement and photos of places we visit when capturing Pokémons, which certifies with a high accuracy where we are. These are just a few examples that show the importance of the physical location and movement of people.

Mobility data can be referred to as *moving object trajectories*, and can represent the movement of people, animals, vehicles, ships, hurricanes, etc. Trajectories are sequences of points located in space and time, representing the evolution of the location of the object. This data evolved over the years, becoming more meaningful and revealing more patterns about moving objects. As the data become more detailed, they become more complex, and the data mining tasks are more challenging. Figure 2 exemplifies how mobility data become more meaningful, more complex, and revealing more patterns about moving objects.

Before 2007, as can be seen in Figure 2, mobility data were called *raw trajectories*, and represented by sequences of points with the dimensions space and time. The space dimension is represented by a position (e.g. latitude and longitude) sequentially ordered by time.

When the concept of *semantic trajectories* emerged in 2008, the points started to be

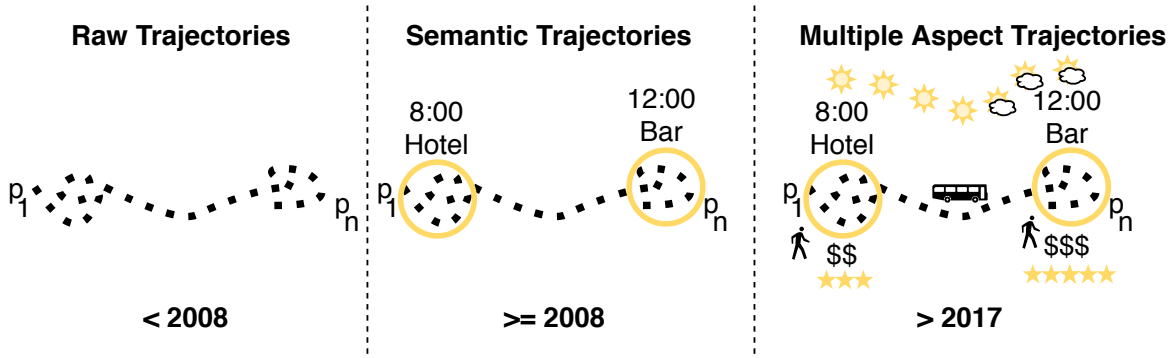


Figure 2 – Trajectory Representation along the years.

annotated as stops and moves, which are the important and semantically rich parts of trajectories (SPACCAPIETRA et al., 2008). Stops are the parts of a trajectory where the object stays for a minimal amount of time, while the moves are the remaining parts. The spatial dimension of the stops was basically annotated with the attribute POI, the Point Of Interest that corresponds to the name of a place the user visited. The moves, for instance, could be annotated with an activity or transportation mean. It is important to notice that trajectories are multidimensional by definition, having at least the spatial and temporal dimensions. Semantic trajectories have an additional dimension, the semantics.

Previous models and attempts to enrich trajectories with more semantic information and different aspects, have led to a very detailed trajectory. In 2018, emerged the concept of *Multiple Aspect Trajectories* (MAT) (FERRERO et al., 2018; MELLO et al., 2019), representing a broader concept than stops and moves, where the spatio-temporal points started to be associated with any type of semantic information. By semantic information we mean any type of data that is neither spatial nor temporal, representing any aspect as the name of a visited place or Point of Interest (POI), the rate or price of the place, the transportation mode, the weather condition, the mood of a person, etc. MAT became more complex because each point has at least three types of dimensions (space, time, and semantics). The MAT allows the representation of the trajectory shown in Figure 3. As can be observed from the figure, a trajectory has very detailed information about the moving object, which is a challenge to trajectory data mining since the object to be analyzed becomes complex. In this example, the trajectory is associated with different semantic aspects that can be related to the object. For instance, at home, the heart and sleeping rate of the user is collected. When he/she moves on foot to work, a Tweet that expresses his mood is captured. He/She works at a smart office where sensors collect environmental information about noise, temperature, and pollution. At night when he/she goes from work to a restaurant, the characteristics of the restaurant as price and reviews can also be collected, creating a new complex type of movement and semantic dimensions that are interconnected.

A fundamental concept for this thesis is the subtrajectory. In general, we define a subtrajectory as a continuous part of a trajectory that might include both spatio-temporal information and semantics. A stop or a move are examples of subtrajectories. However, it is

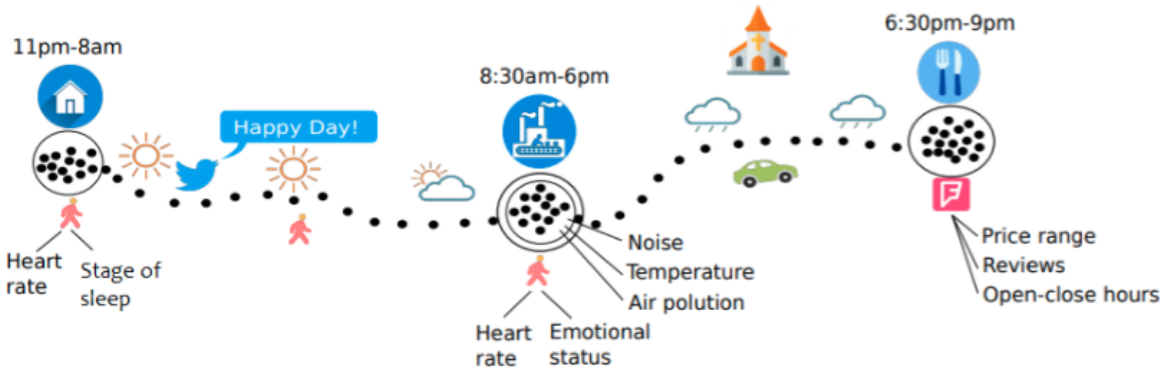


Figure 3 – An example of Multiple Aspect Trajectory (MELLO et al., 2019).

important to highlight that a subtrajectory might contain also a subset of dimensions. Figure 4 shows an example of MAT and clarifies the concept of subtrajectories with different dimensions. Trajectory T_1 has 6 points with three dimensions (weather condition, POI, and time), and two subtrajectories, where $s_{1,2}$ is a subtrajectory of size 2 because it has two points and 3 dimensions (weather, POI name and time), and s_5 is a subtrajectory of size 1 with 1 point and 2 dimensions (POI and time).

A subtrajectory can represent a relevant behavior of a moving object. For instance, the subtrajectories $s_{1,2}$ and s_5 of Figure 4 are very common patterns in the trajectories of tourists. The POI sequence *Hotel* to *Museum A* and the *Cafe* at *18h* are patterns that appear frequently when analyzing tourist trajectories. When analyzing an unknown trajectory with these same or similar subtrajectories, it is possible to infer that it belongs to a tourist.

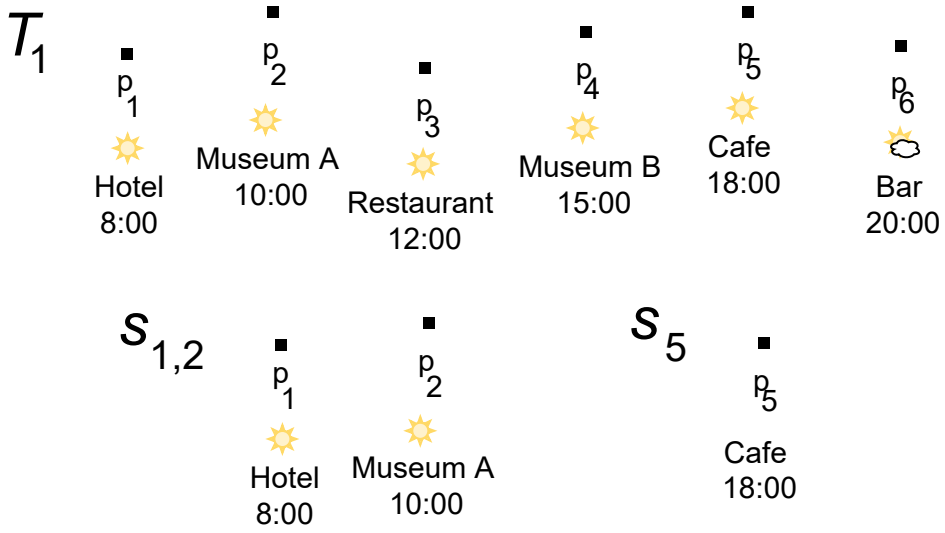


Figure 4 – Trajectory and subtrajectories example.

The analysis of mobility data is useful in several application domains, like transportation, ecology, and maritime studies to name a few. Some well-known analysis tasks include:

clustering, classification, prediction, anomaly detection, and frequent pattern mining (ZHENG, 2015; LEITE; PETRY; BOGORNY, 2019). In this thesis, we focus on trajectory classification which is the task of assigning a trajectory to a class (LEE et al., 2008). Examples are the inference of the transportation mode of a moving object (e.g. car, bus, bike) (ETEMAD; Soares Júnior; MATWIN, 2018), the strength level of a hurricane (LEE et al., 2008), a vessel type (e.g. cargo, fish, tourism, etc) (LEE et al., 2008), the user identity of a given trajectory (FERRERO et al., 2020), etc. Very recently, the importance of trajectory classification has been revealed in health applications. From the driving movement patterns, it is possible to predict early signs of Alzheimer disease (BAYAT et al., 2021a), and from outdoor mobility patterns, to predict dementia in older adults (BAYAT et al., 2021b).

Classification of trajectory data is particularly complex due to the fact that movement data has at least two dimensions: space and time. With semantic or multiple aspect trajectories the task becomes even more complex due to the several dimensions involved.

The first challenge in trajectory classification methods is to define a set of relevant features that better discriminate the class. Depending on the type of class, different types of trajectory features should be extracted for accurate classification. For instance, for transportation mode classification, features such as speed and acceleration may be discriminant, while in user type classification (e.g. tourist, commuter, resident), features as type of visited places, time of the visit, or duration of the visit may be the most discriminant. The features can be local when related to a subtrajectory or a single point (trajectory parts), or global, when related to the entire trajectory (e.g. trajectory average speed) (FERRERO et al., 2018). Most features are local, but global features as the average speed of the entire trajectory may be discriminant for some problems.

The classification task of discovering the user identity of a given trajectory is often referred to as Trajectory User Linking (TUL) in the literature (GAO et al., 2017; ZHOU et al., 2018). The user identity (name or identifier) is the label for the classification task, in which the goal is - from a subset of trajectory data - to identify the user that generated the trajectories. For TUL, the number of classes can be around dozens or hundreds, which makes the classification task very challenging and hard to achieve high accuracy.

A big issue in MAT classification is the large number and the heterogeneity of the types of dimensions that can be associated with each trajectory point. The classification problem complexity is increased by the fact that, for proper classification, trajectories must be segmented into subtrajectories, because an entire trajectory may not characterize a real movement behavior to discriminate classes (LEE et al., 2008). For instance, consider a classification task that analyses the trajectory of individuals to classify trajectories into students or not. A trajectory following the sequence of POIs $\langle$ *Home*, *School*, *Park* $\rangle$ might be too specific and not characterize the general movement of students, while its subtrajectory $\langle$ *Home*, *School* $\rangle$ can better represent the students mobility behavior. Therefore, this subtrajectory can be the discriminant to classify a student trajectory.

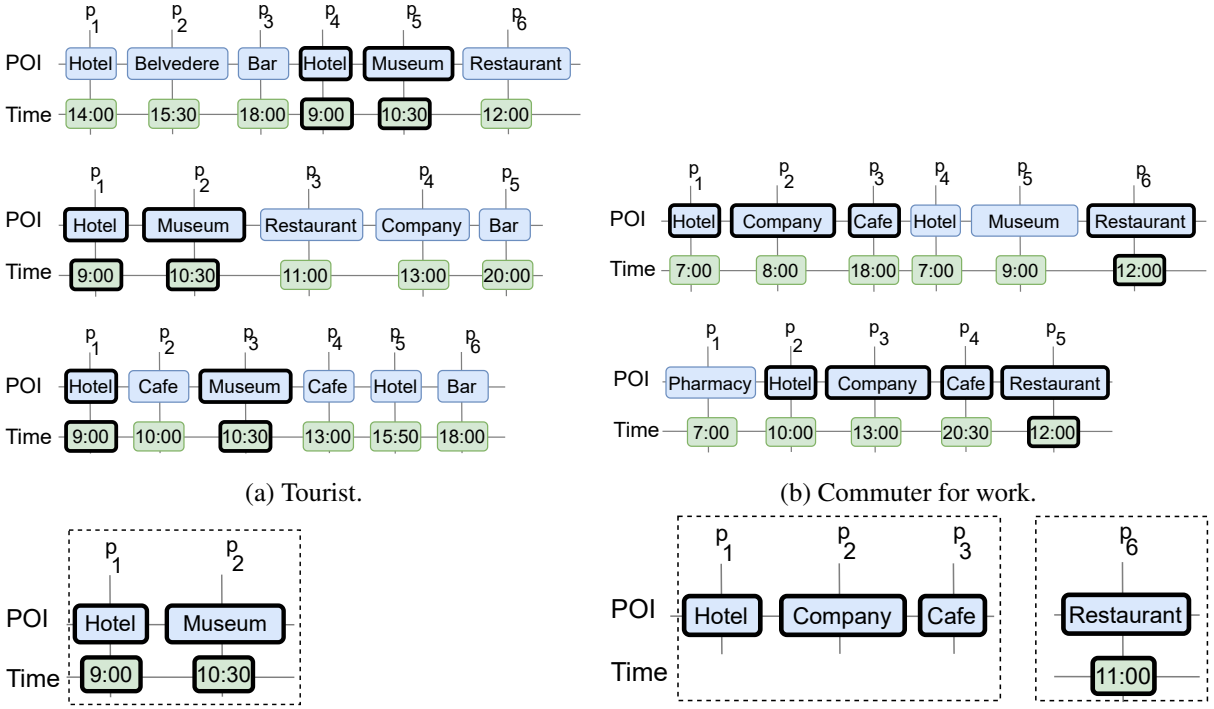
1.1 PROBLEM STATEMENT

For accurate trajectory classification in high dimensional trajectory datasets, the main challenge is to discover, in a feasible way, the most relevant subtrajectories and a subset of their dimensions that better characterize a class. The methods proposed by Ferrero in (FERRERO et al., 2018) and (FERRERO et al., 2020) have outperformed most existing methods for trajectory classification in terms of accuracy, showing that the best trajectory features for classification problems are subtrajectories of different sizes (e.g. one point, two points, etc) and some of their dimensions or dimension combinations.

Figure 5 exemplifies the trajectory classification problem, presenting one trajectory of a tourist (a) and another of a business commuter (b). The highlighted POI and Time attributes discriminate each different user. For example, to better discriminate between the tourist and the commuter, both visiting *Hotel*, the tourist discriminant subtrajectory is the sequence $\langle$ *Hotel at 9 am, Museum at 10:30* $\rangle$ while the commuter is $\langle$ *Hotel, Company, Cafe* $\rangle$, one with a subtrajectory of two POIs and the other with three POIs. Both users execute the sequence of POIs $\langle$ *Hotel, Museum* $\rangle$, but this sequence is less common to the commuter, i.e. a less relevant subtrajectory. The relevant subtrajectory of the tourist user (Figure 5 c) is also discriminant in the time dimension, as this sequential pattern happens in similar moments in time throughout the user trajectories. However, for the commuter, the relevant subtrajectory $\langle$ *Hotel, Company, Cafe* $\rangle$ highlighted on Figure 5 (d) is not time discriminant. Moreover, the tourist relevant subtrajectory in Figure 5 (c), has the same subtrajectory points visited in the user third trajectory, although the exact sequence does not occur. In the example, we notice that for different classes (the users), the relevant subtrajectories are of different sizes (different numbers of POIs) and a different number of dimensions (for one user the time is discriminant in the subtrajectory and for the other it is not). In this simple example, we notice that discovering relevant subtrajectories for each class is a challenging problem due to the vast number of subtrajectory and dimension combinations. Indeed, as we have seen in the example, the most discriminant subtrajectory size and its dimensions can be unrelated between different classes, presenting distinct ways of understanding their behaviors.

Figure 5 (d) presents the subtrajectories: $\langle$ *Hotel, Company, Cafe* $\rangle$ and $\langle$ *Restaurant at 11am* $\rangle$ that are most discriminant to identify the commuter user. To the best of our knowledge, the only method in the literature that automatically discovers the relevant subtrajectories and their dimensions is MASTERMovelets (FERRERO et al., 2020). MASTERMovelets is inspired by shapelets from time series literature, which are discriminant subsequences (ZHANG et al., 2018).

MASTERMovelets is currently the only method in the literature that presents a systematic strategy that automatically explores all possible subtrajectories (or subsequences) of any size (e.g. one point, two points, three points, etc.) and explores all dimension combinations (e.g. only space; space and time; space, time and POI category, etc), while seeking for the best feature to *each* class. Although it has a strategy for properly defining the most relevant features



(a) Tourist.

(b) Commuter for work.

(c) Relevant subtrajectories of (a).

(d) Relevant subtrajectories of (b).

Figure 5 – Examples of trajectories and subtrajectories of different users.

for MAT classification, its brute force makes it unfeasible for real datasets.

MASTERMovelets outperformed previous works for MAT classification with respect to accuracy (FERRERO et al., 2020; LEITE; PETRY; BOGORNY, 2019). However, MASTERMovelets is computationally expensive because it generates a large number of subtrajectories and all dimension combinations. Therefore, it suffers from the course of dimensionality due to the large number of movelet candidates. This problem leads to our research question: **Can we develop new algorithms for discovering relevant subtrajectories that become movelets that are faster than MASTERMovelets while maintaining or improving classification accuracy?** Our hypothesis is that we can reduce the explosion of dimension combinations and accelerate the segmentation process for movelet extraction.

1.2 OBJECTIVES AND CONTRIBUTIONS

The main objective of this thesis is to propose new methods for multiple aspect trajectory classification that may cope with the high computational cost and the large number of movelets of the state-of-the-art method MASTERMovelets while maintaining accuracy and reducing the computational cost. We aim to identify which are the most relevant *subtrajectories* that are likely to become movelets without exploring the entire dataset, thus reducing the search space for movelet discovery. As specific objectives we will:

- Propose and implement techniques for efficient multiple aspect trajectory feature extraction for classification reducing the search space and dimensionality;

- Evaluate the proposed techniques in several datasets and compare them to the state-of-the-art methods;
- Propose a tool to support the classification task of multiple aspect trajectories, specifically providing forms of visualizing *movelets* and trajectory data.

The main contribution of this thesis is a method for fast discovery of movelets and a significant reduction of the curse of dimensionality in terms of movelets that will be submitted to a classifier, called HiPerMovelets. A secondary contribution is the RandomMovelets and Ultra-Movelets for reducing the search space. The last is the prototype for multiple aspect trajectory data and movelet visualization.

1.3 METHODOLOGY AND THESIS STRUCTURE

To accomplish the objectives proposed in this thesis we adopted the following methodology:

1. Perform a literature review in trajectory classification, focusing on works that perform classification in multiple aspect trajectory data;
2. Propose and implement algorithms for reducing the MASTERMovelets search space and running time;
3. Evaluate the behavior of the proposed methods using real and synthetic trajectory datasets by including the classification accuracy, scalability, computational time, classification time, and the total number of generated movelet candidates and evaluated movelets;
4. Define and prepare the datasets to evaluate the methods and compare them to the state of the art;
5. Implement or adapt the state-of-the-art methods that have to be compared to our proposal;
6. Propose trajectory and subtrajectory visualization tools with support to classification and analysis of multiple aspect trajectories;
7. Write articles describing new methods for reducing the search space for extracting the relevant subtrajectories, and benchmark experiments;
8. Write the thesis by describing the main necessary concepts of trajectory data, the trajectory classification problem, the state-of-the-art, the description of the proposed solutions, experimental evaluations, and the conclusions obtained.

We limit the scope of this thesis to the definition of new optimization techniques for exploring relevant subtrajectories, in order to improve the performance of feature extraction for

trajectory classification over state-of-the-art methods. The rest of this thesis is structured as follows.

Chapter 2: BASIC CONCEPTS AND RELATED WORKS

We introduce and discuss the main concepts for understanding our methods, and presents the concepts of trajectories and subtrajectories, movelet candidates, best alignment, rankings, and movelets. We present the main works related to multiple aspect trajectory classification and the metrics commonly used for trajectory classification evaluation.

Chapter 3: STRATEGIES FOR REDUCING THE SEARCH SPACE IN MOVELET DISCOVERY

Presents a new high performance method for extracting movelets and classifying trajectories approach, called HiPerMovelets (High-performance Movelets). We present experimental results showing that HiPerMovelets is 10 times faster than MASTERMovelets, reduces the high dimensionality problem, is more scalable, and presents a high classification accuracy in all evaluated datasets with both raw and multiple aspect trajectories. We also present two new movelet extraction approaches UltraMovelets and RandomMovelets for multidimensional sequential data. UltraMovelets is an 'a priori' algorithm (AGRAWAL; SRIKANT, 1994) to movelet extraction which is parameter-free and able to reduce memory use. RandomMovelets evaluates movelet candidates randomly achieving good results in different types of datasets. We present experimental evaluation in several trajectory datasets (raw, semantic, and multiple aspect trajectories). In addition, we tested these methods in other domains like time series. Preliminary experiments demonstrate that RandomMovelets is promising to achieve good results in different dataset types. The strategy of UltraMovelets demonstrates limiting the combinatorial explosion without any need for parametrization.

Chapter 4: AUTOMATIZE: A PLATFORM FOR MOVELETS ANALYSIS

The chapter presents a tool, called AUTOMATIZE, to support the user in the classification task of multiple aspect trajectories, specifically for extracting and visualizing the *movelets*, the parts of the trajectory that better discriminate a class. The AUTOMATIZE integrates into a unique platform the fragmented approaches available in the literature for multiple aspects trajectories and, in general, for multidimensional sequence classification into a unique web-based and python library system. We present an analysis of movelets from the experimental evaluation of Subsection 3.2.2. We discuss the number of movelets, the average number of attributes, size, and quality of the movelets in several datasets. We also employ AUTOMATIZE visualization tools to discuss the attribute use of each movelet-based method and the attribute use by class labels on the Foursquare NYC dataset.

Chapter 5: CONCLUSION AND FUTURE WORKS

In the last chapter, we summarize the findings of this thesis and discuss future research opportunities in trajectory classification.

2 BASIC CONCEPTS AND RELATED WORKS

In this chapter, we first present the basic concepts to understand the movelet discovery for trajectory classification methods and discuss the related works summarizing their main characteristics. We then introduce and discuss the main concepts that will be used throughout the thesis, most of which are based on (FERRERO et al., 2020). The symbols used in this thesis are summarized in List of Symbols.

2.1 BASIC CONCEPTS

The first concept to understand this work is the trajectory data. A trajectory represents the movement of an object, called the *moving object*, during a certain period of time. The simplest form of representation is a Raw Trajectory, and it is defined by three attributes: x and y, denoting spatial dimension, and attribute t of time dimension, as presented in Definition 1. By dimensions we mean the different types or categories of the trajectory attributes space, time, and semantics.

Definition 1. *Raw Trajectory.* A raw trajectory T_i consists of a sequence of n points $T_i = \langle p_1, p_2, ..., p_n \rangle$, in which $p = \{(x,y),t\}$, where (x,y) is the position of the moving object in space and t is the timestamp that the point was collected.

Recently, with the definition of Multiple Aspect Trajectory (MAT) (FERRERO; ALVARES; BOGORNY, 2016; MELLO et al., 2019), besides space and time information, several heterogeneous attributes can be associated to each individual trajectory point, the *aspects*. This enrichment of trajectory data is especially important for social media as check-in data from Foursquare or Facebook. We discussed in Chapter 1 the increasing complexity of the trajectory data, resulting in very rich trajectories. For instance, the spatial dimension type is composed of the latitude and the longitude attributes, which must be considered together to represent a real position in space, and when associated with time, can generate numerical features such as speed, acceleration, traveled distance, etc. The spatial position of a point can be associated with a *Point of Interest (POI)* name or *category* (e.g. Hotel, School, Restaurant), and each POI can be associated with *price* and *rating*. Besides, the trajectory points are ordered in time, and the sequence is another crucial aspect that must be considered. A more formal definition of MAT is given in Definition 2.

Definition 2. *Multiple Aspect Trajectory.* A multiple aspect trajectory T_i is a sequence of m points $T_i = \langle p_1, p_2, ..., p_m \rangle$, with $p_i = (x,y,t,A)$ being the i-th point of the trajectory at location (x,y) at timestamp t, described by the set of l attributes $A = \{a_1, a_2, ..., a_l\}$, also called aspects.

In trajectory classification problems, the entire trajectory or its attributes are not sufficient to discriminate a class (FERRERO et al., 2018). It is necessary to discover the trajectory parts that are the most relevant and that represent the best features that characterize each class.

Therefore, trajectories need to be segmented into smaller parts, called subtrajectories. Trajectories can be segmented in many ways as period of time, change of direction, visiting area, etc. Subtrajectories are subsequences of a trajectory with a specific set of attributes, as shown in Definition 3.

Definition 3. *Subtrajectory.* Given a trajectory T_i of size m, a subtrajectory $s_{a,b} = \langle e_a, e_{a+1}..., e_b \rangle$ is a contiguous subsequence of T_i starting at element e_a and ending at element e_b, where $1 \leq a \leq b \leq m$. The subtrajectory $s_{a,b}$ can be represented by all attributes A or a subset of attributes $A' \subseteq A$. The length of the subtrajectory is defined as $w = |s_{a,b}|$. In addition, the set of all subtrajectories of length w in T_i is represented by $S^w_{T_i}$, and the set of all subtrajectories of all lengths in T_i is $S^*_{T_i}$.

It is important to point out that a subtrajectory is a part of the trajectory both in a subset of points and number of attributes. Behavior patterns are frequently represented as trajectory parts, i.e., subtrajectories and some of their attributes. Intuitively, if a subtrajectory is typical of a moving object, it can characterize its behavior. For instance, the movement from *Hotel* to *Touristic Place* is a subtrajectory characterizing a tourist behavior, while a subtrajectory from *Home* to *University* at *7 am* can characterize the behaviour of a student. Notice that these subtrajectories have different attributes, the first one has POI attribute while the other has POI and time.

Relevant subtrajectories or discriminant subtrajectories are examples of features that describe the movement behavior of an object of a certain class, and these relevant subtrajectories are called *movelet candidates* (Definition 4). Movelet candidate is one of the most important concepts in trajectory classification.

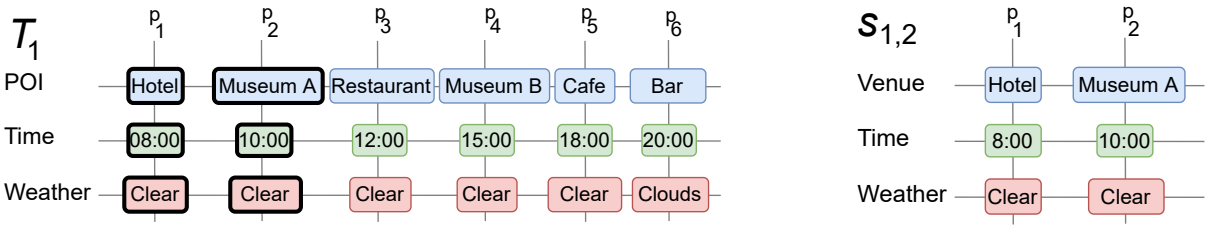
Definition 4. *Movelet Candidate.* A movelet candidate $\mathcal{M}$ from a subtrajectory $s_{start,end}$ is a tuple $\mathcal{M} = (T_i, start, w, C, \mathbb{W}, quality, sp)$, where T_i is a trajectory of the dataset $\mathbf{T}$; $start$ is the position in T_i where the subtrajectory begins, and w is the subtrajectory length ($w = |s_{start,end}|$); C is a subset of attributes such that $C \subseteq A$; $\mathbb{W}$ is a set of pairs $(W^s_{T_j}, class_{T_j})$, where $W^s_{T_j}$ is a distance vector of the subtrajectory $s_{(start,end)}$ to each trajectory T_j in $\mathbf{T}$. The set $W^s_{T_j}$ is the distance vector of the best alignment from the movelet candidate subtrajectory to each trajectory T_j, in each attribute of C. The best alignment reflects the position in a trajectory where the distance to a candidate $\mathcal{M}$ is minimized. The distances are calculated using the best alignment between $s_{(start,end)}$ and each trajectory T_j; *quality* is a relevance score given to the candidate $\mathcal{M}$; sp is a vector of distance values from $\mathbb{W}$ set, called split points, with one value for each attribute that better divides the classes (used to measure the candidate relevance).

The relevance of a movelet candidate is computed based on its distance to all subtrajectories of the same size, i.e., the same number of points, in the entire dataset.

In Ferrero et al. (2018) and Ferrero et al. (2020), movelet candidates are generated from all subtrajectory sizes (e.g. one point, two points, three points, etc) and attribute combinations. Most movelet-based methods generate candidates until the natural logarithm of the trajectory

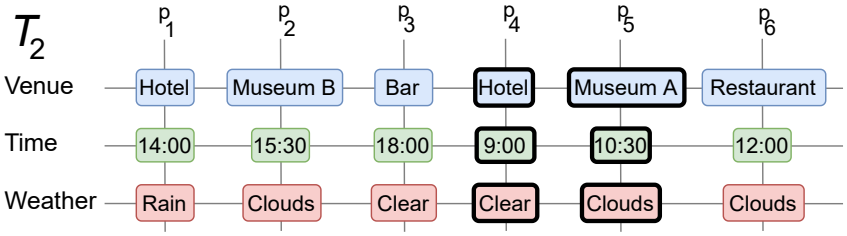
size for efficiency. The candidate generation creates an explosion of attribute combinations. This means that in a trajectory with ten points and three attributes, 55 subtrajectories will be generated, with 7 possible attribute combinations ($2^l - 1$, where l is the number of attributes), resulting in 385 *movelet candidates*. The high number of data dimensions leads to the exponential increase in computational efforts for its processing, which is a concept in data mining called the *Curse of Dimensionality*.

An important step in discovering the movelets is to find a subtrajectory that is more similar to another subtrajectory of the same size - with the same number of points - either in a subset or in all attributes. This is done by computing the distances between trajectories and subtrajectories Ferrero et al. (2020). When computing distances of trajectory and subtrajectories with several semantic attributes we have to take into account the nature of the attributes (i.e. numerical, textual, spatial, etc.). Therefore we need to use different distance metrics for each attribute. With the calculated distances we can compare subtrajectories in the multidimensional space. Thus, to compare trajectories it is essential to find two similar subtrajectories. The similarity of a subtrajectory to a trajectory is defined by the lowest distance values to the trajectory. This step is called *best alignment*, and is detailed in Figure 6. To illustrate the best alignment, consider the subtrajectory $s_{1,2}$ from trajectory T_1 (Figure 6 a and b), and the trajectory T_2 (Figure 6 c); $s_{1,2}$ best alignment in T_2 is represented by the sequence in which the subtrajectory attributes are most similar, highlighted in Figure 6 (c). The subtrajectories with the lowest distance values to the corresponding *best alignments* in other trajectories are similar subtrajectories because present the most similarities. Similar subtrajectories in the same class characterize a behavior pattern of the moving object.



(a) Trajectory T_1.

(b) Subtrajectory $s_{1,2}$.



(c) Subtrajectory best alignment highlighted in the trajectory.

Figure 6 – Example of subtrajectory best alignment in trajectory.

As illustrated by the example in 7 the *best alignment* consists in ranking the distances for all possible alignments and getting the average rank position with the lowest distance value. Figure 7 (a) exemplifies the distance of subtrajectory $s_{1,2}$ alignment in each starting position of

T_2 (represented as vectors). At position p_4 (Figure 7 a) the distance of the attribute *Venue* is 0, the distance in *Time* is 90 minutes (8:00-9:00 + 10:00-10:30) and the distance in *Weather* is 1. Figure 7 (b) shows the ranking where p_4 is the first for *Venue*, p_1 is the second, and p_2, p_3 and p_5 are the third. The best alignment corresponds to position 4, which has the lowest average rank.

Distance	Starting Position				
	p_1	p_2	p_3	p_4	p_5
Venue	1	2	2	0	2
Time	690	930	660	90	270
Weather	2	1	0	1	2
Vector	$\mathbf{V}_1$	$\mathbf{V}_2$	$\mathbf{V}_3$	$\mathbf{V}_4$	$\mathbf{V}_5$

(a) Distance values.

ranked by the lowest value $\Rightarrow$

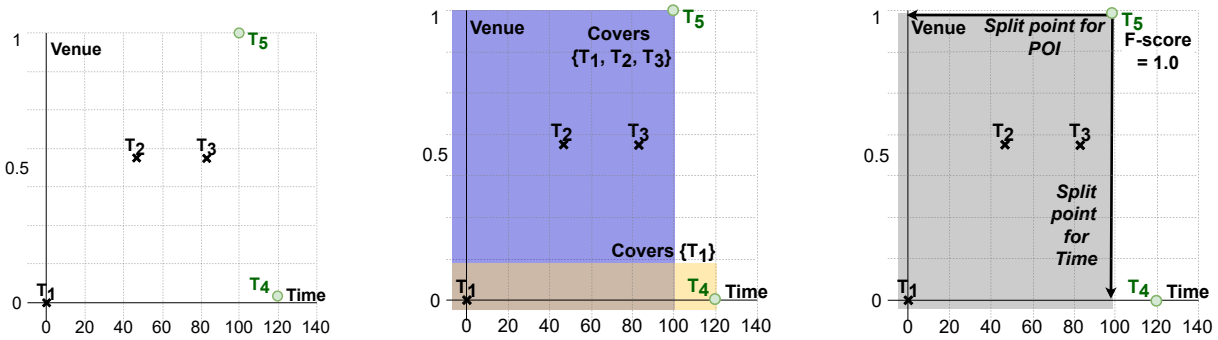
Rankings	Starting Position				
	p_1	p_2	p_3	p_4	p_5
Venue	2	3	3	1	3
Time	4	5	3	1	2
Weather	4	2	1	2	4
Avg. Rank	3.3	3.3	2.3	1.3	3.0

(b) Distance rankings.

Figure 7 – Finding the best alignment from the distance vectors.

The *movelet candidates* are evaluated by quality, and the better the quality is, the more discriminant they are for each class. The measure used to evaluate the *movelet candidate* quality is *F-score*. F-score is the harmonic average of Precision and Recall, where the Precision is the proportion of the trajectories of the class covered by the given *split point* values (a given point that divides the classes). The Recall is the proportion of trajectories of the class, covered by that *split point*, considering all the trajectories of the class in the dataset. A formal definition of F-score is given in the Subsubsection 2.2.0.1.

For finding the split point, Ferrero et al. (2020) considers all attributes of the movelet candidate being evaluated. Figure 8 (a) shows an example for two attributes (*POI* and *Time*). Paper (FERRERO et al., 2020) evaluates the distances from the movelet candidate (target class) to trajectories of other classes (non-target classes), which is a similar concept used in Support Vector Machines (SVM).



(a) Two-dimensional orderline.

(b) Evaluation of split points.

(c) Best split point.

Figure 8 – Example of finding split points in a *multidimensional orderline*.

Figure 8 (b) shows the distances of T_5 in both attributes (1 and 100) covering the trajectories from the target class (represented as $\mathbf{x}$ in the example) better than distances of T_4, that covers only T_1 while T_5 covers T_1, T_2 and T_3. So the split point values are defined by

the trajectory distances that give the best *F-score* (Figure 8 c). The split point, that is T_5 in the example, is the point that better separates the classes considering the attributes *POI* and *Time*. The *F-score* value represents the capability that a movelet candidate has to discriminate trajectories of its class from every other class. A movelet candidate that better discriminates a class is called *movelet*, as presented in Definition 5. Movelets are the subtrajectories with the highest quality without overlapping points in the trajectory they originated from.

Definition 5. *Movelet.* Given a trajectory T_i, and a movelet candidate $\mathcal{M}_x$ containing a subtrajectory $s_{a,b}$ ($s_{a,b} \subseteq T_i$), $\mathcal{M}_x$ is a movelet if for each movelet candidate $\mathcal{M}_y$ containing a subtrajectory $u_{f,g}$ with $u_{f,g} \subseteq T_i$ that overlaps $s_{a,b}$ in at least one point, $\mathcal{M}_x.quality > \mathcal{M}_y.quality$.

Once the movelets are extracted from the trajectory dataset it is necessary to organize them to apply the classifier. The input for classifiers is a matrix $|\mathbf{T}| \times |\mathbf{M}|$, where each row of the matrix is a trajectory from the set $\mathbf{T}$, and each column is a movelet from the set of movelets $\mathbf{M}$. The matrix values are 1 and 0, to represent the presence or absence of a movelet in a trajectory. A movelet is present in a trajectory when its best alignment distances to that trajectory are lower than the split point values. This means that the movelet *covers* the trajectory considering the movelet attributes (C) and split point (sp) values.

Table 1 shows the visual representation of the attribute-value matrix of movelets to use as input into a classifier, proposed by (FERRERO et al., 2020). The $v_{i,j}$ is the distance from trajectory T_i to *movelet* $\mathbf{M}_j$, and the attribute *class* represents the class label $class_{T_i}$ of the trajectory T_i. It is also possible to combine the movelet transformation matrix with other global and local features.

Table 1 – Attribute-value representation of movelet transformation matrix (FERRERO et al., 2020).

Trajectory	$\mathcal{M}_1$	$\mathcal{M}_2$	...	$\mathcal{M}_{\lvert\mathbf{M}\rvert}$	Class Label
T_1	$v_{1,1}$	$v_{1,2}$	...	$v_{1,\lvert\mathbf{M}\rvert}$	$class_{T_1}$
T_2	$v_{2,1}$	$v_{2,2}$	...	$v_{2,\lvert\mathbf{M}\rvert}$	$class_{T_2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
T_n	$v_{n,1}$	$v_{n,2}$	...	$v_{n,\lvert\mathbf{M}\rvert}$	$class_{T_n}$

2.2 TRAJECTORY CLASSIFICATION

As we have discussed before, movelets are important parts of trajectories, also called relevant features, used as input to classifiers in order to achieve the best accuracy possible. In this section, we briefly introduce the classification task and the main algorithms to be used in this work.

Classification is a well known task in data mining aiming to distinguish classes in a dataset. The objective of classification algorithms is to train a model able to assign a correct

label to an unlabeled set of dataset elements with minimal classification error, to identify the correct label of the moving object of a given trajectory (LEE et al., 2008; LEITE; PETRY; BOGORNY, 2019). The model is trained with labeled dataset elements, which requires the dataset to be partitioned in *training* and *testing* subsets. The class labels are known by the classification algorithm in the *training* subset, and are used to induce learning of the classification model. Instead, the *testing* subset class labels are unknown, and this subset is used to validate the classification model ability to correctly label elements. This dataset partition is commonly made in two ways: (a) a *hold-out* division of 70% elements to training and 30% elements to testing sets, or (ii) in a *k-fold cross validation* that splits the dataset into k parts: the training set is composed of $k-1$ parts and one part is left for testing. This method repeatedly changes the sets to build classification models aiming to test the classification algorithm in all data. The important aspect in partitioning the dataset is to respect the class balance in each subset so there will be enough examples in both training and testing subsets.

Several surveys pointed to trajectory classification as one of the most important tasks in data mining due to the complexity of learning the movement patterns of objects (ZHENG, 2015; BIAN et al., 2019; LEITE; PETRY; BOGORNY, 2019). Most works in the literature propose to build classification models from a feature representation of trajectories, which is a set of global and local features extracted from trajectories. Importantly, most trajectory classification methods do not propose new classifiers, but new ways of extracting features from trajectories.

There are several methods that build classifiers starting from the feature representation of trajectories, including Random Forests, Decision Trees, Neural Networks, and Support Vector Machines, to name a few. Below we briefly introduce the ones we used in this thesis and report on related works.

Random Forests (RF) are an ensemble of **Decision Trees (DT)** (BREIMAN, 2001). DTs are algorithms for classification or regression that build a tree of simple decision rules from the parent attribute value. The rules are learned from the data features to build the tree with a set of if-then-else decision rules. The deeper the tree, the more complex the model. The leaf elements of the tree represent the classes, and a path from the root element to a leaf represents the set of rules applied in the classification process.

Neural Networks (NN) are biologically inspired models from synaptic activities of the neural system (KLEENE, 2016). A NN simulates the learning process of the brain with the proposal of neuron models. Several approaches to neural networks and optimization techniques have been developed in recent years, examples are Multi-layer Perceptron, Recurrent Neural Networks, and Word Embeddings. Besides less interpretability, in general, NN performs better in high dimensional spaces than symbolic models. The NN models capture the relation between the movelets and the classes better than RF models according to (FERRERO, 2020).

***k*-Nearest Neighbors (KNN)** is one of the first methods of trajectory classification (SHARMA et al., 2010). It is an algorithm that labels the trajectories from the testing set based on the trajectories proximity in the training set according to a specific distance or similarity measure. The KNN calculates the distance of non-labeled trajectories to all elements of the

training set, and the label is assigned by the majority labels of k trajectories with the lower distance to it. The support of efficient techniques for measuring the distance between trajectories is essential to the KNN classification.

Support Vector Machines (SVM) are classifiers based on finding the boundaries that better divide the classes (DRUCKER; WU; VAPNIK, 1999). It works by sorting elements in a space they are inserted and identifying the regions shared by elements of the same class. SVMs can be computationally expensive in datasets with large number of features.

Other traditional methods employ some **Statistical Analysis** to do classification, such as Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression (LR), and Bayes probabilistic models (JúNIOR; RENSO; MATWIN, 2017; ETEMAD; Soares Júnior; MATWIN, 2018; BIAN et al., 2019). In general, statistical analysis has to satisfy some assumptions of the data distribution to work properly.

2.2.0.1 Classification Metrics

Regarding the evaluation of the classification model, the most commonly used metrics are Accuracy (ACC) and F-Score. Accuracy measures the model ability to correctly label elements in the testing set. It is the total correctly labeled elements by the number of elements, as shown in Equation 2.1.

$$ACC = \frac{|correctly\ labeled\ examples|}{|examples\ in\ the\ test\ set|} \tag{2.1}$$

The accuracy is a ratio between the number of correctly labeled examples of the test set by the total number of examples being tested. The *F–Score* measures the capability of the classifier to label the elements inside each class. It is an harmonic mean of the precision and recall of all the classes, previously presented in Equation 2.2. The *Precision* (Equation 2.3) aims to measure how precise the model is, and *Recall* (Equation 2.4) metric is how much elements were correctly classified. The precision consists of the number of correctly classified samples in a class divided by the total number of samples classified as that class, while recall is the number of correctly classified samples in a class divided by the total number of samples of that class in the dataset.

$$F\text{-}score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall} \tag{2.2}$$

$$Precision = \frac{|true\ positive|}{|true\ positive| + |false\ positive|} \tag{2.3}$$

$$Recall = \frac{|true\ positive|}{|true\ positive| + |false\ negative|} \tag{2.4}$$

2.3 RELATED WORKS

Despite much research work has been done regarding trajectory classification, the focus has been on raw trajectories (BIAN et al., 2019). Multiple aspect trajectories present particular and specific challenges that require special treatment (FERRERO et al., 2018). A recent survey of the state of the art for both raw and multiple aspect trajectory classification can be found in Leite, Petry e Bogorny (2019).

Table 2 summarises trajectory classification works, and compares: the datasets used in each work, the evaluated classifiers, the classification purpose, and the compared methods of the state of the art. The table gives an overall view of the many trajectory classification approaches available in the literature, the kind of datasets, and classifiers they use. This offers intuition to choose the classifiers and datasets to use. We will describe the most related methods developed for raw trajectory data, and methods able to use some semantic information. We will discuss some techniques used in time series classification, a similar problem to multiple aspect trajectories. We will present some traditional similarity measures, and other statistical analyses, that have been used to perform trajectory classification. Then, we present the methods based on movelet discovery that are the most related to our approach.

Some works for trajectory classification were developed for solving a single problem, as transportation mode inference (ETEMAD; Soares Júnior; MATWIN, 2018; DABIRI; HEASLIP, 2018; XIAO et al., 2017), while the works of Santos, Jr e Alvares (2011), Patel et al. (2012), Ferrero et al. (2018), May Petry et al. (2020), Ferrero et al. (2020) were developed for general purposes, such as level of hurricane, animal types, transportation modes, trajectory users, etc.

Trajectory classification can also be performed with similarity measures, such as KNN classification. Examples are similarity measures developed for sequential data are Longest Common Subsequences (LCSS) (VLACHOS; KOLLIOS; GUNOPULOS, 2002), Edit Distance for Real Sequences (EDR) (CHEN; ÖZSU; ORIA, 2005), Dynamic Time Warping (DTW) (HOLT et al., 2007; SHOKOOHI-YEKTA et al., 2017). Other similarity measures were developed specifically for trajectory data as Uncertain Movement Similarity (UMS) (FURTADO et al., 2018), Multidimensional Similarity Measuring (MSM) (FURTADO et al., 2016), Stops and Moves Similarity Measure (SMSM) (LEHMANN; ALVARES; BOGORNY, 2019), Multiple Aspect Trajectory Similarity (MUITAS) (PETRY et al., 2019).

Methods for raw trajectory classification support only the spatial dimension, as TraClas (LEE et al., 2008), or support only space and time, considering features such as speed, traveled distance, acceleration, etc. The works that consider only space and time are different in considering: (i) local features (BOLBOL et al., 2012; DABIRI; HEASLIP, 2018; SOLEYMANI et al., 2014), (ii) global features (ZHENG et al., 2008; SHARMA et al., 2010; JúNIOR; RENSO; MATWIN, 2017), and both (iii) local and global features (DODGE; WEIBEL; FOROOTAN, 2009; SANTOS; JR; ALVARES, 2011; PATEL et al., 2012; XIAO et al., 2017; JIANG et al., 2017; DODGE; WEIBEL; FOROOTAN, 2009; ETEMAD; Soares Júnior; MATWIN, 2018).

Table 2 – Datasets and classifiers used by each state of the art method

Technique	Datasets	Evaluated Classifier	Classification Purpose	Compares to
TraClass (LEE et al., 2008)	Animals, Vessels, Hurricanes, Synthetic Dataset	SVM	General	None
(Zheng, 2008) (ZHENG et al., 2008)	Geolife	DT	Transportation Mode	None
(Dodge, 2009) (DODGE; WEIBEL; FOROOTAN, 2009)	Open Street Map and Eye-Track	SVM	General	None
NNTC[1] (SHARMA et al., 2010)	Milan	KNN	Road Vehicles	None
(Lee, 2011) (LEE et al., 2011)	Taxis from San Francisco, Synthetic Dataset	SVM	Road Vehicles	None
TRACTS (SANTOS; JR; ALVARES, 2011)	Animals, Vessels, Hurricanes	SVM, NN, Bayes	General	(LEE et al., 2008)
TCPR[2] (PATEL et al., 2012)	Geolife, Animals, Hurricanes, School Buses	SVM, DT, Bayes	General	None
(Bolbol, 2012) (BOLBOL et al., 2012)	Private Dataset	SVM	Transportation Mode	None
(Soleymani, 2014)[3] (SOLEYMANI et al., 2014)	Private Dataset	SVM	Medicated Fish	None
(Tragopoulou, 2014) (TRAGOPOULOU; VARLAMIS; EIRINAKI, 2014)	Private Dataset	RF, DT	Transportation Mode	None
CSCA[4] (VARLAMIS, 2015)	Private Dataset	RF, DT, KNN, SVM, NN	Transportation Mode	None
(Xiao, 2017) (XIAO et al., 2017)	Geolife	KNN, DT, SVM, RF	Transportation Mode	(ZHENG et al., 2008; DODGE; WEIBEL; FOROOTAN, 2009)
TrajectoryNet (JIANG et al., 2017)	Geolife	NN	Transportation Mode	(ZHENG et al., 2008)
ANALYTiC (JúNIOR; RENSO; MATWIN, 2017)	Animals, Vessels, Geolife	DT, Bayes, KNN, RF, LR	General	None
TULER, BiTULER (GAO et al., 2017)	Brightkite, Gowalla	SVM, NN, LDA	Users	None
(Dabiri, 2018) (DABIRI; HEASLIP, 2018)	Geolife	NN	Transportation Mode	(ZHENG et al., 2008; ENDO et al., 2016; WANG et al., 2017)
(Etemad, 2018) (ETEMAD; Soares Júnior; MATWIN, 2018)	Geolife	DT, RF, NN, Bayes, QDA	Transportation Mode	(ZHENG et al., 2008; ENDO et al., 2016; XIAO et al., 2017; JIANG et al., 2017; DABIRI; HEASLIP, 2018)
TULVAE (ZHOU et al., 2018)	Foursquare, Gowalla, Brightkite	LDA, DT, RF, NN	Users	(GAO et al., 2017)
MARC (May Petry et al., 2020)	Brightkite, Gowalla, Foursquare	NN	General	(GAO et al., 2017; ZHOU et al., 2018; FERRERO et al., 2018)
POI-F (VICENZI et al., 2020)	Brightkite, Gowalla, Foursquare	NN	General	(GAO et al., 2017; ZHOU et al., 2018; FERRERO et al., 2018)
DeepeST (A. de Freitas. et al., 2021)	Bightkite, Gowalla, Private Dataset	NN	General	(CHEN; GUESTRIN, 2016; BREIMAN, 2001; GAO et al., 2017)
Movelets (FERRERO et al., 2018)	Animals, Athens Vehicles, Hurricanes and Geolife	Bayes, DT, SVM	General	(LEE et al., 2008; DODGE; WEIBEL; FOROOTAN, 2009; ZHENG et al., 2008; XIAO et al., 2017)
MASTERMovelets (FERRERO et al., 2020)	Brightkite, Gowalla, Foursquare	NN, RF	General	(GAO et al., 2017; ZHOU et al., 2018)
SUPERMovelets (PORTELA et al., 2021)	Brightkite, Gowalla, Foursquare	NN	General	(FERRERO et al., 2020)
HiPerMovelets (ours) (PORTELA; CARVALHO; BOGORNY, 2022)	Brightkite, Gowalla, Animals, GoTrack, Vehicles	NN	General	(FERRERO et al., 2020)
RandomMovelets (ours)	Brightkite, Gowalla, Animals, GoTrack, Vehicles, and more	NN, RF	General	(VICENZI et al., 2020; PETRY et al., 2019; FERRERO et al., 2020; PORTELA; CARVALHO; BOGORNY, 2022)
UltraMovelets (ours)	Brightkite, Gowalla, Animals, GoTrack, Vehicles, and more	NN, RF	General	(VICENZI et al., 2020; PETRY et al., 2019; FERRERO et al., 2020; PORTELA; CARVALHO; BOGORNY, 2022)

Movelets (FERRERO et al., 2018) has outperformed the previous state of the art methods for classification RB-TB (LEE et al., 2008), TCPR (PATEL et al., 2012), Dodge (DODGE; WEIBEL; FOROOTAN, 2009), Zheng (ZHENG et al., 2010), Xiao (XIAO et al., 2017), and the similarity measures for raw trajectories: LCSS, EDR, and DTW.

The works of Tragopoulou, Varlamis e Eirinaki (2014) and Varlamis (2015) are the first

Table 3 – Movelet-based methods characteristics.

Method	Main Strategy	Subtrajectory Size	Number of Dimensions	Movelet Candidates
Movelets (FERRERO et al., 2018)	Full dataset comparison (points)	All	All	All
MASTERMovelets (FERRERO et al., 2020)	Full dataset comparison (points and dimensions)	All	All	All
MASTERMovelets-Log (FERRERO et al., 2020)	Limit size to the natural log of trajectory	Log limit	All	Reduced
MASTERPivots (Leite da Silva, 2020)	Selected pivot points for movelet candidate generation	Log limit	All	Reduced
SUPERMovelets (PORTELA et al., 2021)	Selected trajectory parts for movelet candidate generation	Pivots	All	Reduced
SUPERMovelets-λ (PORTELA et al., 2021)	Learned dimension combination limit	Pivots and Log limit	Learned limit	Reduced
HiPerMovelets (PORTELA; CARVALHO; BOGORNY, 2022) (ours)	In-class frequency ranking for movelet candidates selection	Log limit	All	10% at first
HiPerPivots (PORTELA; CARVALHO; BOGORNY, 2022) (ours)	In-class frequency selection of pivot points for movelet candidate generation	Pivots and Log limit	All	10% at first
RandomMovelets (ours)	Random pruning of movelet candidates	Log limit	All	10% at first
UltraMovelets (ours)	Quality criteria to limit movelet candidates generation by recursive search	Learned limit	Learned limit	Learned

to use some semantic information for classification. They use latitude, longitude, altitude, and time from smartphones, and use derived features like matching bus or metro lines, the speed, and others to classify the user movement in walking, running, or driving.

The algorithms Movelets (FERRERO et al., 2018) and MASTERMovelets (FERRERO et al., 2020) discover *relevant subtrajectories*, called movelets, without the need of extracting other features. Table 3 summarises the characteristics of movelet-based works for trajectory classification, including the new methods proposed and discussed in this thesis (see Chapter 3). The main idea of these methods is to find subtrajectories that characterize the classes, that are interpretable and discriminant behaviors. The MASTERMovelets achieves much better accuracy than Movelets. While Movelets encapsulates the distances of all trajectory dimensions in a single distance value, MASTERMovelets keeps distances in a vector using all data dimensions with the best combination. Ferrero (FERRERO et al., 2020), also proposed MASTERMovelets-Log, which reduces the running time of the MASTERMovelets by limiting the size of the movelet candidates to the natural logarithm of the trajectory size. However, the processing time and computational cost are still extremely high. MASTERMovelets (FERRERO et al., 2020) was developed for multiple aspect trajectory classification and has largely outperformed state of the art methods in terms of accuracy. Moreover, the *movelets* are interpretable results, giving insights about the data.

In Portela et al. (2021) we extended MASTERMovelets, called SUPERMovelets. This method optimizes the movelet search by selecting relevant subtrajectories that repeat totally or partially in a trajectory for extracting movelets. Repeated subtrajectories often represent the behaviors of a class. Thus, SUPERMovelets employs a pivot-based approach to reduce the search space, that selects only the most promising trajectory points to extract movelets (PORTELA et al., 2021). Additionally, it provides a method to limit the number of semantic

dimensions for movelets further improving the performance.

The work of Vicenzi et al. (2020) can perform much faster than Movelets (FERRERO et al., 2018), with similar classification accuracy for a single dimension. However, in order to use several dimensions, the user must manually test and select the best dimensions combination, while MASTERMovelets automatically selects the best dimension combinations with heterogeneous and different numbers of dimensions to generate *movelets*.

A recent work, DeepeST (A. de Freitas. et al., 2021) uses deep learning neural network method for trajectory classification. DeepeST method look for the optimal set of hyperparameters for each model. It encodes the spatial points by splitting the space into grid cells. MARC, proposed by May Petry et al. (2020), is another recent work that uses word embeddings (MIKOLOV et al., 2013) in a neural network classification. It encapsulates all trajectory dimensions: space, time, and semantics, and uses them as input to a neural network classifier. It is the first work to use the geoHash (NIEMEYER, 2008) on the spatial dimension, combined with other dimensions. MARC outperforms the Movelets (FERRERO et al., 2018) with very high accuracy when using all dimensions, as Movelets is unable to choose the best dimension. However, the resulting patterns of MARC are not interpretable, as the classifier is limited to neural networks.

Movelets and MASTERMovelets are inspired by shapelets from time series literature. Time series is a similar problem to multiple aspect trajectory, as it is a sequence of discrete time ordered data. For this reason, we introduce some other optimization techniques from time series literature. Several optimization techniques are used in time series shapelets, for instance, early abandoning (YE; KEOGH, 2011; MUEEN; KEOGH; YOUNG, 2011), SAX encoding techniques to reduce the search space and run time (RAKTHANMANON; KEOGH, 2013), and sampling the dataset for fast candidate search (JI et al., 2019). The early abandoning technique is not useful for movelets because multiple aspect trajectories are in general more sparse than time series, and this technique is interesting for sequences with high number of points.

Some works used statistical analysis, such as an Analysis of Variance A(NOVA) Zuo, Zeitouni e Taher (2019). In time series, numerical values and statistical analysis can be used to identify regions from which to extract shapelets that have discriminant power over other classes and to filter redundant and low-quality candidates. Works as Zhang et al. (2018) used Local Fisher Discriminant Analysis (LFDA) to determine the most relevant parts of the time series from which to extract shapelet candidates that are most likely to achieve good classification results. The works in time series, as those mentioned above, generally search discriminant shapelets in one dimension individually, while MASTERMovelets evaluates the dimensions individually and combined with others.

2.4 SUMMARY

In this chapter we presented the basic concepts related to trajectory and relevant subtrajectory representations, movelet discovery, and evaluation metrics. We discussed how movelets

are evaluated, the search for best alignment, and split points. Those parts are important steps to movelet discovery, which is a similar way to separating the classes as done by the Support Vector machines. We presented how movelets represent the complex trajectory data in features being transformed into an attribute-value matrix for the classifier.

In this chapter, we also discussed several techniques for trajectory classification. We summarized the techniques by the datasets employed, the evaluated classifier, its classification purpose, and to which works it is compared. We started describing methods developed for raw trajectory data, and methods able to use some semantic information. We discussed some techniques used in time series classification, a similar problem to multiple aspect trajectories. We presented some common similarity measures, and other statistical analyses, that have been used to perform trajectory classification. Then, we presented methods based on movelet discovery, other recent works, and methods developed for transportation mode inference. We finally presented the characteristics of movelet-based methods, including our methods presented in Chapter 3.

We observe that most works in the literature focus on extracting local or global features to perform trajectory classification, which is not very informative of the behaviors of a class. Another drawback is that only a few works in the literature can use multiple trajectory dimensions other than space and time. Discovering relevant subtrajectories for classification, such as movelets, has the potential to reveal behavior patterns of classes. However, it is a computationally expensive task. To the best of our knowledge, our work is filling a gap in the literature providing efficient methods for discovering movelets, and ways to better understand behavior patterns.

3 STRATEGIES FOR REDUCING THE SEARCH SPACE IN MOVELET DISCOVERY

As discussed in Chapter 2, existing methods for movelet extraction explore all combinations of subtrajectories of different lengths and their attributes to find the most discriminant movelets for classification problems. That implies a combinatorial explosion and requires more computational resources as the volume of data increases.

This chapter presents the new methods developed during the Ph.D. for discovering movelets with different strategies to reduce the search space. The main idea is to avoid comparing all subtrajectories in the dataset and, as a consequence, improve the computational performance compared to the state-of-the-art methods. The first method, introduced in Section 3.1, is called **HiPerMovelets** (PORTELA; CARVALHO; BOGORNY, 2022), which explores the trajectories inside a class pruning the redundant ones. We evaluate its performance by comparing the accuracy and running time to the state-of-the-art and its scalability with increasing points, trajectories, and dimensions.

HiPerMovelets outperforms state-of-the-art methods in terms of computing time while preserving high accuracy. However, not always the best discriminant patterns are frequent. For this reason, in Sections 3.2.1 and 3.2.2, respectively, we introduce and discuss two other methods, called RandomMovelets and UltraMovelets, to limit the search space and the number of attribute comparisons. Preliminary experiments show that both methods are very promising for trajectories and sequential data from other domains.

3.1 HIPERMOVELETS: IN CLASS PRUNING AND FREQUENCY-BASED STRATEGY FOR MOVELET DISCOVERY

The method presented in this section is the main contribution of the thesis. A new algorithm, called HiPerMovelets, reduces the search space in movelet discovery by looking for relevant subtrajectories inside a class and does an early redundant trajectory pruning. We recall that relevant subtrajectories mean trajectory parts that can discriminate classes. In general, the method (i) discovers relevant subtrajectories using a frequency-based ranking in the subset of trajectories of each class and (ii) prunes the entire trajectories from the search when a trajectory meets the coverage criteria by the movelets already discovered. HiPerMovelets (PORTELA; CARVALHO; BOGORNY, 2022) performs faster than MASTERMovelets even in raw trajectory datasets and significantly reduces the number of movelets and the classification running time. HiPerMovelets is inspired by the greedy search algorithm for feature selection proposed in Gigli et al. (2016), and its main advantage is simplicity.

Figure 9 gives an overview of HiPerMovelets, which is composed of four phases named: *HiPerMovelet Candidate Generation, HiPerMovelet Candidate Pruning, HiPerMovelet Generation, and Trajectory Pruning*, each consisting of one or more steps. Given a dataset of **T** trajectories, the algorithm performs the following steps:

- **Step 1:** selects the trajectories **T'** of one class;
- **Step 2:** for each trajectory T_i in the class trajectory dataset **T'**, all remaining steps from 2 to 9 are applied;
- **Step 3:** generates all HiPerMovelet candidates for the trajectory T_i, where a HiPerMovelet candidate is each subtrajectory with one of the possible dimension combinations;
- **Step 4:** for each HiPerMovelet candidate it computes the relative frequency with respect to the trajectories **T'** of the same class, which is computed by the quality defined in Equation 3.2;
- **Step 5:** does the HiPerMovelet candidate pruning, where candidates with quality lower than τ are removed;
- **Step 6:** The remaining candidates are compared to all trajectories in the dataset **T** to define which of the candidates become HiPerMovelets for T_i;
- **Step 7:** Their *F-Score* quality is computed and then HiPerMovelets with low *F-Score* are pruned. The HiPerMovelet pruning is the selection of the HiPerMovelet candidates with *F-Score* higher than zero and with none overlapping points, which are the resulting HiPerMovelets of T_i;
- **Step 8 and 9:** with the resulting HiPerMovelets of T_i, their covered trajectories are removed from the **T'**, thus reducing the number of trajectory comparisons.

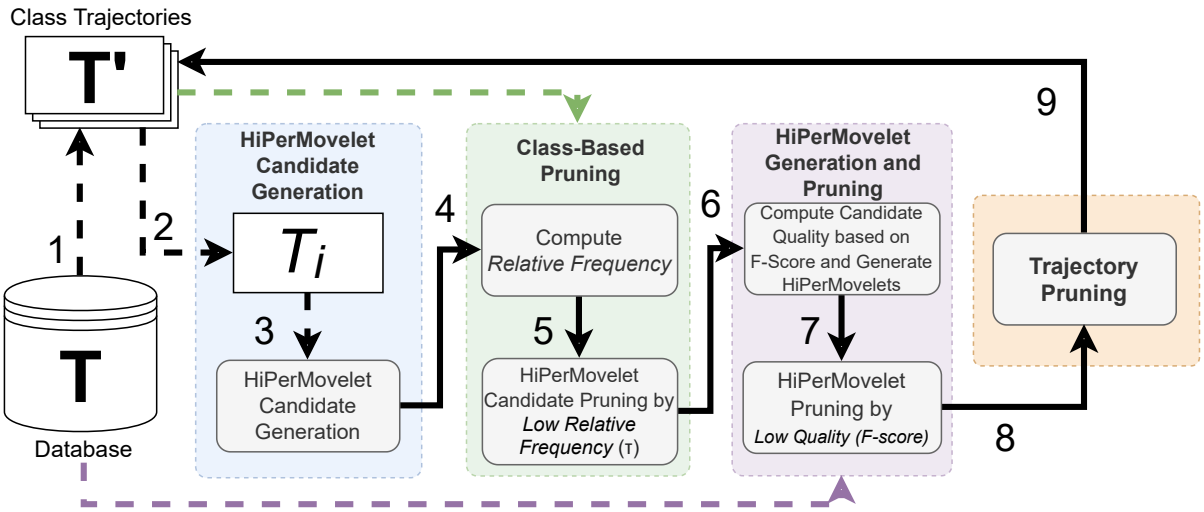


Figure 9 – Overview of the HiPerMovelets method.

Algorithm 1 summarizes the HiPerMovelets method. The input is a set of trajectories **T**, and the output is the set of *HiPerMovelets* **M** extracted from each class. It starts searching for relevant subtrajectories inside the trajectories of each class (line 2); it fills a queue with the trajectories **T'** (line 3), and extracts movelet candidates of one trajectory at a time, with

subtrajectories of trajectory T_i (line 6). We propose two implementations to movelet candidate generation, with pivots and without pivots, as detailed in Subsection 3.1.1. Each *movelet candidate* is qualified with its relative frequency inside the class. The movelet candidates are then filtered by the value of τ, which is learned for each trajectory and consists of 90% of the relative frequency from the best qualified candidate (lines 8 and 9). This step is known as early pruning. Only the movelet candidates with τ relative frequency or higher are compared to all subtrajectories of $\mathbf{T}$ for discovering the *movelets*. This filter reduces the number of candidates, keeping only the most probable to become movelets. The movelet candidates are computed and filtered with the *F-Score* quality (lines 13-14). It is necessary to compare each candidate with the entire dataset, and by reducing the number of movelet candidates, this task can be done significantly faster. The candidates with the best *F-Score* quality that do not share points become HiPerMovelets.

Algorithm 1: HiPerMovelets

Input: $\mathbf{T}$ Trajectory dataset
Output: $\mathbf{M}$ Set of *HiPerMovelets*

1 $\mathbf{M} \leftarrow \emptyset$
2 **foreach** *class trajectories* $\mathbf{T'} \in \mathbf{T}$ **do**
3 $queue \leftarrow \mathbf{T'}$
4 **while** $queue \neq \emptyset$ **do**
5 $T_i \leftarrow queue.pop()$
/* 1. HiPerMovelet Candidate Generation: */
6 $candidates_{T_i} \leftarrow$ MOVELETCANDIDATEGENERATION$(T_i, \mathbf{T'})$ /* Subsection 3.1.1 */
/* 2. HiPerMovelet Candidate Pruning - by frequency (or 10% of the candidates): */
7 $candidates_{T_i}.sort()$
8 $\tau \leftarrow candidates_{T_i}[0].quality * 0.9$ /* τ is learned based on highest frequency */
9 $candidates_{freq} \leftarrow \{\mathcal{M}_i \in candidates_{T_i} \mid \mathcal{M}_i.quality \geq \tau \text{ and } i \leq |candidates_{T_i}| * 0.1\}$
/* 3. HiPerMovelets Generation: */
10 $\mathbf{M}^{T_i} \leftarrow \emptyset$
11 **while** $\mathbf{M}^{T_i} = \emptyset$ ***and*** $candidates_{T_i} \neq \emptyset$ **do**
12 **foreach** $\mathcal{M}$ ***in*** $candidates_{freq}$ **do**
13 $\mathcal{M}.quality \leftarrow$ QUALITY$_{F-Score}(\mathcal{M}, \mathbf{T})$
/* HiPerMovelet Pruning - by *F-Score*: */
14 $\mathbf{M}^{T_i} \leftarrow$ MOVELETPRUNING$(candidates_{freq}, \mathbf{T})$
15 **if** $\mathbf{M}^{T_i} = \emptyset$ **then** /* Gets next subset of most frequent HiPerMovelet candidates */
16 $candidates_{T_i} \leftarrow candidates_{T_i} - candidates_{freq}$
17 $candidates_{freq} \leftarrow \{\mathcal{M}_i \in candidates_{T_i} \mid i \leq |candidates_{freq}| * 2\}$
/* 4. Covered Trajectory Pruning: */
18 $covered_{T_i} \leftarrow$ TRAJECTORYPRUNING$(\mathbf{M}^{T_i})$ /* Subsection 3.1.2 */
19 $queue \leftarrow queue - covered_{T_i}$ /* Removes trajectories already covered */
20 $\mathbf{M} \leftarrow \mathbf{M} \cup \mathbf{M}^{T_i}$
21 **return** $\mathbf{M}$

In its original formulation presented in Ferrero et al. (2020), *Movelet candidates* represent subtrajectories with different sizes and any dimension combination. In our proposal, we extend the definition of movelet candidate as given in Definition 6. In our method, a HiPerMovelet candidate is a subtrajectory that is frequent in the trajectories of a class and not in the entire dataset, significantly decreasing the relevant subtrajectory search space.

Definition 6. *HiPerMovelet Candidate.* A HiPerMovelet candidate is a tuple $\mathcal{M} = (T_i, s_{(start,end)}, C, \mathbb{W}, quality, sp, \mathbf{T}^{\mathcal{M}})$, where T_i is a trajectory from the set of class trajectories $\mathbf{T'}$; $s_{start,end}$ is

a subtrajectory extracted from T_i; C is a subset with the candidate attributes, and $C \subseteq A$ (where A is set of the trajectory attributes); $\mathbb{W}$ is a set of pairs $(W^s_{T_j}, class_{T_j})$, where $W^s_{T_j}$ is a distance vector that contains the distances of the best alignment of $\mathcal{M}$ to every trajectory **T'**, that are the same class as T_j. The distances are calculated using the best alignment between $s_{(start,end)}$ and each trajectory T_j in the attributes A; $\mathbf{T}^{\mathcal{M}}$ are the *covered trajectories*, i.e., a subset of class trajectories that contain the candidate $\mathcal{M}$ ($\mathbf{T}^{\mathcal{M}} \subseteq$ **T'**); sp is a set of distances $W^s_{T_j}$, with one value for each attribute, that better divide the classes (used to measure the candidate relevance), and *quality* is a relative frequency score given to the candidate $\mathcal{M}$.

The *quality score* of a HiPerMovelet candidate is based on the relative frequency (a proportion) that appears in its class trajectories. For each dimension of a HiPerMovelet candidate, we define a relative frequency as presented in Equation 3.1, which considers each dimension separately. Equation 3.1 describes the relative frequency function for a dimension k ($freq_k$) from a *HiPerMovelet candidate* $\mathcal{M}_j$ to the trajectories **T'** of a class:

$$freq_k(\mathcal{M}_x) = \frac{\sum_{i=1}^{i=|\mathbf{T'}|} \left(max(\mathbf{M}.\mathbb{W}[k]) - W^s_{T_i}[k] \right)}{max(\mathbf{M}.\mathbb{W}[k]).|\mathbf{T'}|} \tag{3.1}$$

Since the distances of the different dimensions can be in a different scale, to make the comparisons more accurate, we normalize the distance value by subtracting it from the maximum distance value ($max(\mathbf{M}.\mathbb{W}[k])$) in that dimension. The frequency is the average between the sum of the distances and the maximum possible sum of distances ($max(\mathbf{M}.\mathbb{W}[k]).|\mathbf{T'}|$). The quality of a HiPerMovelet candidate, given in Equation 3.2, aggregates the relative frequency of dimensions in one quality value and is the average proportion that a candidate has in trajectories **T'** of its class in each dimension k.

$$Quality_{RF} = \frac{\sum_{k=1}^{k=|C|} freq_k(\mathcal{M}_j, \mathbf{T'})}{|C|} \tag{3.2}$$

As there are different dimension combinations for a candidate, we use a '*soft*' approach for measuring its frequency and a '*hard*' approach to do *trajectory pruning* (Subsection 3.1.2). The $freq_k$ is a weighted average of all distances in each dimension. This way, even in cases where HiPerMovelet candidates are not exact matches (an exact match is when distances are all zeros), the method can still find the most similar ones since the quality is proportional to the maximum distances.

HiPerMovelet candidates are selected based on their *quality* (defined by the relative frequency), and only those with high (concerning a given threshold) relative frequency will become *HiPerMovelets*. Our method evaluates the HiPerMovelet candidates with at least τ relative frequency ($\mathcal{M}.quality \geq \tau$), where τ is calculated from the HiPerMovelet candidate with the highest score. This strategy in the proposed method limits the search for HiPerMovelets in the set of HiPerMovelet candidates with high relative frequency. The candidates quality is then evaluated using *F-Score* considering all trajectories in the dataset. The formal definition of a *HiPerMovelet* is given in Definition 7.

Definition 7. *HiPerMovelet.* Given a trajectory T_i, and a HiPerMovelet candidate $\mathcal{M}_x$ containing a subtrajectory $s_{a,b}$ ($s_{a,b} \subseteq T_i$), $\mathcal{M}_x$ is a movelet if for each HiPerMovelet candidate $\mathcal{M}_y$ containing a subtrajectory $u_{f,g}$ with $u_{f,g} \subseteq T_i$ that overlaps $s_{a,b}$ in at least one point, $\mathcal{M}_x.quality_{RF} \geq \tau$ and $\mathcal{M}_x.quality_{\text{F-Score}} > \mathcal{M}_y.quality_{\text{F-Score}}$.

Another essential strategy employed in the proposed method to reduce the trajectory comparisons is not to generate HiPerMovelet candidates from all trajectories in the database. Trajectories that contain the same HiPerMovelets are considered similar and therefore do not need to be analyzed several times. We call these trajectories *covered trajectories*, as defined in Definition 8.

Definition 8. *Covered Trajectory.* A trajectory T_j is covered by a HiPerMovelet candidate of a trajectory T_i when the same subtrajectory exists in T_j.

The HiPerMovelet candidates of a trajectory that cover another trajectory of the same class are likely to represent both trajectories. Therefore, it is unnecessary to search for HiPerMovelets in the covered trajectory. Covered trajectories are then ignored by our method, thus avoiding searching for HiPerMovelets in trajectories of the same class.

3.1.1 HiPerMovelet Candidate Generation

In the *HiPerMovelet Candidate Generation* step (Figure 9), HiPerMovelets generates all subtrajectories from trajectory T_i and only computes distances to trajectories **T'** of the same class. Thus its quality is based on the frequency inside the class. Although there is a combinatorial explosion, this step is limited to the number of class trajectories.

We propose two approaches for HiPerMovelet candidate generation: (i) without pivots and (ii) using pivots. The first one, detailed in Algorithm 2, generates all *HiPerMovelet candidates* of any size limited to the natural log of trajectory size. Starting from size one to the limit size m (line 3), it generates the HiPerMovelet candidates (line 4) and then calculates the relative frequency to each candidate (line 7).

Algorithm 2: MOVELETCANDIDATEGENERATION$(T_i, \mathbf{T'})$ Without Pivots

Input: T_i, A trajectory
T' Trajectory set for the class
Output: $candidates_{freq}$ Set of high relative frequency *HiPerMovelet candidates*

```
1 candidates_Ti ← ∅
2 m ← log2(Ti.length)
3 for size ← 1 to m do
4     candidates_size ← FINDCANDIDATES(Ti, ∅, size)
5     candidates_Ti ← candidates_Ti ∪ candidates_size
6 foreach M in candidates_Ti do
7     M.quality ← QUALITY_RF(M, T')
8 return candidates_Ti
```

The second strategy, using pivots, is exemplified in Figure 10, which shows a trajectory with seven points (Figure 10 a). This strategy finds *HiPerMovelet candidates* of size ranging

from one to the natural log of the trajectory size, and at each step, pruning the candidates by relative frequency. The first time, the algorithm generates the candidates of size one and filters them by relative frequency, as shown in Figure 10 (b) (suppose these points are p_2 and p_6). Subsequently, the algorithm looks for candidates of size two in the nearest neighbors (see Figure 10 (c)), either starting or ending at the points of the previous candidates of size one (the neighbor points p_1, p_3, p_5, and p_7). Then, the algorithm generates the candidates of size 2 ($p_1 \rightarrow p_2$, $p_2 \rightarrow p_3$, $p_5 \rightarrow p_6$, and $p_6 \rightarrow p_7$), prunes again by low relative frequency (candidate $p_2 \rightarrow p_3$ in Figure 10 d), and searches new candidates of size three in the neighborhood points (p_1 and p_4). Each candidate will generate two new candidates, one including the previous point and the other with the next point, similar to the idea of the nearest neighbor. Then, the algorithm removes candidates with overlapping points, keeping candidates with higher relative frequency.

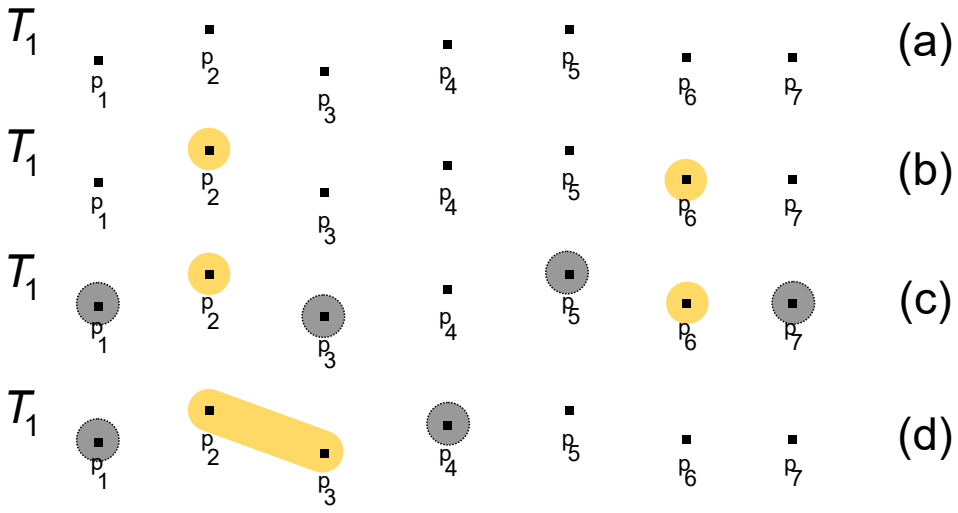


Figure 10 – (a) Trajectory T_1, (b) The pivots of size one, (c) The pivots neighborhood, and (d) one pivot of size two neighborhood.

Algorithm 3: MOVELETCANDIDATEGENERATION(T_i, **T'**) With Pivots

Input: T_i, A trajectory
T' Trajectory set for the class
Output: $candidates_{T_i}$ Set of high relative frequency *HiPerMovelet candidates*

1 $candidates_{T_i} \leftarrow \emptyset$
2 $m \leftarrow \log_2(T_i.length)$
3 **for** $size \leftarrow 1$ **to** m **do**
4 $\quad candidates_{size} \leftarrow$ FINDCANDIDATES$(T_i, candidates_{size}, size)$
5 $\quad$ **foreach** $\mathcal{M}$ ***in*** $candidates_{size}$ **do**
6 $\quad\quad \mathcal{M}.quality \leftarrow$ QUALITY$_{RF}(\mathcal{M}, \boldsymbol{T'})$
7 $\quad \tau \leftarrow candidates_{size}[0].quality * 0.9$
8 $\quad candidates_{freq} \leftarrow \{\mathcal{M}_i \in candidates_{size} \mid \mathcal{M}_i.quality \geq \tau\}$
9 $\quad candidates_{T_i} \leftarrow candidates_{T_i} \cup candidates_{freq}$
10 **return** $candidates_{T_i}$

Algorithm 3 presents the pivots strategy. First, this strategy generates the candidates of size one (line 4), assesses their relative frequency (line 6), and filters those with relative frequency lower than τ (line 8). Then, it restarts the loop looking for candidates of size two

from the neighborhood of the previous candidates of size one (line 4), and prunes again by low relative frequency (line 8). The FINDCANDIDATES function generates subtrajectories only from the previous size candidates by adding the nearest neighbor point before and after, increasing the size until it reaches the trajectory size limit m.

3.1.2 Class-based Pruning

From the discovered *movelets* of each trajectory, HiPerMovelets evaluates the covered trajectories (*Class-based Pruning* step in Figure 9). Algorithm 4 shows how covered trajectories are found from the movelets of a trajectory T_i (line 3). First, each trajectory is mapped with the number of occurrences in the movelets coverage (lines 4 and 5). Then, it selects the trajectories covered by at least half of the HiPerMovelets (lines 6-8), a reasonable value not to limit the search space excessively. If a class contains similar trajectories, then HiPerMovelets search space can be reduced as the HiPerMovelets will cover more examples (trajectories) in the class. However, classes with distant trajectories will need more examples to discover *HiPerMovelets*.

Algorithm 4: TRAJECTORYPRUNING($\mathbf{M}^{T_i}$)

```
   Input:  M^{T_i}                    Set of HiPerMovelets of trajectory T_i
   Output: covered                    Set of covered trajectories
1  covered ← ∅ ;
2  covered_count ← Map[] ;
3  foreach M ∈ M^{T_i} do
4      foreach T_x ∈ M.T^M do
5          covered_count[T_x]++ ;
6  foreach T_y ∈ covered_count do
7      if covered_count[T_y] > (|M^{T_i}|/2) then
8          covered ← covered ∪ T_y ;
9  return covered
```

3.1.3 Experimental Evaluation

We evaluate the proposed method with the Trajectory-User Linking (TUL), described in detail in Gao et al. (2017). This classification problem consists of classifying which user generated a given trajectory based on historical and labeled data. We also perform the tasks of transportation mean and animal species classification.

We compare HiPerMovelets (HM) and HiPerPivots (HP) computational cost and classification power w.r.t MASTERMovelets (MM). The method Movelets has already outperformed all similarity measures on KNN classification in Ferrero et al. (2018). Therefore we do not compare HiPerMovelets with similarity measures. First, we evaluate the computational cost of learning the model and compare the running times with MASTERMovelets-Log. Then, we compare the classification power with MASTERMovelets-Log and MARC on raw and MAT datasets, the datasets on which MASTERMovelets achieved the best results. We compare the

number of movelets and resulting movelets, from now on also referring to HiPerMovelets. In the following sections, we describe the datasets, the setup of the experiments, the obtained results, and the scalability of HiPerMovelets.

3.1.3.1 Datasets

We evaluate our method on five publicly available datasets[1] of multiple aspect trajectories (Gowalla, Brightkite), and raw trajectories (Animals, GoTrack, and Vehicles) also used in May Petry et al. (2020), Ferrero (2020), Leite da Silva (2020). We evaluate our method on raw trajectories consisting of space and time dimensions to demonstrate that it can achieve good results also in these types of data.

The multiple aspect trajectory datasets are from two different Location-Based Social Networks (LBSN) composed of check-in trajectories enriched with semantic attributes. The trajectories of all five datasets were split by week to increase the trajectory examples of each class label. Table 4 presents the characteristics of each dataset, with the average trajectory size, the number of trajectories, points, classes, and the attributes of the dataset.

Table 4 – Summary of the used trajectories datasets.

Dataset	Description	Dimensions
Gowalla	Traj Size: 18.42 ± 8.05 # of Traj.: 5,329 # of Points: 98,158 # of Classes: 300 Class Label: User	Lat, Lon, POI, Time, Weekday, User ID
Brightkite	Traj Size: 16.50 ± 7.07 # of Traj.: 7,911 # of Points: 130,494 # of Classes: 300 Class Label: User	Lat, Lon, POI, Time, Weekday, User ID
Animals	Traj Size: 146.96 ± 144.96 # of Traj.: 102 # of Points: 14,990 # of Classes: 3 (elk, deer, and cattle) Class Label: Animal Species	Lat, Lon, Time, Species
GoTrack	Traj Size: 111.09 ± 534.91 # of Traj.: 163 # of Points: 18,107 # of Classes: 2 (bus and car) Class Label: Transportation Mean	Lat, Lon, Time, Class ID
Vehicles	Traj Size: 467.98 ± 627.02 # of Traj.: 381 # of Points: 178,299 # of Classes: 2 (bus and truck) Class Label: Transportation Mean	Lat, Lon, Time, Class ID

Brightkite: dataset from Brightkite social media, collected between April 2008 and October 2010 (CHO; MYERS; LESKOVEC, 2011). It used a total of 300 users for analysis, randomly selected, with a total of $7,911$ trajectories, a minimum of 10 points, and a maximum of 50 points per trajectory for consistency. Trajectories provide the anonymized user of the check-in, the POI, space, and time information, enriched with the semantic weekday information.

[1] https://github.com/bigdata-ufsc/datasets

Gowalla: dataset from the Gowalla Location-Based Social Network (LSBN), collected between February 2009 and October 2010 (CHO; MYERS; LESKOVEC, 2011). It is a multiple aspect trajectory check-in dataset with anonymized users worldwide. We used a total of 300 random users for analysis, limiting the trajectory sizes minimum of 10 and a maximum of 50 points, resulting in a total of $5,329$ trajectories. It is composed of the exact dimensions of the Brightkite dataset, including the enriched semantic information of the weekday.

The raw trajectory datasets have much longer trajectories than the previous datasets, ranging from 100 points up to 600 in the Vehicles dataset. The **Animals**[2] dataset contains trajectories of three animal species that are the class labels: elk, deer, and cattle. The **GoTrack**[3] dataset from UCI repository contains GPS trajectories from the GO!Track app in the Northeast of Brazil. Trajectories are labeled by two transportation means: bus or car. The **Vehicles**[4] dataset contains trajectories of school buses and trucks collected in the Athens metropolitan area, and the class labels are bus and truck.

3.1.3.2 Experimental Setup

The datasets are split in a 5-fold hold-out proportion of 80% of the trajectories for training and 20% testing respecting the class balance (similar number of trajectories of each class). The reported results are the average of the 5-fold cross-validation. We evaluated two classification algorithms: the Neural Network Multilayer-Perceptron (NN) and the Random Forest (RF), as they are commonly used and achieved the best results in (FERRERO et al., 2020; May Petry et al., 2020; Leite da Silva, 2020). The classifiers were implemented using the Python language, with the `keras`[5] package in the same way as (FERRERO et al., 2020; Leite da Silva, 2020). The NN has a fully-connected hidden layer with 100 units, a Dropout Layer rate of 0.5, a learning rate of $10^{-}3$, and an Output Layer with *softmax* activation. We also applied Adam Optimization to improve the learning time and to avoid categorical cross-entropy loss, the same as in the work of Leite da Silva (2020), with 200 of batch size and a total of 200 epochs for each training. The Random Forest was built with 300 decision trees.

In MASTERMovelets and HiPerMovelets configurations, the distance was calculated in each dimension by using: (i) euclidean distance for the space, (ii) difference for the numerical, and (iii) simple equality (if is equal or not) for the semantics. In both methods, the size of the extracted subtrajectories was limited to the natural logarithm of the trajectory size as in (FERRERO et al., 2020; LEITE; PETRY; BOGORNY, 2019). For Gowalla and Brightkite datasets, the τ limit is 90% of the highest frequency, and in raw trajectory datasets, the τ is 50% in HiPerMovelets configurations.

The experiments were performed in an *Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz*,

[2] http://www.fs.fed.us/pnw/starkey/data/tables/
[3] https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories
[4] http://www.chorochronos.org/
[5] https://keras.io/

with 8 cores (limited to 4 threads) and **32GB** of main memory. The scalability experiments were performed in *Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz*, with 8 cores, limited to 4 threads, and **32GB** of RAM in Windows 10 Pro 64-bit.

3.1.3.3 Accuracy, Number of Movelets, and Processing Time

In this section, we compare the number of *movelet candidates* generated by the MASTER-Movelets-Log, HiPerMovelets and HiPerPivots, the number of movelets, the time spent in this task, the accuracy, the classification time of each classifier, and the number of compared trajectories by HiPerMovelets methods. The classification results in both classifiers, NN and RF, are given by the Accuracy metric. We compare the running time of HiPerMovelets and MASTER-Movelets-Log, limiting the computer resources to 4 threads for fair time comparison. MARC was the fastest method in all the experiments, but it had available all computer resources, with eight cores and GPU acceleration, thus, its running time is reported but not compared.

3.1.3.3.1 Results for Multiple Aspect Trajectory Datasets

We present the results for each dataset in Table 5, where the highest value for each metric is in bold, and the second best is underlined. We also present the comparison for the total number of evaluated movelet candidates and the number of movelets in Figure 11. The first observation is that NN achieves the same or better accuracy than RF. For both datasets, the accuracy values are very similar, but HiPerPivots achieves higher accuracy than HiPer-Movelets, generating significantly less *movelet candidates*, although it obtains more *movelets*. Consequently, with less *movelet candidates*, it is faster to extract movelets but slower when its movelets are used in the classification task because it generates a higher number of movelets. HiPerMovelets is mainly based on subtrajectory frequency for discovering the movelets, and therefore the patterns are different from MASTERMovelets in two ways: (i) the movelets represent the most frequent patterns that are discriminant for classes, and (ii) the reduced number of movelets, but still accurate results show that these patterns are more meaningful for trajectory classification.

Figure 12 presents the running time to discover movelets. Compared with MASTER-Movelets-Log, for the Gowalla dataset, HiPerMovelets running time reduced from hours to minutes, from more than 7 hours to 42 minutes, a reduction of more than 90%. The classification accuracy increased from 95.051 of MASTERMovelets-Log to 95.245 with HiPerMovelets and 95.271 with HiPerPivots. HiPerPivots performed better in both datasets, with an average time of 1h31 in the 5-fold experiments on Brightkite; and 33 minutes on Gowalla, a reduction in comparison to MASTERMovelets-Log of 91.45% (17 hours and 52 minutes) and 92.78% (7 hours and 39 minutes), respectively.

The reduction in running time is directly related to the reduction in the number of evaluated movelet candidates. Figure 13 shows the total number of compared trajectories (top)

Table 5 – Results for 5-fold cross-validation (MAT datasets).

Dataset		MARC	MASTER-Movelets	HiPerMovelets	HiPerMovelets-Pivots
Gowalla (specific)	Candidates	-	4,795,992	2,474,286	**297,338**
	Movelets	-	62,584	**30,851**	44,178
	ACC NN	94.760	95.051	95.245	**95.271**
	ACC RF	-	93.017	92.472	92.703
	Time (Movelets)	8m18s	7h39m50s	42m29s	**33m13s**
	Time (NN)	-	14m38s	**7m30s**	10m44s
	Time (RF)	-	15s	**10s**	12s
	Trajs. Compared	4,262	4,262	**2,325**	3,163
	Trajs. Pruned	-	-	1,937	1,099
Brightkite (specific)	Candidates	-	6,152,724	3,084,087	**579,544**
	Movelets	-	89,038	**42,907**	77,811
	ACC NN	96.052	**96.669**	96.171	96.387
	ACC RF	-	95.959	95.446	95.607
	Time (Movelets)	6m05s	17h52m03s	1h41m47s	**1h31m37s**
	Time (NN)	-	38m27s	**16m38s**	34m40s
	Time (RF)	-	22s	**15s**	21s
	Trajs. Compared	6,328	6,328	**3,405**	5,585
	Trajs. Pruned	-	-	2,923	743

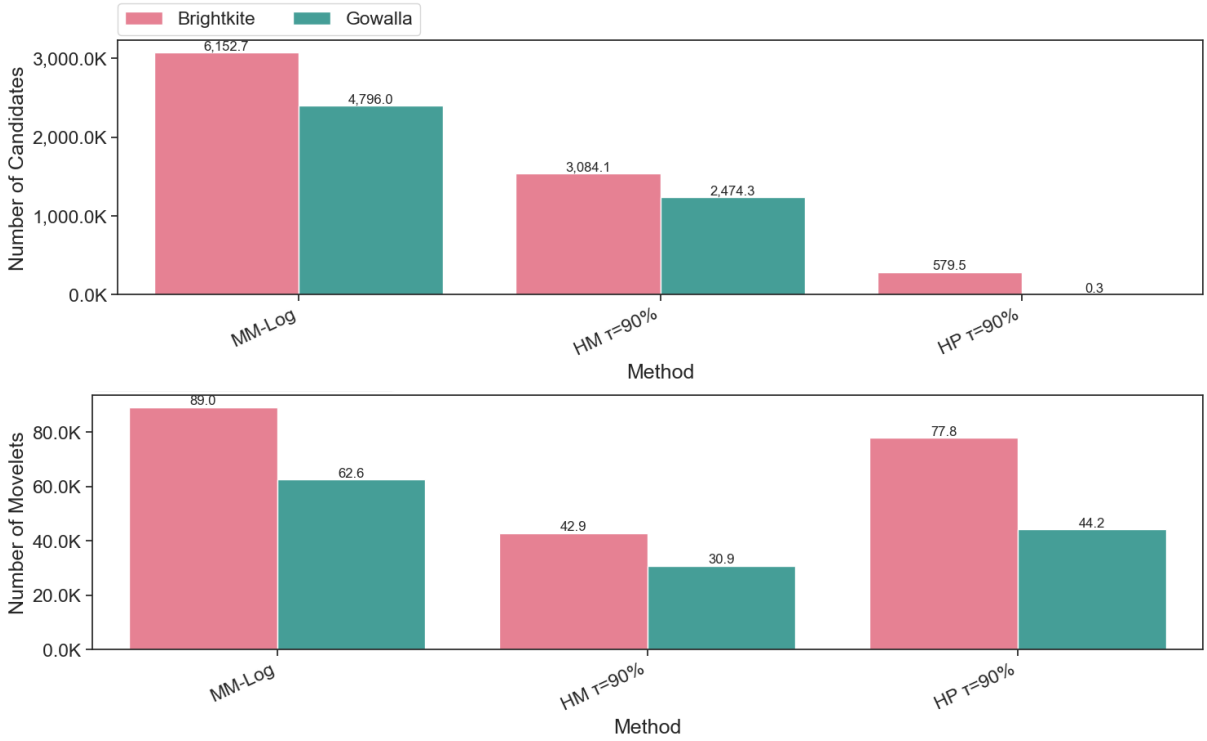


Figure 11 – Number of movelet candidates (top), and number of movelets (bottom) bar plots for all methods in multiple aspect trajectory datasets.

and the total number of pruned trajectories (bottom) of our methods. We can observe that HiPerMovelets prunes almost half of the trajectories in both datasets, reducing the number of movelet candidates by approximately half (Figure 11). Although HiPerPivots prunes a lower number of trajectories, it explores less movelet candidates because of the pivot strategy, thus resulting in faster running time.

In conclusion, the results confirmed that HiPerMovelets and HiPerPivots significantly

reduce the number of movelet candidates, the number of movelets, the movelet extraction time, and the classification processing time. It happens because of the HiPerMovelets candidate pruning strategy and because distances are computed among significantly fewer trajectories. The general processing time of HiPerMovelets reduced by more than 90% in both check-in datasets, decreasing the accuracy only in the Brightkite dataset and in less than 0.3% (from 96.669 to 96,387).

Table 6 – Results for 5-fold cross-validation (spatio-temporal datasets).

Dataset		MARC	MASTER-Movelets	HiPerMovelets	HiPerMovelets-Pivots
Animals	Candidates	-	232,896	216,531	**47,901**
	Movelets	-	6,523	**5,513**	7,943
	ACC NN	80.509	87.175	**89.281**	87.276
	ACC RF	-	88.276	86.170	88.075
	Time (Movelets)	23s	**1m05s**	3m15s	1m27s
	Time (NN)	-	**6s**	**6s**	7s
	Time (RF)	-	**1s**	**1s**	**1s**
	Trajs. Compared	81	81	75	**72**
	Trajs. Pruned	-	-	6	9
GoTrack	Candidates	-	286,447	242,146	**62,802**
	Movelets	-	6,873	**5,103**	8,516
	ACC (NN)	74.822	83.497	**84.636**	80.465
	ACC (RF)	-	79.838	82.853	76.750
	Time (Movelets)	1m02s	**1m28s**	7m09s	3m40s
	Time (NN)	-	**6s**	**6s**	7s
	Time (RF)	-	**1s**	**1s**	**1s**
	Trajs. Compared	130	130	**119**	123
	Trajs. Pruned	-	-	11	6
Vehicles	Candidates	-	3,294,900	709,367	**144,208**
	Movelets	-	98,670	**20,672**	26,876
	ACC (NN)	74.535	**98.947**	98.421	98.428
	ACC (RF)	-	98.147	97.887	98.147
	Time (Movelets)	3m43s	9h14m37s	3h05m24s	**1h09m15s**
	Time (NN)	-	35s	**11s**	13s
	Time (RF)	-	**1s**	**1s**	**1s**
	Trajs. Compared	304	304	111	**103**
	Trajs. Pruned	-	-	193	201

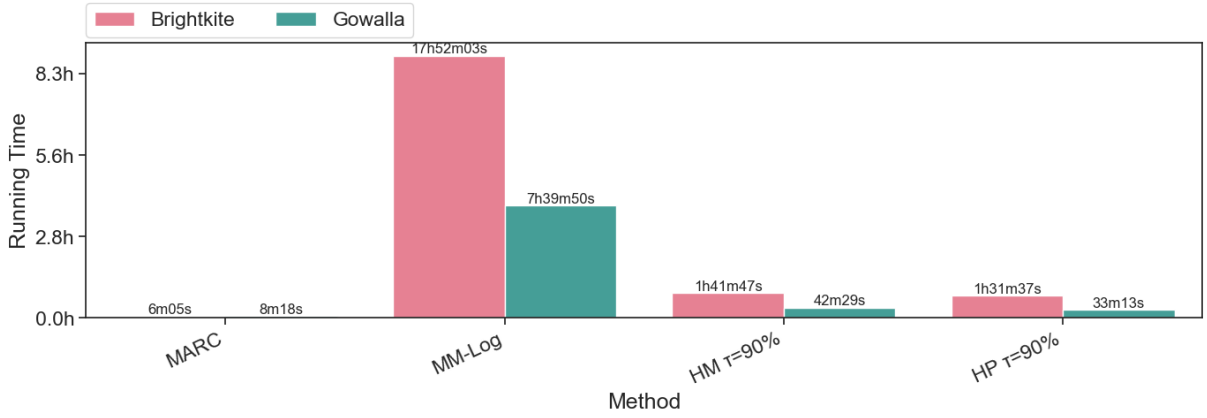


Figure 12 – Running time bar plots for all methods in multiple aspect trajectory datasets.

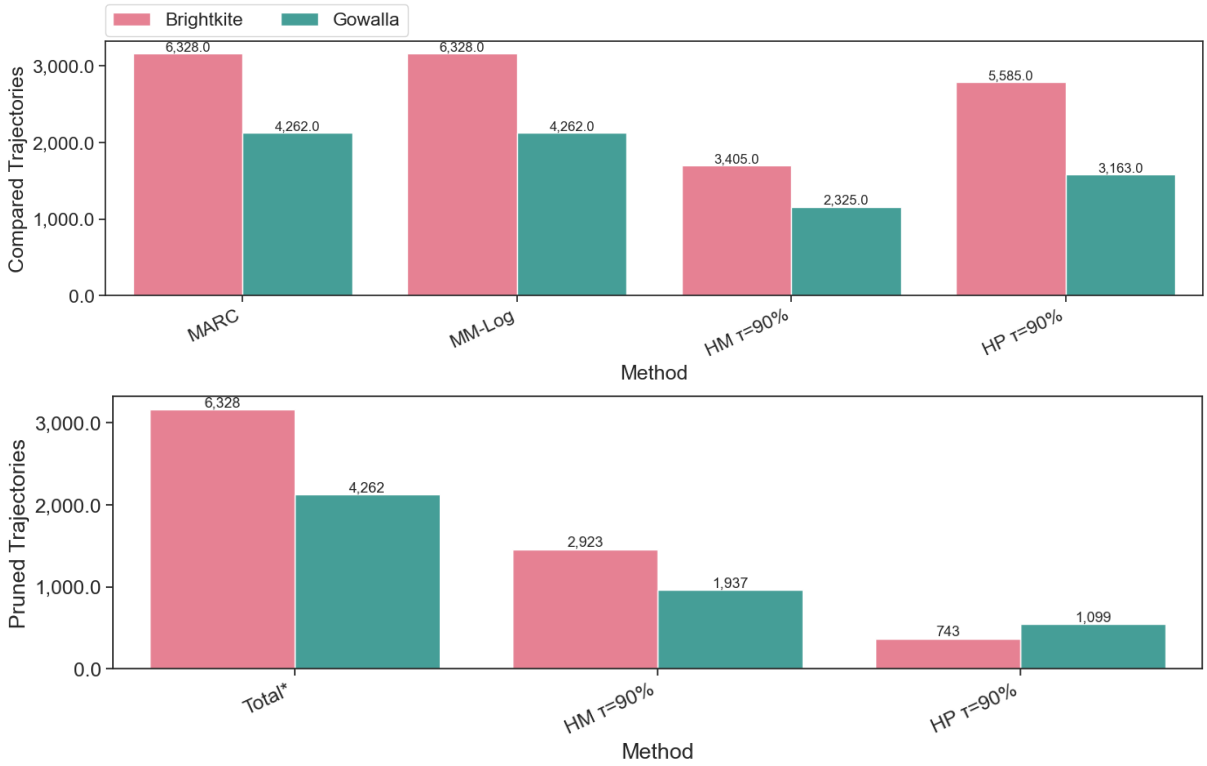


Figure 13 – Total compared trajectories (top), and pruned trajectories (bottom) in multiple aspect trajectory datasets.

3.1.3.3.2 Results for Raw Trajectory Datasets

In the raw trajectory datasets presented in Table 6, HiPerMovelets achieves either the same or higher accuracy results than MASTERMovelets. On the one hand, our method achieved a higher accuracy for smaller datasets such as Animals and GoTrack, although it was initially designed with a focus on performance. On the other hand, for these specific cases, the processing time of HiPerMovelets is greater when compared to MASTERMovelets (Figure 14). It happens because the low number of trajectories does not justify the computational overhead introduced by the additional calculations and mechanisms that try to reduce the amount of data analyzed for movelet extraction. The processing time of HiPerMovelets reduced by more than 66% in the Vehicles dataset, decreasing the accuracy only in 0.5%. Both HiPerMovelets and HiPerPivots classification accuracy is higher than MARC in all datasets. From this experiment, we note that the processing time is not a problem for small datasets. Both methods run in a few minutes, but our approach achieves the best accuracy, which is the most important in classification problems. When looking at the results on a larger dataset as the Vehicles, we can observe how HiPerMovelets took only one-third of the processing time and achieved a similar level of accuracy as MASTERMovelets.

The Animals and GoTrack datasets do not present a significant reduction in the number of evaluated trajectories or pruned trajectories, which explains the increase in running time. However, the vehicles dataset presents a more significant reduction in the number of compared trajectories (Figure 15) and, consequently, the running time. We observe that both HiPer-

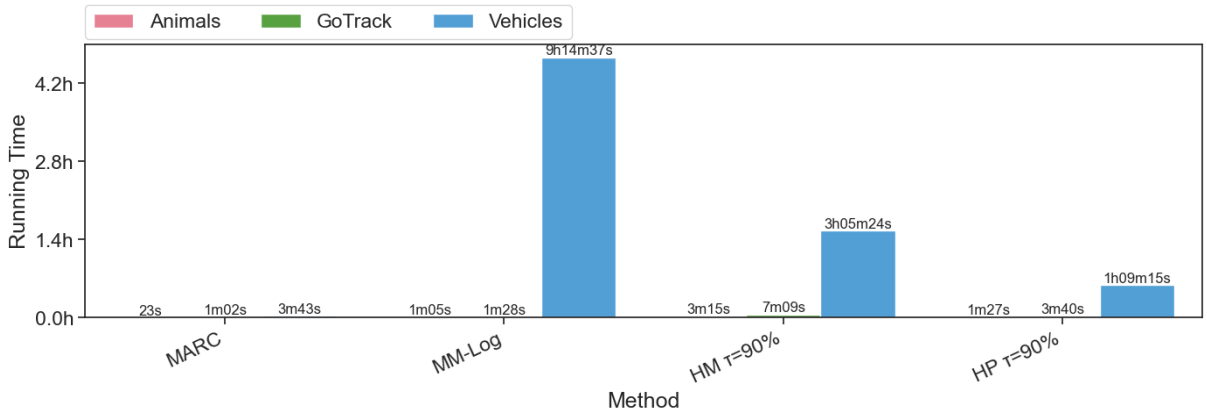


Figure 14 – Running time bar plots for all methods in raw trajectory datasets.

Movelets and HiPerPivots prune almost two third of the trajectories in the Vehicles dataset, significantly reducing the number of movelet candidates and movelets (Figure 16).

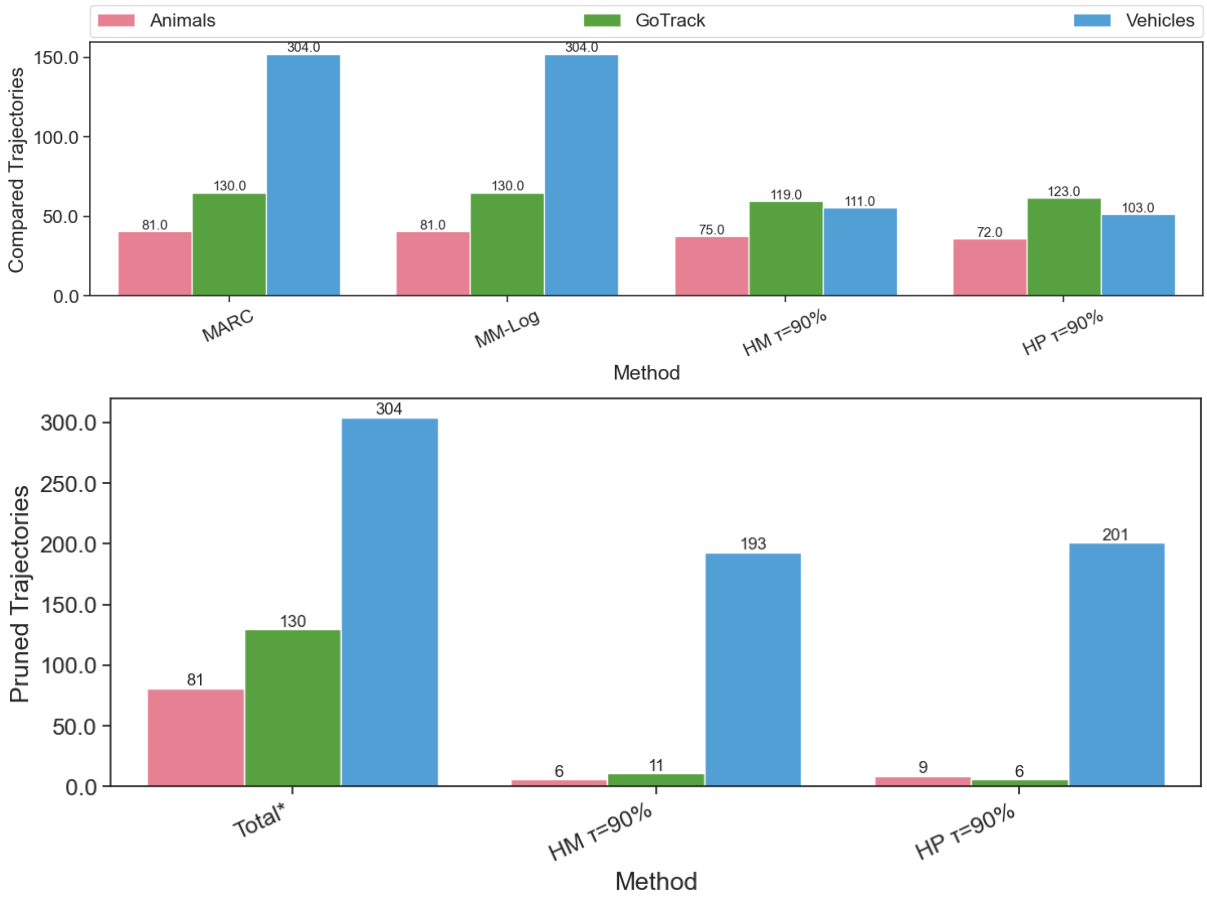


Figure 15 – Total compared trajectories (top), and pruned trajectories (bottom) in raw trajectory datasets.

HiPerMovelets and HiPerPivots present the most significant reduction in the number of movelet candidates and the number of movelets in the Vehicles dataset. The more significant reduction in the number of movelet candidates in that dataset is 95% for HiPerPivots and 72% in the number of resulting movelets. In conclusion, our methods can reduce the search space by early pruning the covered trajectories as an efficient strategy for discovering movelets.

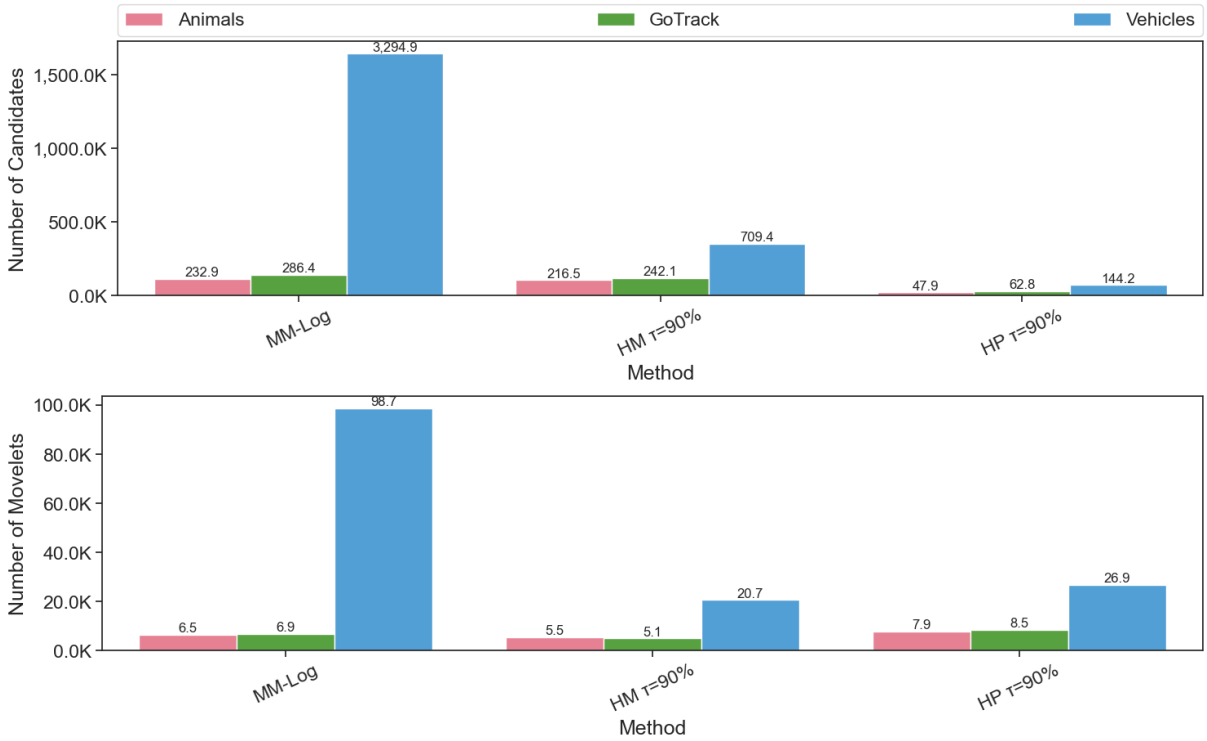


Figure 16 – Number of movelet candidates (top), and number of movelets (bottom) bar plots for all methods in raw trajectory datasets.

3.1.3.4 Scalability Analysis

In this experiment, we evaluate how HiPerMovelets scales in datasets with different characteristics, comparing the computational time spent in the *movelets* extraction. We used three synthetic datasets with different configurations inspired by the methodology adopted in Ye e Keogh (2011) and the same design as in Ferrero et al. (2020). We use three datasets: (i) with a fixed number of 200 trajectories and varying the trajectory size from 10 to 400 points and 1 dimension in every trajectory, (ii) with fixed trajectory size of 50 points and varying the number of trajectories from 100 to 2000 trajectories with 1 dimension, and (iii) with a fixed number of 200 trajectories and fixed trajectory size of 50 points, but varying the number of dimensions from 1 to 5. Fig. 17 shows the performance results.

In all experiments in Figure 17 we can see that both HiPerMovelets and HiPerPivots scale well as the dataset grows, spending less computational time in comparison to MASTERMovelets-Log. In Figure 17 (a), the time reduction of HiPerMovelets and for HiPerPivots is more than 99% in comparison to MASTERMovelets-Log when the number of points increases. In Figure 17 (b), we can see that HiPerMovelets presents a reduction of 17% when the number of trajectories increases from 100 to 2000, and HiPerPivots a reduction of 53.95%.

In Figure 17 (c), we compare the running time for our methods and MASTERMovelets-Log to show that MASTERMovelets-Log needs more time as the number of dimensions increases because it generates movelet candidates with all dimensions combinations. HiPerMovelets shows a reduction of 77.77% compared to MASTERMovelets-Log with five dimen-

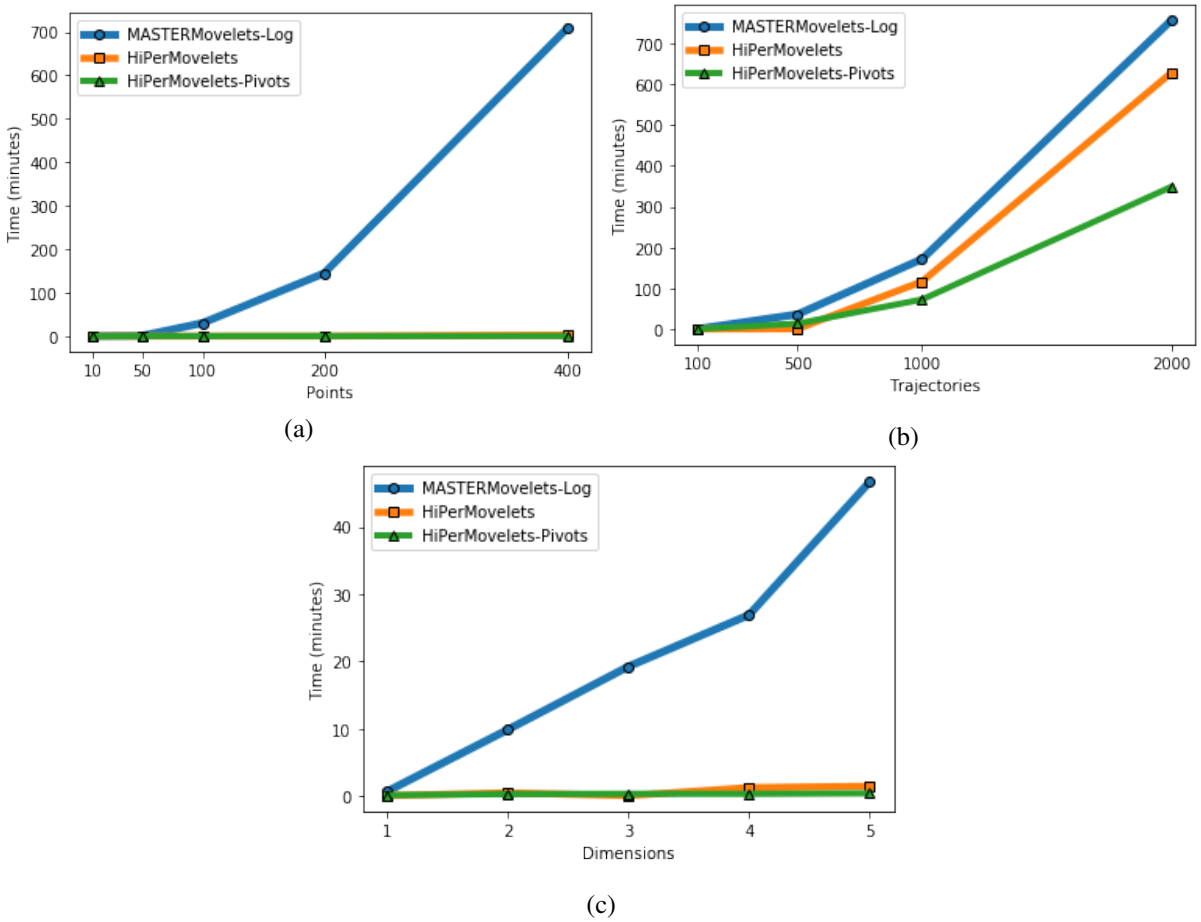


Figure 17 – Scalability analysis of running time varying (a) the number of trajectory points (b) the number of trajectories and (c) the number of dimensions.

sions. HiPerPivots, on the other hand, shows a reduction of 97.20%, and is not much affected as the number of dimensions increase. The higher the number of dimensions in a subtrajectory, the less common/frequent it will be because it requires a repetitive movement with exactly the same characteristics (e.g., visit Gym 8 am, three times a week, on weekdays). This is the main advantage of our proposal, as the frequency of subtrajectories tends to decrease as the number of dimensions increase, and they will either be pruned or not selected, the running time of HiPerMovelets grows from one to two dimensions (Figure 17 c), and then decreases.

We observed that HiPerMovelets performed very well in raw and multiple aspect trajectory datasets. The scalability of HiPerMovelets, especially HiPerPivots, allowed for a reduction in computational time compared to the state-of-the-art method for movelet discovery. It does not need to evaluate all movelet candidates but only the most frequent in class. However, we also observed in raw datasets (Table 6) that HiPerMovelets has some limitations when the value of the frequency bound τ is high, needing to adjust in order to select more movelet candidates for evaluation. In trying to overcome this limitation, we investigated new methods. We, therefore, investigated different movelet extraction criteria trying to reduce the search space in movelet discovery. We introduce and discuss in the following sections the UltraMovelets and RandomMovelets approaches.

3.2 REDUCING THE SEARCH SPACE AND ATTRIBUTE COMPARISON IN MOVELET DISCOVERY

Dealing with one class at a time can limit the performance depending on the number of classes, especially in binary classification. Another problem is that in some datasets, the frequency strategy adopted by HiPerMovelets might not be the best one, due to less frequent relevant subtrajectories. Indeed, most frequent subtrajectories are short in size. For such datasets, the frequency might not be the best strategy to identify discriminant movelets, and a random strategy gives the movelets the same chances to be selected but more efficiently.

Another problem is that the classification task becomes more challenging in generalized datasets. Generalized datasets are anonymized with the detailed information removed (e.g., use the label Hotel instead of Ibis Hotel; or remove the geographic coordinates) to make the problem more challenging or for privacy protection (PORTELA; VICENZI; BOGORNY, 2019), which implies that less specific information is available. For example, the dataset has the category of the places (e.g., Hotel) but not the specific instance name (e.g., North Beach Hotel). In generalized datasets, the most frequent patterns might not describe a class behavior due to the lower variability of the data. It means some datasets have a trade-off where the best discriminant subtrajectories are infrequent.

In this section, we introduce two more methods that we have investigated to cope with the limitations of HiPerMovelets: RandomMovelets and UltraMovelets. These methods are domain-independent, and the evaluation results are preliminary. The **RandomMovelets** main idea is a random pruning of movelet candidates of a trajectory before comparing them to the entire dataset. In this method, we do not use a ranking strategy to select movelet candidates but randomly choose a percentage of the candidates to calculate *F-Score* quality. That can limit the search space by drastically reducing the number of evaluated movelets. Random selection increases the chances of quickly selecting relevant movelets for general-purpose classification. Moreover, it is not class dependent as HiPerMovelets since it does not prune trajectories based on class movelets, processing each trajectory independently.

UltraMovelets selects subtrajectory pivots with high F-Score quality as starting points. The pivots are subtrajectories of one point and one dimension that recursively are incremented in points and dimensions, generating new movelet candidates. That allows the evaluation of the F-Score quality in each step, and to stop the search at any time based on the quality value. When the quality criteria is met, the movelets are selected abandoning, the search on longer subtrajectory combinations of points and attributes. Thus, this method can discover movelets without configuring a limit to the search by the size of the subtrajectory or the number of attribute combinations. UltraMovelets selects only the best combination of dimensions and subtrajectory size. It is parameter-free, essential since parameter definition is a well-known data mining problem, as parameters are difficult to estimate and directly impact the results. In addition, it generates subtrajectories one at a time, thus reducing memory use and making scalability easier.

A fundamental difference between the methods is that UltraMovelets extracts movelets that can share the same trajectory point but not the exact dimensions. On the other hand, RandomMovelets and HiPerMovelets do not extract movelets that share the same point. The following sections detail RandomMovelets and UltraMovelets.

3.2.1 RandomMovelets: A Random-based Movelet Candidate Pruning Strategy for Movelet Discovery

We believe that randomly selecting movelet candidates can increase the chances of finding movelets faster without the need of evaluating all possible movelet candidates. By randomly evaluating movelet candidates, we can reduce the search space without needing extra processing steps, such as frequency calculations and ranking.

Inspired by (RAKTHANMANON; KEOGH, 2013), we introduce a new method for fast discovery of *movelets*, called RandomMovelets. It is not purely random since the movelet candidates quality is evaluated, and the best ones become movelets. The main idea is to iteratively select an α number of movelet candidates to be qualified with *F-Score*, independent of any ranking strategy for candidate selection.

Figure 18 presents an overview of the RandomMovelets method. At first, for each trajectory T_i, it extracts all movelet candidates (steps 1-3), which are of any size and any combination of dimensions. Then, a α percentage of those candidates are chosen randomly (step 4). The *F-Score* quality is calculated for each selected candidate (step 5), and the ones that have insufficient quality are pruned (step 6). If movelets are not found, it will restart the third step by randomly selecting new candidates, twice as many, to increase the chances of finding movelets (same as HiPerMovelets). Movelets are found when the randomly evaluated candidates have good quality scores (step 7).

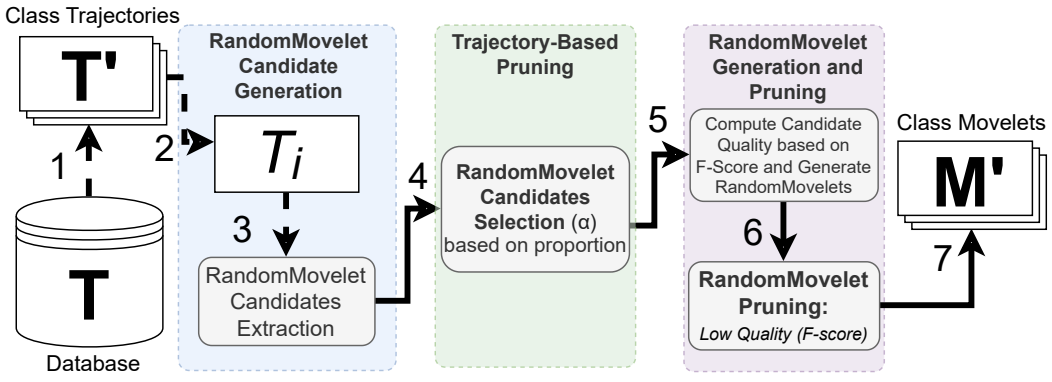


Figure 18 – Overview of the RandomMovelets method.

Before we go into the details of the algorithm, we must define the concept of RandomMovelet Candidate and RandomMovelet. The RandomMovelets candidates (Definition 9) are selected randomly from a set of all possible subtrajectories of a trajectory T_i.

Definition 9. *RandomMovelet Candidate.* A RandomMovelet candidate $\mathcal{M}$ from a subtrajectory $s_{start,end}$ is a tuple $\mathcal{M} = (T_i, start, w, C, \mathbb{W}, quality, sp)$, randomly selected from T_i, which is a trajectory of the dataset **T**; All terms are identically defined as the movelet in Definition 4.

The quality of a RandomMovelet candidate is then evaluated using *F-Score* considering all trajectories in the dataset. The best qualified randomly chosen candidates are *RandomMovelets*, a formal definition is given in Definition 10.

Definition 10. *RandomMovelet.* Given a trajectory T_i, and a randomly selected RandomMovelet candidate $\mathcal{M}_x$ containing a subtrajectory $s_{a,b}$ ($s_{a,b} \subseteq T_i$), $\mathcal{M}_x$ is a RandomMovelet if for each other randomly selected RandomMovelet candidate $\mathcal{M}_y$ containing a subtrajectory $u_{f,g}$ with $u_{f,g} \subseteq T_i$ that overlaps $s_{a,b}$ in at least one point, $\mathcal{M}_x.quality > \mathcal{M}_y.quality$.

Algorithm 5 gives more details of the method and is divided into three main steps: (i) the movelet candidates generation, (ii) the random selection of α percentage of movelet candidates, and (iii) the movelets evaluation. First, subtrajectories with all possible combinations of dimensions are generated up to the natural log size of the trajectory (lines 3-7). Then, only a α percentage of the previously generated candidates are randomly selected to be evaluated with F-Score quality (line 8). The third step iteratively evaluates movelet candidates until it finds movelets (lines 9-17), following the steps:

Algorithm 5: RandomMovelets

Input: **T** Trajectory dataset
α Percentage of movelet candidates to evaluate.
Output: **M** Set of *movelets*

```
1  M ← ∅
2  foreach trajectory T_i ∈ T do
       /* 1. Movelet Candidate Generation:                                   */
3      candidates_{T_i} ← ∅
4      m ← log_2(T_i.length)
5      for size ← 1 to m do
6          candidates_size ← FINDCANDIDATES(T_i, size)
7          candidates_{T_i} ← candidates_{T_i} ∪ candidates_size
       /* 2. Movelet Candidate Pruning - Random Selection of α candidates (α=10%): */
8      candidates_R ← RANDOM(candidates_{T_i}, |candidates_{T_i}| * α)
       /* 3. Movelets Evaluation:                                            */
9      M^{T_i} ← ∅
10     while M^{T_i} = ∅ and candidates_{T_i} ≠ ∅ do
11         foreach M_j in candidates_R do
12             M_j.quality ← QUALITY_{F-Score}(M_j, T)
13         candidates_R.sort()
           /* Movelet Pruning - by F-Score and duplicates:                   */
14         M^{T_i} ← MOVELETPRUNING(candidates_R, T)
15         if M^{T_i} = ∅ then          /* Gets next subset of random movelet candidates */
16             candidates_{T_i} ← candidates_{T_i} − candidates_R
17             candidates_R ← RANDOM(candidates_{T_i}, |candidates_R| * 2)
18     M ← M ∪ M^{T_i}
19 return M
```

i. Evaluate selected movelet candidates with *F-Score* (line 12) and order them by quality (line 13);

ii. Prune candidates with zero quality and the ones that share the same points and dimensions with lower quality value (line 14). This movelet pruning is the same described in HiPer-Movelets (Section 3.1);

iii. If no movelets are found in this iteration, randomly select a new set of movelet candidates with twice the size of the previous set, and repeat until movelets are found or all candidates have been evaluated (lines 15-17).

A limitation of RandomMovelets is that the α percentage parameter may need to be changed. We try to mitigate this by doubling this number in each iteration (if movelets are not found). Another limitation is that besides reducing the search space for evaluating movelet candidates, it still extracts all subtrajectories up to the natural log of trajectory size. We, therefore, propose another strategy for movelet candidate generation and evaluation as an alternative to solve some of these limitations, which is called UltraMovelets.

3.2.2 UltraMovelets: A Recursive Strategy for Efficient Movelet Candidates Generation

In this method, the movelet candidates are computed starting in simple pivot subtrajectories of one point and one dimension, and then each one is extended with another point (or attribute) at a time and tested against the data. Combined with an early abandoning strategy for searching relevant subtrajectories reduces the number of generated movelet candidates.

The extraction of movelets from data with many dimensions has the problem of *Curse of Dimensionality*, which is the concept in data mining that describes the exponential growth in computational processing when increasing the number of data dimensions. In subtrajectory analysis, the longer the subtrajectory (in number of points) or the higher the dimensionality (number of attributes) that composes a subtrajectory is, the more computational resources are needed for the processing.

Figure 19 gives an overview of the UltraMovelets method. For each trajectory of a class (steps 1 and 2), UltraMovelets starts by generating pivot subtrajectories (step 3), which are all the subtrajectories with one dimension and one point from trajectory T_i. First, it computes the *F-Score* quality of each pivot candidate (step 5) and discards pivot candidates with zero quality, becoming the pivot subtrajectories. Second, for each pivot, UltraMovelets recursively:

1. Increments one point and evaluates the *F-Score*;
2. Increments one dimension and evaluates the *F-Score*;
3. Selects the best subtrajectory with the point added, the dimension added or the original candidate (steps 6, 7, and 8).

The movelet candidate generation is done recursively, so it will combine points and dimensions to the pivot until reaching the end of the trajectory and all the dimensions together, or it reaches a point where the quality does not increase (step 9) and stops.

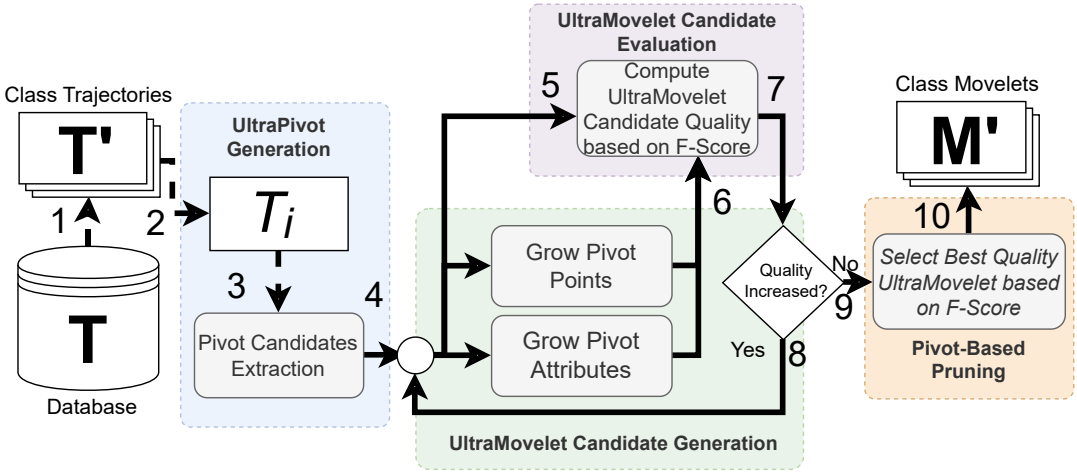


Figure 19 – Overview of the UltraMovelets method.

The method will stop adding points or dimensions to a subtrajectory when the *F-Score* quality reduces. Indeed, since continuing to add information will not generate better quality, this is a point for abandoning the search. Thus, UltraMovelets neither requires criteria to limit the number of dimensions nor the size of subtrajectories to search, as it automatically selects the best combination of dimensions and subtrajectory size. While other approaches limit the search by the natural log of the trajectory size (as MASTERMovelets-Log, SUPERMovelets, and HiPerMovelets), or limit the number of attributes of movelets (as SUPERMovelets-λ), UltraMovelets will stop by the quality criteria. Similarly, it does not limit the number of attribute combinations for searching movelets since adding more attributes will not increase the movelet candidate quality. Indeed, if adding any other attributes reduces the movelet candidate quality, then the best combination of attributes for the subtrajectory is found.

An HiperMovelet Candidate is first evaluated with relative frequency in class trajectories, the RandomMovelet Candidate is selected randomly without previous evaluation, and UltraMovelet Candidates (Definition 11) are generated from pivots with *F-Score* quality greater than zero. The three types of candidates we defined are selected as movelets by evaluating *F-Score* quality in the same manner as HiPerMovelets.

Definition 11. *UltraMovelet Candidate.* A UltraMovelet candidate $\mathcal{M}$ from a subtrajectory $s_{start,end}$ is a tuple $\mathcal{M} = (T_i, start, w, C, \mathbb{W}, quality, sp)$, from T_i, which is a trajectory of the dataset **T**; All terms are identically defined as the movelet in Definition 4. UltraMovelet candidates where $w = 1$ and $|C| = 1$ are called UltraMovelet Pivots. The first point of the UltraMovelet candidate subtrajectory $s_{start,start}$ and the first attribute of the candidate C represents the UltraMovelet Pivot from which originated the candidate.

The three types of candidates we defined are selected as movelets by evaluating *F-Score* quality in the same manner as HiPerMovelets. The UltraMovelet, as Definition 12, differs in concept from a movelet as they can share trajectory points but in different dimensions.

Definition 12. *UltraMovelet.* Given a trajectory T_i, and an UltraMovelet candidate $\mathcal{M}_x$ containing a subtrajectory $s_{a,b}$ ($s_{a,b} \subseteq T_i$), $\mathcal{M}_x$ is an UltraMovelet if for each other UltraMovelet candidate $\mathcal{M}_y$ containing a subtrajectory $u_{f,g}$ with $u_{f,g} \subseteq T_i$ that overlaps $s_{a,b}$ in at least one point, but does not contain the same dimensions $\mathcal{M}_x.C \neq \mathcal{M}_y.C$, the *F-Score* quality is $\mathcal{M}_x.quality > \mathcal{M}_y.quality$.

Algorithm 6 details the UltraMovelets method. The first step (lines 4-6) is to find and evaluate all movelet candidates of one point with one dimension, called pivot candidates. The pivot candidates are pruned by their quality measured with F-Score (line 7), and each one is recursively incremented with the next trajectory point and one of the other dimensions (line 9).

Algorithm 6: UltraMovelets

```
   Input:  T                          Trajectory dataset
   Output: M                          Set of movelets
 1 M ← ∅
 2 foreach trajectory T_i ∈ T do
       /* 1. Pivot Candidates Generation:                                          */
 3     candidates_{T_i} ← ∅
 4     candidates_1 ← FINDCANDIDATES(T_i, 1)   /* Generates pivot candidate subtrajectories */
 5     foreach M_j in candidates_1 do
 6         M_j.quality ← QUALITY_{F-Score}(M_j, T)
 7     pivots_{T_i} ← {M_j ∈ candidates_1 | M_j.quality > 0.0}
       /* 2. Movelet Candidate Recursive Search:                                   */
 8     foreach M_j in pivots_{T_i} do
           /* Recursively increment points and dimensions to the pivot, returns the one with best F-Score */
 9         M_j ← GROWPIVOT(M_j, T_i, T)
10         M ← M ∪ M_j
11 return M
```

The essence of UltraMovelets is the recursive function *GrowPivot*, detailed in Algorithm 7. Each pivot subtrajectory serves as a base to construct related subtrajectories, branching out as a tree of new subtrajectory combinations of the base pivot. The first step is to create a pivot copy with the next point of trajectory T_i , evaluate the quality (lines 1-2), and compare with the base pivot. If the quality increases, the new pivot will be the base for recursively searching movelets (line 3). In the second step, the pivot is recursively incremented with dimensions. A copy of the pivot is incremented with another dimension (lines 6-7), the quality is evaluated, and it is compared with the best pivot found so far (line 9). The algorithm finishes by returning the best movelet candidate found recursively (line 10).

The main advantage of the recursive strategy of UltraMovelets is the possibility at each step to eliminate the branches of combinations that will not increase the quality of a movelet candidate. The method also deals with one subtrajectory at a time, except for parallelization, thus reducing the memory load.

UltraMovelets discovers movelets that share the same point in the trajectory but not the same dimensions. The other movelet-based methods only select movelets without overlapping points. Thus, the resulting number of movelets of UltraMovelets is expected to be higher, being at most the product of the number of trajectory points n and the number of dimensions l ($n.l$).

Algorithm 7: GROWPIVOT

Input: $\mathcal{M}_j$ Movelet candidate pivot
T_i Trajectory of $\mathcal{M}_j$
$\mathbf{T}$ Trajectory dataset
Output: $\mathcal{M}$ The best found movelet

/* Recursively increment points to the pivot, returns the one with best F-Score */
$\mathcal{M}_{jp} \leftarrow$ copy $\mathcal{M}_j$ with next neighbour point from T_i
$\mathcal{M}_{jp}.quality \leftarrow \texttt{QUALITY}_{F-Score}(\mathcal{M}_{jp}, \boldsymbol{T})$
$\mathcal{M}_{jp} \leftarrow \begin{cases} \texttt{GROWPIVOT}(\mathcal{M}_{jp}, T_i, \boldsymbol{T}), & if \quad \mathcal{M}_{jp}.quality >= \mathcal{M}_j.quality, \\ \mathcal{M}_j, & else. \end{cases}$
/* Recursively increment dimensions to the pivot, returns the one with best F-Score */
$\mathcal{M}_{jd} \leftarrow \mathcal{M}_j$
foreach A ***in*** $T_i.D$ **do**
  $\mathcal{M}_{ja} \leftarrow$ copy $\mathcal{M}_j$
  $\mathcal{M}_{ja} \leftarrow \mathcal{M}_{ja}.C \cup A$
  $\mathcal{M}_{ja}.quality \leftarrow \texttt{QUALITY}_{F-Score}(\mathcal{M}_{ja}, \boldsymbol{T})$
  $\mathcal{M}_{jd} \leftarrow \begin{cases} \texttt{GROWPIVOT}(\mathcal{M}_{ja}, T_i, \boldsymbol{T}), & if \quad \mathcal{M}_{ja}.quality >= \mathcal{M}_{jd}.quality, \\ \mathcal{M}_{jd}, & else. \end{cases}$
/* Select the best movelet candidate: */
$\mathcal{M} \leftarrow \begin{cases} \mathcal{M}_{jp}, & if \quad \mathcal{M}_{jp}.quality >= max(\mathcal{M}_j.quality, \mathcal{M}_{jd}.quality), \\ \mathcal{M}_{jd}, & if \quad \mathcal{M}_{jd}.quality >= max(\mathcal{M}_j.quality, \mathcal{M}_{jp}.quality), \\ \mathcal{M}_j, & else. \end{cases}$
return $\mathcal{M}$

3.2.3 Experimental Evaluation

We evaluate the computational cost and classification power of the proposed methods w.r.t HiPerMovelets. This evaluation compares accuracy in an $80\% - 20\%$ hold-out train test split validation, the running time, and the number of generated movelet candidates.

The very specific nature of some data, such as the spatial position, can make the classification task easier. However, its patterns may not frequently repeat in the trajectories of a specific class. It is unlikely for a general-purpose classification method to perform well in a variety of datasets. However, we show that our methods HiPerMovelets, RandomMovelets, and UltraMovelets can achieve good results in a variety of domains. Therefore, we select datasets from domains other than raw and MAT trajectories, including genetic data and time series.

We also performed a scalability analysis and measured memory use because reducing the search space for movelets and the combinatorial exploration of subtrajectories directly impact the memory resources needed. These are preliminary results since more experiments are needed to evaluate the methods properly: (i) k-fold cross-validations to mitigate the accuracy differences, (ii) repetitions of RandomMovelets with different seed values, and (iii) extensive scalability experiments.

In the following sections, we describe the datasets, and the setup configurations of the experiments, discuss the obtained results and evaluate the scalability. We organize the results in four parts: the experiments with multiple aspect trajectory datasets, experiments with raw trajectory datasets, genetic sequence datasets, time series datasets, a general discussion of the results, and scalability experiments.

Table 7 – Complementary summary of the used trajectories datasets for Table 4.

Dataset	Description	Dimensions
Foursquare NYC	Traj Size: 21.75 ± 122.25 # of Traj.: 3,079 # of Points: 66,962 # of Classes: 193 Class Label: User	**Specific**: Lat, Lon, POI, POI Category, Time, Weekday, Weather, Price, Rating, User ID. **Generic**: POI Category, Time, Weekday, Weather, Price, Rating, User ID.
Promoters	Traj Size: 57 ± 0.0 # of Traj.: 106 # of Points: 6,042 # of Classes: 2 Class Label: Positive or Negative	Sequence, Protein Name, Class Name
SJGS	Traj Size: 60 ± 0.0 # of Traj.: 3,190 # of Points: 191,400 # of Classes: 3 Class Label: Exon/Intron boundaries	Sequence, Protein Name, Class Name

3.2.3.1 Datasets

We evaluate RandomMovelets and UltraMovelets on the same five datasets reported in Section 3.1, but we also use more datasets to show how these methods are promising in other domains[6]. In particular, we added two MAT dataset variations from Foursquare NYC data: one with specific attributes and the other with generic attributes; one raw trajectory dataset (Hurricanes); two datasets of genetic sequences (Promoters and SJGS), that are problems of binary classification; multivariate and univariate time series datasets to compare the performance of the methods in other domains with sequential data. Globally, we employed four types of datasets that include raw trajectories, genetic sequences, multiple aspects, and multivariate time series: (i) three raw trajectory datasets (Animals, GoTrack, and Vehicles) as described in Subsubsection 3.1.3.1, (ii) four multiple aspect trajectory datasets (Brightkite, Gowalla, Foursquare NYC with specific and generic attributes) also used in May Petry et al. (2020), Ferrero (2020), Leite da Silva (2020), Portela, Carvalho e Bogorny (2022), (iii) two semantic-only datasets of genetic sequences (Promoters, and SJGS), iv) eight multivariate time series datasets, and (v) twenty five univariate time series datasets. Table 7 presents descriptions of the datasets included in this evaluation (complementary to Subsubsection 3.1.3.1).

Foursquare NYC: Dataset from the Foursquare social media with multiple aspect trajectories in New York (YANG et al., 2015). The points were enriched with the semantic information of the weekday, and the Foursquare API[7] enabled to collect the semantic information of the POI category and the numerical information of the price and rating of the POIs. Indeed, the Weather Wunderground API [8] enabled to enrich the points with the weather situation. The resultant dataset Foursquare NYC has a total of 3079 trajectories, with trajectory sizes varying from 10 to 144, and Foursquare Global has a total of $20,911$ trajectories, with trajectory sizes

[6] https://github.com/bigdata-ufsc/datasets
[7] https://developer.foursquare.com/
[8] https://www.wunderground.com/weather/api/

varying from 10 to 211. We employed a second **generic** version of Foursquare NYC that has been further anonymized with suppression of specific attributes (Lat, Lon, and POI).

Promoters Gene Sequences[9]**:** Promoter Gene Sequences dataset represents 263 DNA sequences that have been compiled and analyzed with known transcriptional start points for E. Coli genes. This dataset is E. Coli promoter gene sequences (DNA) with partial domain theory.

Splice-junction Gene Sequences (SJGS)[10]**:** Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as EI sites) and recognizing intron/exon boundaries (IE sites).

Univariate and Multivariate Time Series: we employed eight multivariate time series datasets (ArticularyWordRecognition, AtrialFibrillation, BasicMotions, CharacterTrajectories, Epilepsy, ERing, EthanolConcentration, and Handwriting), and twenty five univariate time series datasets (ACSF1, Adiac, AllGestureWiimoteX, AllGestureWiimoteY, AllGestureWiimoteZ, ArrowHead, BME, Beef, BeetleFly, BirdChicken, Car, CBF, Chinatown, ChlorineConcentration, CinCECGTorso, Coffee, Computers, CricketX, CricketY, CricketZ, DiatomSizeReduction, DistalPhalanxOutlineAgeGroup, DistalPhalanxOutlineCorrect, DistalPhalanxTW, and ECG200). The descriptions of each dataset and the train and test split used are the same as stated in the source, without any data preparation. These last large group of datasets is not reported in Table 7 to keep the manuscript readable and since the description is already reported in the original data source[11].

3.2.3.2 Experimental Setup

The datasets are split in a hold-out proportion of 80% of the trajectories for training and 20% testing respecting the class balance (similar number of trajectories of each class). We evaluated the Neural Network Multilayer-Perceptron (NN) and Random Forrest (RF) classification algorithms with the same classifier and method configurations as described in Subsubsection 3.1.3.2. We present the results of Accuracy, Accuracy Top 5, and F-Measure metrics. The latter two metrics and total running time are displayed only with the NN classifier for readability purposes. The Accuracy Top-5 is the accuracy for the model ability to correctly classify in any of 5 highest probability label matches. The Accuracy Top-5 is the accuracy for the model ability to correctly classify in any of the five highest probability label matches. The F-Measure is the average F-Score between class predictions. The F-Measure is calculated from the harmonic mean of precision and recall of the test set as a macro average of all class labels.

[9] https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Promoter+Gene+Sequences

[10] http://archive.ics.uci.edu/ml/datasets/Molecular+Biology+%28Splice-junction+Gene+Sequences%29

[11] http://timeseriesclassification.com

We include the methods for raw trajectory datasets: Xiao (XIAO et al., 2017), Dodge (DODGE; WEIBEL; FOROOTAN, 2009), Movelets (FERRERO et al., 2018), and Zheng (ZHENG et al., 2008). We also included the DeepeST method for trajectory classification that, for each model, look for the optimal set of hyperparameters (A. de Freitas. et al., 2021). The MARC and MASTERMovelets methods follow the same configurations as in Subsection 3.1.3. We set RandomMovelets α to 10% of movelet candidates (default value), as experiments have shown a reasonable value to prune enough candidates improving performance without restricting the search too much.

We conducted a trial experiment for *POI-F* (VICENZI et al., 2020), HiPerMovelets and HiPerPivots methods with different configurations, as presented in Figure 20. For the results of the method *POI-F*, we use the variant *NPOI* as it achieves the best results. For *NPOI*, in each dataset, we chose one attribute of the data with the highest variance. Then, we tested the method with subsequence sizes of 1, 2, and 3 points. We also extended *NPOI* to concatenate the sequences of 1, 2, and 3 points for classification, which is presented as *NPOI (1+2+3)*. The *NPOI* with 1 point achieves good classification results. However, it does not take the sequence of the points into account. *NPOI (1+2+3)* can perform well in most cases as it has the attribute information and small sequences are considered together. In the following sections, we present the results for the *NPOI (1)* and *NPOI (1+2+3)* variations.

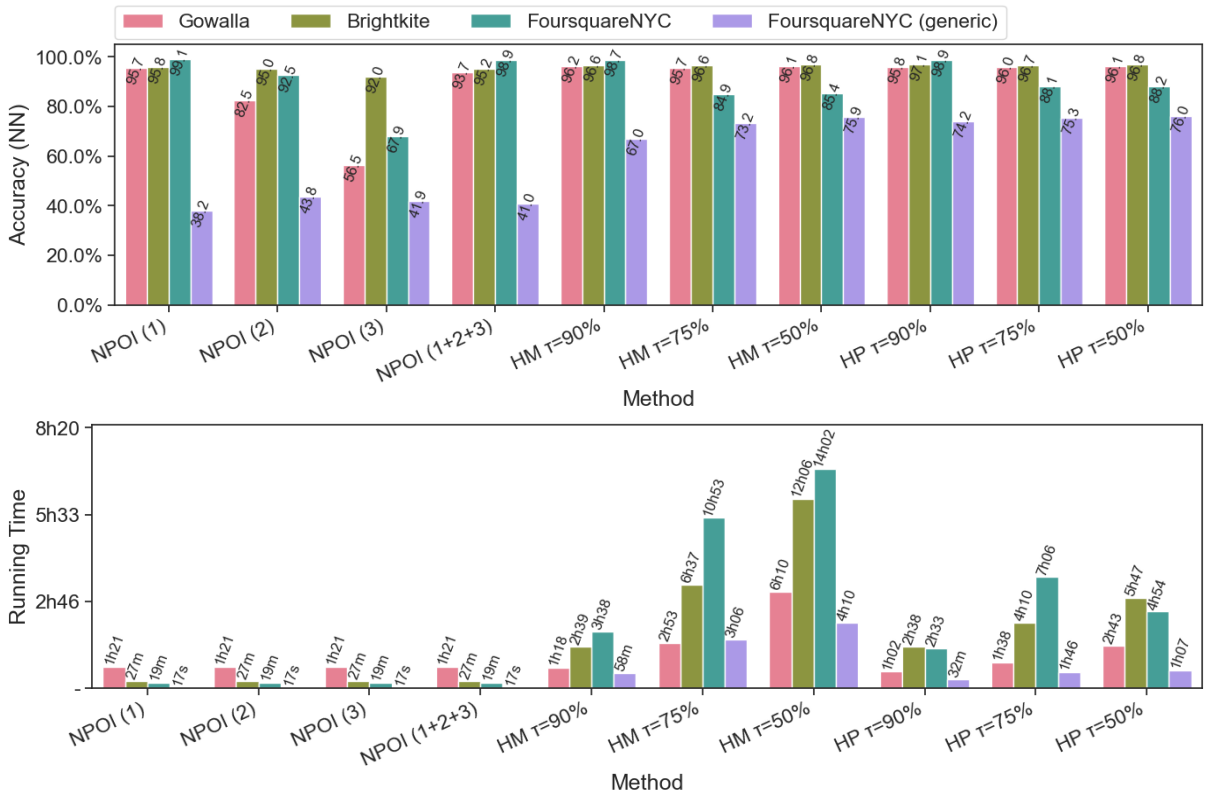


Figure 20 – Test configuration of *NPOI*, HiPerMovelets, and HiPerPivots for accuracy with the classifier NN (top) and running time (bottom) in multiple aspect trajectory datasets.

For HiPerMovelets and HiPerPivots trial experiments, we evaluate both approaches with three settings of τ parameter for movelet candidates selection: 50%, 75%, and 90% percent

of movelet candidate relative frequency. In terms of accuracy (Figure 20 top), there is a trade-off between the value of τ and the HiPerMovelets performance on specific and generic datasets. High values of τ work better on specific datasets where the class most frequent patterns are more discriminant, while in the generic datasets, they are not so discriminant. However, the running time increase (Figure 20 bottom) as the value of τ changes because more movelet candidates are evaluated. Therefore, in each following section, we present the results for HiPerMovelets and HiPerPivots with the τ value that achieved the best performance in general, considering accuracy and lower running time.

The experiments were performed in five computers with the configurations and the respective datasets as described in Table 8. The computer resources are limited, as described in the table for a fair time comparison among the movelet-based methods. We also set a timer of seven days as a limit for all experiments to complete and three days for the scalability experiments.

Table 8 – Summary of the experimental setup.

Machine	**Processor**	**Operating System**	**RAM**	**Cores**	**RAM Limit**	**Thread Limit**	**Datasets**
PC-1	Intel(R) Xeon(R) Platinum 8276L CPU @ 2.20GHz	Ubuntu 18.04.6 LTS	504GB	112	400GB	4	FoursquareNYC, FoursquareNYC generic, Promoters, SJGS
PC-2	Intel(R) Xeon(R) Platinum 8276L CPU @ 2.20GHz	Ubuntu 18.04.5 LTS	512GB	112	250GB	4	Animals, GoTrack, Vehicles
PC-3	AMD EPYC 7702 64-Core	CentOS Linux 7 (Core)	1TB	256	700GB	4	Gowalla, Brightkite, *Multivariate Time Series*
PC-4	Intel Core i7-6700 CPU @ 3.40GHz		32GB	8	30GB	4	*Univariate Time Series*
PC-5	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz	Ubuntu 16.04.7 LTS	64GB	8	60GB	2	*Scalability datasets*

3.2.3.3 Preliminary Results of Accuracy, Processing Time, and Number of Movelet Candidates

This section compares the accuracy, processing time, and the number of generated movelet candidates. The classification results of Multilayer-Perceptron Neural Network (NN) and Random Forest (RF) are given by the Accuracy metric to evaluate the performance of extracted features. We compare the running time of MASTERMovelets-Log, HiPerMovelets, RandomMovelets, and UltraMovelets due to the nature of the optimization problems of movelet-based methods. We compare the number of generated movelet candidates to indicate the method performance. The results are summarized in *bar plots*. The detailed results for each dataset are presented in Chapter 6. We only included results from experiments that finished. Xiao and Dodge did not run on the GoTrack dataset.

3.2.3.3.1 Results with Multiple Aspect Trajectory Data

In this first part we compare DeepeST (A. de Freitas. et al., 2021), NPOI (VICENZI et al., 2020), MASTERMovelets-Log (MM) (FERRERO et al., 2020), MARC (May Petry et al., 2020), HiPerMovelets (HM), HiPerPivots (HP), RandomMovelets (RM), and UltraMovelets (UM) methods. To compare the methods performance, we present the Accuracy metric, Accuracy Top-5 and F-Measure of all classification results as bar plots. The experiments are performed in four MAT datasets: Gowalla, Brightkite, Foursquare NYC, and Foursquare NYC generic. These datasets are prepared in a hold-out of 80% train and 20% test split. The unfinished experiments have been omitted from the figures, and the detailed results are presented in Chapter 6.

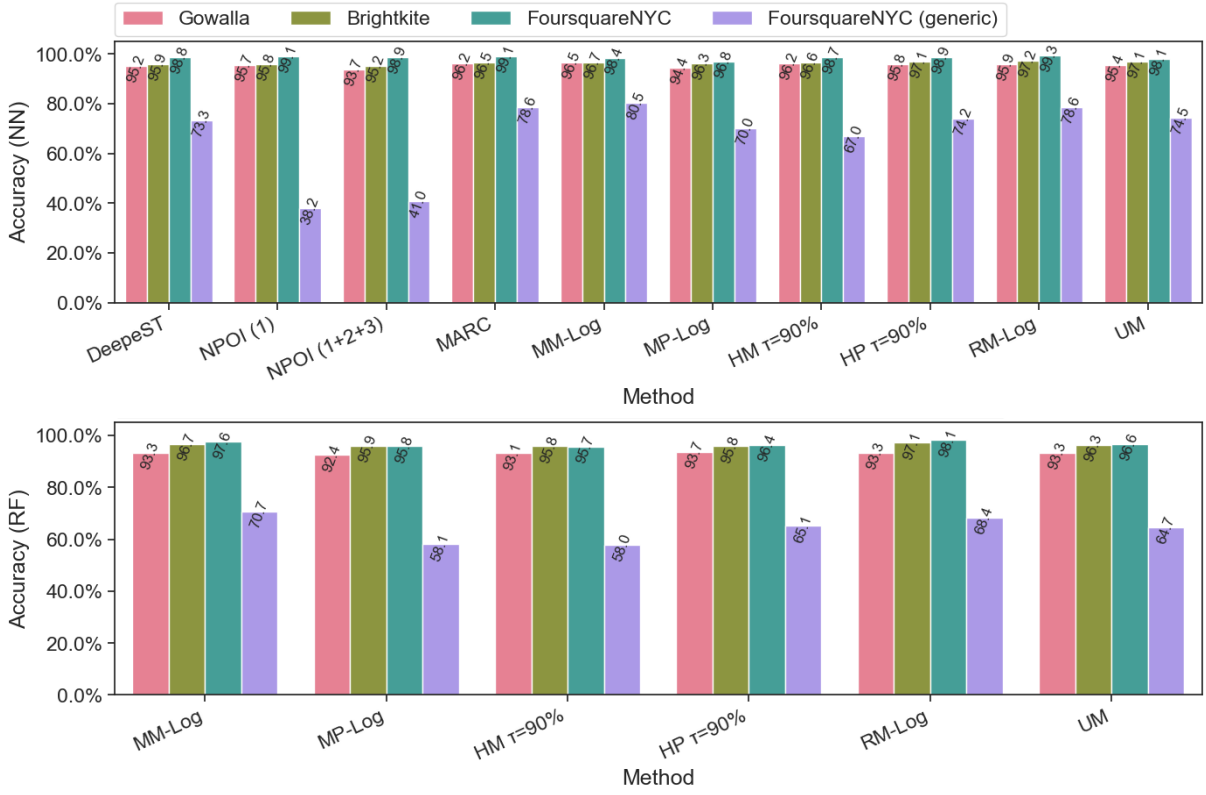


Figure 21 – Accuracy bar plots for all methods, with the classifiers NN (top) and RF (bottom) in multiple aspect trajectory datasets.

Figure 21 presents the Accuracy metric. Both the NN and the RF classifiers achieved similar accuracy values. However, the NN classifiers performed better in general. All methods present similar accuracy for the specific datasets (Gowalla, Brightkite, Foursquare NYC), with all accuracies over 90%, except in the generic dataset. The main observation in this preliminary experiment is that the methods proposed in this thesis have similar accuracy on multiple aspect trajectory data, compared to MASTERMovelets with improved running time. We observe that *NPOI (1+2+3)* improved the accuracy compared to *NPOI* with subtrajectories of sizes 2 and 3 because subtrajectories of shorter size can be more discriminant as they tend to have a higher frequency. In the Foursquare NYC generic, higher accuracy is achieved with

subsequences concatenated (*NPOI (1+2+3)*) that indicate the sequence provides important information to discriminate classes. Indeed, concatenating different sizes of subtrajectories lead the classification to achieve higher accuracy. These results are interesting, considering that only one dimension was used for each dataset.

In Figure 22, the Accuracy Top-5 metric presents the model's ability to classify the correct class label in any of the five highest probability matches. As expected, these results showed improvements in accuracy, and the methods behavior is similar to the results presented in Figure 21. In the Foursquare NYC generic, the accuracy is expected to drop because of the removal of specific dimensions, and it is the dataset with more differences in terms of accuracy. However, for movelet-based methods and MARC, in the generic dataset the Accuracy Top-5 performance improved. Thus, indicating that embeddings and movelets are techniques more likely to rank the correct class label between the first five results.

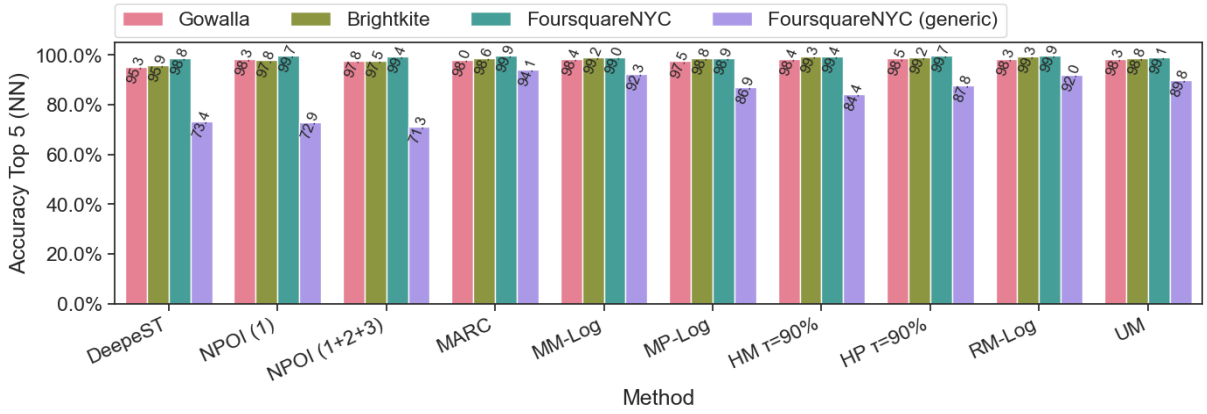


Figure 22 – Accuracy top 5 bar plots for all methods, with NN classifiers in multiple aspect trajectory datasets.

Figure 23 shows the bar plot results regarding the macro F-Measure, the harmonic mean of precision and the recall average between class predictions. For this metric, the best results are close to 1.0. Results in both Figures 21 and 23 are similar, demonstrating these different metrics comparability. The lower F-Measure for the Foursquare NYC generic dataset further shows it is a more challenging classification problem.

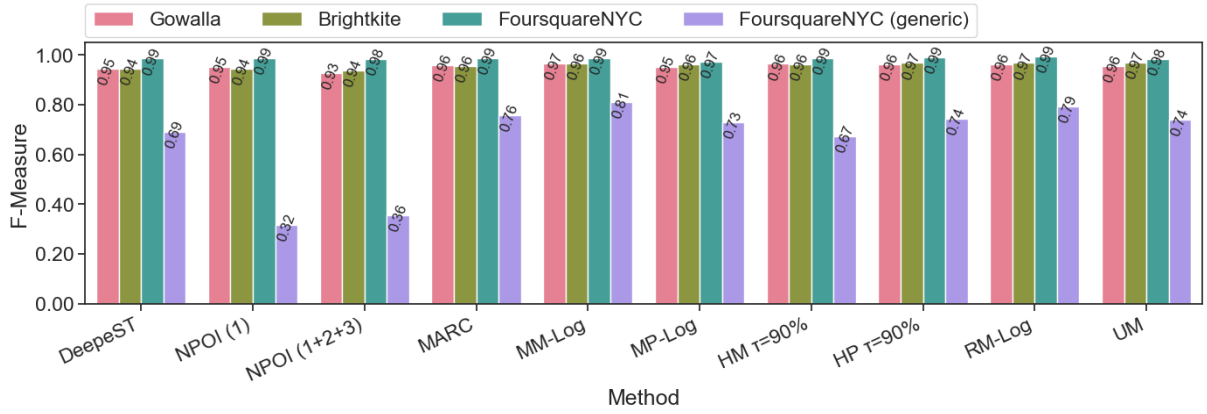


Figure 23 – Macro F-Measure bar plots for all methods, with NN classifier in multiple aspect trajectory datasets.

Figure 24 presents the total running time considering the sum of the feature extraction and the classification time on NN classifiers. DeepeST and MASTERMovelets-Log are the slowest methods, with higher processing times. We can highlight that NPOI and MARC are the most efficient methods in terms of total running time on trajectory datasets, followed by HiPerMovelets methods (with τ set to 90%) and UltraMovelets.

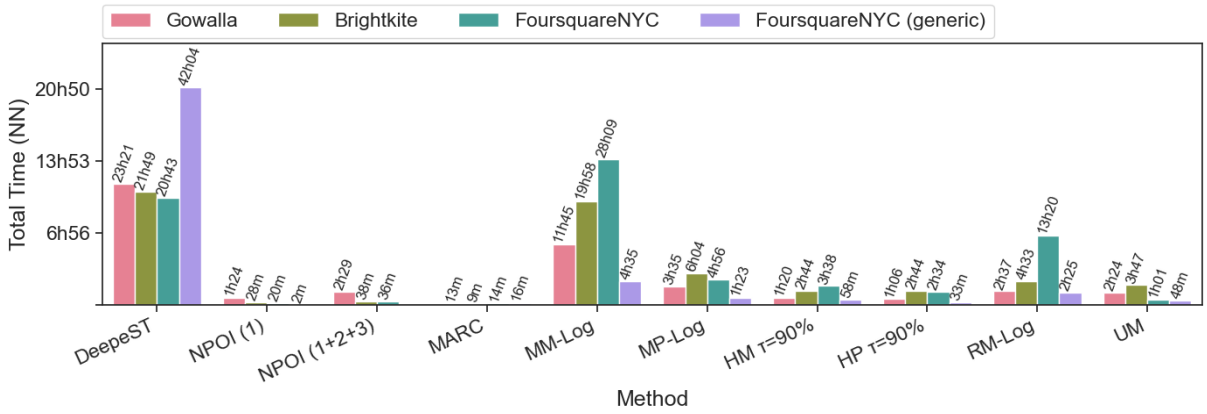


Figure 24 – Total running time (in hours) bar plots for all methods, sum of running time and classification time with NN classifier in multiple aspect trajectory datasets.

The evaluation of movelet candidates determines the method running time because finding the best alignment, defining the split point, and the ranking procedures for each candidate are expensive in computer resources. Our methods reduce the number of movelet candidates that need to be analyzed, thus reducing the running time. Indeed, as Figure 24 shows, all our methods performed faster than MASTERMovelets. Another observation is that the methods with pivot strategies present the lowest total running time among the movelet-based algorithms, and HiPerPivots with $\tau = 90\%$ was the fastest among all our strategies. RandomMovelets and UltraMovelets achieved good results. Although losing in total running time to HiPerPivots, they achieved higher accuracy.

We now compare the distribution of the number of movelet candidates compared to MASTERMovelets. In Figure 25 (top), the characteristics of each method in movelet candidate generation are evident. We present the number of movelet candidates evaluated with F-Score, the most time-consuming step in finding movelets. First, MASTERMovelets, HiPerMovelets, and RandomMovelets generate all possible subtrajectories. However, only MASTERMovelets evaluates all subtrajectories to find movelets. RandomMovelets has similar results in number of candidates as MASTERPivots. HiPerMovelets has similar results independently of the value of τ, because it also explores all subtrajectories but not on all trajectories. Therefore less movelet candidates are produced compared to MASTERMovelets. MASTERPivots, HiPerPivots, and UltraMovelets are the methods that generate less movelet candidates. We point out that UltraMovelets does not limit the search by the natural log of the trajectory size as the other methods but still produces less movelet candidates. That shows how promising the UltraMovelets strategy is to evaluate movelet candidates. More analysis on movelets obtained in this experimental

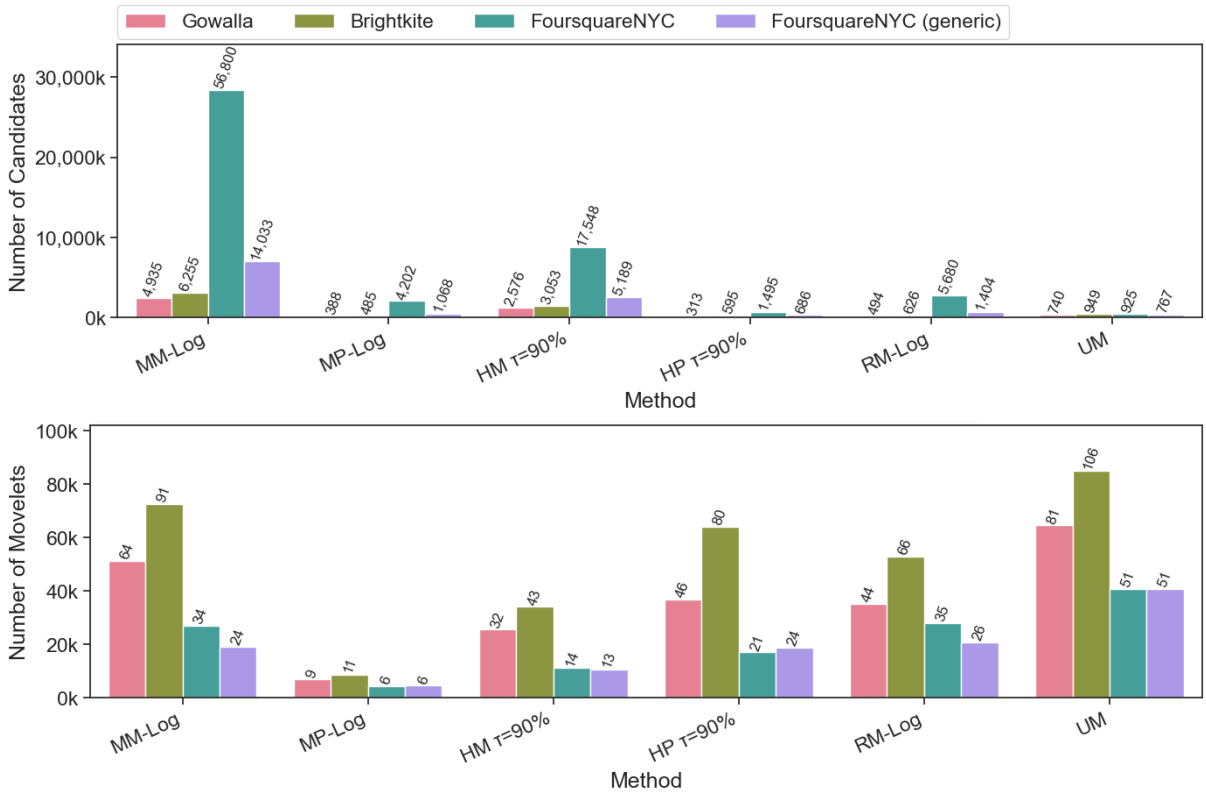


Figure 25 – Bar plots of number of movelet candidates (top), and movelets (bottom), in thousands, for movelet-based methods in multiple aspect trajectory datasets.

evaluation are presented in Chapter 4 considering the number of movelets, their quality, the number of attributes, and their size.

3.2.3.3.2 Results with Raw Trajectory Data

In this evaluation, we included methods developed for raw trajectory data: Dodge (DODGE; WEIBEL; FOROOTAN, 2009), Xiao (XIAO et al., 2017), and Zheng (ZHENG et al., 2008), developed for spatio-temporal trajectories. We also included Movelets (FERRERO et al., 2018) that better perform in this type of data. We present the same evaluations as in the previous section, except we do not include the Accuracy Top 5 measure since all the datasets evaluated have less than five class labels. We performed experiments in three datasets of raw trajectories: Animals, GoTrack, and Vehicles described in Subsubsection 3.1.3.1. The results are for 5-fold cross-validation prepared in a hold-out of 80% train and 20% test split.

We present the accuracy distribution bar plots in Figure 26. All movelet-based methods mean accuracy is in 80% to 100% in the NN classifier, a better performance than Dodge, Zheng, Movelets, DeepeST, NPOI, and MARC. Specifically, MARC, NPOI, and DeepeST showed a more significant drop in accuracy distribution on these raw trajectory data compared to results on MAT datasets. That is expected since they were developed for multiple aspect trajectories. These results show that movelet-based methods perform well in raw trajectory datasets. The lesser difference between NN and RF classifiers is in movelet-based methods, which indicates

that movelets are good features independent of the classifier used.

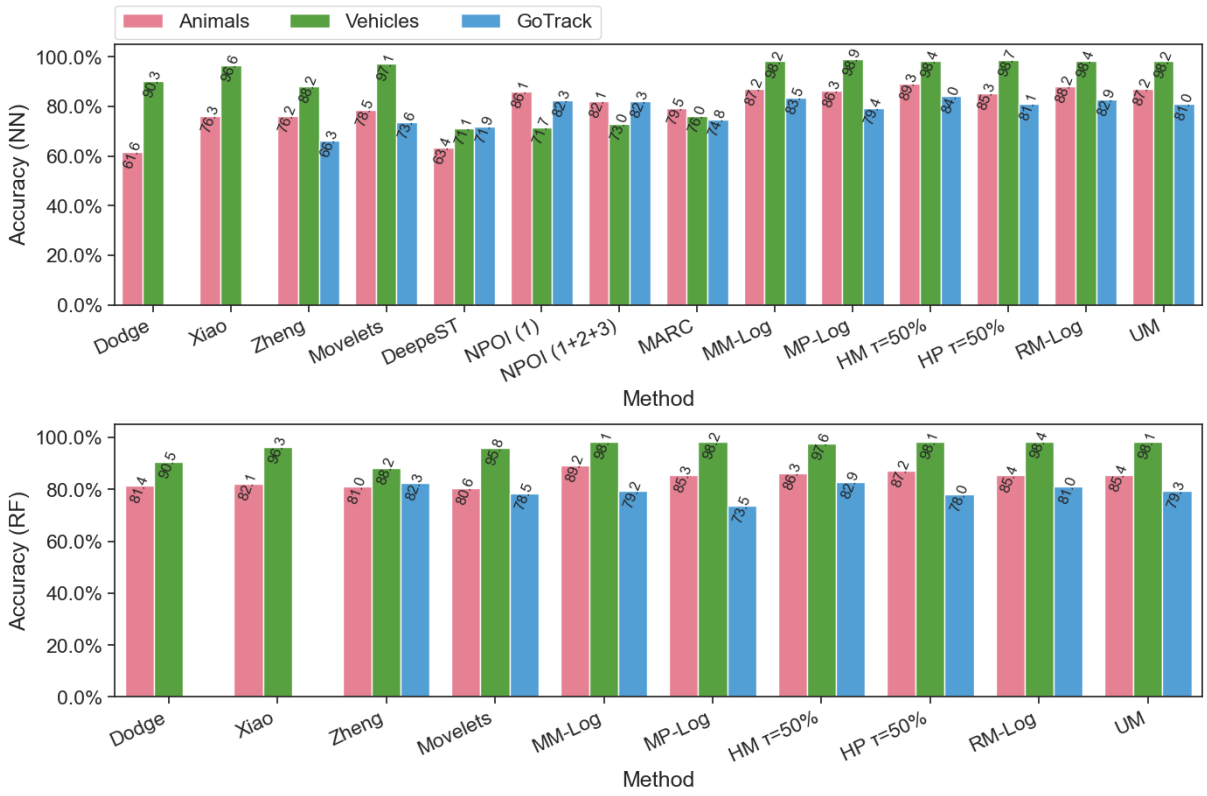


Figure 26 – Accuracy bar plots for all methods, with the classifiers NN (top) and RF (bottom) in raw trajectory datasets.

Regarding the F-Measure, presented in Figure 27 with NN classifiers, the promising results of movelet-based methods are most evident compared to other approaches in raw trajectory data. F-Measure results are similar to the Accuracy results presented in Figure 26.

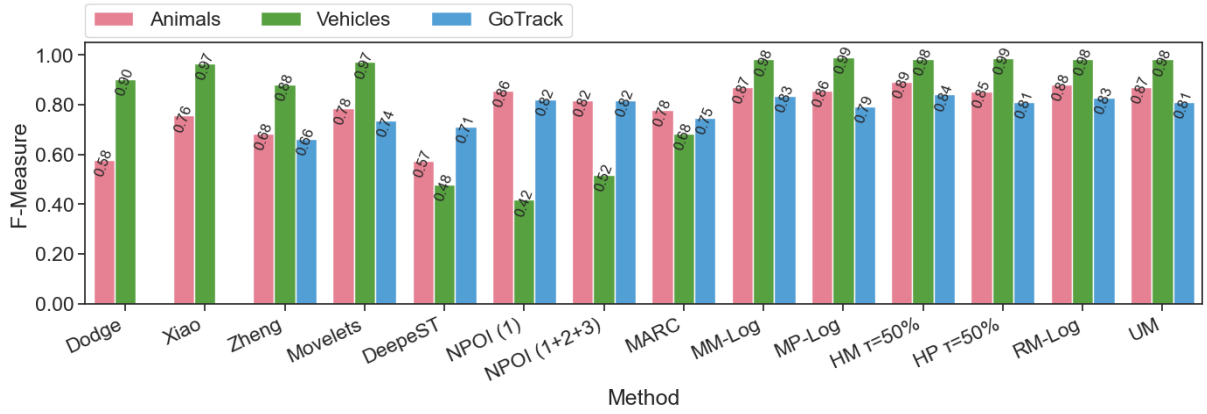


Figure 27 – Macro F-Measure bar plots for all methods, with NN classifiers in raw trajectory datasets.

The total running time results in Figure 28 are the sum of running time and classification time with the neural network classifier for raw trajectory datasets. The methods Xiao, Dodge, Zheng, NPOI, MARC, MASTERPivots, HiPerPivots (with τ set to 50%), RandomMovelets, and UltraMovelets presented the lower total running times. However, Xiao and Dodge could not run on the GoTrack dataset. Moreover, the running time on raw trajectory

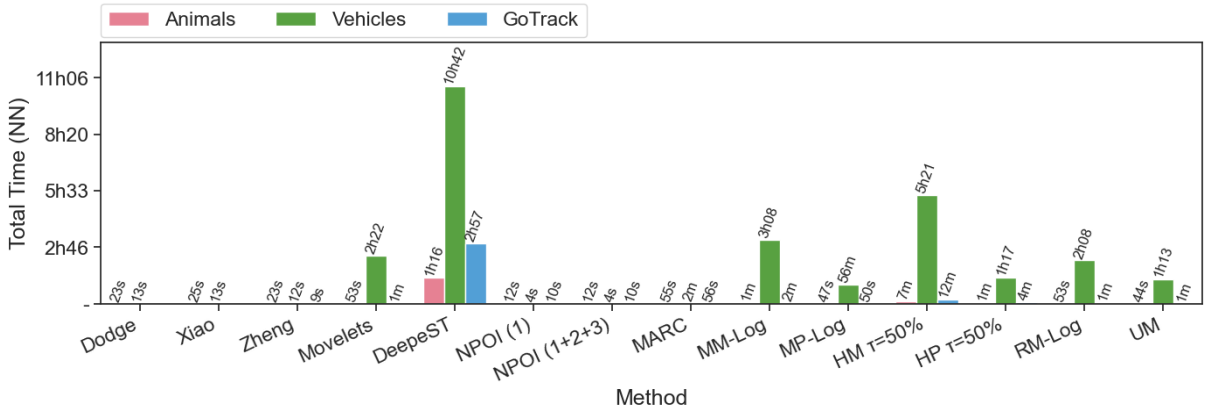


Figure 28 – Total running time (in hours) bar plots for all methods, sum of running time and classification time with NN classifier in raw trajectory datasets.

datasets is much lower for all methods than MAT datasets, primarily due to the fewer trajectories and dimensions (only space and time). Indeed, the number of dimensions impacts the running time primarily because of the exponential combinatorial explosion as more dimensions a dataset contains. Another observation is the total running time of DeepeST, which is higher than all other methods. It runs all combinations of hyperparameters on the train set, then chooses the best configuration to create the model for the test set evaluation. Moreover, except for the DeepeST, the running times of all other compared methods are in minutes (or less) for Animals and GoTrack datasets.

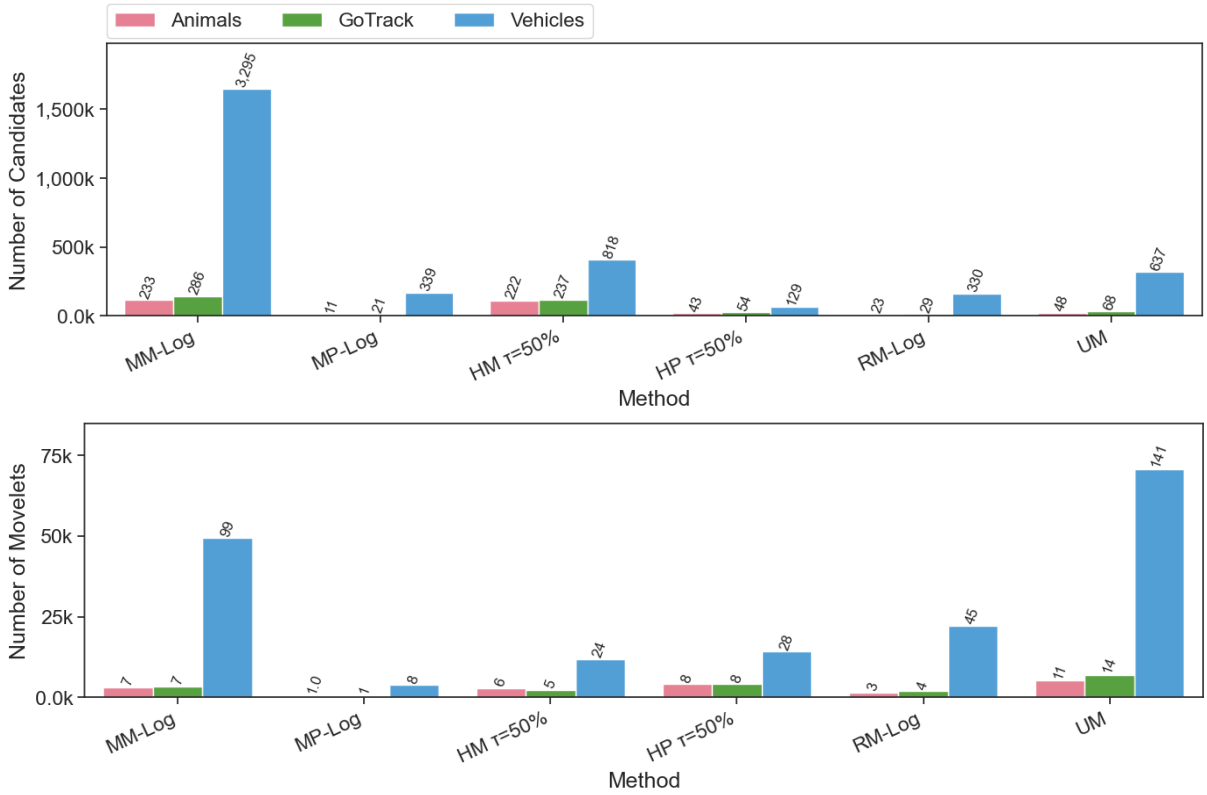


Figure 29 – Bar plots of number of movelet candidates (top), and movelets (bottom), in thousands, for movelet-based methods in raw trajectory datasets.

The number of movelet candidates in raw trajectory datasets follows similar behavior compared to MAT datasets (Figure 29). Comparatively, the number of candidates is much lower, which is expected as the datasets are smaller in the number of trajectories and their dimensions. The number of movelet candidates for MASTERMovelets is much greater than for other methods, similar to the results on MAT datasets. HiPerMovelets in Animals and GoTrack datasets generated as many movelet candidates as MASTERMovelets, indicating the reason for a higher running time. As mentioned before, it is expected that HiPerMovelets methods to perform better in larger datasets, as in the Vehicles dataset, where our methods generated less candidates. In most cases, the methods generate a similar number of movelets, but MASTERPivots, HiPerMovelets, HiPerPivots, RandomMovelets, and UltraMovelets generate less candidates than MASTERMovelets.

3.2.3.3.3 Results with Genetic Sequence Data

In this preliminary evaluation, we performed experiments in two datasets of genetic sequences: Promoters and SJGS. Both datasets are prepared in a hold-out proportion of 80% train and 20% test split. This section and the following section on time series datasets are evaluations of our methods on other domains for which movelet-based methods were not developed. The main idea is to evaluate and compare the methods performance on other types of sequential data. The genetic sequences are only semantic datasets with one attribute.

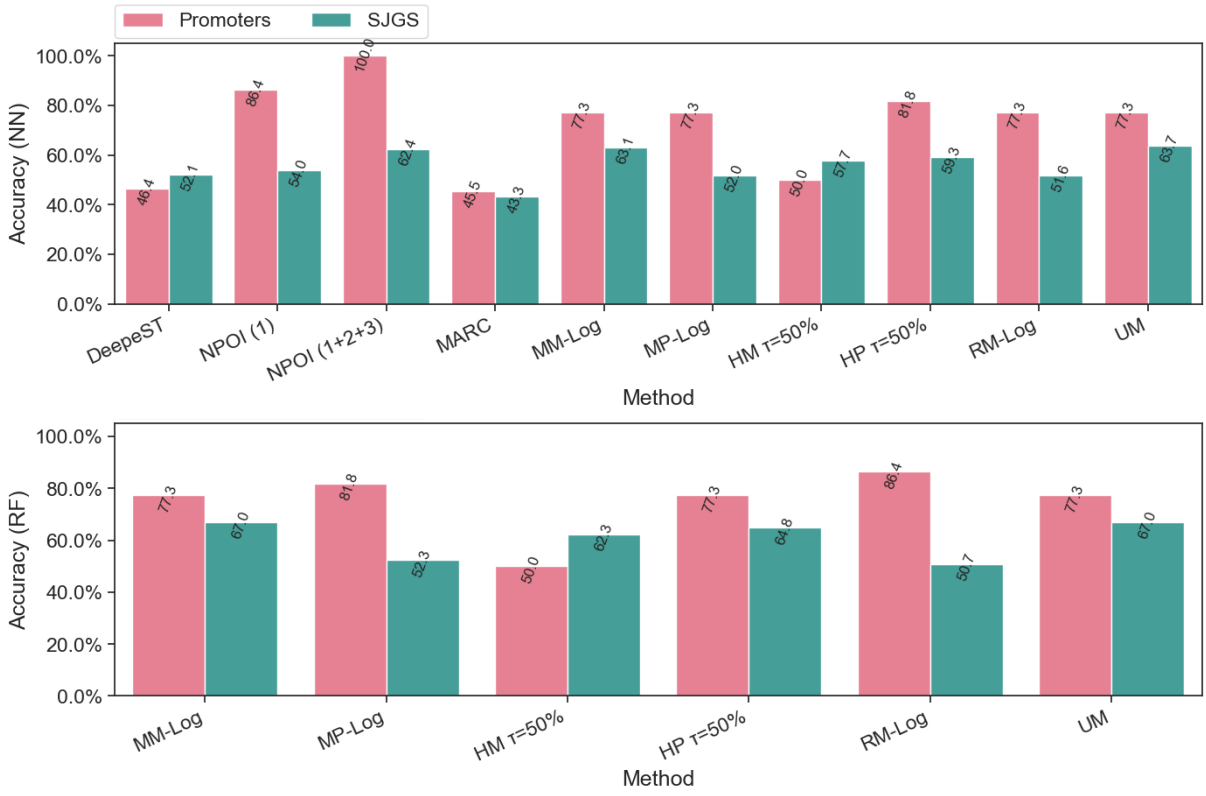


Figure 30 – Accuracy bar plots for all methods, with the classifiers NN (top) and RF (bottom) in genetic sequence datasets.

Accuracy results in genetic sequence datasets are mostly between 50% to 80%, as presented in Figure 30. An interesting observation is that NPOI with sequences of size three achieved good results. Consequently, *NPOI (1+2+3)* presents the best classification results. A possible explanation is that in molecular biology, the proteins are encoded by sequences of three-nucleotide codons. Thus, sequences of size three can be the most discriminant for these classification problems.

Both F-Measure (Figure 31) and Accuracy (Figure 30) results are comparable, showing very similar behaviors for the classification results. MARC and DeepeST performed worse in F-Measure and Accuracy, lower than 50%. Genetic sequences are a very different domain from multiple aspect trajectories, and reasonably the methods perform worse. However, in this data type, the RF classifier performed better, a disadvantage for methods based on neural networks such as MARC and DeepeST.

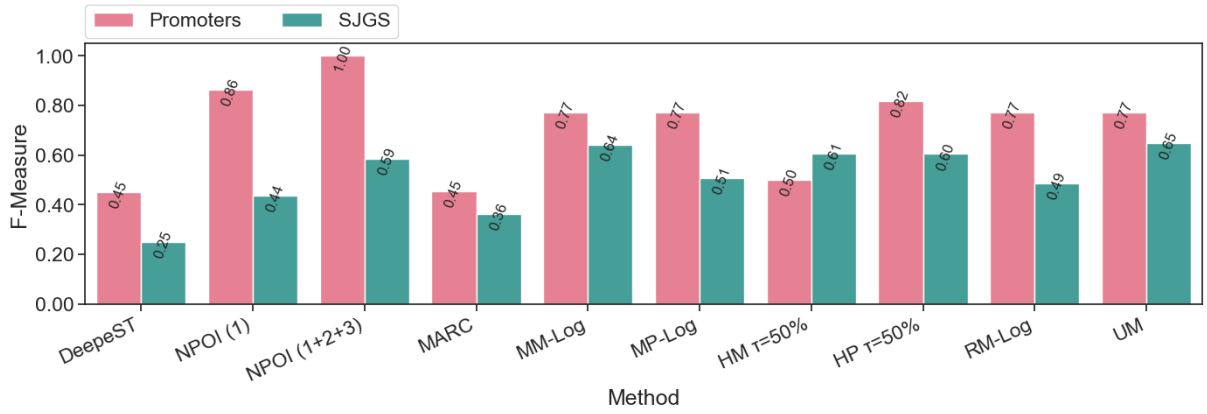


Figure 31 – Macro F-Measure bar plots for all methods, with NN classifier in genetic sequence datasets.

The total running times, presented in Figure 32, are similar to evaluations in previous sections. It is important to point out that in the Promoters dataset, most methods run in less than one minute as it is a small dataset.

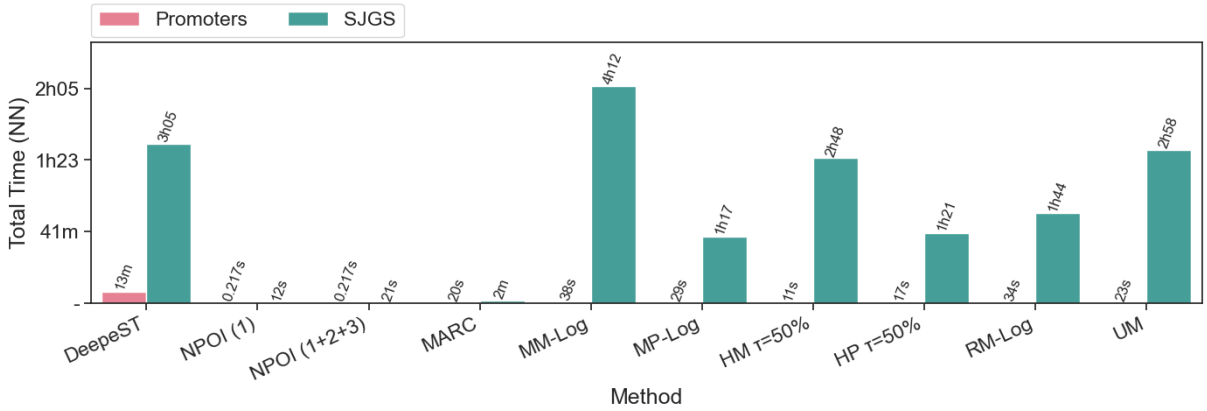


Figure 32 – Total running time (in hours) bar plots for all methods, sum of running time and classification time with NN classifier in genetic sequence datasets.

In terms of number of movelet candidates (Figure 33), MASTERPivots, HiPerPivots, and RandomMovelets generate less candidates. All our methods still generate less movelet can-

didates than MASTERMovelets. Another observation is that despite HiPerMovelets generating around half the number of candidates compared to MASTERMovelets, it still outputs nearly the same number of movelets. That indicates our method ability to reduce the search space for finding the movelets.

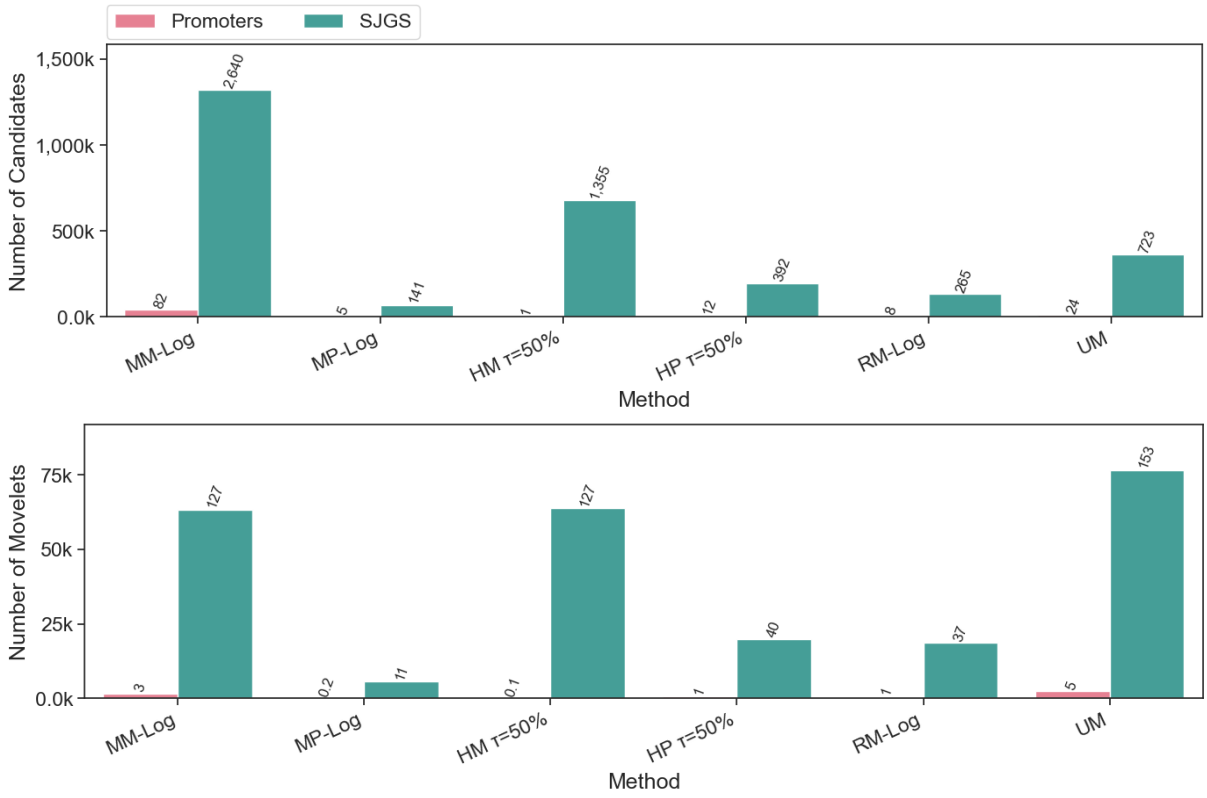


Figure 33 – Bar plots of number of movelet candidates (top), and movelets (bottom), in thousands, for movelet-based methods in genetic sequence datasets.

3.2.3.3.4 Results with Multivariate and Univariate Time Series Data

Multivariate and Univariate Time Series datasets are predominantly numerical and highly sampled data. In this analysis, we compile results from datasets with a diverse number of time series and attributes. Some time series datasets present a complex classification problem that general-purpose methods might not be enough to solve. However, as far as the classification problem is concerned, they can be considered similar to trajectories. As described in Section 3.2, the proposed movelet extraction methods can be used and adapted to multidimensional sequential data. Therefore, we compare the mean accuracy between our methods on these two kinds of data, as presented in Figure 34. In this evaluation, we included the best accuracy reported in the literature with methods created specifically for classifying time series data (BAGNALL et al., 2017; RUIZ et al., 2021). In Figure 34, the mean accuracy results are presented as TS Best for multivariate and univariate datasets, except for one dataset (CharacterTrajectories) for multivariate time series, and four datasets (ACSF1, AllGestureWiimoteX, AllGestureWiimoteY, AllGestureWiimoteZ) of univariate time series.

For our methods, the NN and RF classifiers have very similar results. RandomMovelets and HiPerPivots (with $\tau = 50\%$) show the best performance in classification for these different domains, as they present higher mean accuracy (near 75%). The *NPOI* method presents the highest variation in accuracy results and the lowest values. In Multivariate datasets, we expect our methods to perform better than in univariate time series since they combine dimensions to find discriminant subsequences (the movelets).

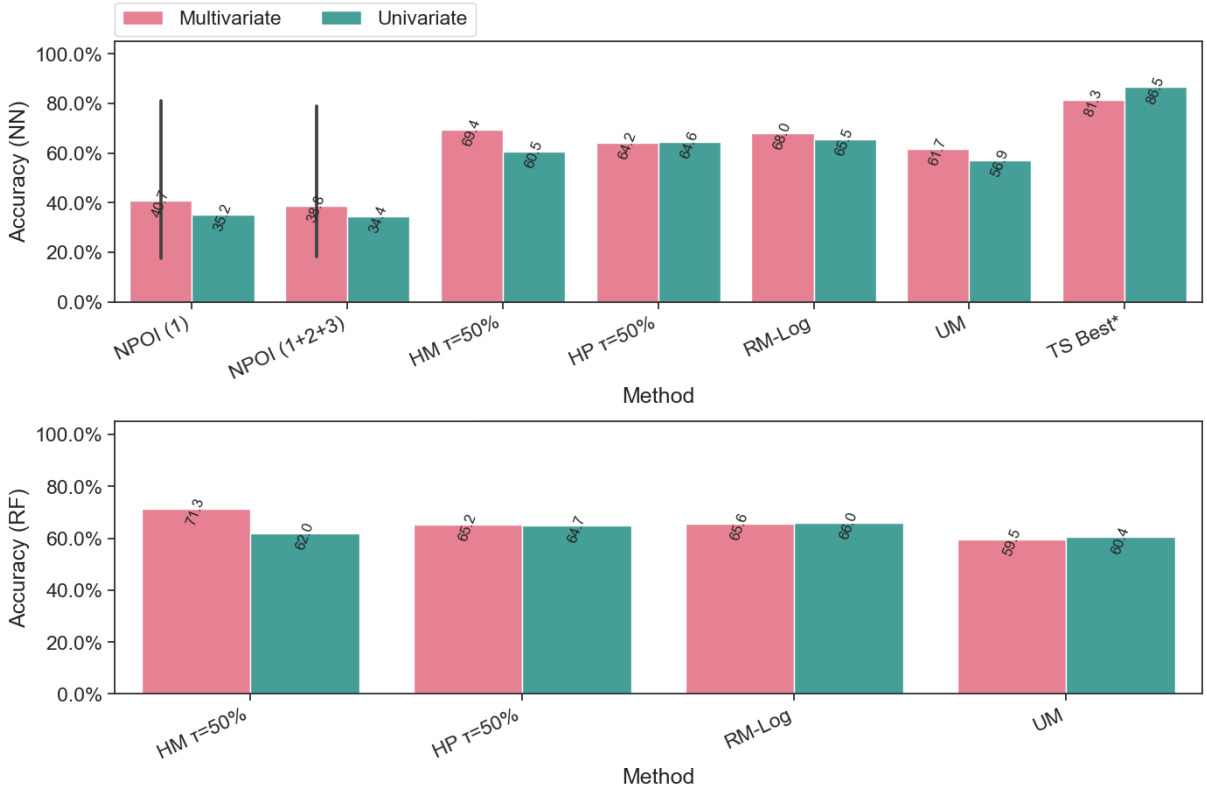


Figure 34 – Accuracy bar plots for all methods, with the classifiers NN (top) and RF (bottom) in time series datasets.

Comparing the difference in F-Measure between our methods on these two kinds of data shows similar distribution as accuracy scores, as presented in Figure 35.

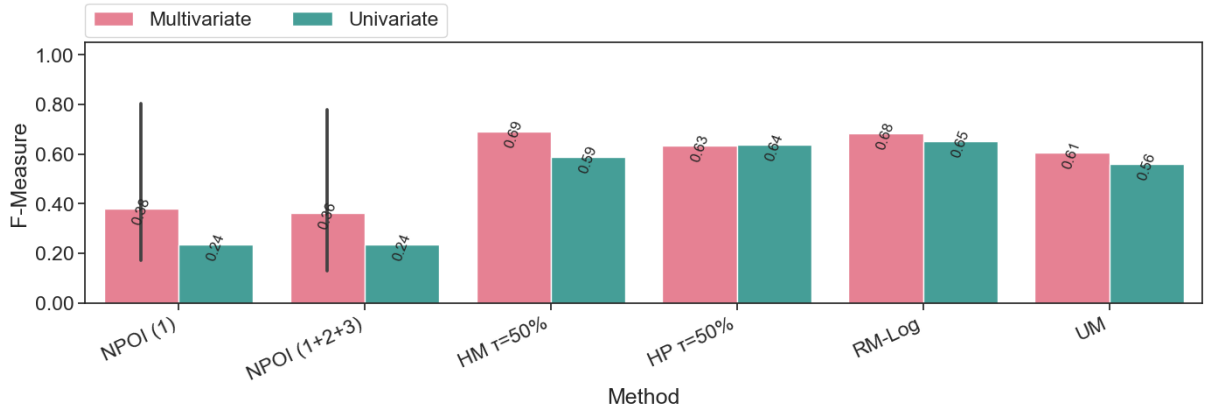


Figure 35 – Macro F-Measure bar plots for all methods, with NN and RF classifiers in time series datasets.

Regarding running time, presented in Figure 36, HiPerMovelets is the method that performed better, with a lower running time, also HiPerPivots and UltraMovelets have similar performances, despite some datasets that take much more time to run. Although further experimentation is necessary, those methods present good results for general-purpose classification. The total running time in time series datasets shows different results of our methods as in MAT datasets (Figure 24). We notice that some running times are outside of interquartile distributions, meaning that in some datasets, the running time is much higher than the average. That is expected since larger datasets were used. Another important observation is that UltraMovelets running times display less variation. It is a promising result that UltraMovelets can be better suited for larger datasets.

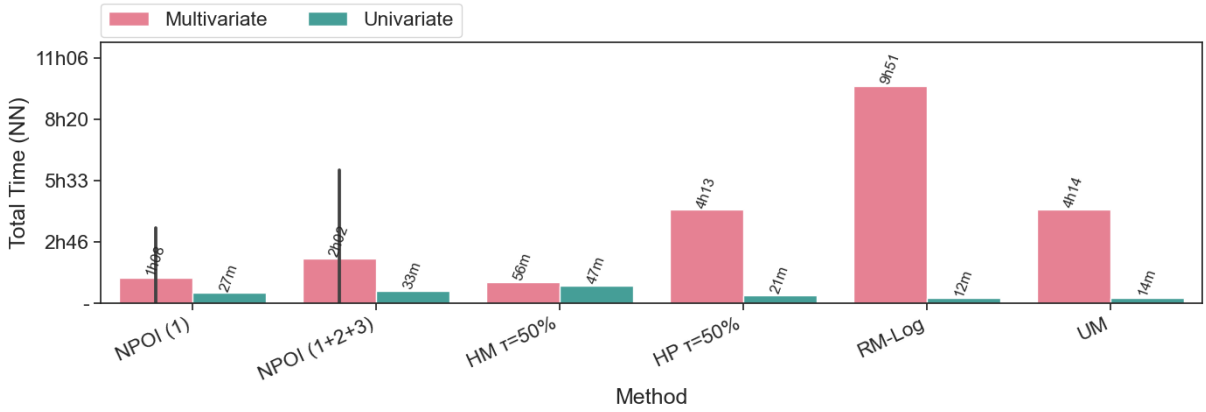


Figure 36 – Total running time (in hours) bar plots for all methods, sum of running time and classification time with NN classifier in time series datasets.

So far, we can conclude that the results show that movelet-based methods are promising to reduce the search space. Generating fewer movelet candidates is a clear advantage for reducing the running time. In this sense, we compare the distribution box plot results over the number of generated movelet candidates on all datasets, presented in Figure 37. From the two figures, we can draw two main conclusions about our methods: (i) RandomMovelets is the method that generates fewer movelet candidates, (ii) UltraMovelets also generates much less candidates, compared to HiPerMovelets, a similar number to candidates of RandomMovelets. Another observation is that we tested the τ parameter with 50%, 75%, and 90% configurations, concluding that it has little impact on the number of generated movelet candidates and, consequently, in the number of resulting movelets.

3.2.3.3.5 Analysis with All Datasets including data from other Domains

To clarify the results obtained, we compare differences between pairs of methods: MASTERMovelets-Log, as the state-of-the-art reference; The HiPerPivots that is our published method with the best performance; RandomMovelets and UltraMovelets, also proposed in this chapter. We calculated the average difference between every two methods for the total running time, classification accuracy, number of movelet candidates, and number of output movelets.

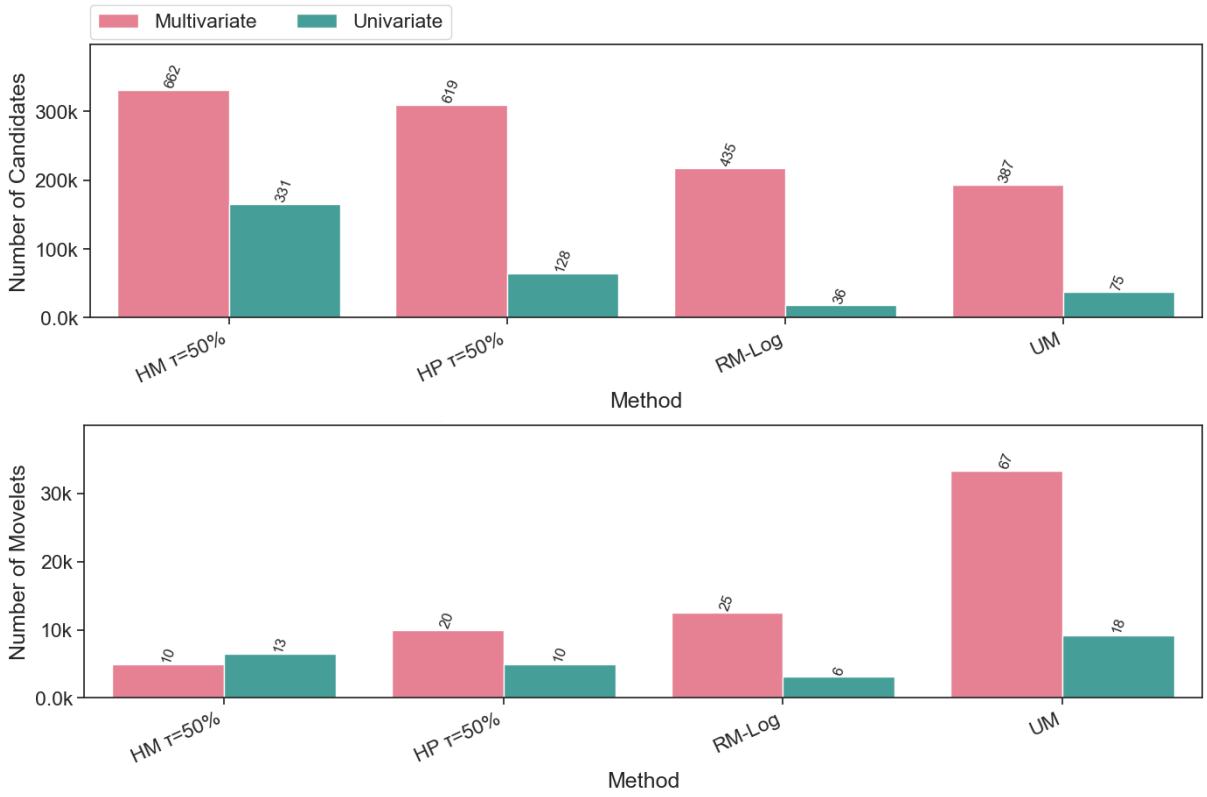


Figure 37 – Bar plots of number of movelet candidates in thousands for our methods in time series datasets.

The Table 9 shows the differences, where the negative results represent a reduction, and the positive values are a percentage increase. The first part of the table considers only trajectory datasets for comparing with MASTERMovelets, and in the second part, we compare to HiPer-

Table 9 – Result differences of average total time, accuracy, number of movelet candidates, and number of movelets, in percent.

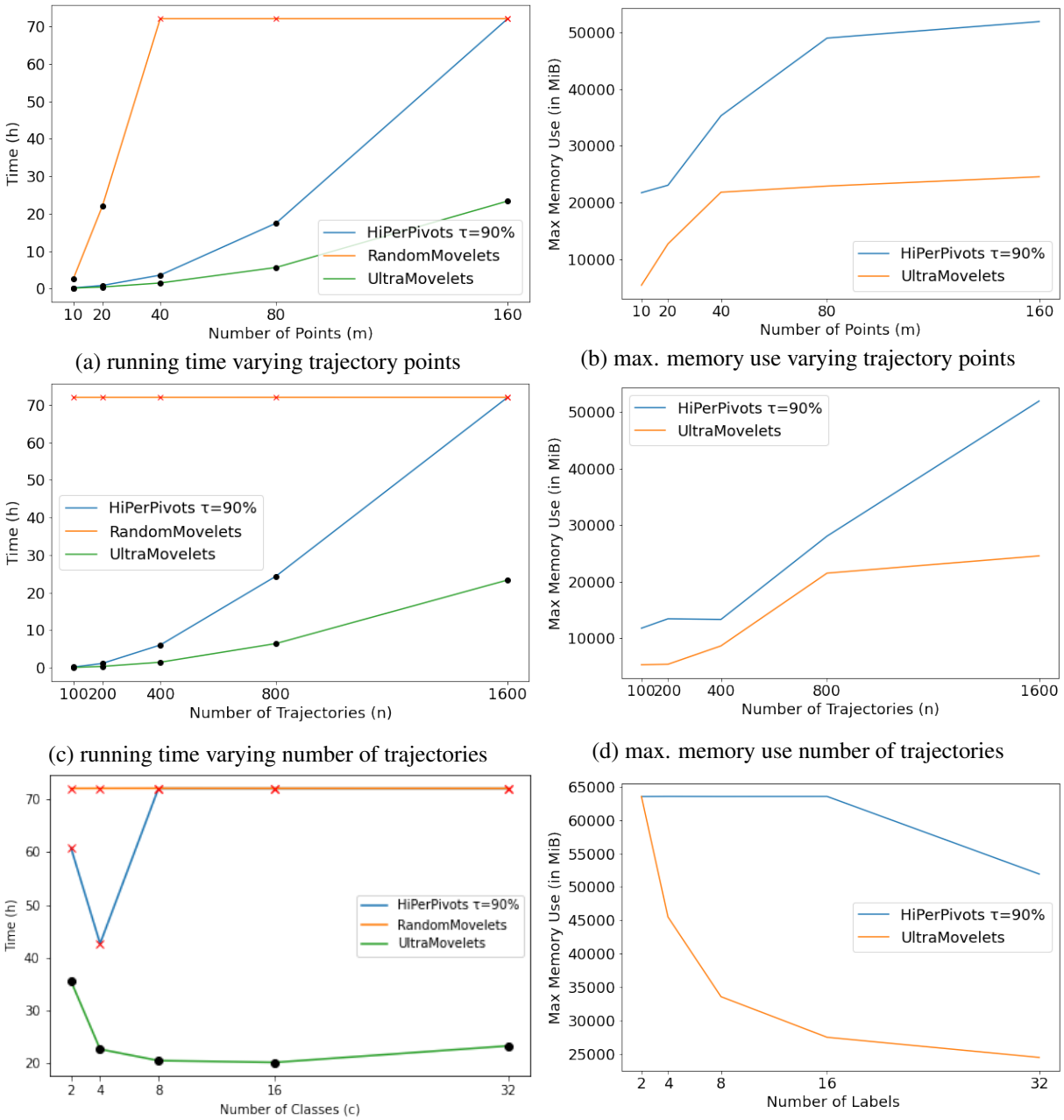
Reduction or increase (%) in trajectory datasets:				
	Total Time	**ACC (NN)**	**# of Candidates**	**# of Movelets**
MASTERMovelets-Log to HiPerPivots $\tau = 90\%$	**-50.72**	-3.17	**-91.99**	**-22.79**
MASTERMovelets-Log to RandomMovelets	**-41.42**	-2.20	**-89.99**	**-34.03**
MASTERMovelets-Log to UltraMovelets	**-65.62**	-1.01	**-87.25**	50.59
HiPerPivots $\tau = 90\%$ to RandomMovelets	146.69	**0.98**	128.74	0.18
HiPerPivots $\tau = 90\%$ to Ultra-Movelets	19.42	**2.39**	79.72	160.17
RandomMovelets to Ultra-Movelets	**-31.46**	**1.62**	27.36	164.57
Reduction or increase (%) in all datasets, including genetic sequences and Time Series:				
HiPerPivots $\tau = 90\%$ to RandomMovelets	50.24	**10.63**	6.73	22.27
HiPerPivots $\tau = 90\%$ to Ultra-Movelets	311.99	-6.15	51.93	291.94
RandomMovelets to Ultra-Movelets	423.75	-12.78	102.65	178.93

Pivots in all datasets. We highlighted in bold where the method compared to perform better. The first observation is in trajectory datasets that compared to MASTERMovelets-Log all three of our methods reduced the running time and the number of movelet candidates in at least 41% and 87%. Moreover, our methods are, on average, 6 to 10 times faster than MASTERMovelets, thus, indicating a significant reduction in the search space for movelets. UltraMovelets is the only method that increased the number of movelets, an average of 53%, while HiPerPivots and RandomMovelets reduced around 35%. It is essential to notice that the impact on accuracy is not a significant difference (reduction of 3.1% and lower 1.0%). However, when compared to HiPerPivots in trajectory datasets, RandomMovelets and UltraMovelets improved accuracy. When comparing to HiPerPivots in all datasets (which includes time series), another aspect that calls attention is that RandomMovelets improved accuracy.

3.2.3.4 Discussion of Preliminary Scalability Results

In this experiment, we evaluate how HiPerPivots, RandomMovelets, and UltraMovelets scale in different datasets and with different characteristics, comparing the computational time spent in the *movelets* extraction. We choose HiPerPivots as it shows the best performance from experimental evaluation in Section 3.1 and (PORTELA; CARVALHO; BOGORNY, 2022). We use synthetic datasets with different configurations inspired by the methodology adopted in Ye e Keogh (2011), Ferrero et al. (2020). In particular, we use three datasets: (i) with a fixed number of 1600 trajectories and varying the trajectory size from 10 to 160 points, 8 attributes in every trajectory, and 32 labels; (ii) with fixed trajectory size of 160 points and varying the number of trajectories from 100 to 1600 trajectories with 8 attributes, and 32 labels; and (iii) with a fixed number of 1600 trajectories, trajectory size of 160 points, 8 attributes in every trajectory, and varying the number of labels from 2 to 32. The configurations use a logarithmic scale. Figure 38 shows the performance results. The black point markers (on the left) indicate experiments that finished in the setup configurations, while the elements with an 'x' could not complete. We omitted the RandomMovelets from memory use evaluation as most experiments did not finish.

When observing the scalability in Figure 38 (a) and (c), we can notice that UltraMovelets performs much better than HiPerPivots, both in increasing trajectory points and the number of trajectories. HiPerPivots achieves the time limit with 1600 trajectories and 160 points, and RandomMovelets could not finish in most cases. This difference is expressive by comparing the maximum use of RAM (Figure 38 b and d). UltraMovelets appears to be more stable on memory use compared to HiPerPivots. Another interesting preliminary result is presented when varying the number of classes (Figure 38 f), the maximum memory use decreases as the number of classes increases. That happens as HiPerPivots has to keep movelets on memory while it finishes processing the trajectories of the class. It impacts the running time (Figure 38 e). However, in all three evaluations, UltraMovelets performs better under the curve of HiPerPivots.



(a) running time varying trajectory points

(b) max. memory use varying trajectory points

(c) running time varying number of trajectories

(d) max. memory use number of trajectories

(e) running time varying number of labels

(f) max. memory use varying number of labels

Figure 38 – Scalability analysis of running time (left) and maximum memory use (right).

3.3 SUMMARY

In this chapter, we focused on the problem of reducing the search space and the running time for movelet extraction. The main contribution is the proposed new HiPerMovelets method that reduces up to 90% of processing time compared to the state-of-the-art MASTERMovelets, keeping similar or high accuracy. We stated several results on the accuracy, number of movelets, and processing time comparing MARC, MASTERMovelets, HiPerMovelets, and HiPerPivots methods, and the last is ten times faster than MASTERMovelets. The method significantly reduced the number of movelet candidates, the number of movelets, the movelet extraction time, and the classification time. It is a significant result that it can achieve good classification results

with fewer movelets since the discriminant power of the movelets is not as much related to the quantity. Moreover, we showed that the scalability of HiPerMovelets performed better than MASTERMovelets when increasing the number of points, trajectories, and dimensions. The results in this chapter appeared in the journal paper (PORTELA; CARVALHO; BOGORNY, 2022).

We then investigated how we can use movelets in other domains to classify multidimensional sequential data. Therefore, in this chapter, we also proposed new RandomMovelets and UltraMovelets methods for movelet extraction. The RandomMovelets was proposed to be independent of the domain, aiming to overcome a limitation of HiPerMovelets on generalized datasets. The preliminary experimental results indicate that the random strategy of this method can achieve good results in several types of data. Furthermore, we attempt to deal with another significant limitation of previous methods (including HiPerMovelets and RandomMovelets), which is the high need for computational resources. We propose UltraMovelets as a method independent of parameters that reduce the search space and the number of generated movelet candidates. We, therefore, presented preliminary experimental results for classification in several datasets of raw, semantic, and multiple aspect trajectories, as well as in multivariate and univariate time series datasets. We compared several results of accuracy, processing time, number of generated movelets, and number of output movelets. In preliminary experimentation, UltraMovelets shows very promising in reducing the search space and mitigating the curse of dimensionality. More experiments are necessary to confirm these preliminary results. Based on the assessed results, performing fine adjustments to improve UltraMovelets would be possible. The code to reproduce the experiments is publicly available at https://github.com/bigdata-ufsc/HiPerMovelets.

4 AUTOMATIZE: A PLATFORM FOR MOVELETS ANALYSIS

We can observe from the experimental results presented in the previous chapter how movelets can be several in number and complex to understand due to the high dimensionality. As a further result of this thesis, we developed a tool to support the analyst in properly visualizing and analyzing movelets. In this chapter, we present a tool called AUTOMATIZE to give insights into the behavior of the classes. We describe the details of this tool and show its usefulness by discussing the movelets obtained in the experimental evaluation of Subsection 3.2.3. The content of this chapter has been presented in (PORTELA et al., 2022) and was awarded as the best demo runner-up (in International Conference on Mobile Data Management, 2022).

4.1 MULTIPLE ASPECT TRAJECTORY DATA MINING TOOL LIBRARY

Frameworks capable of trajectory data visualization as SCIKIT-MOBILITY[1] and GEOPANDAS[2] provide managing and analysis tools based on the raw spatio-temporal data. However, due to the intrinsic complexity of the data, the multiple aspect trajectory analysis needs more specific tools that could offer visualization associated with patterns discovered from multidimensional data. Moreover, another fundamental issue is how to make the patterns extracted from such high-dimensional data understandable.

Movelet-based trajectory data mining faces the following major challenges:

1. Visualizing the high dimensional data;
2. Visualizing the movelets associated to the trajectory data;
3. Providing the user with a unique platform for accessing the different tools available for movelet extraction and trajectory classification.

This section introduces the AUTOMATIZE framework (PORTELA et al., 2022), an interactive web system and python library that provides a friendly interface to run multiple aspect trajectory classification and analysis. More precisely, the system provides tools to (1) configure the experimental environment, (2) select movelets extraction methods, (3) run different classifiers, and (4) summarize and visualize the results. For example, a data analyst can use AUTOMATIZE for wildlife monitoring since he/she can visualize and compare movelets extracted from animal trajectories to classify species moving patterns.

The present tool can be useful for multiple aspect trajectories and in a more general setting of Multivariate Time Series data mining.

[1] https://github.com/scikit-mobility/scikit-mobility
[2] https://doi.org/10.5281/zenodo.705645

As a contribution to the research community, we make available AUTOMATIZE[3] as an online web application and as a Python library to provide developers and data analysts with a comprehensive platform for classification of multidimensional data.

4.1.1 System Architecture

The AUTOMATIZE framework is available as a library in Python that can be easily imported in scripts or notebook coding and as an interactive web platform that integrates public datasets, classification methods, and experimental results. The prototype web platform is currently working with public datasets of multiple aspect trajectories, spatio-temporal trajectories, and time series datasets[4].

AUTOMATIZE has a friendly web interface that allows the user to interact with most of the functions from the Python library. Thus, the user can perform the following activities: a) trajectory and movelet visualization, b) public dataset review and analysis, c) results exploration, d) experimental environment preparation, and e) related publications review. It is essential to highlight that the web-interface scalability is limited to the browser and web server resources. It is a limit that the user has to manage, as large datasets will be very hard to load and visualize.

The architecture of AUTOMATIZE is shown in Figure 39, is composed of five main modules: Data Preprocessing, Scripting, Analysis, Results, and Visualization Tools, described below.

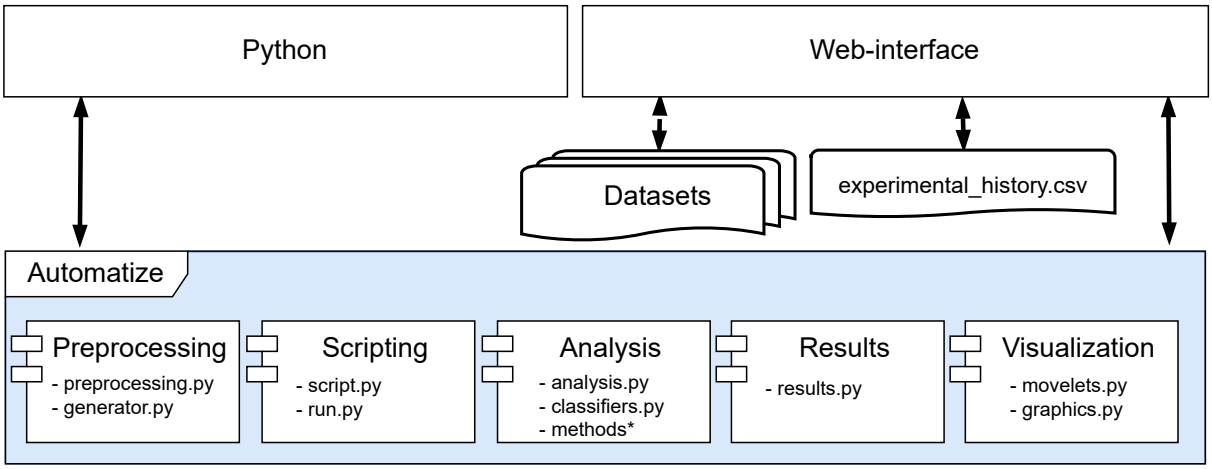


Figure 39 – The architecture of AUTOMATIZE platform.

Data Preprocessing: module capable of reading and writing trajectory datasets in three formats: (i) sequential data in comma-separated values, (ii) sequential data in cache format zip files, and (iii) multivariate time series files format. It provides methods for hold-out splitting and joining the data, *k*-fold data split, format conversions, and dataset statistics. This module

[3] Platform and source codes: https://github.com/ttportela/automatize
[4] From https://archive.ics.uci.edu/ and http://timeseriesclassification.com/

also is capable of generating trajectory datasets based on random or sampling data with control of the number of trajectories, points, dimensions, and classes;

Scripting: the scripting module provides functions to generate command-line scripts for running methods and classifiers. Available methods to generate scripts include methods developed in: (XIAO et al., 2017; ZHENG et al., 2010; DODGE; WEIBEL; FOROOTAN, 2009; FERRERO et al., 2018; VICENZI et al., 2020; May Petry et al., 2020; FERRERO et al., 2020; PORTELA et al., 2021; PORTELA; CARVALHO; BOGORNY, 2022), but it can be easily extended with other methods;

Analysis: provides python implementation of classifiers for multiple aspect trajectories and movelet-based methods: MARC (May Petry et al., 2020), POI-F (VICENZI et al., 2020), Multi-layer Perceptron (MLP), Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM) (implemented in (FERRERO et al., 2020));

Results: the module provides functions to read and compile statistics from the experimental results file, such as running time and accuracy. The web interface will read a result file pre-computed in (PORTELA; CARVALHO; BOGORNY, 2022). Four types of results visualization are provided: critical difference diagrams, box plots of the results, average ranks, and filtered raw results. Again, this can be easily extended to other results files;

Movelets Visualization: library of tools that provides visualization schemes for movelets as Sankey diagrams[5], Markov chain[6], class heat map per class, and a movelet tree visualization.

4.1.2 Visualization Tools

The AUTOMATIZE platform provides the following four types of trajectory and movelet visualizations:

Statistics: display statistics of the trajectories (number of trajectories, attributes, trajectories by class, trajectory sizes, and descriptive statistics by each dimension). When importing movelet files, it displays descriptive statistics about the movelets by class, such as the quality, sizes, and used features;

Trajectories and Movelets: displays the trajectory data in each dimension, ordered by the sequence of points. The trajectory points present in movelets (for each dimension) are highlighted. The user can interact by selecting the range of trajectories and the dimensions to display. It is important to highlight that the interface allows the user to visualize trajectories and their movelets together;

Movelets: this interface displays the movelets of one selected trajectory (Figure 40). The trajectory points are here displayed aligned with its movelets. This visualization shows the position in the trajectory where the movelets were extracted or they best fit. The user can look for insights from the movelet patterns that could characterize the class behavior;

[5] https://en.wikipedia.org/wiki/Sankey_diagram
[6] https://setosa.io/ev/markov-chains/

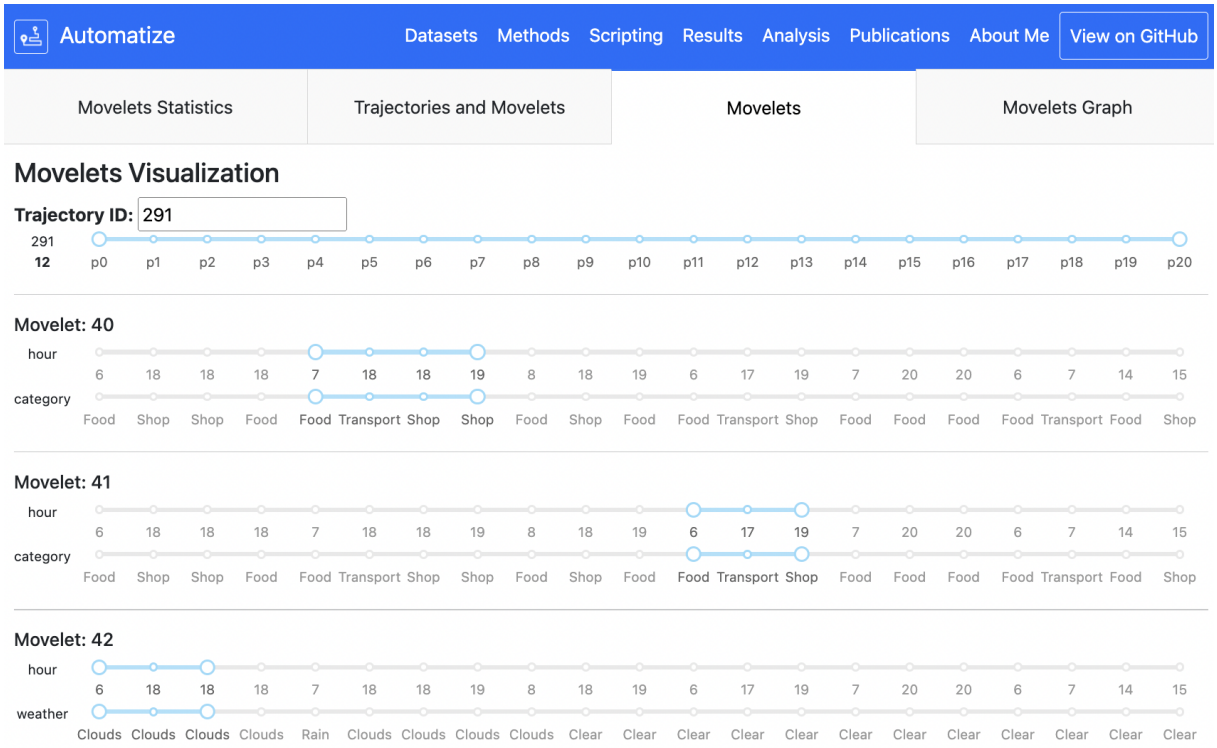


Figure 40 – Movelets visualization screen for selected trajectory.

Movelets Graph: provides three forms of graphical visualization of the movelets, where the user can select the movelet range, the dimension to display, and the visualization type:

- *The Movelet Sankey Diagram* is a visualization tool that shows the movelet sequences flowing from one set of values to another. It depicts the movelet sequences, starting with the class label, in an interactive way to show how the movelets can inter-connect;
- *The Movelet Markov Chain* shows how the movelet patterns flow from one point to another as a directed graph network. The user can visualize the points and the sequences that connect these points inside movelets;
- *The Class Heat Map* provides a heat map view of the movelets attributes frequency by the class label. It displays the frequency of an attribute that appears in the movelets of each class. Thus, helping to understand which attributes are best to discriminate the classes;
- *The Movelet Tree* provides a tree view of the movelets ordered by the higher quality value computed as F-Score (FERRERO et al., 2020) and aggregated based on overlapping elements. In other words, it displays the movelets in a top-bottom order of the most relevant subtrajectories to the less significant. It is a simple way of organizing the movelets to help the user understand a class behavior.

Results Exploration: The Results Exploration interface provides ways to summarize, filter, and explore the obtained results with experimental evaluations. The user can display a

predefined results file or load a new one in comma-separated values. The user can filter results and build rankings of methods by selecting datasets, methods, and classifiers. The rankings are built by accuracy, running time of the method execution, and classification running times. Four types of result visualizations are provided: critical difference diagrams, box plots of the results, average ranks, and filtered raw results. The displayed results can be filtered and exported to a file.

4.2 MOVELETS ANALYSIS AND STATISTICS

Multidimensional and sequential data is not trivial by definition, thus, complex to analyze. Movelets present an opportunity to understand the patterns of this data. In this section, we analyze the statistics of the resulting movelets of Subsection 3.2.3 obtained with the methods: MASTERMovelets, HiPerMovelets, RandomMovelets, and UltraMovelets. First, we analyze the number of movelets, the average number of features, size and quality of generated movelets by each method (Subsection 4.2.1). Since the number of movelets is very high, we presented the distribution box plot of the average on each dataset for the number of features, size, and quality of generated movelets. That is to maintain a balanced visual representation between the datasets. Then, we discuss the attribute dependency by the class label in the Foursquare NYC dataset (Subsection 4.2.2). We present this analysis only in trajectory datasets to compare with MASTERMovelets.

4.2.1 Number of Movelets, Features, Size, and Quality

To better understand how each movelet-based method works, we present an analysis of movelet characteristics in this section. We present the distribution of the number of movelets and the average number of features, size, and quality of generated movelets by each method on trajectory datasets, and the two datasets of genetic sequences. The results presented in this section correspond to one evaluation of each method on each dataset, to be fair, since raw datasets are evaluated in five folds. We included the raw, semantic, and multiple aspect trajectory datasets from experimental evaluation in Subsection 3.2.3.

The movelets are input to a classifier, which in very large datasets will impact the classifier running time. As presented in Figure 41, the methods with more extracted movelets are MASTERMovelets, and UltraMovelets. Indeed, UltraMovelets is expected to output more movelets since it can select subtrajectories with the same points in different dimensions, different from other compared methods. However, as shown in Subsection 3.2.3, it is a disadvantage of the method since the classification accuracy is not significantly higher compared to our other methods. The method with fewer extracted movelets is MASTERPivots, followed by HiPerMovelets and HiPerPivots, as we consider the mean values. For both HiPerMovelets variations, the value of τ seems to have little impact on the total number of output movelets. HiPer-

Movelets methods are designed to generate fewer movelets due to their pruning strategies, and the reduction, compared to MASTERMovelets, is significant.

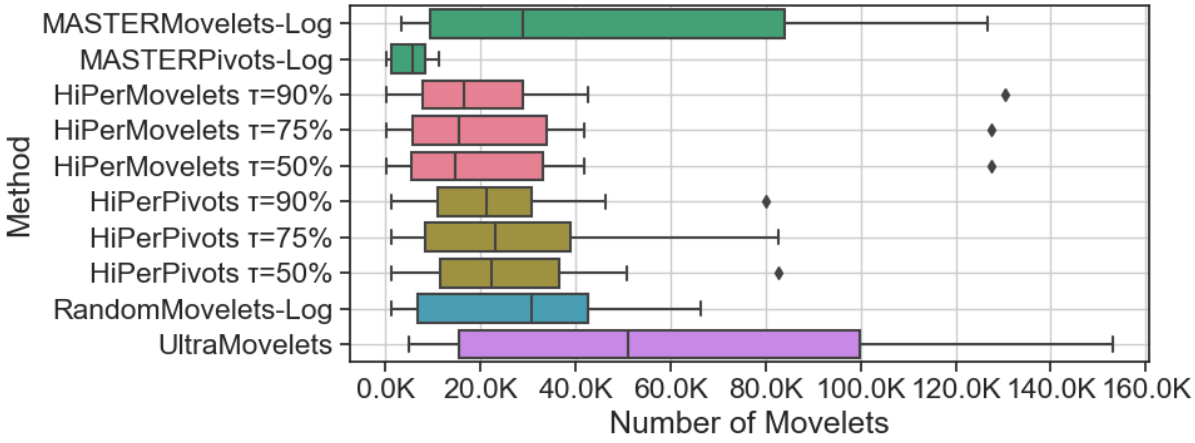


Figure 41 – Distribution box plots of the number of movelets by method.

Movelets are subtrajectories that combine different dimensions and points to represent relevant patterns in the trajectories of a class. The number of features in movelets is the combined dimensions in that subtrajectory. We analyze the average number of features of movelets extracted in each dataset, as presented in Figure 42. On average, all methods generate movelets with less than 3.5 combined dimensions, most with a mean average below 2 dimensions. That indicates that no much more than 2 dimensions combined are necessary to discriminate behaviors. On average, the UltraMovelets extract movelets with fewer dimensions combined, as it does only the necessary combinations while quality improves. The methods seem to present the most differences when comparing the average size of the subtrajectories. As shown in Figure 43, on average, HiPerPivots is the method that extracts the shorter movelets, followed by HiPerMovelets and UltraMovelets, respectively. RandomMovelets and MASTERMovelets extract movelets with longer subsequences of the trajectories that have more variation of the average size between datasets (larger interquartile range). On average, subtrajectory sizes are all under three points, meaning that most subtrajectories up to three points appear to be the most discriminant, an important aspect of behavior analysis. That can indicate that short subtrajectories can be the most discriminant.

The quality of movelets represents how discriminant the subtrajectory is for classification purposes. Since all compared methods employ *F-score* as a quality measure, we compare the average quality of the movelets presented in Figure 44. Between the presented methods, the quality is very similar. Most average quality values are between 40% to 60%. The quality alone, however, is not enough to tell how well a classifier will perform on a set of movelets. Indeed, it is important to how representative the movelets are, which depends on whether they cover sufficient examples of the moving object trajectories and how distinct they are to the classifier to reduce confusion.

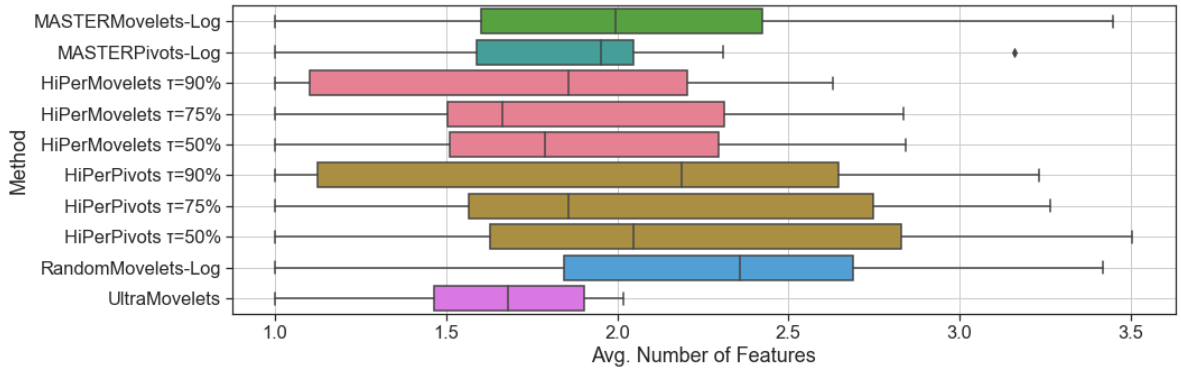


Figure 42 – Distribution box plots of average movelet number of attributes by method.

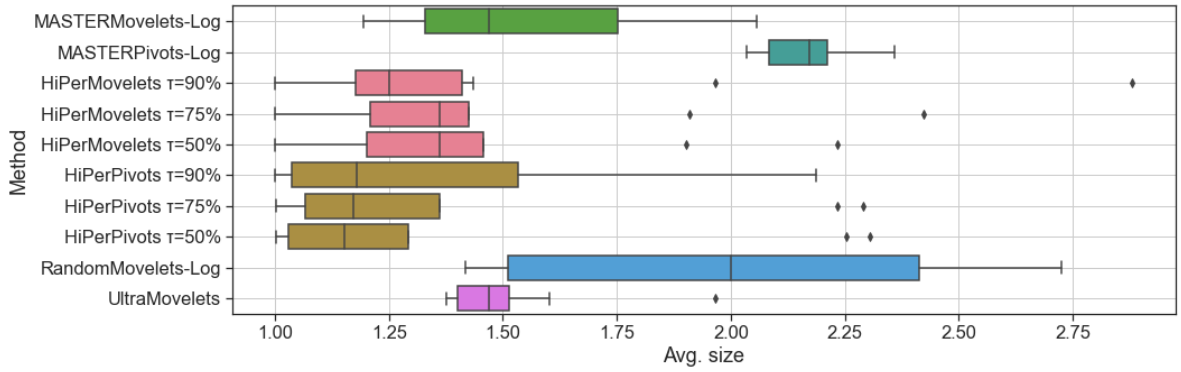


Figure 43 – Distribution box plots for average movelet number of points by method.

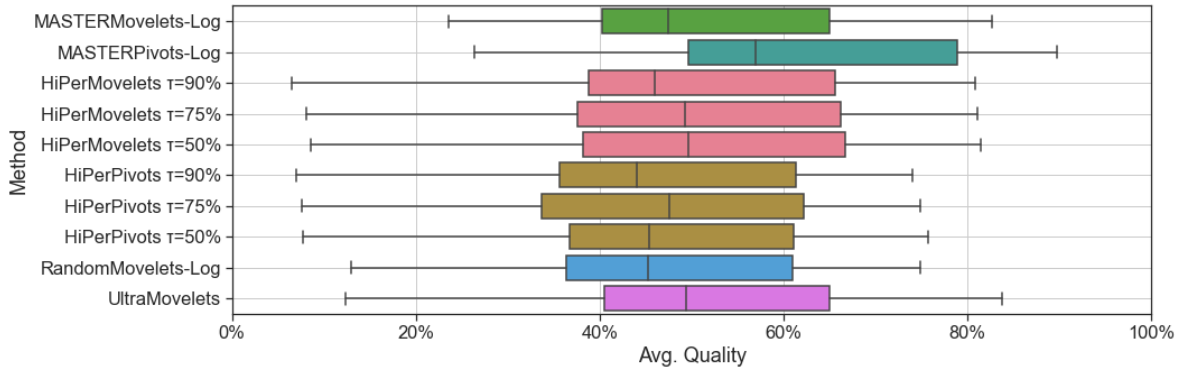


Figure 44 – Distribution box plots of average movelet quality by method.

4.2.2 Attribute Confidence Analysis

The movelets can indicate the most discriminant attributes on a given dataset. When an attribute is good at separating the classes, it is expected to appear more in movelets. We analyze the attribute confidence of our methods in comparison to MASTERMovelets by the proportion of each attribute appearing on extracted movelets. We employ AUTOMATIZE tool to generate heat map plots in a proportion scale of 0 to 1, representing the ratio of attribute occurrence by the total number of movelets. First, we present the movelet attribute use on the Foursquare NYC dataset with specific and generic configurations by each movelet-based method. Second,

we present the proportion of each attribute by the 30 first users.

Figure 45 – Heat map for movelet attribute use by method in Foursquare NYC specific.

Figure 46 – Heat map for movelet attribute use by method in Foursquare NYC generic.

Figure 45 presents the attribute frequency (from 0 to 1) for Foursquare NYC specific dataset. Latitude and Longitude (lat and lon) appear equal in proportion, as they are always treated as one spatial attribute, never split by the methods. MASTERMovelets relies most on spatial and POI attributes, while other methods rely most on Spatial data, followed by the hour, category, and day in almost half of the movelets. The spatial information is very specific for classification, thus, is expected to be very present in movelets. POI depends on the spatial attribute and is specific enough to be frequently selected. The Hour and Day attributes appear to be very important in discriminating the behaviors of the classes. Indeed, Foursquare NYC represents check-in trajectories of people segmented by week spam, which can indicate people behaviors repeat at specific times and days of the week. Moreover, when removing the very specific data (the spatial and POI attributes), the movelets reliance on Hour and Day attributes increase, as presented in Figure 46. In this scenario, Category and Rating also appear more frequently. The category attribute represents the POI generic categorization and depends on spatial information. The Rating and Price depend on the POI, but the place Rating appears more

discriminant of the class behaviors than the Price for some methods. The weather condition, however, has the least impact on discriminating classes, less common in the movelets.

We chose to detail the attribute reliance of the first 30 class labels for movelets extracted with HiPerPivots on Foursquare NYC generic, as presented in Figure 47. The first observation is that the day of the week attribute is present in over 50% of movelets in all classes. However, the weather condition seems much less relevant for any class labels. The Hour and Category also appeared more consistently in the classes, except for users 172 and 642, mainly distinguished by the day of the week. That indicates the different importance of each attribute to represent the behaviors of such classes.

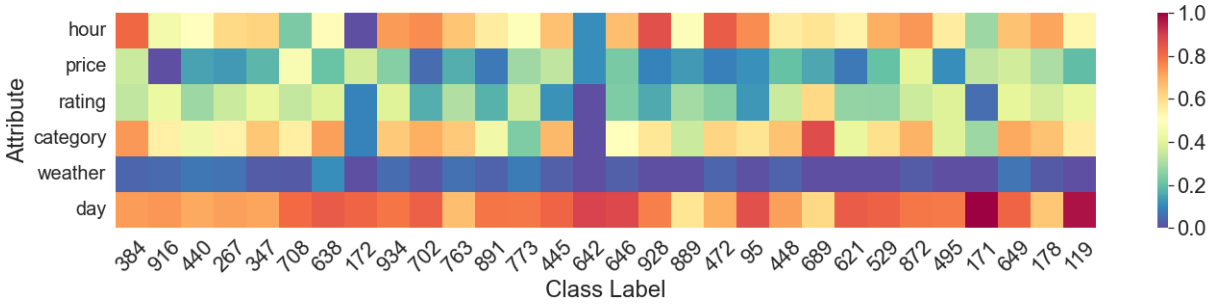


Figure 47 – Heat map for movelet attribute use by class in Foursquare NYC generic.

We argue that we can discover knowledge from the movelets not only by visualizing the subtrajectories but also by analyzing the characteristics of the movelets of the classes. It can reveal important information about class behaviors. Visualizing the movelets and the multidimensional trajectory is a powerful way of investigating classification results. Thus, the platform described in this chapter is a valuable tool to support analysts and researchers on trajectory data mining.

4.3 SUMMARY

In this chapter, we presented the AUTOMATIZE platform, both a web interface and a python library that provides tools to perform trajectory and multidimensional data classification tasks. AUTOMATIZE is tailored for multiple aspect trajectories and more general multidimensional sequential datasets and offers several options to visualize both trajectories and movelets. That speeds up the task of creating interactive visualizations that data analysts need to compare and offers unique views of the movelets that can give insights to the data analyst. The main features of this platform are supporting the analyst, or researcher, to develop experimental environments, retrieve classification results and visualize the trajectory classification data. The tool also supports the preprocessing of data, including generating synthetic datasets.

We discussed the movelets obtained in several trajectory datasets with MASTER-Movelets, HiPerMovelets, RandomMovelets, and UltraMovelets. Using the tool, we analyzed the distribution of the number of movelets, and the distribution averages of the number of features, size, and quality of generated movelets, presenting differences and similarities of each

method. This statistical analysis of movelets aims to give insights into the knowledge to be discovered in movelets. We presented the frequency of appearance in movelets of attributes from the Foursquare NYC dataset in both specific and generic data. This analysis reveals what dimensions better discriminate the classes and the fundamental differences between them. For instance, a person that is almost only distinguished by the attribute time can indicate that his behavior is particularly governed by the periods of time it moves and is less dependent on the visited places. We discussed the differences between methods, attribute dependencies, and possibilities to analyze discriminant attributes by the class label. The platform and source codes are available at https://github.com/ttportela/automatize.

5 CONCLUSION AND FUTURE WORKS

In this thesis, we addressed the problem of multiple aspect trajectory (MAT) classification, an important topic in trajectory data mining. Existing trajectory classification methods have mainly been developed for raw trajectory data, considering only space and time dimensions, ignoring the large number of semantic attributes of MATs.

Among the works that can deal with MAT trajectories and that show the best accuracy are MARC and MASTERMovelets. MARC is based on embeddings of the dimensions, and MASTERMovelets extracts discriminant subtrajectories called *movelets*. MASTERMovelets generates interpretable patterns, i.e., trajectory parts that better discriminate a class. Although the method presents high accuracy, it has a high computational cost, making it almost impossible to apply to large datasets.

Trying to answer our research question "**Can we develop new algorithms for discovering movelets that are faster than Movelets and MASTERMovelets maintaining or improving classification accuracy?**" we tackled the trajectory classification problem proposing faster, scalable, and still accurate strategies for finding *movelets*. We propose a main method called HiPerMovelets, two methods that have shown promising results for other domains called RandomMovelets and UltraMovelets, and a tool for facilitating trajectory classification and movelet analysis.

HiPerMovelets is an in-class pruning and frequency-based strategy for movelet discovery. HiPerMovelets uses a greedy approach with a frequency-based technique to find the most discriminant subtrajectories for classification problems. It does not compute the distance matrix of a trajectory point to all other trajectory points in the dataset, but to the points of the trajectories of a single class. It extracts *movelets* only from subtrajectories that occur more in the trajectories of the same class and then evaluates the subtrajectories by calculating their relative frequency of occurrence in the trajectories of the class. HiPerMovelets also limits the number of trajectories searched in the dataset by counting the covered trajectories from the *movelets*. HiPerPivots is a variation that can further reduce the search space by selecting pivot points more likely to generate movelets. Experimental results show that HiPerMovelets and HiPerPivots are 10 times faster than MASTERMovelets, reduce the high dimensionality problem, are more scalable, and present a high classification accuracy in all evaluated datasets with both raw and multiple aspect trajectories. These methods, specially HiPerPivots, are more scalable than MASTERMovelets when comparing increasing number of trajectories, trajectory points, and dimensions. In comparison to MARC, our proposals are more accurate and generates interpretable patterns.

A limitation to HiPerMovelets is that not always frequency is the best strategy for efficiently searching movelets. In some datasets as generalized data, where the very specific dimensions are omitted or replaced for categorical information, the most discriminant patterns are less frequent. As secondary contributions we proposed RandomMovelets and UltraMovelets methods to reduce the search space for movelets.

RandomMovelets uses a random strategy that gives the movelets the same chances to be selected without other pre selection calculations. Movelet candidates are randomly selected and then evaluated for movelet discovery. It reduces the search space by limiting the number of candidates compared to the dataset. We investigate the method performance in raw, semantic, multiple aspect (including a generic dataset), and time series data. Preliminary results indicate RandomMovelets has good classification accuracy and a reasonable running time. Compared to MASTERMovelets it reduces the running time by half on average. It also reduces the number of movelet candidates and movelets keeping similar accuracy as MASTERMovelets and improved accuracy compared to HiPerPivots. The RandomMovelets method shows improvements in accuracy also for other domains compared HiPerPivots, and in the generic dataset.

The UltraMovelets method recursively selects the best combination of subtrajectory points and dimensions to evaluate movelets, thus reducing the high number of explored candidates. It early stops the combinatorial explosion of points and dimensions from subtrajectories that will not increase movelet quality. It shows promising results in reducing the generated movelet candidates, does not require any parameter configuration, reduces the generated movelet candidates, and scales better than HiPerPivots. Reducing the combinatorial exploration of subtrajectories shows a direct impact on reducing memory resources. UltraMovelets performed better in scalability experiments comparing to HiPerPivots, the method that previously showed better scalability results.

Both RandomMovelets and UltraMovelets in preliminary experiments showed improvement in accuracy in detriment of running time, which shows a trade-off between the time to search for the movelets and the classification accuracy. RandomMovelets is a strategy best suited for problems where the dataset characteristics are unknown or little known. The random uncertainty is a limitation of the method, but in some level mitigated by the iterative search of movelets, which will evaluate all movelet candidates if necessary. The UltraMovelets method, instead, is purely based on the *F-Score* quality to search movelets, which is more reliable to the patterns discovered. The UltraMovelets are different patterns from the discovered from HiPerMovelets and RandomMovelets considering the size and dimensions present in the output movelets. It can represent a different knowledge extracted from the data.

In order to facilitate the trajectory classification task, we proposed AUTOMATIZE. It is a tool to support the user in the classification task of multiple aspect trajectories, specifically for extracting and visualizing the *movelets*. It integrates into a unique platform the fragmented approaches available in the literature for multiple aspect trajectories and, in general, for multidimensional sequence classification into a unique web-based and python library system. The tool provides resources to:

1. Configure the experimental environment;
2. Select movelets extraction methods;
3. Run different classifiers;

4. Summarize and visualize the results:

5. Visualize the multidimensional data and the movelets;

6. Preprocess the data and generate synthetic datasets.

The AUTOMATIZE tool is tailored for multiple aspect trajectories but also to more general multidimensional sequential datasets. It offers several options to visualize trajectories and movelets, by hiding many details of the python coding. Due to the intrinsic complexity of the multiple aspect trajectory data, the analysts need more specific tools that could offer visualization associated with patterns discovered from multidimensional data. Multiple aspects trajectories is a model general enough to represent other domains, such as the multivariate time series, event logs, or genetic sequences. AUTOMATIZE is an open source platform that was designed to be easily extended for other types of data and feature visualizations.

It is important to highlight that movelets are interpretable and so far, the only representation of sequential patterns from the multidimensional data. Movelets present an opportunity for data analysts to understand the behaviors of multiple aspect trajectories, the discriminant sequential patterns, and which dimensions characterize the classes. Movelets can be several in number and complex to understand due to their high dimensionality, therefore, a proper tool for visualizing and analyzing movelets such as AUTOMATIZE is needed to give insights about the behavior of the classes for data analysts.

Another fundamental issue we tackled is how to make understandable the patterns extracted from the high dimensional trajectory data. We explored the discovered movelets comparing MASTERMovelets, HiPerMovelets, RandomMovelets, and UltraMovelets in several datasets. From the distribution of the number of movelets, the number of sequential points, the number of attributes, and the quality to have insights about the characteristics of the methods. The main conclusions is, from the reduced number of movelets generated by HiPerMovelets, it enables to the production of faster classification models. RandomMovelets presents more variability in movelets size and number of attributes, being able to perform well in datasets with very different characteristics (like the specific and generic data). The UltraMovelets method, however, produces simpler subtrajectories that are smaller in size and with fewer attributes. In terms of the average quality of the movelets, the methods present low variability, which the results indicate their ability to perform high accuracy classification.

In order to overcome some limitations of our methods, we propose the following future works.

For trajectory classification it is important to find a way of counting the presence or absence of the movelet in a trajectory, and if it can be done without the need of finding the split point, or best alignments, it is possible to improve the method performance. Thus, proposing a new strategy to do the split point and the best alignment that avoids calculating distances for every position in all trajectories is a solution. The split point is a multidimensional point that better separates trajectories from a target class and non-target classes. Finding the split

point requires comparing the distances of a movelet candidate to every trajectory in the dataset. The best alignment is done by calculating the distances (point-to-point) of that candidate to all possible positions in a trajectory (by aligning the points). Thus, for evaluating the *F-score* quality, first, point-to-point distances are calculated; the minimal distance of a candidate is chosen as the best alignment for each trajectory; and one of the best alignments is selected as the split point as it provides the higher *F-Score*. Those are complex steps that can be improved.

Movelets are evaluated with *F-score* quality, which works as an internal classifier to measure the relevance of the subtrajectories for discriminating classes. A solution might be neither using the best alignment nor the *F-score* but another efficient quality measure, as for instance, the Gini (that measures the inequality among different groups) or TF-IDF (term frequency and inverse document frequency). Other classification methods could be used to evaluate the subtrajectories quality like random forests, decision trees, linear or logistic regression, covariance, neural networks, etc.

The performance of extracting movelets was improved by not comparing all trajectory points, or combining all attributes in movelet discovery. Another solution could be a new method for distributed processing of trajectories. A parallel distributed method for movelets discovery can divide the dataset into many parts for distributed processing, thus, allowing to grow scalability for Big Data. The challenge to this is to discover movelets which have to be compared to subtrajectories of the entire dataset, thus requiring a new method for movelet evaluation.

Many classifiers are being used for trajectory classification. Random Forest and Multilayer Perceptron Neural Networks, Random Forests, and Support Vector Machines have been used for trajectory classification using movelets. Other methods have employed embeddings, deep learning, extracting features, and other recent techniques to classify trajectories. Another architecture for a classifier can be used to improve the results, as for instance the use of deep learning, ensemble methods, and convolutional neural networks with the aggregation of movelets. A new direction is to investigate new classifiers with movelets or the combination of movelets and new classification techniques to improve the classification results. It could be done with the combination of movelets, other features and ensemble methods.

The potential of movelet-based methods goes beyond the classification of multiple aspect trajectories. Indeed, research can benefit from experimental evaluations on any multidimensional sequential data and other domains. Movelets can potentially be a way of discovering interpretable patterns in other complex types of data.

The scalability experiments give insights into methods behaviors on increasing volumes of data. Extensive scalability experimentation can help identifying problems, and lead to solutions for more scalable methods. It is necessary to do scalability experiments on the number of trajectories, number of points, number of dimensions, number of classes, and threading parallelization.

Movelets are the patterns that discriminate classes. Movelets can have many applications beyond the task of trajectory classification. By identifying what discriminates classes

can be possible to do other mining tasks, such as: anonymization, by hiding what reveals the user identity; prediction, for instance, the level of a hurricane with the beginning of the storm trajectories; and clustering trajectories by finding similar subtrajectories. Thus, another future work is investigating other application domains for movelets.

BIBLIOGRAPHY

A. de Freitas., N. et al. Using deep learning for trajectory classification. In: INSTICC. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*. [S.l.]: SciTePress, 2021. p. 664–671. ISBN 978-989-758-484-8. ISSN 2184-433X.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: **Proceedings of the 20th International Conference on Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. (VLDB '94), p. 487–499. ISBN 1558601538.

BAGNALL, A. et al. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. **Data Mining and Knowledge Discovery**, v. 31, n. 3, p. 606–660, May 2017. ISSN 1573-756X. Disponível em: https://doi.org/10.1007/s10618-016-0483-9.

BAYAT, S. et al. GPS driving: a digital biomarker for preclinical alzheimer disease. **Alzheimer's Research & Therapy**, Springer Science and Business Media LLC, v. 13, n. 1, jun. 2021. Disponível em: https://doi.org/10.1186/s13195-021-00852-1.

BAYAT, S. et al. A GPS-based framework for understanding outdoor mobility patterns of older adults with dementia: An exploratory study. **Gerontology**, S. Karger AG, p. 1–15, abr. 2021. Disponível em: https://doi.org/10.1159/000515391.

BIAN, J. et al. Trajectory data classification: a review. **ACM Transactions on Intelligent Systems and Technology**, v. 10, n. 4, 2019. ISSN 21576912.

BOLBOL, A. et al. Inferring hybrid transportation modes from sparse gps data using a moving window svm classification. **Computers, Environment and Urban Systems**, Elsevier, v. 36, n. 6, p. 526–537, 2012.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: https://doi.org/10.1023/A:1010933404324.

CHEN, L.; ÖZSU, M. T.; ORIA, V. Robust and fast similarity search for moving object trajectories. **Proceedings of the ACM SIGMOD International Conference on Management of Data**, p. 491–502, 2005. ISSN 07308078.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Disponível em: https://doi.org/10.1145/2939672.2939785.

CHO, E.; MYERS, S. A.; LESKOVEC, J. Friendship and mobility: User movement in location-based social networks. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 1082–1090, 2011.

DABIRI, S.; HEASLIP, K. Inferring transportation modes from GPS trajectories using a convolutional neural network. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 86, n. August 2017, p. 360–371, 2018. ISSN 0968090X. Disponível em: https://doi.org/10.1016/j.trc.2017.11.021.

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 Ltd, v. 33, n. 6, p. 419–434, 2009. ISSN 01989715. Disponível em: http://dx.doi.org/10.1016/j.compenvurbsys.2009.07.008.

DRUCKER, H.; WU, D.; VAPNIK, V. N. Support vector machines for spam categorization. **IEEE Transactions on Neural networks**, IEEE, v. 10, n. 5, p. 1048–1054, 1999.

ENDO, Y. et al. Deep feature extraction from trajectories for transportation mode estimation. In: SPRINGER. **Pacific-Asia Conference on Knowledge Discovery and Data Mining**. [S.l.], 2016. p. 54–66.

ETEMAD, M.; Soares Júnior, A.; MATWIN, S. Predicting transportation modes of GPS trajectories using feature engineering and noise removal. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 10832 LNAI, n. ii, p. 259–264, 2018. ISSN 16113349.

FERRERO, C. A. **Discovering Relevant Subtrajectories for Multidimensional Trajectory Classification**. 118 p. Tese (PhD thesis) — Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, 2020.

FERRERO, C. A.; ALVARES, L. O.; BOGORNY, V. Multiple aspect trajectory data analysis: Research challenges and opportunities. **Proceedings of the Brazilian Symposium on GeoInformatics**, v. 2016-Novem, p. 56–67, 2016. ISSN 21794847. Disponível em: http://mtc-m16c.sid.inpe.br/col/sid.inpe.br/mtc-m16c/2017/02.21.11.53/doc/56-67ferrero.pdf.

FERRERO, C. A. et al. Movelets: Exploring relevant subtrajectories for robust trajectory classification. In: **Proceedings of the 33rd Annual ACM Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2018. (SAC '18), p. 849–856. ISBN 9781450351911. Disponível em: https://doi.org/10.1145/3167132.3167225.

FERRERO, C. A. et al. MasterMovelets: discovering heterogeneous movelets for multiple aspect trajectory classification. **Data Mining and Knowledge Discovery**, v. 34, n. 3, p. 652–680, 2020. Disponível em: https://doi.org/10.1007/s10618-020-00676-x.

FURTADO, A. S. et al. Unveiling movement uncertainty for robust trajectory similarity analysis. **International Journal of Geographical Information Science**, Taylor & Francis, v. 32, n. 1, p. 140–168, 2018. Disponível em: https://doi.org/10.1080/13658816.2017.1372763.

FURTADO, A. S. et al. Multidimensional similarity measuring for semantic trajectories. **Transactions in GIS**, v. 20, n. 2, p. 280–298, 2016. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12156.

GAO, Q. et al. Identifying human mobility via trajectory embeddings. **IJCAI International Joint Conference on Artificial Intelligence**, p. 1689–1695, 2017. ISSN 10450823.

GIGLI, A. et al. Fast feature selection for learning to rank. **ICTIR 2016 - Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval**, p. 167–170, 2016.

HOLT, G. T. et al. Multi-Dimensional Dynamic Time Warping for Gesture Recognition. **Annual Conference of the Advanced School for Computing and Imaging**, 2007.

JI, C. et al. A fast shapelet selection algorithm for time series classification. **Computer Networks**, v. 148, p. 231–240, 2019. ISSN 13891286.

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. C. Representation of events in nerve nets and finite automata. In: ____. **Automata Studies. (AM-34), Volume 34**. Princeton: Princeton University Press, 2016. p. 3–42. Disponível em: https://doi.org/10.1515/9781400882618-002.

LEE, J. G. et al. TraClass: Trajectory classification using hierarchical region based and trajectory based clustering. **Proceedings of the VLDB Endowment**, v. 1, n. 1, p. 1081–1094, 2008. ISSN 21508097.

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

LEHMANN, A. L.; ALVARES, L. O.; BOGORNY, V. Smsm: a similarity measure for trajectory stops and moves. **International Journal of Geographical Information Science**, Taylor & Francis, v. 33, n. 9, p. 1847–1872, 2019. Disponível em: https://doi.org/10.1080/13658816.2019.1605074.

LEITE, C.; PETRY, L. M.; BOGORNY, V. A Survey and Comparison of Trajectory Classification Methods. **2019 8th Brazilian Conference on Intelligent Systems (BRACIS) (submitted)**, p. 788–793, 2019.

Leite da Silva, C. **Pivot-based approaches for Movelets and MASTERMovelets Optimizations**. 88 p. Tese (Master thesis) — Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, 2020.

May Petry, L. et al. MARC: a robust method for multiple-aspect trajectory classification via space, time, and semantic embeddings. **International Journal of Geographical Information Science**, 2020. ISSN 13623087.

MELLO, R. d. S. et al. MASTER: A multiple aspect view on trajectories. **Transactions in GIS**, 2019. ISSN 14679671.

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.

MUEEN, A.; KEOGH, E.; YOUNG, N. Logical-shapelets: An expressive primitive for time series classification. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 1154–1162, 2011.

NIEMEYER, G. **Geohash**. 2008. [Online]. Available: https://en.wikipedia.org/wiki/Geohash.

PATEL, D. et al. Incorporating duration information for trajectory classification. In: **Proceedings of the 2012 IEEE 28th International Conference on Data Engineering**. USA: IEEE Computer Society, 2012. (ICDE '12), p. 1132–1143. ISBN 9780769547473. Disponível em: https://doi.org/10.1109/ICDE.2012.72.

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.

PORTELA, T. T. et al. Automatise: Multiple aspect trajectory data mining tool library. In: **2022 23rd IEEE International Conference on Mobile Data Management (MDM)**. Online: [s.n.], 2022. p. 282–285. **Awarded: MDM 2022 Best Demo Runner-Up**. **[Detailed in Chapter 4]**.

PORTELA, T. T.; CARVALHO, J. T.; BOGORNY, V. Hipermovelets: high-performance movelet extraction for trajectory classification. **International Journal of Geographical Information Science**, Taylor & Francis, v. 0, n. 0, p. 1–25, 2022. **[Detailed in Section 3.1]**. Disponível em: https://doi.org/10.1080/13658816.2021.2018593.

PORTELA, T. T. et al. Fast movelet extraction and dimensionality reduction for robust multiple aspect trajectory classification. In: BRITTO, A.; DELGADO, K. V. (Ed.). **2021 10th Brazilian Conference on Intelligent Systems (BRACIS)**. Cham: Springer International Publishing, 2021. p. 468–483. ISBN 978-3-030-91702-9. **[Detailed in Chapter 2]**.

PORTELA, T. T.; VICENZI, F.; BOGORNY, V. Trajectory data privacy: Research challenges and opportunities. In: FILHO, J. L.; MONTEIRO, A. M. V. (Ed.). **XX Brazilian Symposium on Geoinformatics - GeoInfo 2019, São José dos Campos, SP, Brazil, November 11-13, 2019**. MCTIC/INPE, 2019. p. 99–110. Disponível em: http://urlib.net/rep/8JMKD3MGPDW34R/3UFDFDB.

RAKTHANMANON, T.; KEOGH, E. Fast shapelets: A scalable algorithm for discovering time series shapelets. **Proceedings of the 2013 SIAM International Conference on Data Mining, SDM 2013**, p. 668–676, 2013.

RUIZ, A. P. et al. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. **Data Mining and Knowledge Discovery**, v. 35, n. 2, p. 401–449, Mar 2021. ISSN 1573-756X. Disponível em: https://doi.org/10.1007/s10618-020-00727-3.

SANTOS, I. J. P. dos; JR, I.; ALVARES, L. O. TRACTS: um método para classificação de trajetórias de objetos móveis usando séries temporais. In: **XXXI Congresso da Sociedade Brasileira de Computação**. [S.l.: s.n.], 2011.

SHARMA, L. K. et al. Nearest neighbour classification for trajectory data. In: SPRINGER. **International Conference on Advances in Information and Communication Technologies**. [S.l.], 2010. p. 180–185.

SHOKOOHI-YEKTA, M. et al. Generalizing DTW to the multi-dimensional case requires an adaptive approach. **Data Mining and Knowledge Discovery**, Springer US, v. 31, n. 1, p. 1–31, 2017. ISSN 1573756X.

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 and Knowledge Engineering**, Elsevier, v. 65, n. 1, p. 126–146, 2008. ISSN 0169023X.

TRAGOPOULOU, S.; VARLAMIS, I.; EIRINAKI, M. Classification of movement data concerning user's activity recognition via mobile phones. **ACM International Conference Proceeding Series**, 2014.

VARLAMIS, I. Evolutionary data sampling for user movement classification. **2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings**, IEEE, p. 730–737, 2015.

VICENZI, F. et al. Exploring frequency-based approaches for efficient trajectory classification. **Proceedings of the ACM Symposium on Applied Computing**, p. 624–631, 2020.

VLACHOS, M.; KOLLIOS, G.; GUNOPULOS, D. Discovering similar multidimensional trajectories. **Proceedings - International Conference on Data Engineering**, p. 673–684, 2002. ISSN 10844627.

WANG, H. et al. Detecting transportation modes using deep neural network. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 100, n. 5, p. 1132–1135, 2017.

XIAO, Z. et al. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. **ISPRS International Journal of Geo-Information**, v. 6, n. 2, 2017. ISSN 22209964.

YANG, D. et al. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, n. 1, p. 129–142, 2015.

YE, L.; KEOGH, E. Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. **Data Mining and Knowledge Discovery**, v. 22, n. 1-2, p. 149–182, 2011. ISSN 13845810.

ZHANG, Z. et al. Discriminative extraction of features from time series. **Neurocomputing**, v. 275, p. 2317–2328, 2018. ISSN 18728286.

ZHENG, Y. Trajectory data mining: An overview. **ACM Transactions on Intelligent Systems and Technology**, v. 6, n. 3, p. 1–41, 2015. ISSN 21576912.

ZHENG, Y. et al. Understanding transportation modes based on GPS data for web applications. **ACM Transactions on the Web**, v. 4, n. 1, 2010. ISSN 15591131.

ZHENG, Y. et al. Understanding mobility based on gps data. In: ACM. **Proceedings of the 10th international conference on Ubiquitous computing**. [S.l.], 2008. p. 312–321.

ZHOU, F. et al. Trajectory-user linking via variational autoencoder. **IJCAI International Joint Conference on Artificial Intelligence**, v. 2018-July, p. 3212–3218, 2018. ISSN 10450823.

ZUO, J.; ZEITOUNI, K.; TAHER, Y. Exploring interpretable features for large time series with se4tec. In: **22nd International Conference on Extending Database Technology, EDBT 2019**. [S.l.: s.n.], 2019.

6 EXPERIMENTAL RESULTS FOR SEVERAL DATASETS

In this appendix, we present a more detailed description of the results obtained in the experimental evaluation done during the Ph.D. period. The following sections provide a summary of the results obtained in experimental evaluations. First, we present the compilation of results in tables for all datasets. Then, we present a summary of the results as box plots, organized into four parts: experiments with multiple aspect trajectory datasets, experiments with raw trajectory datasets, genetic sequence datasets, and time series datasets.

6.1 ACCURACY RESULTS FOR ALL DATASETS WITH NEURAL NETWORK CLASSIFIER

Table 10 – Results of Accuracy with neural network classifier for all dataset.

Dataset	Dodge	Xiao	Zheng	Movelets	RF	XGBoost	DeepeST	NPOI				MARC	MASTERMovelets		HiPerMovelets			HiPerPivots			Random-	Ultra-
								1	2	3	1+2+3		Log	Log+Pivots	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	Movelets	Movelets
Animals	61.569	76.256	76.150	78.456	80.175	77.885	63.397	86.123	57.717	37.226	82.113	79.456	87.175	86.323	85.471	87.376	**89.281**	83.466	86.471	85.271	*88.228*	87.175
GoTrack	**	**	66.347	73.553	73.871	78.257	71.905	82.321	80.446	74.398	82.265	74.804	83.497	*83.515**	82.909	83.497	**84.049**	79.288	80.447	81.053	82.854	81.035
Vehicles	90.293	96.589	88.166	97.108	91.307	90.821	71.077	71.657	72.436	71.657	72.970	60.817	98.161	**98.947**	98.268	*98.687*	98.421	*98.687*	*98.687*	*98.687*	98.421	98.168
Gowalla					70.525	60.944	95.205	95.698	82.476	56.453	93.704	*96.222*	**96.537**	94.439	*96.222*	95.698	96.118	95.803	96.013	96.118	95.908	95.383
Brightkite					91.831	86.093	95.950	95.779	95.031	91.967	95.167	96.460	96.732	96.256	96.596	96.596	96.801	*97.073*	96.732	96.801	**97.209**	*97.073*
FoursquareNYC (generic)					17.489	18.135	73.271	38.164	43.759	41.894	41.033	**99.713**	*80.488*	70.014	67.001	73.171	75.897	74.175	75.323	76.040	78.623	74.462
FoursquareNYC					48.795	45.882	98.766	*99.139*	92.539	67.862	98.852	*99.139*	98.422	96.844	98.709	84.935	85.366	98.852	88.092	88.235	**99.283**	98.135
Promoters					**100.000**	**100.000**	46.364	86.364	*95.455*	**100.000**	**100.000**	45.455	77.273	77.273	50.000	50.000	50.000	77.273	81.818	81.818	77.273	77.273
SJGS					*94.444*	**95.321**	52.128	53.991	57.121	63.224	62.441	43.349	63.067	51.956	50.704	57.433	57.746	55.399	58.998	59.311	51.643	63.693
ArticularyWordRecognition								9.333	7.000	4.333	10.000				95.667	**98.000**	97.333	94.000	**98.000**	**98.000**	*97.667*	96.667
AtrialFibrillation								**40.000**	**40.000**	*26.667*	*26.667*				13.333	20.000	20.000	20.000	20.000	13.333	20.000	13.333
BasicMotions								*32.500*	25.000	25.000	27.500				**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**
CharacterTrajectories								8.565	9.958	9.262	8.426				91.086	**93.245**	92.618	90.460	*92.758*	92.618	91.574	88.928
Cricket								16.667	18.056	18.056	22.222				91.667*	88.889*	90.278*	*94.444*	*94.444*	*94.444*	**97.222**	93.056
ERing								16.667	16.667	16.667	16.667				66.296	*71.111*	69.259	67.037	63.704	67.778	**78.148**	51.481
Epilepsy								81.159	76.087	56.522	78.986				93.478	*94.928*	93.478	94.203	94.203	94.203	**97.101**	*94.928*
EthanolConcentration								34.601	33.080	25.475	36.882				*39.163*	38.403*	**39.924***	33.460	36.502	33.840	34.981	35.741
Handwriting								5.176	5.059	5.059	4.588				12.471	12.588	13.294	13.059	*15.176*	14.000	**24.706**	12.706
ACSF1								27.000	10.000	10.000	23.000				73.000	85.000	86.000	86.000	*87.000*	**88.000**	**88.000**	81.000
Adiac								3.325	1.790	2.302	2.813				34.271	43.223	45.013	43.223	47.570	*48.849*	**50.384**	19.182
AllGestureWiimoteX								32.571	35.286	33.714	37.857				28.857	28.571	29.857	35.571	38.429	*39.286*	**40.286**	37.571
AllGestureWiimoteY								38.714	39.143	34.143	37.857				34.857	35.286	34.571	40.000	*40.714*	40.143	**41.429**	39.571
AllGestureWiimoteZ								*35.143*	31.714	27.571	32.286				25.857	28.429	27.857	31.286	33.714	*35.143*	**35.857**	32.857
ArrowHead								29.714	39.429	30.286	30.286				57.714	57.143	57.714	*66.286*	65.714	**67.429**	65.143	57.143
BME								39.333	38.667	42.000	45.333				45.333	44.667	46.667	45.333	48.667	**52.667**	*50.667*	48.000
Beef								23.333	20.000	20.000	16.667				*60.000*	53.333	**63.333**	53.333	53.333	*60.000*	53.333	*60.000*
BeetleFly								50.000	50.000	50.000	50.000				80.000	80.000	80.000	*90.000*	**95.000**	**95.000**	85.000	*90.000*
BirdChicken								50.000	50.000	50.000	50.000				**90.000**	**90.000**	*80.000*	**90.000**	**90.000**	**90.000**	**90.000**	75.000
CBF								33.111	33.111	33.111	33.111				60.111	60.667	60.444	60.333	60.778	*63.000*	**72.111**	62.222
Car								25.000	23.333	23.333	30.000				58.333	*66.667*	65.000	65.000	**71.667**	**71.667**	*66.667*	60.000
Chinatown								53.936	32.945	27.405	50.437				92.711	*93.878*	93.294	93.003	**95.627**	*93.878*	91.837	83.673
ChlorineConcentration								53.229	53.255	53.255					57.891	60.156	*60.417*	57.682	58.385	58.333	**64.193**	**
CinCECGTorso								25.942	24.783	24.855	24.493				58.551	58.333	57.101	*61.812*	58.478	57.391	**64.855**	54.638
Coffee								53.571	53.571	53.571	53.571				89.286	89.286	*92.857*	*92.857*	85.714	*92.857*	**96.429**	85.714
Computers								50.000	50.000	50.000	50.000				68.000	71.600	73.200	76.000	*81.600*	**82.000**	74.400	65.600
CricketX								8.205	6.923	7.692	6.667				25.385	30.513	31.538	24.615	33.590	*40.256*	**44.359**	25.385
CricketY								8.718	8.462	8.462	9.231				31.026	35.385	36.667	29.744	34.872	*39.231*	**45.128**	25.897
CricketZ								7.692	8.462	6.154	8.718				23.846	27.949	32.564	24.615	35.641	*40.256*	**43.590**	23.333
DiatomSizeReduction								31.699	30.065	30.065	31.699				73.529	71.569	72.549	72.876	75.490	*78.105*	**85.294**	73.529
DistalPhalanxOutlineAgeGroup								46.763	46.763	46.763	47.482				61.151	71.223	**74.820**	71.223	71.942	72.662	*74.101*	68.345
DistalPhalanxOutlineCorrect								58.333	58.333	58.333	58.333				67.029	75.362	*76.087*	73.551	72.826	74.638	**78.261**	72.464
DistalPhalanxTW								30.216	30.216	30.216	30.935				63.309	*64.748*	**65.468**	62.590	**65.468**	*64.748*	*64.748*	58.273
ECG200								64.000	64.000	64.000	64.000				69.000	69.000	*70.000*	*70.000*	*70.000*	69.000	**72.000**	67.000

6.2 ACCURACY RESULTS FOR ALL DATASETS WITH RANDOM FORREST CLASSIFIER

Table 11 – Results of Accuracy with random forest classifier for all dataset.

Dataset	Dodge	Xiao	Zheng	Movelets	RF	XGBoost	DeepeST	NPOI				MARC	MASTERMovelets		HiPerMovelets			HiPerPivots			Random-	Ultra-
								1	2	3	1+2+3		Log	Log+Pivots	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	Movelets	Movelets
Animals	81.361	82.065	81.013	80.561									**89.228**	85.323	85.318	86.271	86.271	86.371	*87.223*	87.175	85.371	85.419
GoTrack	**	**	*82.285*	78.497									79.232	79.857*	79.250	81.052	**82.853**	77.981	78.606	77.981	80.978	79.250
Vehicles	90.539	96.308	88.180	95.754									98.147	98.154	*98.245*	98.147	97.628	97.887	98.147	98.147	**98.414**	98.147
Gowalla													93.284	92.445	93.075	**93.704**	*93.494*	**93.704**	**93.704**	93.389	93.284	93.284
Brightkite													*96.732*	95.916	95.779	96.188	96.392	95.779	95.984	96.120	**97.141**	96.256
FoursquareNYC (generic)													**70.732**	58.106	57.963	64.706	65.854	65.136	65.997	66.284	*68.436*	64.706
FoursquareNYC													*97.561*	95.839	95.696	94.692	94.692	96.413	94.405	94.548	**98.135**	96.557
Promoters													77.273	*81.818*	50.000	50.000	50.000	77.273	77.273	77.273	**86.364**	77.273
SJGS													**66.980**	52.269	54.147	62.285	62.285	61.346	*65.728*	64.789	50.704	**66.980**
ArticularyWordRecognition															91.667	**96.000**	95.333	91.667	95.000	*95.667*	95.333	93.333
AtrialFibrillation															*40.000*	**46.667**	**46.667**	*40.000*	**46.667**	*40.000*	20.000	6.667
BasicMotions															**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**
CharacterTrajectories															77.437	81.476	**83.983**	78.621	81.476	*83.565*	82.312	82.591
Cricket															*94.444**	91.667*	91.667*	93.056	91.667	90.278	**97.222**	93.056
ERing															60.000	60.741	**63.704**	50.000	53.333	53.333	*63.333*	47.778
Epilepsy															94.928	93.478	94.928	*95.652*	94.203	94.203	**96.377**	90.580
EthanolConcentration															40.304	39.924*	39.163*	39.163	*41.825*	*41.825*	**42.205**	41.445
Handwriting															13.529	13.176	*14.471*	11.529	13.529	13.059	**25.412**	13.294
ACSF1															76.000	86.000	**88.000**	*87.000*	85.000	**88.000**	86.000	83.000
Adiac															29.668	37.340	36.061	35.038	*40.921*	40.153	**45.013**	18.670
AllGestureWiimoteX															32.571	32.286	33.571	**44.286**	42.000	*43.714*	42.857	40.857
AllGestureWiimoteY															41.143	41.857	42.143	44.143	**47.571**	*45.286*	45.143	44.286
AllGestureWiimoteZ															33.286	35.286	34.571	36.429	*37.429*	36.857	**39.571**	37.286
ArrowHead															62.857	60.571	60.571	62.286	**65.714**	*65.143*	64.000	56.571
BME															48.000	48.667	*54.000*	48.000	48.000	**57.333**	50.667	52.667
Beef															53.333	53.333	53.333	*56.667*	**66.667**	*56.667*	53.333	50.000
BeetleFly															80.000	*90.000*	*90.000*	85.000	80.000	**95.000**	85.000	*90.000*
BirdChicken															**90.000**	*85.000*	*85.000*	**90.000**	*85.000*	80.000	**90.000**	**90.000**
CBF															57.556	60.556	59.111	60.000	60.222	*65.667*	**84.667**	65.111
Car															60.000	**66.667**	**66.667**	61.667	63.333	*65.000*	**66.667**	56.667
Chinatown															*96.501*	*96.501*	95.335	95.627	96.210	**96.793**	94.169	93.586
ChlorineConcentration															57.604	60.312	*60.443*	56.276	57.109	57.786	**62.396**	**
CinCECGTorso															55.217	55.217	55.217	57.029	57.754	*57.899*	55.507	**60.870**
Coffee															*96.429*	*96.429*	*96.429*	92.857	92.857	**100.000**	**100.000**	*96.429*
Computers															73.200	**80.000**	78.800	76.000	77.600	*79.600*	75.600	70.400
CricketX															20.256	27.436	30.513	24.872	35.641	*36.667*	**40.513**	31.026
CricketY															25.128	31.538	36.923	25.128	35.385	*40.256*	**42.821**	21.538
CricketZ															21.538	25.641	29.231	23.846	31.282	*38.205*	**40.513**	22.564
DiatomSizeReduction															73.856	73.203	73.203	83.333	78.105	87.582	**89.216**	*88.562*
DistalPhalanxOutlineAgeGroup															*71.942*	69.065	71.223	*71.942*	**73.381**	**73.381**	**73.381**	71.223
DistalPhalanxOutlineCorrect															67.391	*77.174*	**78.623**	71.014	71.377	73.188	75.725	72.826
DistalPhalanxTW															64.748	65.468	65.468	64.748	*66.906*	66.187	**67.626**	62.590
ECG200															71.000	*75.000*	*75.000*	73.000	73.000	72.000	**79.000**	72.000

6.3 TOTAL RUNNING TIME RESULTS FOR ALL DATASETS WITH NEURAL NETWORK CLASSIFIER

Table 12 – Results of total running time (with neural network classifier) for all datasets.

Dataset	Dodge	Xiao	Zheng	Movelets	RF	XGBoost	DeepeST	NPOI				MARC	MASTERMovelets		HiPerMovelets			HiPerPivots			Random-Movelets	Ultra-Movelets
								1	2	3	1+2+3		Log	Log+Pivots	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$		
Animals	23s	25s	**23s**	53s	1h05m24s	2h53m32s	1h16m03s	**12s**	**12s**	**12s**	**12s**	55s	1m42s	47s	5m53s	6m13s	7m14s	2m	1m57s	1m54s	53s	44s
GoTrack	**	**	**9s**	1m02s	1h06m04s	2h48m31s	2h57m50s	**10s**	**10s**	**10s**	**10s**	56s	2m12s	48s*	12m12s	12m22s	12m28s	4m30s	4m28s	4m36s	1m37s	1m19s
Vehicles	13s	13s	**12s**	2h22m56s	1h10m54s	3h54m43s	10h42m32s	**4s**	**4s**	**4s**	**4s**	2m01s	3h08m06s	56m27s	4h23m19s	5h10m59s	5h21m57s	1h05m	1h13m34s	1h17m01s	2h08m17s	1h13m12s
Gowalla					1h12m15s	4h55m57s	23h21m38s	1h24m13s	1h30m20s	1h39m05s	2h29m26s	**13m16s**	11h45m35s	3h35m50s	1h20m12s	2h54m43s	6h11m52s	*1h06m*	1h41m54s	2h46m39s	2h37m17s	2h24m28s
Brightkite					1h17m37s	5h57m15s	21h49m51s	*28m53s*	30m37s	32m39s	38m48s	**9m22s**	19h58m38s	6h04m58s	2h44m21s	6h41m05s	12h09m33s	2h44m29s	4h16m49s	5h53m47s	4h33m02s	3h47m40s
FoursquareNYC (generic)					1h38m50s	69h12m41s	42h04m30s	2m20s	1m50s	*1m18s*	**46s**	11m57s	4h35m25s	1h23m55s	58m39s	3h07m08s	4h11m39s	33m41s	1h01m23s	1h08m17s	2h25m35s	48m37s
FoursquareNYC					1h30m40s	65h14m41s	20h43m53s	*20m33s*	23m16s	24m33s	36m35s	**14m11s**	28h09m36s	4h56m30s	3h38m38s	10h54m08s	14h03m22s	2h34m34s	7h07m44s	4h54m34s	13h20m49s	1h01m07s
Promoters					1h04m32s	50m47s	13m12s	**0.217s**	**0.217s**	**0.217s**	**0.217s**	20s	38s	29s	11s	11s	*11s*	16s	16s	17s	34s	23s
SJGS					1h24m53s	59m10s	3h05m51s	**12s**	15s	*15s*	21s	2m40s	4h12m56s	1h17m27s	1h44m43s	2h10m48s	2h48m38s	1h07s	1h02m16s	1h21m18s	1h44m31s	2h58m40s
ArticularyWordRecognition								**7m34s**	*7m50s*	7m55s	10m39s				22m19s	1h12m10s	1h12m33s	22m58s	1h06m31s	1h17m07s	19m55s	14m44s
AtrialFibrillation								**10s**	12s	12s	*12s*				35s	38s	40s	37s	36s	36s	1m	44s
BasicMotions								10s	**8s**	**8s**	**11s**				3m45s	6m52s	7m08s	3m47s	7m19s	6m59s	1m59s	28s
CharacterTrajectories								3h15m	3h16m41s	3h14m39s	8h38m39s				3h29m41s	5h18m14s	5h10m22s	3h30m53s	5h18m59s	6h04m30s	**1h28m30s**	*2h28m54s*
Cricket								**17m08s**	*17m33s*	17m44s	20m18s				168h15s*	168h14s*	168h15s*	25h37m52s	27h23m02s	27h09m33s	65h40m42s	3h23m32s
ERing								**8s**	**8s**	**8s**	**8s**				23s	23s	26s	16s	**12s**	**12s**	21s	15s
Epilepsy								**1m33s**	*1m43s*	**1m43s**	2m03s				4m27s	1m47s	1m45s	9m07s	7m22s	7m48s	11m02s	6m23s
EthanolConcentration								**6h47m59s**	*7h51m02s*	7h53m52s	9h19m31s				148h34m55s	168h16s*	168h16s*	22h04m02s	24h15m18s	26h12m55s	76h46m08s	31h01m55s
Handwriting								3m59s	3m59s	3m59s	3m59s				2m30s	2m36s	2m34s	**36s**	*37s*	**37s**	1m55s	2m08s
ACSF1								*33m41s*	33m59s	35m02s	41m40s				1h51s	1h41m36s	1h56m33s	57m56s	1h06m58s	1h08m24s	40m01s	**29m23s**
Adiac								26m26s	25m29s	25m28s	39m58s				5m50s	11m50s	12m20s	5m14s	8m58s	9m56s	*4m02s*	**3m25s**
AllGestureWiimoteX								56s	59s	1m27s	1m14s				**27s**	*28s*	29s	56s	59s	1m02s	3m09s	11m53s
AllGestureWiimoteY								**44s**	46s	52s	57s				**23s**	**23s**	**23s**	1m	1m06s	1m07s	3m09s	12m13s
AllGestureWiimoteZ								47s	49s	57s	1m01s				**22s**	**22s**	**23s**	50s	58s	59s	3m	11m42s
ArrowHead								58s	57s	57s	1m05s				38s	41s	41s	*34s*	37s	37s	**24s**	**34s**
BME								15s	15s	18s	19s				14s	14s	**13s**	**12s**	**13s**	14s	**12s**	16s
Beef								40s	40s	40s	47s				57s	57s	57s	47s	50s	50s	*29s*	**23s**
BeetleFly								**21s**	**21s**	**21s**	25s				1m04s	1m05s	1m04s	54s	57s	56s	27s	**22s**
BirdChicken								**22s**	**23s**	**22s**	25s				1m	1m01s	59s	51s	52s	53s	25s	**22s**
CBF								1m15s	1m16s	1m15s	1m24s				30s	30s	31s	*26s*	27s	27s	**24s**	43s
Car								3m43s	3m43s	3m43s	4m35s				6m	6m18s	6m18s	4m04s	4m13s	4m17s	*2m04s*	**1m38s**
Chinatown								**7s**	**6s**	**6s**	**7s**				8s	8s	8s	8s	8s	8s	8s	8s
ChlorineConcentration								2h57m43s	2h50m34s	2h50m42s					1h47m54s	2h32m50s	2h32m43s	*1h40m59s*	1h44m29s	1h50m38s	**56m20s**	168h*
CinCECGTorso								3h52m23s	3h50m24s	3h51m59s	4h20m30s				1h36m39s	1h35m24s	1h37m52s	1h22m46s	*1h22m35s*	1h22m53s	**1h09m29s**	2h11m52s
Coffee								**18s**	**18s**	*18s*	20s				36s	40s	40s	29s	34s	35s	18s	**17s**
Computers								**2m33s**	*2m34s*	2m37s	2m42s				6h57m03s	7h31m45s	7h55m52s	1h03m17s	1h15m50s	1h21m37s	1h15m28s	1h06m25s
CricketX								1h01m13s	1h37s	1h56s	2h15m37s				1h20m57s	1h32m02s	1h32m57s	46m48s	49m52s	53m02s	*17m02s*	**16m05s**
CricketY								1h02m16s	59m53s	1h45s	2h13m47s				1h16m40s	1h25m51s	1h26m10s	44m53s	47m33s	48m46s	*17m11s*	**16m14s**
CricketZ								1h18s	1h01m17s	59m46s	2h24m33s				1h22m32s	1h34m49s	1h33m52s	48m20s	50m55s	51m51s	*16m55s*	**16m29s**
DiatomSizeReduction								1m15s	1m12s	1m12s	1m31s				26s	29s	28s	**18s**	20s	*19s*	22s	29s
DistalPhalanxOutlineAgeGroup								3m53s	3m49s	3m49s	5m41s				6m21s	12m06s	13m41s	*3m18s*	4m34s	4m40s	**2m56s**	7m13s
DistalPhalanxOutlineCorrect								10m20s	10m21s	10m21s	17m50s				12m02s	24m21s	28m30s	*7m52s*	11m19s	11m47s	**7m44s**	23m46s
DistalPhalanxTW								3m52s	3m52s	3m53s	5m44s				6m25s	10m20s	9m56s	**2m16s**	3m23s	3m31s	*2m21s*	3m48s
ECG200								36s	*32s*	33s	44s				1m12s	1m27s	1m28s	48s	49s	49s	**25s**	38s

6.4 NUMBER OF MOVELET CANDIDATES FOR ALL DATASETS

Table 13 – Results of movelet candidates for all datasets.

Dataset	MASTERMovelets		HiPerMovelets			HiPerPivots			Random-	Ultra-
	Log	Log+Pivots	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	Movelets	Movelets
Animals	232,896	**10,921**	220,000	221,131	222,093	*41,176*	42,170	42,695	232,896	47,782
GoTrack	286,447	16,908*	241,105	238,329	236,647	**50,996**	*52,443*	53,718	286,447	67,602
Vehicles	3,294,900	339,485	746,435	805,937	818,187	**117,206**	*125,472*	129,373	3,294,900	637,082
Gowalla	4,934,955	388,395	2,576,295	2,628,435	2,645,265	**313,378**	*434,108*	652,387	4,934,955	740,301
Brightkite	6,254,925	**484,650**	3,053,010	3,034,425	3,054,150	*594,766*	868,477	1,172,644	6,254,925	948,650
FoursquareNYC (generic)	14,033,061	1,067,976	5,188,680	5,699,547	5,699,547	**685,687**	931,331	963,701	14,033,061	*766,741*
FoursquareNYC	56,800,485	4,201,635	17,548,080	18,500,760	18,572,925	*1,495,363*	2,814,986	3,027,446	56,800,485	**925,373**
Promoters	82,404	*5,292*	**1,008**	**1,008**	**1,008**	7,278	7,877	12,031	82,404	23,790
SJGS	2,640,285	**141,354**	1,354,581	1,354,581	1,354,581	*241,188*	256,833	392,235	2,640,285	723,457
ArticularyWordRecognition			*1,634,325*	*1,634,325*	*1,634,325*	*1,634,325*	*1,634,325*	*1,634,325*	*1,634,325*	**179,539**
AtrialFibrillation			*61,104*	*61,104*	*61,104*	*61,104*	*61,104*	*61,104*	229,140	**36,709**
BasicMotions			*1,474,200*	*1,474,200*	*1,474,200*	*1,474,200*	*1,474,200*	*1,474,200*	*1,474,200*	**48,377**
CharacterTrajectories			*1,019,744*	*1,019,744*	*1,019,744*	*1,019,744*	*1,019,744*	*1,019,744*	*1,019,744*	**261,687**
Cricket			53,438,049*	50,732,325*	52,085,187*	*3,274,586*	3,295,615	3,295,617	73,054,548	**1,157,437**
ERing			168,750	168,750	168,750	**9,463**	10,059	10,059	168,750	*12,008*
Epilepsy			218,834	109,417	119,364	34,886	**30,488**	*30,496*	1,362,739	139,626
EthanolConcentration			9,465,246	9,135,063*	9,135,063*	**614,998**	*700,718*	706,457	28,725,921	2,393,156
Handwriting			156,450	156,450	156,450	**17,847**	*18,366*	18,495	156,450	24,882
ACSF1			1,310,400	1,310,400	1,310,400	*316,728*	392,041	413,926	1,310,400	**200,169**
Adiac			472,290	472,290	472,290	*89,589*	157,833	172,414	472,290	**72,437**
AllGestureWiimoteX			*19,232*	**18,533**	**18,533**	20,865	21,363	23,008	236,508	90,548
AllGestureWiimoteY			**14,443**	**14,443**	**14,443**	*22,739*	24,308	25,058	245,148	105,648
AllGestureWiimoteZ			*13,220*	**12,617**	**12,617**	17,312	20,149	21,286	238,482	92,652
ArrowHead			62,496	62,496	62,496	*20,028*	22,893	23,450	62,496	**13,883**
BME			22,590	22,590	22,590	**5,902**	*6,289*	8,214	22,590	6,734
Beef			111,960	111,960	111,960	*26,772*	28,404	28,859	111,960	**18,515**
BeetleFly			81,360	81,360	81,360	*24,576*	26,215	26,548	81,360	**18,427**
BirdChicken			81,360	81,360	81,360	*25,226*	26,269	26,572	81,360	**19,435**
CBF			22,590	22,590	22,590	*6,924*	8,777	9,089	22,590	**6,218**
Car			275,280	275,280	275,280	*90,674*	96,181	97,568	275,280	**48,510**
Chinatown			2,200	2,200	2,200	**514**	*802*	1,056	2,200	903
ChlorineConcentration			532,847	532,847	532,847	**203,997**	*236,862*	242,016	532,847	102,653*
CinCECGTorso			588,600	588,600	588,600	*170,649*	173,046	173,903	588,600	**95,159**
Coffee			55,468	55,468	55,468	*15,550*	20,109	21,271	55,468	**14,594**
Computers			1,421,536	1,421,536	1,421,536	*493,532*	565,726	587,209	1,433,000	**343,067**
CricketX			810,810	810,810	810,810	*313,715*	338,202	345,226	810,810	**133,234**
CricketY			810,810	810,810	810,810	*316,296*	340,353	345,711	810,810	**133,839**
CricketZ			810,810	810,810	810,810	*315,239*	338,878	345,359	810,810	**133,252**
DiatomSizeReduction			38,304	38,304	38,304	**3,690**	*4,925*	5,107	38,304	8,188
DistalPhalanxOutlineAgeGroup			186,000	186,000	186,000	**50,354**	*65,534*	66,854	186,000	67,100
DistalPhalanxOutlineCorrect			279,000	279,000	279,000	**70,384**	98,224	102,661	279,000	*95,333*
DistalPhalanxTW			186,000	186,000	186,000	**44,762**	62,574	64,162	186,000	*59,149*
ECG200			56,100	56,100	56,100	**17,029**	*17,838*	18,163	56,100	18,492

6.5 NUMBER OF MOVELETS FOR ALL DATASETS

Table 14 – Results of number of movelets for all datasets.

Dataset	MASTERMovelets		HiPerMovelets			HiPerPivots			Random-	Ultra-
	Log	Log+Pivots	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	$\tau = 90\%$	$\tau = 75\%$	$\tau = 50\%$	Movelets	Movelets
Animals	232,896	**10,921**	220,000	221,131	222,093	*41,176*	42,170	42,695	23,326	47,782
GoTrack	286,447	16,908*	241,105	238,329	236,647	**50,996**	*52,443*	53,718	28,718	67,602
Vehicles	3,294,900	339,485	746,435	805,937	818,187	**117,206**	*125,472*	129,373	329,624	637,082
Gowalla	4,934,955	388,395	2,576,295	2,628,435	2,645,265	**313,378**	*434,108*	652,387	493,848	740,301
Brightkite	6,254,925	**484,650**	3,053,010	3,034,425	3,054,150	*594,766*	868,477	1,172,644	625,803	948,650
FoursquareNYC (generic)	14,033,061	1,067,976	5,188,680	5,699,547	5,699,547	**685,687**	931,331	963,701	1,404,239	*766,741*
FoursquareNYC	56,800,485	4,201,635	17,548,080	18,500,760	18,572,925	*1,495,363*	2,814,986	3,027,446	5,680,294	**925,373**
Promoters	82,404	*5,292*	**1,008**	**1,008**	**1,008**	7,278	7,877	12,031	8,316	23,790
SJGS	2,640,285	**141,354**	1,354,581	1,354,581	1,354,581	*241,188*	256,833	392,235	265,304	723,457
ArticularyWordRecognition			1,634,325	1,634,325	1,634,325	1,634,325	1,634,325	1,634,325	**163,625**	*179,539*
AtrialFibrillation			61,104	61,104	61,104	61,104	61,104	61,104	**22,920**	*36,709*
BasicMotions			1,474,200	1,474,200	1,474,200	1,474,200	1,474,200	1,474,200	*147,440*	**48,377**
CharacterTrajectories			1,019,744	1,019,744	1,019,744	1,019,744	1,019,744	1,019,744	**102,699**	*261,687*
Cricket			53,438,049*	50,732,325*	52,085,187*	*3,274,586*	3,295,615	3,295,617	7,305,552	**1,157,437**
ERing			168,750	168,750	168,750	**9,463**	*10,059*	*10,059*	16,890	12,008
Epilepsy			218,834	109,417	119,364	34,886	**30,488**	*30,496*	136,315	139,626
EthanolConcentration			9,465,246	9,135,063*	9,135,063*	**614,998**	*700,718*	706,457	2,872,827	2,393,156
Handwriting			156,450	156,450	156,450	*17,847*	18,366	18,495	**17,010**	24,882
ACSF1			1,310,400	1,310,400	1,310,400	316,728	392,041	413,926	**135,033**	*200,169*
Adiac			472,290	472,290	472,290	89,589	157,833	172,414	**47,702**	*72,437*
AllGestureWiimoteX			*19,232*	**18,533**	**18,533**	20,865	21,363	23,008	23,797	90,548
AllGestureWiimoteY			**14,443**	**14,443**	**14,443**	*22,739*	24,308	25,058	24,652	105,648
AllGestureWiimoteZ			*13,220*	**12,617**	**12,617**	17,312	20,149	21,286	23,986	92,652
ArrowHead			62,496	62,496	62,496	20,028	22,893	23,450	**6,264**	*13,883*
BME			22,590	22,590	22,590	*5,902*	6,289	8,214	**2,280**	6,734
Beef			111,960	111,960	111,960	26,772	28,404	28,859	**11,220**	*18,515*
BeetleFly			81,360	81,360	81,360	24,576	26,215	26,548	**8,140**	*18,427*
BirdChicken			81,360	81,360	81,360	25,226	26,269	26,572	**8,140**	*19,435*
CBF			22,590	22,590	22,590	6,924	8,777	9,089	**2,280**	*6,218*
Car			275,280	275,280	275,280	90,674	96,181	97,568	**27,540**	*48,510*
Chinatown			2,200	2,200	2,200	*514*	802	1,056	**220**	903
ChlorineConcentration			532,847	532,847	532,847	*203,997*	236,862	242,016	**53,705**	102,653*
CinCECGTorso			588,600	588,600	588,600	170,649	173,046	173,903	**58,880**	*95,159*
Coffee			55,468	55,468	55,468	15,550	20,109	21,271	**5,572**	*14,594*
Computers			1,421,536	1,421,536	1,421,536	493,532	565,726	587,209	**143,500**	*343,067*
CricketX			810,810	810,810	810,810	313,715	338,202	345,226	**81,120**	*133,234*
CricketY			810,810	810,810	810,810	316,296	340,353	345,711	**81,120**	*133,839*
CricketZ			810,810	810,810	810,810	315,239	338,878	345,359	**81,120**	*133,252*
DiatomSizeReduction			38,304	38,304	38,304	**3,690**	4,925	5,107	*4,560*	8,188
DistalPhalanxOutlineAgeGroup			186,000	186,000	186,000	*50,354*	65,534	66,854	**18,800**	67,100
DistalPhalanxOutlineCorrect			279,000	279,000	279,000	*70,384*	98,224	102,661	**28,200**	95,333
DistalPhalanxTW			186,000	186,000	186,000	*44,762*	62,574	64,162	**18,800**	59,149
ECG200			56,100	56,100	56,100	*17,029*	17,838	18,163	**5,700**	18,492

7 MOVELETS ANALYSIS

In this appendix, we present the statistics of the resulting movelets obtained with the methods: MASTERMovelets, HiPerMovelets, RandomMovelets, and UltraMovelets. First, we present the number of movelets, the average number of features, size, and quality of generated movelets by dataset. Then, we present the attribute confidence by method in multiple aspect trajectory datasets.

7.1 MOVELETS STATISTICS BY DATASET

Table 15 – Movelets statistics on all datasets.

Dataset	Method	Movelets	Quality			Size			Number of Attributes		
Animals	MASTERMovelets-Log	5,367	45.447	11.765	92.593	1.8	1.0	7.0	1.9	1.0	3.0
	MASTERPivots-Log	887	52.738	12.121	93.103	2.2	2.0	7.0	2.0	1.0	3.0
	HiPerMovelets $\tau = 90\%$	4,427	42.758	10.000	92.000	2.0	1.0	7.0	1.0	1.0	3.0
	HiPerMovelets $\tau = 75\%$	4,843	47.039	10.811	92.857	1.9	1.0	7.0	1.6	1.0	3.0
	HiPerMovelets $\tau = 50\%$	5,084	49.639	10.811	93.333	1.9	1.0	7.0	1.8	1.0	3.0
	HiPerPivots $\tau = 90\%$	4,351	38.973	10.811	90.909	1.6	1.0	7.0	1.0	1.0	3.0
	HiPerPivots $\tau = 75\%$	5,582	41.928	10.811	91.803	1.4	1.0	7.0	1.7	1.0	3.0
	HiPerPivots $\tau = 50\%$	5,980	43.507	10.811	91.803	1.3	1.0	7.0	1.8	1.0	3.0
	RandomMovelets	3,156	46.673	10.811	92.000	2.7	1.0	7.0	2.0	1.0	3.0
	UltraMovelets	10,218	49.205	10.811	93.103	1.4	1.0	8.0	1.7	1.0	3.0
GoTrack	MASTERMovelets-Log	6,243	23.576	5.195	66.055	1.9	1.0	8.0	2.0	1.0	3.0
	MASTERPivots-Log	949	26.312	5.263	65.455	2.4	2.0	15.0	2.1	1.0	3.0
	HiPerMovelets $\tau = 90\%$	5,780	26.977	5.263	66.667	2.9	1.0	8.0	1.9	1.0	3.0
	HiPerMovelets $\tau = 75\%$	5,565	28.928	5.263	66.667	2.4	1.0	8.0	2.1	1.0	3.0
	HiPerMovelets $\tau = 50\%$	5,364	28.957	5.263	66.667	2.2	1.0	8.0	2.0	1.0	3.0
	HiPerPivots $\tau = 90\%$	7,809	24.086	5.263	66.055	1.4	1.0	8.0	2.0	1.0	3.0
	HiPerPivots $\tau = 75\%$	8,209	24.442	5.263	66.055	1.3	1.0	8.0	2.3	1.0	3.0
	HiPerPivots $\tau = 50\%$	8,256	24.772	5.263	66.055	1.3	1.0	8.0	2.2	1.0	3.0
	RandomMovelets	4,057	26.026	5.000	66.055	2.6	1.0	8.0	2.0	1.0	3.0
	UltraMovelets	13,426	27.341	5.063	65.455	1.4	1.0	7.0	1.8	1.0	3.0
Vehicles	MASTERMovelets-Log	98,283	38.983	1.786	96.386	1.5	1.0	8.0	1.5	1.0	3.0
	MASTERPivots-Log	7,717	60.677	2.715	96.386	2.2	2.0	12.0	1.9	1.0	3.0
	HiPerMovelets $\tau = 90\%$	19,814	37.397	1.754	96.386	1.3	1.0	8.0	1.3	1.0	3.0
	HiPerMovelets $\tau = 75\%$	23,753	37.509	1.778	96.386	1.4	1.0	8.0	1.5	1.0	3.0
	HiPerMovelets $\tau = 50\%$	25,292	38.217	1.770	96.386	1.5	1.0	8.0	1.5	1.0	3.0
	HiPerPivots $\tau = 90\%$	21,223	34.449	1.770	96.386	1.1	1.0	8.0	1.4	1.0	3.0
	HiPerPivots $\tau = 75\%$	24,382	33.671	1.778	96.386	1.1	1.0	8.0	1.6	1.0	3.0
	HiPerPivots $\tau = 50\%$	26,194	34.475	1.778	96.386	1.1	1.0	8.0	1.6	1.0	3.0
	RandomMovelets	45,517	34.464	1.778	96.386	2.5	1.0	8.0	1.8	1.0	3.0
	UltraMovelets	141,440	39.839	1.786	96.386	1.4	1.0	10.0	1.5	1.0	3.0

Table 16 – Movelets statistics on all datasets (continued).

Dataset	Method	Movelets	Quality			Size			Number of Attributes		
			Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.
Gowalla	MASTERMovelets-Log	61,886	66.311	0.236	100.000	1.3	1.0	5.0	2.5	1.0	5.0
	MASTERPivots-Log	8,407	82.559	0.466	100.000	2.1	2.0	5.0	1.9	1.0	5.0
	HiPerMovelets $\tau = 90\%$	32,048	66.583	0.365	100.000	1.2	1.0	5.0	2.4	1.0	5.0
	HiPerMovelets $\tau = 75\%$	33,790	66.244	0.297	100.000	1.2	1.0	5.0	2.5	1.0	5.0
	HiPerMovelets $\tau = 50\%$	33,030	66.678	0.365	100.000	1.2	1.0	5.0	2.5	1.0	5.0
	HiPerPivots $\tau = 90\%$	46,212	63.156	0.365	100.000	1.2	1.0	5.0	2.7	1.0	5.0
	HiPerPivots $\tau = 75\%$	49,984	62.184	0.226	100.000	1.2	1.0	5.0	2.7	1.0	5.0
	HiPerPivots $\tau = 50\%$	50,778	62.360	0.226	100.000	1.2	1.0	5.0	2.7	1.0	5.0
	RandomMovelets	44,159	61.102	0.218	100.000	1.6	1.0	5.0	2.7	1.0	5.0
	UltraMovelets	80,780	65.799	0.244	100.000	1.6	1.0	22.0	1.9	1.0	5.0
Brightkite	MASTERMovelets-Log	90,209	82.694	0.248	100.000	1.2	1.0	5.0	2.3	1.0	5.0
	MASTERPivots-Log	10,824	88.052	0.349	100.000	2.1	2.0	5.0	2.0	1.0	5.0
	HiPerMovelets $\tau = 90\%$	42,646	80.784	0.248	100.000	1.2	1.0	5.0	2.3	1.0	5.0
	HiPerMovelets $\tau = 75\%$	41,668	81.057	0.248	100.000	1.2	1.0	5.0	2.3	1.0	5.0
	HiPerMovelets $\tau = 50\%$	41,747	81.457	0.248	100.000	1.2	1.0	5.0	2.3	1.0	5.0
	HiPerPivots $\tau = 90\%$	80,113	73.995	0.248	100.000	1.2	1.0	5.0	2.9	1.0	5.0
	HiPerPivots $\tau = 75\%$	82,644	74.820	0.218	100.000	1.1	1.0	5.0	2.9	1.0	5.0
	HiPerPivots $\tau = 50\%$	82,708	75.669	0.171	100.000	1.1	1.0	5.0	2.9	1.0	5.0
	RandomMovelets	66,300	74.882	0.065	100.000	1.5	1.0	5.0	2.7	1.0	5.0
	UltraMovelets	106,213	83.676	0.229	100.000	1.5	1.0	25.0	2.0	1.0	5.0
Foursquare NYC (generic)	MASTERMovelets-Log	22,092	23.514	0.370	100.000	2.1	1.0	6.0	3.4	1.0	6.0
	MASTERPivots-Log	5,329	31.168	0.457	100.000	2.3	2.0	5.0	3.2	1.0	6.0
	HiPerMovelets $\tau = 90\%$	13,466	6.473	0.200	100.000	1.4	1.0	6.0	1.8	1.0	6.0
	HiPerMovelets $\tau = 75\%$	15,397	8.093	0.200	100.000	1.4	1.0	6.0	1.7	0.0	6.0
	HiPerMovelets $\tau = 50\%$	14,541	8.551	0.200	100.000	1.4	1.0	6.0	1.6	0.0	6.0
	HiPerPivots $\tau = 90\%$	23,541	7.003	0.220	100.000	1.0	1.0	6.0	2.4	1.0	6.0
	HiPerPivots $\tau = 75\%$	23,083	7.630	0.220	100.000	1.0	1.0	6.0	1.9	0.0	6.0
	HiPerPivots $\tau = 50\%$	23,086	7.661	0.220	100.000	1.0	1.0	6.0	1.9	0.0	6.0
	RandomMovelets	26,123	12.985	0.200	100.000	1.8	1.0	6.0	2.7	1.0	6.0
	UltraMovelets	51,130	12.355	0.576	100.000	2.0	1.0	15.0	1.5	1.0	5.0
Foursquare NYC (specific)	MASTERMovelets-Log	32,485	60.794	0.306	100.000	1.5	1.0	6.0	2.0	1.0	7.0
	MASTERPivots-Log	5,633	89.749	0.835	100.000	2.0	2.0	4.0	1.5	1.0	6.0
	HiPerMovelets $\tau = 90\%$	13,960	62.735	0.200	100.000	1.2	1.0	6.0	2.6	1.0	8.0
	HiPerMovelets $\tau = 75\%$	14,806	64.810	0.220	100.000	1.2	1.0	6.0	2.8	1.0	8.0
	HiPerMovelets $\tau = 50\%$	14,622	65.157	0.220	100.000	1.2	1.0	6.0	2.8	1.0	8.0
	HiPerPivots $\tau = 90\%$	21,355	55.734	0.200	100.000	1.0	1.0	3.0	3.2	1.0	7.0
	HiPerPivots $\tau = 75\%$	21,416	57.197	0.200	100.000	1.0	1.0	3.0	3.3	1.0	8.0
	HiPerPivots $\tau = 50\%$	21,516	57.281	0.200	100.000	1.0	1.0	3.0	3.3	1.0	8.0
	RandomMovelets	35,184	62.060	0.303	100.000	1.4	1.0	6.0	3.4	1.0	8.0
	UltraMovelets	51,130	62.735	0.597	100.000	1.4	1.0	15.0	1.9	1.0	6.0
Promoters	MASTERMovelets-Log	3,255	67.617	21.277	82.192	1.5	1.0	6.0	1.0	1.0	1.0
	MASTERPivots-Log	220	67.557	61.157	75.789	2.2	2.0	3.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 90\%$	104	66.667	66.667	66.667	1.0	1.0	1.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 75\%$	104	66.667	66.667	66.667	1.0	1.0	1.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 50\%$	104	66.667	66.667	66.667	1.0	1.0	1.0	1.0	1.0	1.0
	HiPerPivots $\tau = 90\%$	1,096	67.649	56.075	75.789	2.2	1.0	3.0	1.0	1.0	1.0
	HiPerPivots $\tau = 75\%$	1,283	66.113	41.758	75.789	2.3	1.0	3.0	1.0	1.0	1.0
	HiPerPivots $\tau = 50\%$	1,308	65.653	35.789	75.789	2.3	1.0	3.0	1.0	1.0	1.0
	RandomMovelets	1,151	60.443	8.511	82.192	2.2	1.0	6.0	1.0	1.0	1.0
	UltraMovelets	4,788	67.510	66.667	75.510	1.5	1.0	3.0	1.0	1.0	1.0
SJGS	MASTERMovelets-Log	126,574	49.305	0.600	68.301	1.4	1.0	5.0	1.0	1.0	1.0
	MASTERPivots-Log	11,312	48.539	34.416	67.777	2.1	2.0	4.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 90\%$	130,391	49.109	38.761	68.301	1.3	1.0	3.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 75\%$	127,476	49.294	38.761	68.301	1.4	1.0	3.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 50\%$	127,476	49.294	38.761	68.301	1.4	1.0	3.0	1.0	1.0	1.0
	HiPerPivots $\tau = 90\%$	33,061	48.617	30.894	68.301	2.1	1.0	3.0	1.0	1.0	1.0
	HiPerPivots $\tau = 75\%$	38,887	47.571	26.964	68.301	2.2	1.0	3.0	1.0	1.0	1.0
	HiPerPivots $\tau = 50\%$	39,909	47.206	19.061	68.301	2.3	1.0	3.0	1.0	1.0	1.0
	RandomMovelets	37,432	43.852	0.451	68.301	2.2	1.0	6.0	1.0	1.0	2.0
	UltraMovelets	153,056	49.483	0.600	68.301	1.5	1.0	5.0	1.0	1.0	1.0

Table 17 – Movelets statistics on all datasets (continued).

Dataset	Method	Movelets	Quality			Size			Number of Attributes		
			Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	Max.
ArticularyWordRecognition	HiPerMovelets $\tau = 50\%$	18,093	58.192	13.333	100.000	1.9	1.0	6.0	2.2	0.0	3.0
	HiPerMovelets $\tau = 75\%$	18,066	57.919	13.333	100.000	1.9	1.0	6.0	2.2	0.0	3.0
	HiPerMovelets $\tau = 90\%$	13,351	51.836	16.667	100.000	1.8	1.0	6.0	1.8	0.0	3.0
	HiPerPivots $\tau = 50\%$	18,093	58.192	13.333	100.000	1.9	1.0	6.0	2.2	0.0	3.0
	HiPerPivots $\tau = 75\%$	18,066	57.919	13.333	100.000	1.9	1.0	6.0	2.2	0.0	3.0
	HiPerPivots $\tau = 90\%$	13,351	51.836	16.667	100.000	1.8	1.0	6.0	1.8	0.0	3.0
	RandomMovelets	11,823	55.155	17.391	100.000	2.6	1.0	6.0	1.8	0.0	3.0
	UltraMovelets	18,780	50.089	17.391	100.000	1.3	1.0	7.0	1.5	1.0	3.0
AtrialFibrillation	HiPerMovelets $\tau = 50\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	HiPerMovelets $\tau = 75\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	HiPerMovelets $\tau = 90\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	HiPerPivots $\tau = 50\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	HiPerPivots $\tau = 75\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	HiPerPivots $\tau = 90\%$	1,275	52.314	28.571	100.000	1.6	1.0	8.0	1.1	1.0	2.0
	RandomMovelets	2,295	63.010	28.571	100.000	3.0	1.0	8.0	0.8	0.0	2.0
	UltraMovelets	7,380	57.626	28.571	100.000	1.4	1.0	6.0	1.2	1.0	2.0
BasicMotions	HiPerMovelets $\tau = 50\%$	2,723	91.091	26.667	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	HiPerMovelets $\tau = 75\%$	2,721	91.087	26.667	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	HiPerMovelets $\tau = 90\%$	2,562	84.869	25.000	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	HiPerPivots $\tau = 50\%$	2,723	91.091	26.667	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	HiPerPivots $\tau = 75\%$	2,721	91.087	26.667	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	HiPerPivots $\tau = 90\%$	2,562	84.869	25.000	100.000	1.5	1.0	6.0	1.8	0.0	6.0
	RandomMovelets	2,097	86.836	25.000	100.000	1.9	1.0	6.0	1.6	0.0	6.0
	UltraMovelets	3,944	80.924	16.000	100.000	1.4	1.0	7.0	1.5	1.0	4.0
CharacterTrajectories	HiPerMovelets $\tau = 50\%$	41,895	15.243	3.883	100.000	2.7	1.0	7.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 75\%$	39,255	14.988	3.883	100.000	2.6	1.0	7.0	1.0	1.0	1.0
	HiPerMovelets $\tau = 90\%$	28,002	14.551	3.883	100.000	2.5	1.0	7.0	1.0	1.0	1.0
	HiPerPivots $\tau = 50\%$	41,895	15.243	3.883	100.000	2.7	1.0	7.0	1.0	1.0	1.0
	HiPerPivots $\tau = 75\%$	39,255	14.988	3.883	100.000	2.6	1.0	7.0	1.0	1.0	1.0
	HiPerPivots $\tau = 90\%$	28,002	14.551	3.883	100.000	2.5	1.0	7.0	1.0	1.0	1.0
	RandomMovelets	20,444	15.017	3.774	99.408	3.5	1.0	7.0	1.0	1.0	1.0
	UltraMovelets	51,199	16.000	3.738	100.000	1.6	1.0	16.0	1.0	1.0	1.0
Cricket	HiPerMovelets $\tau = 50\%$	41,364	75.897	20.000	100.000	2.0	1.0	9.0	2.6	0.0	6.0
	HiPerMovelets $\tau = 75\%$	35,897	75.942	20.000	100.000	2.0	1.0	9.0	2.6	0.0	6.0
	HiPerMovelets $\tau = 90\%$	41,391	75.782	20.000	100.000	2.0	1.0	9.0	2.7	0.0	6.0
	HiPerPivots $\tau = 50\%$	103,673	69.020	18.182	100.000	1.1	1.0	9.0	2.9	0.0	6.0
	HiPerPivots $\tau = 75\%$	103,673	69.020	18.182	100.000	1.1	1.0	9.0	2.9	0.0	6.0
	HiPerPivots $\tau = 90\%$	103,616	68.990	18.182	100.000	1.1	1.0	9.0	2.9	0.0	6.0
	RandomMovelets	50,795	73.237	19.048	100.000	2.5	1.0	9.0	2.4	0.0	6.0
	UltraMovelets	71,323	49.414	16.000	100.000	1.3	1.0	8.0	1.5	1.0	5.0

7.2 MOVELET ATTRIBUTE CONFIDENCE BY METHOD IN MULTIPLE ASPECT TRAJECTORIES

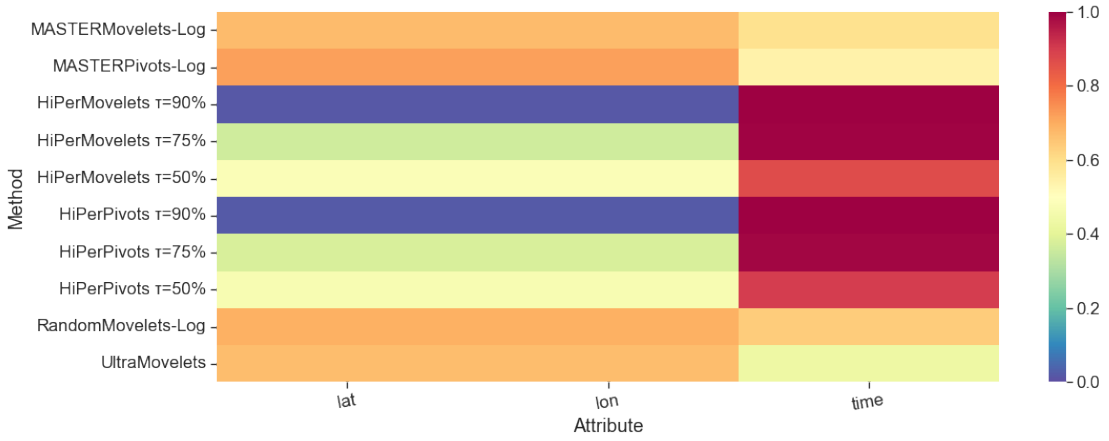


Figure 48 – Heat map for movelet attribute use by class on Animals dataset.

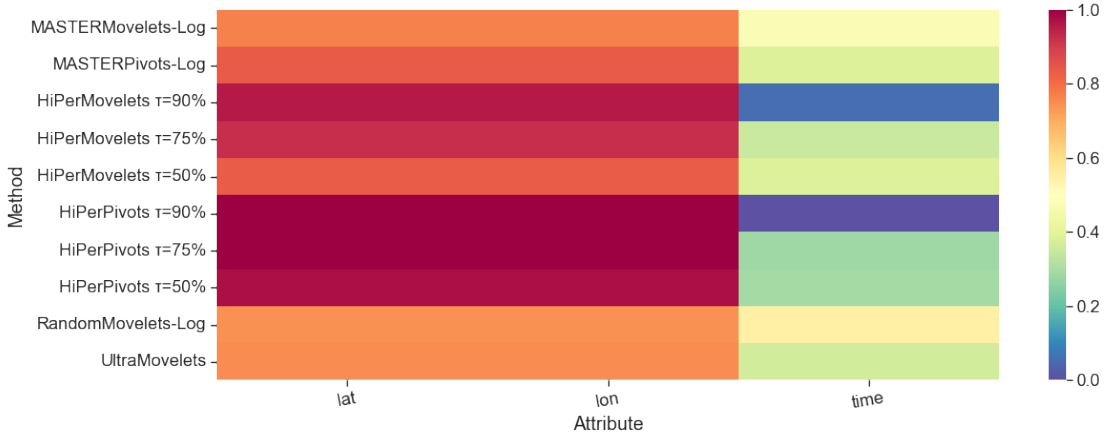


Figure 49 – Heat map for movelet attribute use by class on GoTrack dataset.

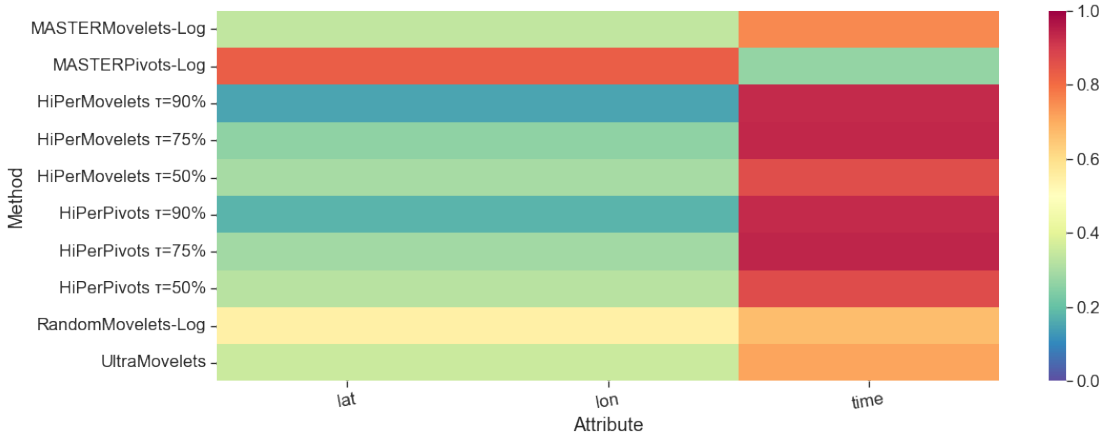


Figure 50 – Heat map for movelet attribute use by class on Vehicles dataset.

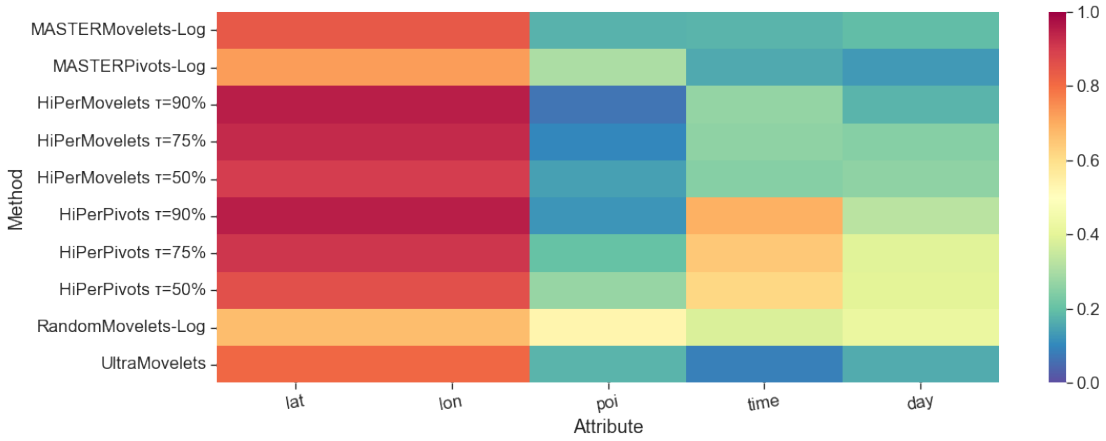


Figure 51 – Heat map for movelet attribute use by class on Brightkite dataset.

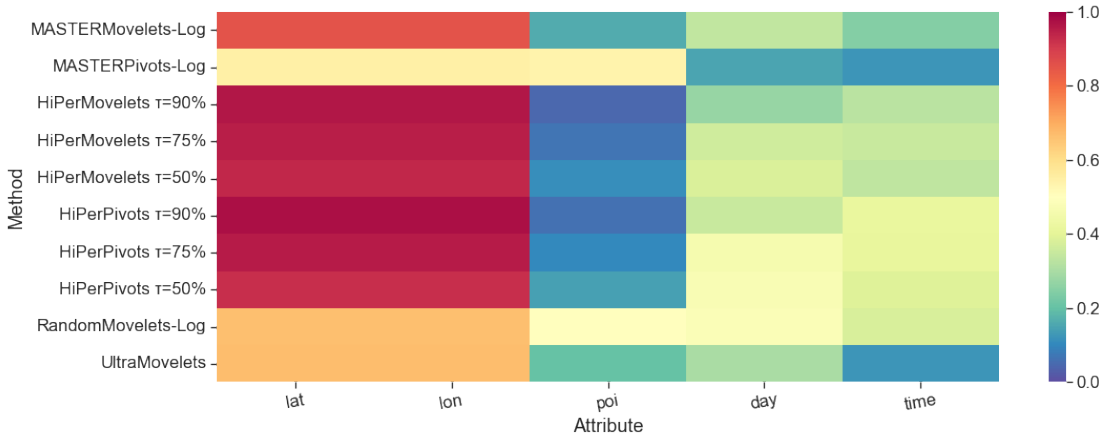


Figure 52 – Heat map for movelet attribute use by class on Gowalla dataset.

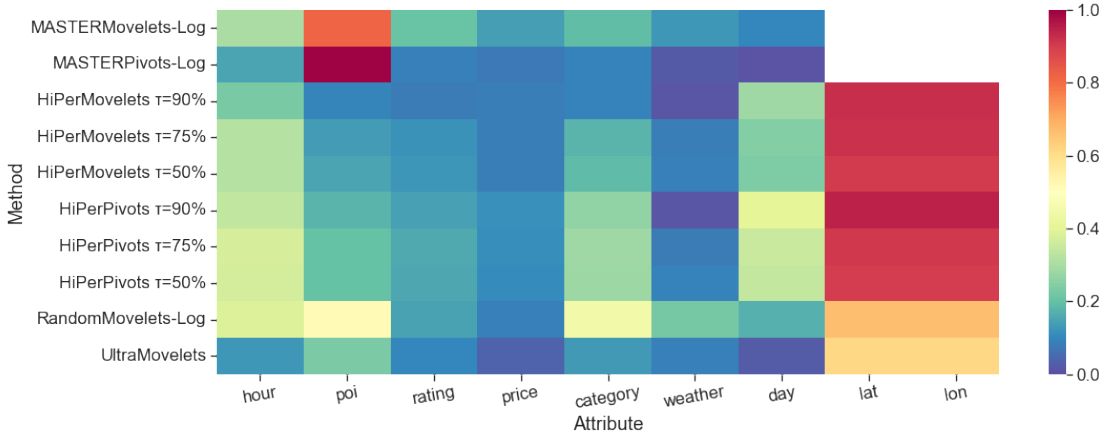


Figure 53 – Heat map for movelet attribute use by class on Foursquare NYC specific.

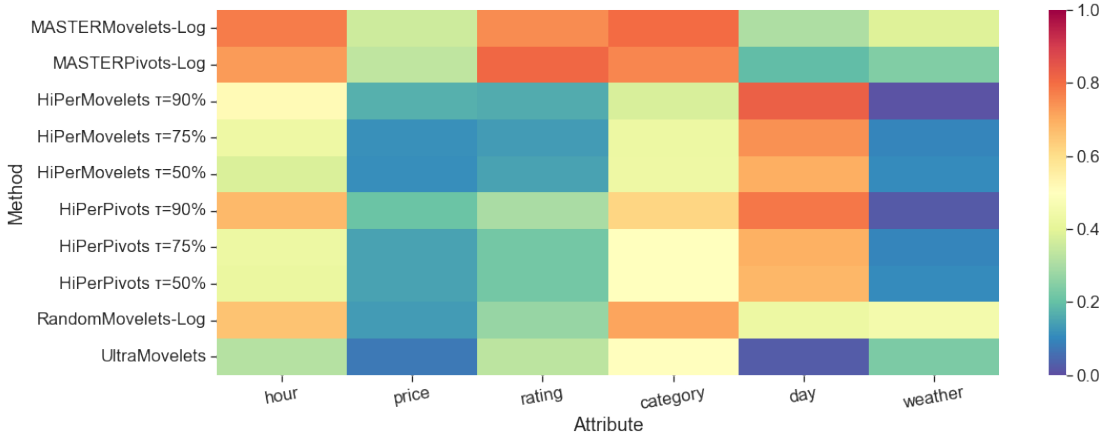


Figure 54 – Heat map for movelet attribute use by class on Foursquare NYC generic.

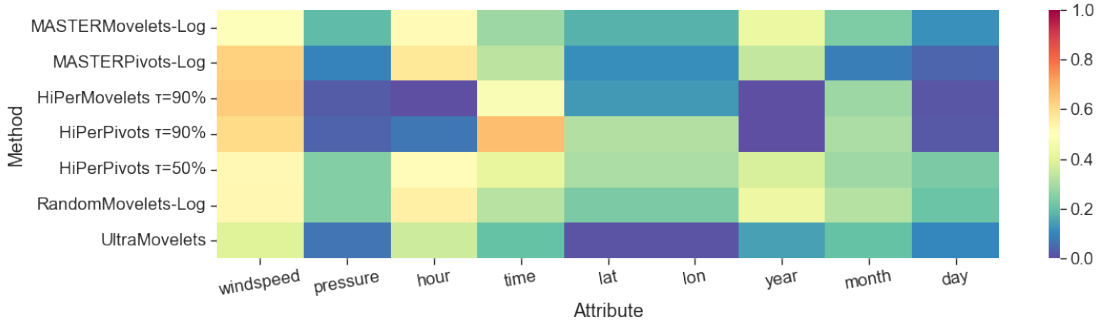


Figure 55 – Heat map for movelet attribute use by class on Hurricanes dataset.