PREVISÃO DE ATIVIDADES HUMANAS PARA DISPOSITIVOS MÓVEIS UTILIZANDO MINERAÇÃO DE PADRÕES SEQUENCIAIS

KLINSMAN MAIA GONÇALVES

PREVISÃO DE ATIVIDADES HUMANAS PARA DISPOSITIVOS MÓVEIS UTILIZANDO MINERAÇÃO DE PADRÕES SEQUENCIAIS

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Eduardo James Pereira Souto

Manaus

Abril de 2019

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Gonçalves, Klinsman Maia
G643p Previsão de atividades humanas para dispositivos móveis utilizando mineração de padrões sequenciais / Klinsman Maia Gonçalves. 2019
138 f.: il. color; 31 cm.

Orientador: Eduardo James Pereira Souto
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. previsão de atividades. 2. previsão de sequência. 3. mineração de padrões sequenciais. 4. reconhecimento de padrões. 5. sensores do smartphone. I. Souto, Eduardo James Pereira II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Previsão de atividades humanas para dispositivos móveis utilizando mineração de padrões sequenciais"

KLINSMAN MAIA GONÇALVES

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Prof. Eduardo James Pereira Souto - PRESIDENTE

Prof. Marco Antonio Pinheiro de Cristo - MEMBRO INTERNO

Prof. Gilbert Breves Martins - MEMBRO EXTERNO

Manaus, 12 de Abril de 2019

"Uma jornada de mil milhas começa com um único passo."
(Lao-Tzu)

Resumo

Os *smartphones* estão transformando a forma como os seres humanos vivem e como se relacionam com as outras pessoas. A medida que o número de sensores embarcados nesses dispositivos aumenta, a quantidade de dados possíveis de serem reconhecidos cresce, tornando esses dispositivos candidatos naturais para monitorar as atividades humanas. Diante disso, há um interesse crescente no desenvolvimento de sistemas inteligentes que utilizam essas informações para reconhecer o comportamento humano, como assistentes virtuais. Uma das formas de observar o comportamento humano é por meio da análise da rotina dos usuários. Na computação, tal identificação pode ser tratada utilizando métodos de mineração de padrões sequenciais (*Sequential Pattern Mining - SPM*). As atividades realizadas pelo ser humano podem ser modeladas em uma sequência simbólica e a partir desta um algoritmo de SPM pode ser utilizado para identificação destes padrões, e com base nestes, identificar qual atividade ocorrerá no futuro. Entretanto, tais tipos de dados ainda são poucos explorados para previsão de eventos futuros. Neste contexto, este trabalho propõe um método para previsão de atividades humanas reconhecidas por meio de dispositivos móveis baseado em mineração de padrões sequenciais. Além disto, para aprimorar a acurácia do método de previsão são utilizados dados de contexto. A utilização destes dados têm sido cada vez mais explorada em sistemas inteligentes com resultados promissores. O método desenvolvido é avaliado utilizando duas bases de dados, alcançando acurácia de até 84,40%. Além disso, os resultados apontam que a utilização de contexto no cenário avaliado aumenta a acurácia do método em até 17,97%.

Palavras-chave: previsão de atividades, previsão de sequência, mineração de padrões sequenciais, reconhecimento de padrões, sensores do smartphone.

Abstract

Smartphones have transformed the way people live and how they relate to others. As the number of sensors embedded in these devices increases, the amount of data that can be recognized grows, which makes these devices natural candidates for monitoring human activities. Therefore, there is a growing interest in the development of intelligent systems, such as virtual assistants, which use this information to recognize human behavior. One way to observe human behavior is by analyzing the routine of smartphone users. In computing, such identification can be done by using sequential pattern mining (SPM) techniques. The activities performed by the human being can be modeled in a symbolic sequence. From this sequence, SPM algorithms can be used to identify these patterns and, based on these, to identify which activity will occur in the future. However, such data are still little explored to predict future events. In this context, this work proposes a method for forecasting human activities recognized through mobile devices, based on sequential pattern mining. In addition, to improve the accuracy of the forecasting method, context data are used. These data have been increasingly explored and used in the field of intelligent systems and have shown promising results. The developed method is evaluated using two databases, reaching an accuracy of up to 84.40 %. In addition, the results indicate that the use of context data in the scenario evaluated increases the accuracy of the method by up to 17.97 %.

Keywords: activity prediction, sequence prediction, sequential pattern mining, pattern recognition, smartphone sensors.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

A previsão de ações futuras é um tópico que tem sido muito explorado na área de inteligência artificial (IA) nos últimos anos devido à sua alta importância [Ahmadi, 1990; Gu et al., 2016; Rodrigues et al., 2019]. Com a utilização de técnicas e métodos de previsão é possível prever: *i)* acidentes em estradas, beneficiando a área de vigilância de transportes [Xu et al., 2015]; *ii)* trajetórias de pessoas em locais lotados, útil para criação de plano de ação em casos de acidente [Alahi et al., 2016]; *iii)* os aplicativos que serão utilizados no *smartphone* [Baeza-Yates et al., 2015]; *iv)* congestionamento de ligações, útil para notificar as operadoras sobre os lugares que devem concentrar seus esforços para melhorar o fluxo [Cici et al., 2016].

Além disso, técnicas de previsão podem ser empregadas para prever comportamentos futuros dos usuários, auxiliando-os na execução de atividades diárias. Esquecer de realizar alguma tarefa durante o seu dia é um fato muito comum para a maioria dos indivíduos, visto que cerca de 77% das informações obtidas pela primeira vez são esquecidas em até 6 dias, caso a informação não seja relembrada [Murre & Dros, 2015]. Alguns problemas de saúde como estresse, ansiedade, depressão e Alzheimer contribuem para que a falta de memória seja ainda mais frequente. Nos Estados Unidos da América, estima-se que a cada 65 segundos alguém desenvolve Alzheimer[1], sendo esta doença a sexta maior causa de morte no país. Dessa forma, auxiliar a antecipação ou lembrança de ações que o usuário tende a esquecer de forma pervasiva é, sem dúvidas, de grande ajuda para muitas pessoas [Bahrainian & Crestani, 2017a].

Uma forma de prever ou antecipar ações humanas de forma ubíqua é com a utilização de *smartphones*. A utilização destes dispositivos, bem como sua capacidade de processamento e sensores embarcados, tem aumentado continuamente. Segundo o por-

[1] http://alz.org

tal alemão de estatística e pesquisa de mercado *Statista*[2], estima-se que em 2020 exista cerca de 2,87 bilhões de usuários de *smartphones* no mundo. Os sistemas embarcados em *smartphones*, utilizados para auxilar e antecipar ações do usuário, são chamados de Assistentes Virtuais, onde destacam-se assistentes como a *Siri*[3], *Cortana*[4], *Bixby*[5] e *Google Assistant*[6]. Tais assistentes tem evoluído cada vez mais em relação à sua capacidade de antecipar ações futuras. No entanto, as ações futuras antecipadas por esses sistemas focam prioritariamente na otimização de ações que o usuário realiza no próprio smartphone, como prever o aplicativo que será utilizado ou prever trajetos e traçar rotas previamente.

Neste contexto, este trabalho foca em prever as atividades físicas que serão realizadas pelo usuário, com base em dados obtidos por meio de sensores vestíveis, em especial por meio de *smartphones*. Diferentes tipos de sensores (e.g.: acelerômetro, giroscópio, magnetômetro e GPS) existentes nos *smartphones* têm sido utilizados para identificar informações de movimento do usuário, como correr, andar, dirigir, sentar, deitar, cair, subir ou descer escadas [Mathie et al., 2004; Lopes et al., 2012; Anguita et al., 2013; Vavoulas et al., 2013; Zhong et al., 2015]. Devido à ampla utilização por parte dos usuários e à capacidade de extrair dados de seus diversos sensores, estes dispositivos têm se tornado candidatos naturais para coletar dados de forma não intrusiva [Fan et al., 2013].

Devido à tal capacidade, uma variedade de aplicações para reconhecimento de atividades humanas tem sido proposta com base nos dados adquiridos por sensores pervasivos. Exemplos dessas aplicações incluem a automação residencial [Reyes-Ortiz et al., 2016], *healthcare* (e.g.: monitoramento de pessoas idosas) [Karantonis et al., 2006], definição de tarefas realizadas por robôs [Magnanimo et al., 2014], sugestão e lembretes de atividades para serem realizadas em casas inteligentes [Moutacalli et al., 2015; Minor et al., 2015; Nazerfard & Cook, 2015], entre outras.

Além das informações relacionadas às atividades, o *smartphone* provê ainda diversos dados de contexto, como as informações sobre o ambiente e a situação do usuário, por meio de sensores de localização, áudio, imagem, entre outros [Nweke et al., 2018]. Além dos dados que podem ser coletados e interpretados por meio dos sensores do *smartphone*, existe ainda a alta quantidade de dados sensíveis e humanamente interpretáveis que o próprio usuário armazena no dispositivo como seus gostos, agenda e lugares que frequenta [Hang et al., 2012].

[2]http://www.statista.com
[3]http://www.apple.com/br/siri
[4]http://www.microsoft.com/pt-br/windows/cortana
[5]http://www.samsung.com/br/apps/bixby
[6]http://assistant.google.com

1.1 Problema e Motivação

Atualmente, existem diversas técnicas e abordagens para previsão de comportamento e atividades humanas. A maioria das soluções propostas realiza predição com base em dados obtidos a partir de vídeos [Xu et al., 2015; Yang et al., 2015; Alahi et al., 2016]. Um dos problemas com esta abordagem está na forma com que os dados são coletados, pois a detecção dos eventos ocorridos são limitados ao ambiente no qual a câmera se encontra e ao seu ângulo de visão. Filmar uma pessoa durante o dia todo em um ângulo que seja favorável para o reconhecimento das ações pode ser uma tarefa muito trabalhosa. Por outro lado, os *smartphones* normalmente são utilizados todos os dias e as pessoas o levam consigo para onde quer que vão. Além disso, tais dispositivos possuem diversos sensores com capacidade para reconhecer as atividades realizadas pelo usuário, mas ainda assim não costumam ser explorados para previsão de atividades humanas.

Prever qual atividade um usuário vai realizar no futuro, com base apenas nos dados do *smartphone* não é uma tarefa trivial. Um dos motivos é que normalmente as pessoas têm rotinas diferentes. Como consequência, as previsões serão diferentes para cada pessoa. Desta forma, não é adequado gerar um modelo padrão baseado em um tipo de rotina e assumir que tal modelo irá se adequar à todos os casos. Logo, é necessário que o método de previsão seja adaptável para cada pessoa, individualmente. Para criar um modelo que seja adaptável, é necessária a obtenção de registros individuais do usuário. De modo geral, existem duas formas de manter e utilizar tal histórico. A primeira abordagem baseia-se no envio dos dados para um serviço externo que os armazene e forneça as previsões. Uma segunda abordagem baseia-se no armazenamento do histórico e treinamento no próprio *smartphone*. Ambas abordagens possuem suas limitações específicas.

A utilização de um serviço externo de previsão requer a utilização significativa de largura de banda e de bateria do dispostivo, pois as coletas são realizadas a todo momento. Além disso, caso a previsão também seja online, o usuário sempre ficaria dependente de uma conexão com a Internet. Na abordagem onde todo o processamento é realizado no *smartphone*, mesmo com todo avanço tecnológico, existem problemas de restrições de armazenamento e processamento de dados dos dispositivos.

Os trabalhos relacionados à previsão de atividades que utilizam dados obtidos por meio de *smartphones* desenvolvidos até então, não abordam a previsão de atividades físicas humanas. Os métodos existentes focam na previsão: *i)* do próximo aplicativo que vai ser utilizado [Bahrainian & Crestani, 2017b]; *ii)* do próximo trajeto [Zong et al., 2019]; *iii)* de qual música será ouvida posteriormente [Sun et al., 2016] e; *iv)* de tópicos

que serão analisados em reuniões futuras, com base no áudio das reuniões passadas [Bahrainian & Crestani, 2018]. Tais métodos também utilizam dados de contexto para melhorar a taxa de acerto da previsão.

Na literatura, diversas técnicas de previsão têm sido propostas para diferentes domínios de aplicação, como previsão de: rotas e trajetos [Zong et al., 2019], ações no mercado financeiro [Zhang et al., 2019] e de comportamento humano [Moreira et al., 2019]. Tais técnicas também podem ser utilizadas para descobrir a próxima ação que um usuário pretende realizar. Contudo, devido à ampla quantidade de dados coletados pelos sensores e a limitação de recursos dos *smartphones* é premissa básica a adoção de modelos de previsão que requeiram poucos recursos. Para atacar estes problemas, este trabalho propõe o emprego de técnicas de representação simbólica para discretizar os reconhecidos por meio dos sensores inerciais de dispositivos móveis. Neste trabalho, a discretização dos dados consiste na transformação de segmentos de sinais dos sensores inerciais em símbolos (ou palavras).

Ao utilizar uma representação simbólica, em vez de dados brutos, a dimensinalidade dos dados é reduzida [Quispe et al., 2018]. Logo, torna-se uma opção ideal para ser utilizada em dispositivos móveis. No entanto, ainda assim a quantidade de padrões existentes na sequência pode excessiva, provocando um longo tempo de processamento. Assim, para lidar com este problema este trabalho utiliza um algoritmo de mineração de padrões sequenciais capaz de extrair somente as palavras (atividades) mais frequentes. Desta forma, a numerosidade dos dados é reduzida, assim como o consumo de memória e tempo de processamento. A partir dos padrões frequentes encontrados é utilizado probabilidade condicional e busca aproximada para prever a atividade futura.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver um método para previsão de atividades humanas por meio de dados obtidos a partir de dispositivos móveis, utilizando algoritmos de mineração de padrões sequenciais. O método deve ser modelado levando em consideração que a utilização de dados de contexto incorporados na sequência de atividades são essenciais, e deve ser projetado para utilizar uma baixa quantidade de recursos computacionais.

Para atingir esse objetivo, pretende-se alcançar os seguintes objetivos específicos:

- Estimar e avaliar os parâmetros e algoritmos de mineração de padrões sequenciais.
- Adaptar e aprimorar o método de mineração de padrões sequenciais utilizado para o problema abordado.

- Organizar os dados de atividade e contexto de forma simbólica sequencialmente.
- Desenvolver um modelo probabilístico baseado nos padrões encontrados nas sequências.
- Avaliar, por meio de testes, a eficácia do método desenvolvido.

1.3 Contribuições

As principais contribuições deste trabalho são:

- Introdução de uma abordagem para previsão de atividades humanas no contexto de dispositivos móveis baseada em mineração de padrões sequenciais. No processo de revisão da literatura não foram encontrados trabalhos que realizassem tal tipo de previsão.
- Aprimoramento de um método de previsão de sequências, tornando-o mais eficaz ao utilizar algoritmos de busca aproximada. Além disso, aprimorando sua eficiência ao utilizar algoritmos de mineração de sequências baseados em busca em profundidade, mapeamento vertical de *bits* e mapas de coocorrência.
- Criação de uma base de dados de atividades diárias inédita na literatura, composta por dados do acelerômetro, giroscópio, GSP e câmera.

1.4 Organização do Trabalho

O restante desta dissertação está organizado da seguinte forma:

- O Capítulo 2 aborda os conceitos teóricos utilizados neste trabalho, bem como os fundamentos que norteiam o desenvolvimento de métodos para previsão de sequência. Entre os conceitos apresentados estão: descrição dos tipos de atividades e como podem ser caracterizadas, classificação dos tipos de dados de contexto, e quais atividades humanas podem ser reconhecidas. Também são introduzidos os principais algoritmos utilizados neste trabalho e em pesquisas correlatas.
- O Capítulo 3 apresenta os principais trabalhos relacionados, citando as principais abordagens utilizadas para previsão de sequências. Os trabalhos são categorizados em três grupos. O primeiro grupo apresenta superficialmente como prever dados em diferentes cenários (mercado financeiro, comportamental, robótica, trajetos). No segundo grupo são apresentadas as pesquisas voltadas para previsão de

atividades humanas e, no terceiro, as pesquisas voltadas para previsão utilizando apenas os dados de dispositivos móveis. Por fim, são apresentadas as discussões e comparações entre os trabalhos citados.

- O Capítulo 4 apresenta uma breve discussão sobre a definição formal do problema de previsão de atividades e sua modelagem. Em seguida, é descrito como ocorre a aquisição dos dados de entrada, bem como a saída do método após o processamento. Logo após, os algoritmos utilizados para a criação do método denominado SOPHIA são descritos em detalhes. O capítulo é encerrado com uma breve discussão sobre alguns dos algoritmos utilizados.

- O Capítulo 5 apresenta o protocolo experimental proposto para avaliar a metodologia abordada, bem como as bases e métricas utilizadas neste trabalho. O protocolo é dividido em duas etapas. A primeira parte apresenta uma comparação entre os métodos de mineração de padrões sequenciais em termos de memória e tempo de processamento. Mais adiante são exibidos os resultados obtidos em relação à taxa de acerto. Dois cenários são utilizados na avaliação, cada um com um tipo (nível) de atividade.

- Por fim, o Capítulo 5 apresenta as conclusões obtidas a partir deste estudo, bem como os principais desafios, melhorias e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo aborda os principais conceitos, seus significados e as principais técnicas utilizadas para previsão de sequências. A Seção 2.1 aborda a definição de atividades humanas e como tais atividades podem ser categorizadas. Mais adiante, na Seção 2.2 são apresentados os conceitos relativos aos dados de contexto, bem como a sua relação com o reconhecimento e previsão de atividades humanas. Na Seção 2.3, são brevemente introduzidas as principais abordagens de reconhecimento de atividades humanas por *smartphones* adotadas na literatura. A Seção 2.4 introduz de forma sucinta a área de mineração de dados como um todo. Logo após, são exibidas as principais técnicas para minerar padrões sequenciais e mais adiante técnicas probabilísticas para previsão de sequências.

2.1 Atividades Humanas

Segundo o dicionário Aurélio [Ferreira, 2004], alguns dos significados da palavra "atividade" são: *i)* realização de uma função ou operação específica; *ii)* exercício ou aplicação dessa capacidade; *iii)* faculdade de exercer a ação. Com base nestas definições, existem diversos trabalhos que classificam as atividades em categorias (ou níveis). Alguns autores classificam as atividades em duas categorias, como por exemplo, atividades simples e atividades complexas [Dernbach et al., 2012], de baixo nível e de alto nível [Khan et al., 2014], micro-atividade e macro-atividade [Yan et al., 2012]. As atividades simples, de baixo nível ou micro-atividades são aquelas que podem ser reconhecidas por meio de um único sensor. Enquanto as atividades complexas, de alto nível ou macro-atividades são as que precisam de informações de diversos sensores (ou de contexto) e da combinação de mais de uma atividade.

Cook & Krishnan [2015] definem que uma atividade é composta de uma sequência de ações e/ou interações realizadas por um ou mais indivíduos. A ação corresponde a uma sequência de estados (ou micro-atividades) de comportamento humano realizada por um indivíduo em um curto período de tempo. Interação corresponde a uma ação ocorrida que envolve mais de um indivíduo, como abraçar e apertar as mãos. Assim, uma atividade é composta por uma sequência de ações e cada ação é composta por um conjunto de estados, onde um estado é um gesto ou movimento qualquer, como andar, correr, sentar, levantar e subir escada.

Han [2013] propõe agrupar as atividades em três níveis (ou categorias), conforme exibido na Figura 2.1. No primeiro nível se encontram as atividades que consistem em gestos ou movimentos, ou seja, são as atividades em seu nível mais granulado. Tais atividades costumam durar um curto período de tempo, normalmente poucos segundos, e costumam ser repetitivas. No segundo nível estão as atividades básicas, que são obtidas a partir de uma diferente combinação de gestos e movimentos. Por exemplo, "andar" (nível 2) é composta por uma sequência de passos (nível 1). Atividades de nível 2 costumam ter uma duração maior que as do nível 1.

No terceiro nível estão as atividades de alto nível, que são chamadas de "atividades específicas". Estas atividades podem ser inferidas a partir da combinação de um conjunto de atividades básicas e um ou mais dados de contexto. Dado um cenário no qual uma pessoa está andando, neste caso é possível utilizar dados de contexto como localização e dia da semana para inferir qual atividade específica a pessoa realizou. Por exemplo, se a pessoa estava andando em um shopping no fim de semana (ou feriado) à noite, é possível inferir que ela estava passeando. Caso ela estivesse andando em um

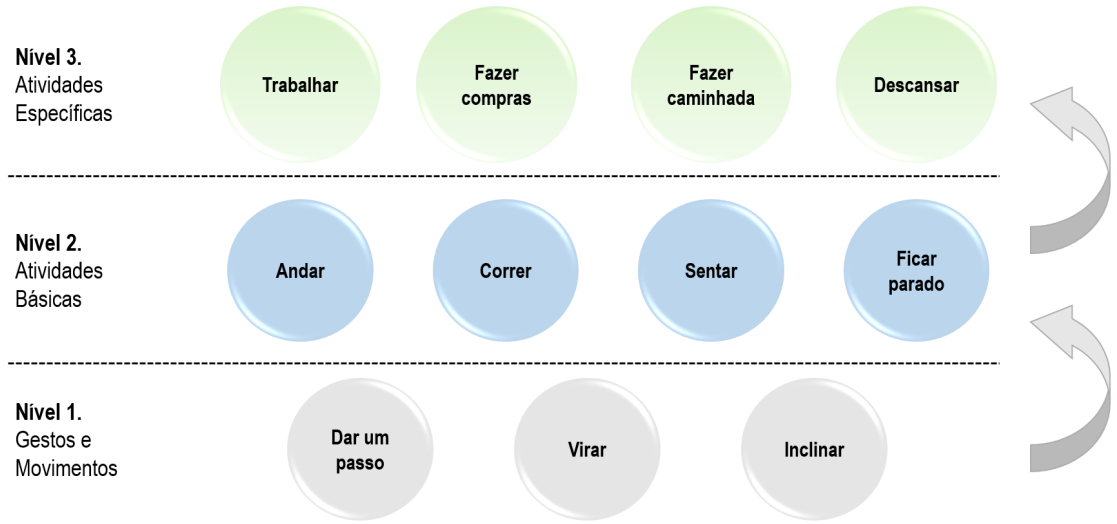


Figura 2.1: Atividades humanas em 3 níveis [Han, 2013].

shopping em um dia de semana, ao meio-dia, é provável que ela esteja almoçando ou comprando algo.

As atividades de nível 1 e 2 podem ser reconhecidas por meio de sensores, de acordo com a identificação dos movimentos realizados. Sensores são dispositivos que respondem a estímulos físicos por meio de sinais. Estes estímulos podem ser relacionados à pressão, umidade, temperatura, entre outros. Os sensores utilizados para reconhecer uma atividade podem ser classificados em sensores de ambiente e sensores vestíveis [Cook & Krishnan, 2015]. Os sensores de ambiente (e.g. umidade, luz, temperatura, pressão, vibração) se caracterizam por estarem disponíveis apenas no local no qual o indivíduo se encontra, e não no próprio indivíduo. Os sensores vestíveis (e.g.: acelerômetro, giroscópio, magnetômetro e GPS) são utilizados em alguma parte do corpo ou na roupa do indivíduo.

As atividades no nível 3, que são as "específicas", costumam ter uma longa duração. Tais atividades costumam ser compostas de atividades de nível 2 e normalmente contam com alguma informação de contexto para serem reconhecidas. Os dados de contexto podem ser obtidos a partir de diversas fontes, como o local no qual o indivíduo se encontra, horário, clima, entre outros. A Seção 2.2 descreve o que é dado de contexto e como isto pode auxiliar no reconhecimento de uma atividade específica.

Este trabalho adota a nomenclatura de Dernbach et al. [2012], que define as atividades como simples e complexas. Seguindo a Figura 2.1, as atividades simples são as de nível 2 e as atividades complexas seriam as de nível 3. Embora outros trabalhos também usem o termo "atividade humana" para referenciar atividades genéricas (e.g. atender telefone, acessar e-mail, acessar aplicativos de mensagens), no contexto deste trabalho atividades humanas correspondem às atividades físicas.

2.2 Dados de Contexto

De acordo com o dicionário Aurélio [Ferreira, 2004], "contexto" pode ser definido como um conjunto de circunstâncias à volta de um acontecimento ou de uma situação. Diversos autores redefinem esta palavra a fim de adaptá-la à realidade computacional. Schilit et al. [1994] definem contexto como localização, identidades, pessoas próximas, objetos próximos e mudanças realizadas nestes objetos. Outros autores definem contexto como os diferentes aspectos da situação do usuário [Hull et al., 1997]. Ryan et al. [1999] definem como a localização, ambiente, tempo, temperatura e as pessoas ao redor do usuário. Abowd et al. [1999] definem contexto, de uma forma mais ampla, como qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade.

A entidade pode ser uma pessoa, um lugar ou um objeto que é considerado relevante para a interação entre o usuário e a aplicação. A "situação da entidade" refere-se ao que a pessoa está fazendo, onde ela está e a condição dela em um determinado momento.

Em outras palavras, contexto são as informações relevantes sobre o que acontece ao redor do usuário. As informações de contexto podem ser entendidas, analisadas e processadas por um sistema. O nome dado aos sistemas que utilizam tais informações é Sistema Sensível ao Contexto (do inglês, *Context-aware system*). Um sistema sensível ao contexto é um sistema que fornece e compreende os contextos disponíveis percebidos dos usuários utilizando informação de sensores e também executa as adaptações de serviço apropriadas com base nesses contextos e suas alterações.

Existem diversos aspectos e classificações para os dados de contexto na literatura. Mayrhofer et al. [2003] categorizam contexto em diferentes aspectos, como: geográfico, físico, organizacional, social, usuário, tecnológico e temporal. Sigg [2008] separa os dados de contexto em 5 aspectos principais (tempo, localização, temperamento, ambiente e identidade). Han [2013] categoriza os dados de contexto em baixo nível e alto nível, como representado na Figura 2.2.

Conforme a Figura 2.2, dados brutos são extraídos a partir dos sensores e, com base nestes, os dados de contexto de baixo nível são obtidos. Por exemplo, o sensor de temperatura é capaz de mensurar a temperatura de um ambiente ou objeto (dado bruto). Esta informação pode ser transformada em alguma unidade, como em graus

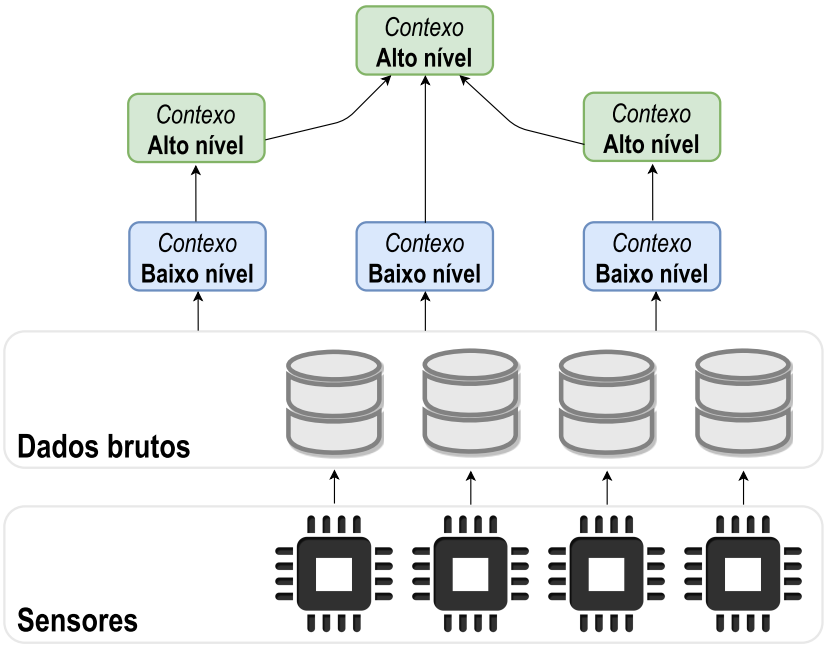


Figura 2.2: Classificação dos níveis de contexto [Han, 2013].

Celsius (e.g. 22°C), que é um dado de contexto de baixo nível. Os dados de contexto de alto nível podem ser obtidos a partir da combinação de um ou mais dados de contexto de baixo ou alto nível ou a partir da combinação de dados de contexto de baixo nível e de dados brutos.

Os dados de contexto são essenciais para inferir as atividades que o indivíduo realiza [Han, 2013; Li & Fu, 2016; Wang et al., 2017b]. De acordo com a ciência cognitiva, as informações de contexto são dados críticos para o compreendimento de atividades humanas [Marszalek et al., 2009; Han et al., 2009]. Por exemplo, dado que um indivíduo se encontra em seu local de trabalho, sentado em frente ao computador, com o computador ligado mas sem utiliza-lo, é possível inferir que naquele momento a pessoa se encontra ocupada (e.g. falando ao telefone). Se o indivíduo estivesse utilizando o computador, seria possível inferir que ele estava trabalhando, por exemplo. Logo, é possível notar que para esta categoria de problema, os dados de contexto aumentam a relevância do reconhecimento de atividades humanas [Han, 2013]. Esta é a razão das atividades complexas (de nível 3) precisarem de informações de contexto para serem reconhecidas, conforme mencionado na seção 2.1.

2.3 Reconhecimento de Atividades Utilizando *Smartphone*

Atividades humanas podem ser reconhecidas por meio de sensores, um dos sensores mais utilizados para este fim é o acelerômetro [Ravi et al., 2005; Lau & David, 2010; Kwapisz et al., 2011; Anguita et al., 2013; Vavoulas et al., 2013; Micucci et al., 2017]. Bao & Intille [2004] utilizaram o acelerômetro em 5 diferentes partes do corpo (braço, pulso, joelho, tornozelo, cintura) para reconhecer 20 atividades. Kern et al. [2007] também utilizam o acelerômetro, mas neste caso em 12 diferentes partes do corpo e para reconhecer 5 atividades básicas. Os dois trabalhos citados alcançaram uma acurácia de cerca de 90%.

O reconhecimento de atividades utilizando sensores vestíveis em diferentes partes do corpo alcançam resultados promissores [Ignatov, 2018]. No entanto, um problema identificado nesta abordagem, é que o uso de sensores fixos em diversas partes do corpo é muito intrusivo e pode fazer com que as pessoas se sintam desconfortáveis [Lau & David, 2010; Ravi et al., 2005; Choudhury et al., 2006]. Uma forma não intrusiva de reconhecer atividades é utilizando os dados dos sensores do *smartphone*. Devido à sua alta utilização, quantidade de sensores (e.g.: acelerômetro e GPS) e poder de processamento, esses dispositivos têm se tornado candidatos naturais para o

reconhecimento de atividades humanas [Fan et al., 2013].

Por esses motivos, diversos pesquisadores têm explorado a utilização de *smartphones* para o reconhecimento de atividades. Yang [2009] e Kwapisz et al. [2011] utilizaram apenas o acelerômetro do *smartphone* para reconhecer 6 atividades básicas. Os dois trabalhos atingiram cerca de 90% de acurácia. Ryder et al. [2009] apresentaram uma aplicação para analisar como pessoas com doenças crônicas (como mal de *Parkinson* e atrofia muscular) se locomovem. As atividades realizadas foram reconhecidas utilizando o acelerômetro e o GPS do *smartphone*.

Uma limitação encontrada em muitos trabalhos de reconhecimento de atividades é que o *smartphone* deveria sempre ficar em um lugar fixo na pessoa, mas no dia a dia as pessoas costumam deixar o celular no bolso em qualquer posição. A fim de resolver esta limitação, Henpraserttae et al. [2011] desenvolveram um método para reconhecer atividades, independente da posição que o *smartphone* se encontre. Para tanto, 16 posições diferentes foram testadas e o método atingiu cerca de 90% de acurácia.

Portanto, pode-se notar que o campo de reconhecimento de atividade humanas tem avançado fortemente, tornando possível o surgimento de um novo campo, o de previsão de atividades humana. Prever uma atividade humana pode ser comparado com o problema de prever uma sequência de eventos com base nos eventos anteriores. Existem inúmeras formas de resolver tal problema, nas seções a seguir é descrito a área de mineração de dados e principais técnicas utilizadas para previsão de sequências. Compreender como tais técnicas funcionam é essencial para entender pesquisas correlatas descritas no capítulo seguinte.

2.4 Mineração de dados

Com o rápido crescimento da disponibilidade de dados em diversos tipos de aplicações (redes sociais, sistemas industriais, comércio eletrônico, entre muitos outros), a análise de todos esses dados se tornou humanamente inviável. Logo, surge a necessidade de estudar e analisar como extrair conhecimento a partir dessas bases de dados, de forma eficaz e automática. O conhecimento extraído pode ser utilizado para futuro planejamento, tomada de decisão e muitas outras aplicações. A mineração de dados é apenas uma das etapas do processo de descoberta de conhecimento. Esta etapa busca aplicar algoritmos em dados previamente preparados a fim de descobrir padrões.

O termo Mineração de Dados (*Data Mining*) possui diversas definições. Muitos autores tratam mineração de dados como um sinônimo para o processo de Descoberta de Conhecimento em Dados (*Knowledge Discovery from Data* - KDD). Han et al. [2011]

definem este termo como o processo de descobrir conhecimento e padrões interessantes a partir de uma grande quantidade de dados. Outros autores tratam mineração de dados como uma das etapas do KDD.

Fayyad et al. [1996] definem KDD como o "processo não trivial de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis em dados". De acordo com esta definição, "dados" são conjuntos de fatos e "padrão" corresponde a uma expressão em alguma linguagem descrevendo um subconjunto dos dados. O termo "processo" informa que o KDD compreende vários passos que envolvem a preparação de dados, busca por padrões e a avaliação do conhecimento obtido, de forma que tais passos possam se repetir em múltiplas interações. O termo "não trivial" indica que o processo não é direto e pode envolver o uso de técnicas de busca e algoritmos.

O processo de KDD é constituído basicamente pelas seguintes etapas [Terzi, 2012], *i*) seleção; *ii*) pré-processamento; *iii*) transformação; *iv*) mineração de dados; *v*) interpretação e avaliação. A Figura 2.3 ilustra estes passos. Os dados de entrada podem ser obtidos por meio de inúmeras fontes. Na etapa de "seleção", os dados mais representativos são selecionados. Na etapa de "pré-processamento" é realizada uma limpeza dos dados, como exclusão de *outliers*, valores inválidos e quaisquer outros tipos de dados que possam atrapalhar o método. Na etapa de "transformação" são realizados os ajustes dos dados para um formato comum, como categorização, discretização ou normalização, por exemplo. Logo após, é executada a principal etapa do processo: a mineração dos dados. Nesta etapa são executados os algoritmos de mineração que encontram padrões e geram modelos que são avaliados na etapa final.

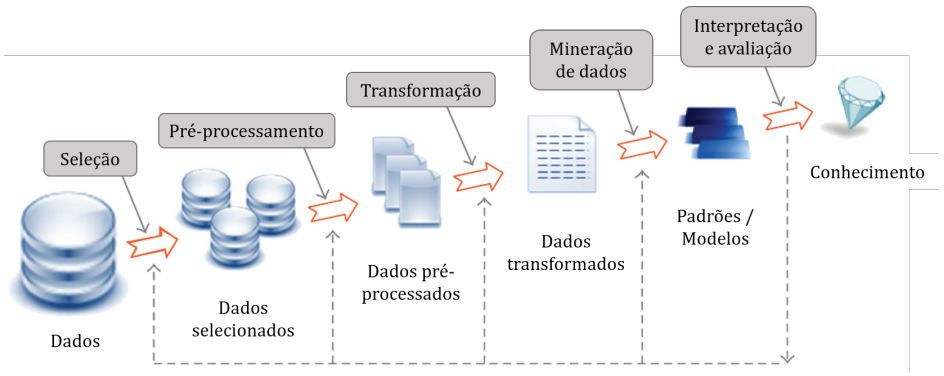


Figura 2.3: Processo de descoberta de conhecimento em dados - KDD (adaptado de Terzi [2012])

De acordo com o problema abordado uma metodologia diferente pode ser utilizada para o solucionar. Algumas das principais tarefas de mineração de dados são as seguintes [Furtado et al., 2005]:

- **Classificação:** A tarefa de classificação supervisionada consiste em criar um modelo ou classificador com base nos dados de treinamento. Os dados de treinamento são aqueles em que se conhece tanto os atributos dos objetos quanto a classe a qual este pertence. O classificador é gerado utilizando regras para determinar a classe de objetos não conhecidos. A classificação é amplamente utilizada para diversas tarefas, como o diagnóstico de doença com base nos sintomas [Vijayarani et al., 2015], análise de crédito [Harris, 2015], entre outros.

- **Agrupamento:** É o processo de agrupar um conjunto de objetos em classes de objetos semelhantes de acordo com os seus atributos. Diferente da classificação, nesta fase não é preciso saber a classe (rótulo) do objeto. Neste processo os grupos são formados de acordo com a similaridade de seus atributos. Uma das suas utilizações pode ser para retornar documentos similares em um site de busca, por exemplo.

- **Mineração de Regras de Associação:** Busca descobrir associações entre objetos. Uma regra de associação possui o seguinte formato, $a_1 \wedge ... \wedge a_i \rightarrow b_1 \wedge ... \wedge b_j$ e informa que o objeto a_i tende a ser encontrado junto com b_j. Tal técnica foi criada com o propósito de analisar "cesta de compras de supermercado" e descobrir os itens que normalmente são comprados juntos [Agrawal et al., 1994]. Desta forma, tal informação pode ser útil para ajudar uma reestruturação de produtos nas prateleiras do mercado [Han et al., 2011].

- **Mineração de Padrões Sequenciais:** Tem como objetivo identificar sequências de eventos que ocorrem frequentemente em um banco de dados temporal. Tal abordagem é utilizada em diversas áreas com inúmeras aplicações como análise de sequências de DNA, evolução de compras realizadas por clientes, análise da sequência que leva o usuário a acessar determinada página, evolução de sintomas de pacientes, entre outros [Fournier-Viger et al., 2017].

No contexto deste trabalho, a mineração de padrões sequenciais é a tarefa mais importante, pois é uma das formas de identificar padrões que podem se repetir no futuro e, desta forma, prever a continuação de uma sequência. Antes de entrar em detalhes sobre tal tipo de mineração, a Seção 2.5 explana como minerar padrões em dados. Mais adiante, as seções 2.6, 2.8 e 2.7 exibem os detalhes e algoritmos que podem ser utilizados para descobrir padrões sequenciais, que é considerado um tópico avançado desta área. Por fim, a Seção 2.10 exibe outros métodos que também são utilizados para previsão de sequências.

2.5 Mineração de Regras de Associação

A descoberta de regras de associação busca encontrar padrões regulares nos dados. Seu objetivo principal é encontrar dependências ou correlação entre os itens da base de dados, ou seja, determinar quais itens tendem a ocorrer juntos em uma transação. Um exemplo clássico é determinar quais produtos costumam ser adquiridos em uma mesma compra no supermercado. Uma regra de associação pode ser da forma, ($cerveja \rightarrow amendoim$) = *suporte*: *2%, confiança: 50%*. Que significa que em 2% de toda a base contém o conjunto de itens (*itemset*) "cerveja, amendoim" e que em 50% das vezes que cerveja ocorre, o item "amendoim" também ocorre.

Existem também as regras de associação que ocorrem em tempos regulares, ou seja, a regra ocorre periodicamente. Estas são chamadas de "regras de associação cíclica". Um exemplo deste tipo de regra é, $café \wedge pão$. Se analisar todas as transações da base de dados, esta regra pode não parecer interessante, mas se limitar nas transações que ocorrem por faixa de tempo, como entre 7:00h e 9:00h, pode-se descobrir que tal regra ocorre com alta frequência.

Uma importante tarefa na descoberta de regras de associação é a mineração de *itemsets* frequentes, que busca encontrar quais são os conjuntos de itens que mais são encontrados na base de dados e a frequência com a qual estes ocorrem. Existem diversos algoritmos para realizar esta tarefa, como Apriori, AprioriTID [Agrawal et al., 1994], FPGrowth [Han et al., 2004], FIN [Deng & Lv, 2014] e PrePost+ [Deng & Lv, 2015]. Todos estes algoritmos encontram os *itemsets* que são mais frequentes que um limiar, chamado de suporte mínimo (*minsup*). O funcionamento de um destes algoritmos é descrito a seguir.

2.5.1 Algoritmo *Apriori*

O algoritmo Apriori foi proposto em 1994 [Agrawal et al., 1994], pelo time de pesquisa do Projeto QUEST da IBM que originou o software *Intelligent Miner*. Este algoritmo, baseia-se na propriedade Apriori, que foi definida por Agrawal et al. [1994] como: nenhum super-conjunto de um *itemset* não frequente pode ser frequente. Ou seja, o subconjunto de um *itemset* frequente também é frequente. Um *itemset* é denotado por $x_1, x_2, ..., x_k$, onde para cada x_i, para $i \in \{1, ..., k\}$, é um item. Um *itemset* com k itens é denominado *k-itemset* (ou *itemset* de comprimento k). O suporte de um *itemset* $\mathcal{I}$ com relação a um banco de dados $\mathcal{D}$, denotado por *sup($\mathcal{I}$)*, corresponde ao número de transações em $\mathcal{D}$ que contem $\mathcal{I}$. Onde $\mathcal{I}$ é um conjunto de itens (*itemset*), denotado por $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ e $\mathcal{D}$ um banco de dados de transações. Um *itemset*

é dito frequente em $\mathcal{D}$ se seu suporte é maior ou igual ao valor do suporte mínimo especificado pelo usuário.

O Apriori é um algoritmo interativo, a cada interação o algoritmo produz *itemsets* candidatos de comprimento $k+1$ a partir dos itemsets frequentes de tamanho k, executando uma operação de junção e em seguida de poda de candidatos. O algoritmo começa percorrendo todas as transações do banco de dados e calculando os 1-*itemsets* frequentes. A partir destes, o conjunto dos 2-*itemsets* candidatos é obtido. O banco de dados é novamente percorrido para obter seus suportes. Os 2-*itemsets* frequentes serão agora utilizados no próximo passo, na obtenção dos 3-*itemsets* candidatos. Este processo é repetido a obter todos os *itemsets* frequentes

O Algoritmo 1 exibe um pseudocódigo do algoritmo Apriori [Agrawal et al., 1994]. C_k representa o conjunto de *k-itemsets* candidatos e F_k o conjunto de $k-$ *itemsets* frequentes. Este algoritmo é composto de por três etapas principais, que são, *i*) geração de candidatos; *ii*) poda; *iii*) cálculo do suporte. Na etapa *i* são gerados os *itemsets* candidatos de tamanho *k+1* a partir dos *itemsets* de tamanho *k*, a função *apriori-generate* é responsável por essa criação. *Itemsets* $\in F_k$ são combinados em uma etapa de junção de forma que é gerado *itemsets* de tamanho *k+1*. Por exemplo, seja $F_2 = \{(a,b),(a,c),(a,d),(b,c),(b,d)\}$ o conjunto é dado por $C_3 = \{(a,b,c),(a,b,d),(a,c,d),(b,c,d)\}$.

Algoritmo 1: APRIORI

Entrada: $\mathcal{D}, minsup$
Saída: Conjunto de *itemsets* frequentes

```
1  início
2      k = 1
3      F_k = {i | i ∈ I ∧ σ({i}) ≥ N × minsup}
4      repita
5          k = k + 1
6          C_k = apriori-generate(F_{k-1})
7          para transação t ∈ D faça
8              C_t = subset(C_k, t)
9              para candidato c ∈ C_t faça
10                 σ(c) = σ(c) + 1
11             fim
12         fim
13         F_k = {c | c ∈ C_k ∧ σ(c) ≥ N × minsup}
14     até F_k = 0;
15 fim
16 retorna ⋃ F_k
```

Na etapa *ii* é executada a eliminação de candidatos que possuem um subconjunto não frequente. Considerando o exemplo anterior, o *itemset* (*b, c, d*) é podado, pois seu subconjunto (c, d) não é frequente. Logo, após a poda $C_3 = \{(a, b, c), (a, b, d), (a, c, d)\}$. Na etapa *iii* a base de dados é percorrida e o suporte de cada *itemset* é calculado.

2.6 Mineração de Sequência

O problema de mineração de padrões sequenciais (*Sequence Pattern Mining - SPM*) foi inicialmente estudado por Agrawal & Srikant [1995]. Nesse trabalho os autores propõem um algoritmo para mineração de padrões sequenciais em uma base de dados de clientes, chamado de *AprioriAll*. Desde então, diversos métodos para aprimorar tal método têm sido desenvolvidos ano após ano [Fournier-Viger et al., 2017].

O problema de minerar padrões sequenciais é de grande importância na área de mineração de dados e possui inúmeras aplicações, tais como, análise de sequências de DNA [Wang et al., 2004], análise de navegação em páginas *web* (utilizada para indicar quais páginas anteriores levaram o usuário a alcançar a página atual) [Fournier-Viger et al., 2012], análise de compras [Srikant & Agrawal, 1996], análise de texto [Pokou et al., 2016], redução de energia em *smarthomes* [Schweizer et al., 2015], entre outros. Existem diversos algoritmos e abordagens para minerar padrões sequenciais. Antes de entrar em detalhes sobre estes, são exibidas, a seguir, algumas definições formais no contexto de SPM.

Definição 2.6.1 (Sequência)**.** Seja I um conjunto de itens. Uma **sequência** s é uma lista ordenada de *itemsets* e é denotada por $s =< s_1 s_2 ... s_n >$ onde cada s_i é um *itemset*, ou seja, $s_i \subseteq I (1 \leq i \leq n)$. Alguns autores definem que, o **comprimento** de uma sequência s, corresponde ao número total de itens nessa sequência, isto é, $\sum_{i=1}^{n} \mid s_i \mid$, onde $\mid s_i \mid$ denota o número de itens do i-ésimo *itemset* de s. Uma sequência de comprimento k é referenciada como k-sequência [Srikant & Agrawal, 1996]. Além disso, em alguns trabalhos os autores definem o tamanho da sequência como a quantidade de *itemsets* presentes na sequência [Agrawal & Srikant, 1995]. Neste trabalho, o tamanho ou comprimento da sequência refere-se a quantidade de *itemsets* existentes na sequência.

Exemplo 2.6.1. Seja $I = \{a, b, c, d, e\}$ um conjunto de itens e $s =< (a, b)(a)(b, c, d) >$. A sequência s possui 3 *itemsets*, possui comprimento 3 e é composta por 4 itens distintos. Um item pode pertencer a mais de um *itemset* e um *itemset* pode ser composto por um ou mais itens. Por exemplo, cada *itemset* pode representar um conjunto de produtos comprados juntos, por exemplo. Desta forma, s representa que os itens a

e b foram comprados juntos, depois somente o item a foi comprado e por fim, foram comprados os itens b, c, e d.

Definição 2.6.2 (Subsequência)**.** Uma sequência $s_a =< A_1, A_2, ..., A_n >$ é dita **subsequência** de $s_b =< B_1, B_2, ..., B_m >$ se e somente se existem inteiros $1 \leq i_1 \leq i_2 < ... < i_n, n \leq m$, tal que, $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{in}$, denotado por $s_a \sqsubseteq s_b$. Também é dito que s_b é uma **supersequência** de s_a e que s_a está contida em s_b ou s_b contém s_a.

Exemplo 2.6.2. As sequências $s_1 =< (b)(a)(b, d) >$ e $s_2 =< (b)(c) >$ são exemplos de subsequências de $s =< (a, b)(a)(b, c, d) >$. Pois todos os *itemset* de s_1 estão contidos nos itemsets de s seguindo a mesma ordem. Entretanto, $s_3 =< (b)(b)(c) >$ e $s_4 =< (a, b, c) >$ não são. Para a sequência s_3 o segundo *itemset* da sequência não está contido no segundo *itemset* de s. A sequência s_4 é composta por apenas um *itemset* e este não está contido em nenhum *itemset* de s.

Definição 2.6.3 (Sequência maximal)**.** Dado um conjunto de sequências C, é dito que a sequência $s \in C$ é **maximal** se não há nenhuma sequência em $C - \{s\}$ contendo s.

Exemplo 2.6.3. Para o conjunto de sequências $A = \{< (a)(b, c)(d) >, < (a, e)(b) >, < (a)(c)(d) >, < (a)(b) >\}$, a sequência $< (a, e)(b) >$ é maximal, pois nenhuma outra no conjunto a contém. Entretanto, a sequência $< (a)(c)(d) >$ não é maximal, pois está contida na sequência $< (a)(b, c)(d) >$.

Definição 2.6.4 (Banco de dados de sequências)**.** Um **banco de dados de sequências** D é um conjunto de tuplas $< sid, s >$ onde s é uma sequência e sid é o identificador da sequência (*Sequence ID*) s. É dito que uma tupla $< sid, s >$ em um banco de dados de sequências D contém uma sequência s_a se s_a é subsequência de s, ou seja, $s_a \sqsubseteq s$.

Exemplo 2.6.4. A Tabela 2.1 exibe um exemplo de um banco de dados de sequências com quatro tuplas.

Tabela 2.1: Representação de um banco de dados de sequência.

SID	**Sequência**
1	$< (a, b)(c)(f, g)(g)(e) >$
2	$< (a, d)(c)(b)(a, b, e, f) >$
3	$< (a)(b)(f, e) >$
4	$< (b)(f, g) >$

Definição 2.6.5 (Suporte)**.** O **suporte absoluto** de uma sequência s_a em um banco de dados de sequência $\mathcal{D}$ é definido como o número de tuplas que contêm s_a e é denotado por $supAbs(s_a)$, isto é, $supAbs(s_a) = |\{s|s \in \mathcal{D} \wedge s_a \sqsubseteq s\}|$. Alguns autores, ao se referir ao **suporte**, *sup*, fazem menção ao suporte absoluto. E para se referir ao suporte relativo o fazem explicitamente, *supRel*. Neste trabalho, **suporte** (ou suporte relativo), corresponde a porcentagem de tuplas em $\mathcal{D}$ que contém s_a, denotado por $sup(s_a) = \frac{|\{s|s \in \mathcal{D} \wedge s_a \sqsubseteq s\}|}{|\mathcal{D}|}$.

Exemplo 2.6.5. A sequência $< (b)(f,g) >$, considerando a base de dados da Tabela 2.1 possui suporte absoluto igual a 2, pois está contida nas sequências 1 e 4. O suporte relativo é 0,5, ou 50%, pois ocorrem $2/4$ vezes na base de dados.

Definição 2.6.6 (Sequência frequente)**.** Dado um banco de dados $\mathcal{D}$ e um limite mínimo de suporte, *minsup*, uma sequência s_a é chamada de **sequência frequente** (ou **padrão sequencial**) se e somente se a porcentagem das tuplas em $\mathcal{D}$ que contém s_a é maior que *minsup*, ou seja, $sup(s_a) \geq minsup$.

Exemplo 2.6.6. A sequência $s_a =< (b)(f,g) >$ possui suporte igual a 0,5 (exemplo 2.6.5). Caso $minsup \leq 0,5$, então s_a é frequente.

O problema de minerar padrões sequenciais pode ser definido como, "dado um banco de dados de sequências $\mathcal{D}$ e um suporte mínimo, *minsup*, encontre todas as sequências frequentes com relação a $\mathcal{D}$ e *minsup*". Este é um problema de enumeração, que consiste em enumerar todas as sequências e selecionar as que possuem suporte superior ao limite estabelecido. Resolver este problema pode ser muito complexo, uma abordagem força bruta seria gerar todas as possíveis subsequências e calcular o suporte de cada uma delas. Entretanto, é ineficiente, pois o número de sequências pode ser grande, uma sequência com q itens em uma base de dados pode ter $2^q - 1$ subsequências distintas. Logo, não é escalável e é uma abordagem inviável para as aplicações do dia a dia.

Diversos algoritmos têm sido desenvolvidos para minerar padrões sequenciais de forma mais eficiente. Os mais populares são, AprioriAll [Agrawal & Srikant, 1995], GSP (*Generalized Sequential Patterns*) [Srikant & Agrawal, 1996], PSP [Masseglia et al., 1998], FreeSpan (*Frequent pattern-projected Sequential pattern mining*) [Han et al., 2000], SPADE (*Sequential PAttern Discovery using Equivalence classes*) [Zaki, 2001], SPAM (*Sequential PAttern Mining*) [Ayres et al., 2002], PrefixSpan (*Prefix-projected Sequential pattern mining*) [Pei et al., 2004], LAPIN-SPAM (*LAst Position INduction sequential pattern mining*) [Yang & Kitsuregawa, 2005], CM-SPAM [Fournier-Viger et al., 2014a], CM-SPADE [Fournier-Viger et al., 2014a], entre outros. Estes algoritmos

sempre retornam o mesmo conjunto de sequencias frequentes para uma mesma base de dados $\mathcal{D}$ e suporte mínimo, *minsup*. Ou seja, a diferença entre eles não é a saída, mas sim a forma como cada um encontra os padrões. Cada um destes algoritmos utiliza uma estratégia ou estrutura de dados diferente do outro. Portanto, em determinados casos, a eficiência de um pode superar a do outro, em termos de uso de memória ou tempo de processamento.

Além dos algoritmos citados, também existem algoritmos que encontram padrões fechados, como o CloSpan (*Closed Sequential pattern mining*) [Yan et al., 2003], ou sequências maximal como o VMSP (*Vertical mining of Maximal Sequential Patterns*) [Fournier-Viger et al., 2014b] e também existem os que encontram os padrões sequenciais negativos, como o F-NSP+ (*Fast Negative Sequential Patterns mining*) [Dong et al., 2018]. Conforme apresentado, embora a saída dos algoritmos seja o mesmo conjunto de subsequências, a forma como cada um funciona é diferente. Nas seções a seguir é exibida a forma como alguns destes algoritmos funcionam.

2.7 Algoritmo AprioriAll

O algoritmo AprioriAll foi proposto por Agrawal & Srikant [1995] e surgiu como um complemento do Apriori, visando atender a necessidade de manter a sequência dos dados. Este algoritmo busca encontrar sequências de itens que ocorrem com uma determinada frequência na base de dados. Neste contexto, a base de dados necessita de uma nova informação, que pode ser uma data ou algum identificador temporal para as transações. O AprioriAll identifica sequências de itens que podem ser representadas como $s =< A, B, C >$ onde A, B e C são *itemsets* frequentes que ocorrem segundo a ordem apresentada. Este algoritmo pode ser resumido em três etapas principais, que são, *i*) encontrar *itemsets* frequentes; *ii*) transformação; *iii*) buscar sequências frequentes. Cada uma destas etapas é detalhada a seguir.

Tabela 2.2: Um banco de dados de sequências $\mathcal{D}$

SID	Sequência
1	$< (3)(9) >$
2	$< (1,2)(3)(4,6,7) >$
3	$< (3,5,7) >$
4	$< (3)(4,7)(9) >$
5	$< (9) >$

- **Encontrar *itemsets* frequentes**: A primeira etapa consiste em encontrar os *itemsets* frequentes dado um suporte mínimo, *minsup*. Para encontrar tais pa-

drões, é utilizado o algoritmo apriori (algoritmo 1), apresentado na Seção 2.5.1 com algumas alterações que são descritas mais adiante.

A Tabela 2.2 exibe uma base de dados de sequências como exemplo. Após encontrar os *itemsets* frequentes, considerando $minsup = 0,4$, os suportes dos *itemsets* mais frequentes são contabilizados conforme a Tabela 2.3 e mapeados para um valor inteiro. Este mapeamento é realizado para agilizar as comparações entre duas sequências. Comparar dois inteiros leva um tempo constante e é bem mais rápido do que comparar se um *itemset* i_a está contido em um *itemset* i_b.

Tabela 2.3: Mapeamento de *itemsets* frequentes.

Itemsets	Suporte	Mapeado para
(3)	0,8	1
(4)	0,4	2
(7)	0,6	3
(4, 7)	0,4	4
(9)	0,6	5

- **Transformação:** Nesta etapa, para cada sequência s em $\mathcal{D}$, cada *itemset* i contido em s é substituído pelo conjunto dos *itemsets* frequentes contidos neste *itemset* i. A partir disso é criada uma nova base, $\mathcal{D}_T$, contendo apenas as sequências que possuem *itemsets* frequentes. Se uma transação (*itemset*) não possui nenhum *itemset* frequente, então, esta transação é removida. Assim como, se uma sequência não contém nenhum *itemset* frequente, é removida da base de dados (mas continua contando para o cálculo da quantidade total de sequências na base). A Tabela 2.4 exibe os resultados após a transformação e mapeamento.

Tabela 2.4: Transformação da base de dados.

SID	Original	Tranformada	Mapeada
1	$< (3)(9) >$	$< \{(3)\}\{(9)\} >$	$< \{1\}\{5\} >$
2	$< (1,2)(3)(4,6,7) >$	$< \{(3)\}\{(4),(7),(4,7)\} >$	$< \{1\}\{2,3,4\}$
3	$< (3,5,7) >$	$< \{(3)\}\{(7)\} >$	$< \{1,3\} >$
4	$< (3)(4,7)(9) >$	$< \{(3)\}\{(4),(7),(4,7)\}\{(9)\} >$	$< \{1\}\{2,3,4\}\{5\} >$
5	$< (9) >$	$< \{(9)\} >$	$< \{5\} >$

- **Buscar sequências frequentes:** Esta etapa funciona de forma muito similar ao algoritmo Apriori, exatamente a mesma ideia. No entanto, o que seria *itemset* no apriori é uma sequência no aprioriAll e o que seria um item é um *itemset*. Além disso, a função *apriori-generate*, que é responsável por gerar os candidatos sofre algumas alterações. Por exemplo, para o apriori, dado que $L_2 = \{1,3\};\{1,4\}$,

então $C_3 = \{1, 3, 4\}$. Entretanto, para o aprioriAll, $C_3 = \{1, 3, 4\}; \{1, 4, 3\}$, pois a ordem sequencial precisa ser contabilizada para ambos. O Algoritmo 2 exibe a função modificada para encontrar as sequências candidatas. As etapas de poda e cálculo do suporte também seguem a mesma ideia do apriori, com as particularidades de tratar os *itemsets* como sequência em ambas operações.

Algoritmo 2: apriori-generate

```
1 insert into C_k
2 select p.item_1, p.item_2, ..., p.item_{k-2}, p.item_{k-1}, q.item_{k-1}
3 from L_{k-1} p, L_{k-1} q
4 where p.item_1 = q.item_1, p.item_2 = q.item_2, ..., p.item_{k-2} = q.item_{k-2};
```

Este algoritmo é capaz de encontrar os padrões sequenciais e serviu como base para criação de vários outros algoritmos de mineração de padrões sequenciais. Os mesmos autores do aprioriAll desenvolveram posteriormente o GSP, que é até 20 vezes mais eficiente e é descrito na seção a seguir.

2.8 Algoritmo GSP

O algoritmo GSP (*Generalized Sequential Patterns*) [Srikant & Agrawal, 1996] foi desenvolvido pelos mesmos autores do AprioriAll e segue a mesma ideia dos algoritmos baseados na propriedade apriori. No entanto, conta com algumas melhorias que fazem o método ser de até 20 vezes mais rápido. Cada iteração é composta por três etapas principais, que são, *i*) geração de candidatos; *ii*) poda; *iii*) cálculo do suporte. Cada etapa é descrita em mais detalhes nas seções adiante.

Uma das diferenças entre o AprioriAll e o GSP é o conceito de k-sequência (definição 2.6.1). Para o AprioriAll, uma k-sequência é uma sequência com k *itemsets*. Porém, no GSP, uma k-sequência é uma sequência com k *itens*. Se um item aparece repetidas vezes em *itemsets* distintos, este item é contado uma vez para cada *itemset* no qual ele está contido. Por exemplo, $< \{1, 2\} >, < \{1\}, \{2\} >, < \{1\}, \{1\} >$, todas as três sequências são consideradas 2-sequências, ou seja, de comprimento 2.

No GSP, à iteração k os conjuntos L_k e C_k são constituídos de sequências de k *itens*. Diferente do AprioriAll, que em cada iteração k os conjuntos L_k e C_k são constituídos de sequências de k *itemsets*. A Figura 2.4 demonstra como os possíveis candidatos são gerados a cada iteração em ambos os algoritmos.

	AprioriAll	GSP
Interação 1	<{*a*}> <{*b*}> <{*c*}>	<{*a*}> <{*b*}> <{*c*}>
	1 itemset	1 item
Interação 2	<{*a, b*}> <{*a, c*}> <{*b, c*}>	<{*a, b*}> <{*a*}, {*b*}> <{*a, c*}> <{*a*}, {*c*}> <{*b, c*}> <{*b*}, {*c*}>
	2 itemsets	2 itens
Interação 3	<{*a, b, c*}>	<{*a, b, c*}> <{*a*}, {*b, c*}> <{*a, b*}, {*c*}> <{*a*}, {*b*}, {*c*}>
	3 itemsets	3 itens

Figura 2.4: Diferença entre GPS e AprioriAll para geração de candidatos. Adaptado de De Amo [2004]

2.8.1 Geração de Candidatos

Na **iteração 1**, o processo de geração de candidatos, cálculo do suporte e poda é muito similar ao do AprioriAll. A Figura 2.5 exibe um exemplo de como pode ser gerado os candidatos para uma base de sequência. Para este exemplo é considerado que o suporte mínimo é 0,50, ou seja, o itemset precisa ocorrer em no mínimo 50% das sequências da base de dados. Nas próximas iterações o processo é bastante diferente e é possível notar nitidamente as disparidades entre GSP e AprioriAll.

ID	Sequências
1	<{2, 1, 4, 6}, {3, 7, 8}, {5}, {9}>
2	<{2, 4, 5}, {1, 6, 7}, {3}>
3	<{2, 4}, {1, 5}>
4	<{1, 3}, {2, 4, 5}>

Itemset	Suporte	
{1}	4	
{2}	4	
{3}	3	
{4}	4	
{5}	4	
{6}	2	
{7}	2	suporte >= 50%
{8}	1	✗
{9}	1	✗

Figura 2.5: Iteração 1 do algoritmo GSP - Geração e poda de candidatos

Na **iteração 2**, para cada itemset frequente em L_1 são geradas todas as 2-sequências possíveis combinando $L_1 X L_1$. Dado que $s_1 \subset L_1$ e $s_2 \subset L_1$, para combinar duas sequências s_1=<{x}> e s_2=<{y}> de um item para gerar uma de dois itens é necessário adicionar o item y de s_2 em s_1 tanto como parte do *itemset* {x} quanto como um *itemset* isolado. Assim a combinação de s_1 com s_2 produz duas sequências

de 2 elementos : <{x, y}> e <{x}, {y}>. A Figura 2.6 exibe como estas sequências são construídas.

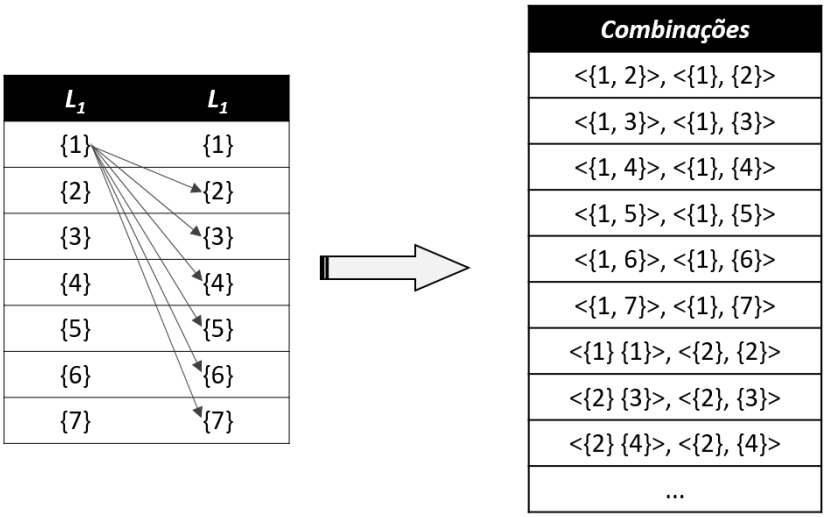


Figura 2.6: Iteração 2 do algoritmo GSP - Geração de candidatos

A partir da **iteração 3**, a geração de candidatos segue uma abordagem bem diferente do AprioriAll. Nesta fase, um conceito importante introduzido é o de sequências *ligáveis*. Dado que L_{k-1} foi gerado na etapa $k-1$. Duas sequências $s =< s1, s2, ..., s_n >$ e $t =< t1, t2, ..., t_m >$ contidas em L_{k-1} são ditas *ligáveis* se, ao retirar o primeiro item de s_1 e o último item de t_m, as sequências resultantes são iguais. Caso seja possível, s e t são ligadas para produzir a sequência v. Onde, se t_m é unitário então $v =< s_1, s_2, ..., s_n, t_m >$, senão, $v =< s_1, s_2, ..., s_n \cup t' >$, onde t' é o último item de t_m. O exemplo a seguir ilustra como pode ser feita a geração de v.

Exemplo 2.8.1. Seja $s =< \{1,2\}, \{3\}, \{5,7\} >$, e $t =< \{2\}, \{3\}, \{5,7,10\} >$. Retirando-se o primeiro item de s_1 (o item 1) e o último item de t_3 (o item 10) é obtida a mesma sequência: $< \{2\}, \{3\}, \{5,7\} >$. Portanto, são *ligáveis* e sua junção produz a sequência $v =< \{1,2\}, \{3\}, \{5,7,10\}$.

2.8.2 Poda dos Candidatos

Pela propriedade Apriori, se uma sequência s é frequente, então toda subsequência de s deve ser frequente. Logo se alguma subsequência de s não estiver em $L_k - 1$ então a sequência s não tem nenhuma chance de ser frequente.

A Figura 2.7 exibe como funciona a etapa de junção e poda de candidatos C_4. A partir de L_3 são geradas as combinações das sequência ligáveis. Para o conjunto L_3 foi possível gerar duas combinações. No entanto, a sequência $< \{1,2\}, \{3\}, \{5\} >$

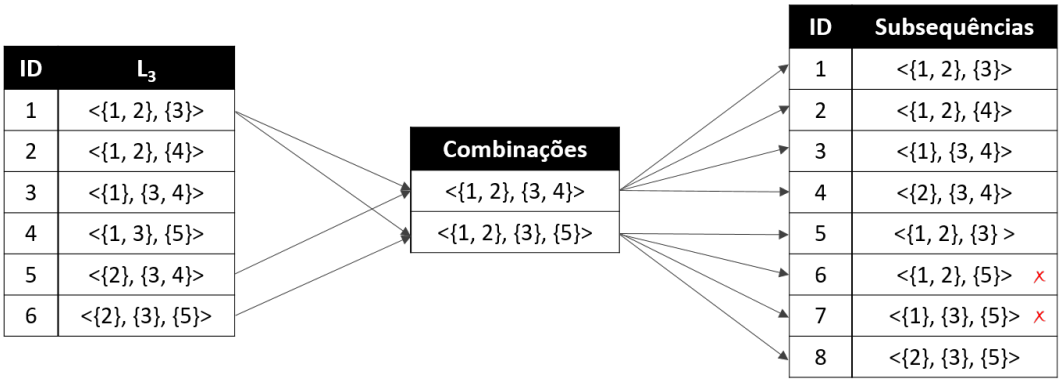


Figura 2.7: Ligação e poda de sequências.

é podada, pois duas de suas subsequências (6 e 7) não são frequentes. Logo, $C_4 =< \{1,2\},\{3,4\} >$.

Após obter o conjunto C_k, as sequências candidatas são adicionadas em uma árvore *hash*. Uma árvore hash é uma árvore onde as folhas armazenam conjuntos de padrões sequenciais e os nós intermediários armazenam tabelas hash com pares *número, ponteiro*. Este tipo de estrutura permite uma poda mais eficiente de elementos a cada iteração. Para contruir esta árvore é necessário definir duas variáveis, M e N. M corresponde ao número de sequências que cabem em um mesmo nó. N é o número de filhos que cada nó pode assumir.

Tabela 2.5: Sequências candidatas de 2 itens.

SID	**Sequência**
1	$< \{1,3\} >$
2	$< \{1\},\{3\} >$
3	$< \{2\},\{3\} >$
4	$< \{3\},\{3\} >$
5	$< \{2,3\} >$
6	$< \{1\},\{4\} >$

A Tabela 2.5 exibe um exemplo de sequências candidatas C_2 com quatro itens distintos. A Figura 2.8 exibe como funciona a inserção das sequências na árvore hash. Para este exemplo foi considerado $M = 3$ e $N = 2$. A função hash utilizada retorna 1, caso o valor seja impar, e 2, caso o valor for par. Por exemplo, $h(1) = 1$, $h(2) = 2$, $h(3) = 1$, $h(4) = 2$. No passo 1, são inseridas as sequências $< \{1,3\} >$, $< \{1\},\{3\} >$, $< \{2\},\{3\} >$ na raiz. Ao inserir a sequência $< \{3\},\{3\} >$, o limite de M é estourado. Logo, o nó é quebrado em dois. As sequências que possuem $h(i_1) = 1$ (primeiro item da sequência é ímpar) vão para o nó esquerdo, e as que possuem $h(i_1) = 2$ vão para o nó direito.

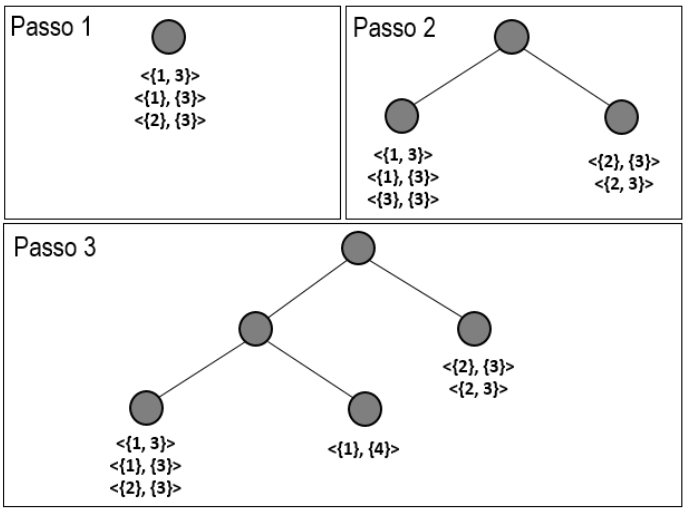


Figura 2.8: Árvore hash das sequências candidatas

No passo 2 é inserida a sequência $< \{2, 3\} >$ no nó que sai da raiz para a direita, pois $h(2) = 2$. No terceiro e último passo, tenta-se inserir $< \{1\}, \{4\} >$, no entanto o valor máximo de sequências M é estourado novamente, portanto o nó é dividido em dois. Para o nó à esquerda vão as sequências que possuem o hash do segundo item igual a 1, $h(i_2) = 1$ e para a direita os que resultarem em $h(i_2) = 2$. Ao final desta fase é executada a contagem do suporte, para obter os itens mais frequentes L_k.

2.8.3 Contagem do Suporte

Para contagem do suporte, é necessário varrer a base de sequências D. Para cada sequência s lida, todos os candidatos em C_k contidos em s têm seu suporte incrementado. Testar se cada candidato está incluído em s para todas as sequências em D é uma tarefa relativamente trabalhosa. Portanto algumas otimizações são feitas para tornar a contagem menos exaustiva.

Uma das otimizações é a utilização da árvore hash para reduzir o número de candidatos que serão contados. Para cada sequência s a árvore é percorrida seguindo a lógica:

- Calcula a função h para o item i e dirige-se para o nó correspondente.
- Caso o nó seja um nó folha é aplicado o procedimento INCLUDE para cada sequência da folha.
- Caso o nó não seja um nó folha, calcula-se o valor da função h para o próximo item $i + 1$ até encontrar um nó folha ou enquanto $i <= m$, onde m é o último

item de s.

O procedimento INCLUDE verifica apenas se um candidato $c \subset C_k$ está contido sequência s. Para isso, é criado um array com o tamanho igual à quantidade de itens distintos presentes na base de sequências. Cada transação (*itemset*) pertence a um tempo t de forma ordenada. Para cada item, é armazenado o tempo no qual este item ocorre na sequencia. A Tabela 2.6 exibe como é feita a representação da sequência $s =< \{1,2\},\{4,6\},\{3\},\{1,2\},\{3\},\{2,4\},\{6,7\} >$. Ao utilizar a Tabela 2.6b é possível saber os tempos (ou transações) na qual cada item ocorre. Desta forma, é possível identificar de forma mais eficiente se uma sequência candidata de C_k está contida em s.

Tabela 2.6: Indexação da transação na qual cada item ocorre

(a) Itens por transação

Tempo	**Itens**
1	1, 2
2	4, 6
3	3
4	1, 2
5	3
6	2, 4
7	6, 7

(b) Tempo em que cada item ocorre

Item	**Tempo/transação**
1	[1, 4]
2	[1, 4, 6]
3	[3, 5]
4	[2, 6]
5	[ ]
6	[2, 7]
7	[7]

O procedimento INCLUDE retorna sim caso uma sequência s_1 está contida em outra sequência s_2. Esta verificação é feita em cada sequência da base de sequências para cada candidata selecionada. A Figura 2.9 ilustra o passo a passo e o resultado obtido utilizando as seguintes regras para testar se um candidato $c = c_1, c_2, ..., c_n$ é suportado por uma sequência s:

- Percorre-se os itens de c_1 e encontra-se o primeiro tempo de cada item nas respectivas listas. Se este tempo for o mesmo para cada item, este será o tempo t_1 da primeira ocorrência de c_1 em s e o processo termina. Senão, o processo é repetido, mas agora tentando encontrar o primeiro tempo maior que o máximo dos primeiros tempos de cada item em c_1.

- Encontra-se a primeira ocorrência de s_2 após o tempo t_1 encontrado no item anterior. Se existir essa ocorrência, esta corresponde a um tempo $t_2 > t_1$. Repete-se o mesmo processo para todos os itens em c.

- Interrompe-se o processo e retorna como uma sequência não suportada quando um dos itens de c possuir a sua lista de tempos vazia. Ou quando um *itemset* $c_i + 1$ no tempo $t_i + 1$ não ocorrer após s_i no tempo t_i

A Figura 2.9 utiliza a sequência s da Tabela 2.6b para verificar se a sequência candidata, $c =< \{2,4\},\{2,4\} >$, é suportada. O primeiro passo é encontrar para a primeira transação ($\{2,4\}$) os tempos no qual cada item ocorre em s. Para o item 2 os tempos são $[1,4,6]$, e para o item 4 são $[2,6]$. O próximo passo é saber se estes itens estão na mesma transação. Para isso, é verificado se os primeiros itens de cada lista são iguais. Neste caso, como não são $(1 \neq 2)$ compara-se o próximo item da lista e repete-se o processo até encontrar o momento no qual os tempos dos itens são iguais ou até o fim da lista. Neste exemplo é verificado que os itens ocorrem no tempo 6. O mesmo se repete para o segundo *itemset* ($\{6,7\}$). Como todos os *itemsets* estão na sequência na mesma ordem $t_1 < t_2 < ... < t_n$, o candidato tem seu suporte incrementado.

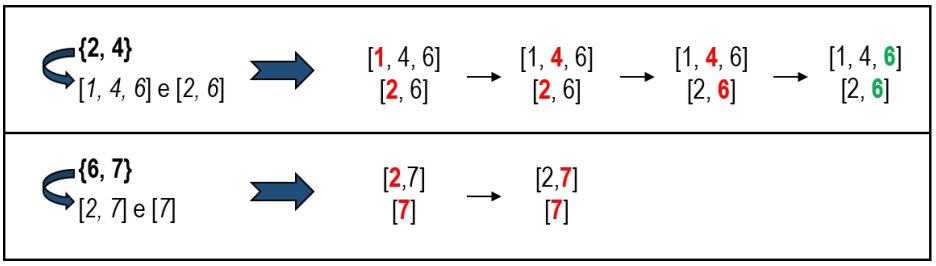


Figura 2.9: Verificação dos candidatos em relação a uma sequência $s \subset D$

2.9 Algoritmo SPAM

O algoritmo SPAM (**S**equential **PA**ttern **M**ining) [Ayres et al., 2002] é mais um dos algoritmos que se baseiam na propriedade Apriori, assim como AprioriAll e GSP. Um dos diferenciais deste algoritmo é que ele realiza uma busca em profundidade para encontrar os padrões frequentes. Outro diferencial é a geração e poda de candidatos, que foi desenvolvida baseando-se em uma árvore lexicográfica. A principal caraterística que distingue o SPAM dos demais algoritmos é a representação da base de sequências, chamada de mapeamento vertical de *bits*.

Tais aspectos fazem com que este algoritmo seja bem mais estável e se adapte melhor, mesmo ao utilizar uma alta quantidade de dados. As seções a seguir exibem o funcionamento básico deste método em mais detalhes. Este método serve como base para o algoritmo utilizado para minerar padrões neste trabalho. As alterações e

melhorias realizadas neste método são descritas no capítulo 4. As seções a descrevem os principais componentes do algoritmo.

2.9.1 Árvore Lexicográfica

O raciocínio formal de como pode ser criada uma árvore de sequência lexicográfica é o seguinte: dado que existe uma ordem lexicográfica nos *itemsets* das sequências, de forma que se o item i ocorre antes de j, denota-se que $I_i \leq I_j$, então esta ordem também pode ser aplicada para sequências definidas como $s_a \leq s_b$ se $s_a \sqsubseteq s_b$. A árvore segue a seguinte estrutura: a raiz é marcada como $\emptyset$; recursivamente, se n é um nó da árvore, então os $n's$ filhos são todos nós n', tal que $n \leq n'$ e $\forall m \in T : n' \leq m \Rightarrow n \leq m$, onde m é o número de transações (*itemsets*) em uma sequência [Ayres et al., 2002].

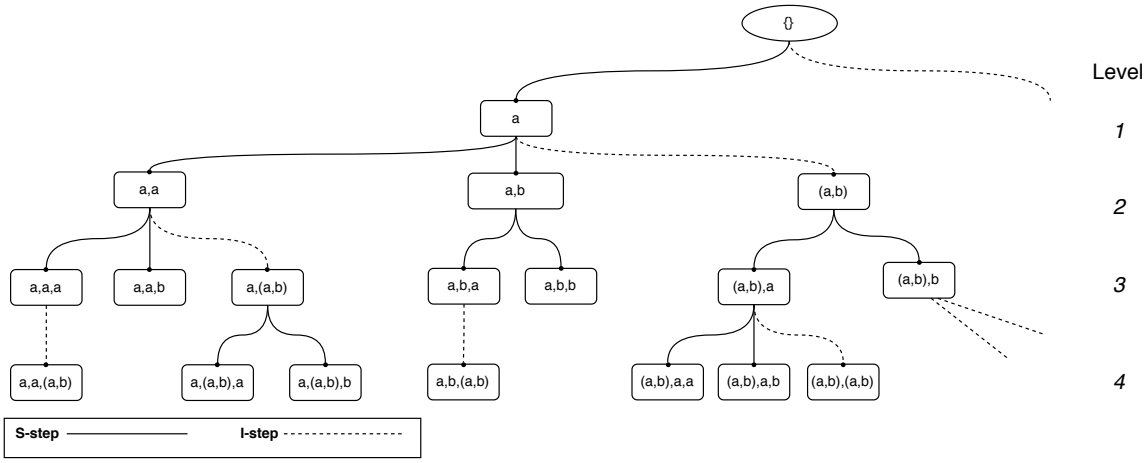


Figura 2.10: Árvore lexicográfica. (Adaptado de Ayres et al. [2002])

Cada sequência na árvore de sequências pode ser considerada como uma extensão de sequência ou como uma extensão de *itemset*. Uma sequência estendida é uma sequência gerada pela adição de uma nova transação, composta por um item, ao fim da sequência do nó pai. Por outro lado, uma extensão de *itemset* é uma sequência gerada pela adição de um item no último *itemset* da sequência do nó pai. A Figura 2.10 exibe como são formados extensões de sequências e *itemsets*. Por exemplo, dado a sequência "$\{a\}\{a\}$", as sequências "$\{a\}\{a\}\{a\}$" e "$\{a\}\{a\}\{b\}$" são duas extensões de sequência enquanto "$\{a\}\{(a,b)\}$" é uma extensão de *itemset*. O passo de estender uma sequência é chamado de S-step (*sequence-extension step*) e o passo para estender um *itemset* é chamado de *I-step* (*itemset-extension step*).

2.9.2 Busca em Profundidade e Poda

O SPAM utiliza uma busca em profundidade em uma árvore lexicográfica similar à arvore apresentada na seção anterior. A cada nó visitado, avalia-se o suporte de cada um dos nós filhos. Caso o suporte do nó seja maior que o suporte mínimo estabelecido, então este nó é guardado e a busca prossegue para os próximos filhos recursivamente. Caso o nó esteja abaixo do suporte, a busca é interrompida para este nó. Pois, seguindo a propriedade apriori da antimonotonia, se o nó pai não é frequente, então os filhos não tem possibilidade de serem frequentes. Para evitar que a busca continue "infinitamente" é possível também limitar o tamanho máximo da sequência. Quando a sequência alcançar este valor, o nó com a sequência de tamanho máximo é considerado nó folha.

Algoritmo 3: BUSCA

Entrada: pad, S_n, I_n $minsup$

início
  $S_{temp} = \emptyset$
  $I_{temp} = \emptyset$
  para *cada item* $i \in S_n$ **faça**
    se $ehFrequente((pad_1, ..., pad_k, \{i\}))$ **então**
      $S_{temp} = S_{temp} \cup \{i\}$
    fim
  fim
  para *cada item* $i \in S_{temp}$ **faça**
    BUSCA ($(pad_1, ..., pad_k, \{i\})$, S_{temp}, $\{e \in S_{temp} | e \succ j\}$, $minsup$)
  fim
  para *cada item* $i \in I_n$ **faça**
    se $ehFrequente((pad_1, ..., pad_k \cup \{i\}))$ **então**
      $I_{temp} = I_{temp} \cup \{i\}$
    fim
  fim
  para *cada item* $i \in I_{temp}$ **faça**
    BUSCA ($(pad_1, ..., pad_k \cup \{i\})$, S_{temp}, $\{e \in I_{temp} | e \succ j\}$, $minsup$)
  fim
fim

É notável que tal abordagem gera quantidade de candidatos extremamente alta, tal fato pode fazer com que o algoritmo leve mais tempo para encontrar todos os padrões frequentes. Uma das formas de evitar que tantos candidatos sejam testados é realizar uma poda mais eficiente, para que os nós gerados a partir de um *S-step* ou *I-step*, sem possibilidade de serem frequentes pelo principio da antimonotonia, sejam podados.

Para podar os filhos gerados a partir de um *S-step*, é utilizada a seguinte técnica. Dado uma sequência s em um nó n de forma que $s_a = (s, \{i_j\})$ e $s_b = (s, \{i_k\})$ são sequências estendidas de s. Seja s_a uma sequência frequente e seja s_b uma sequência não frequente, pelo principio Apriori nem $(s, \{i_j\}, \{i_k\})$ e $(s, \{i_j, i_k\})$ não podem ser frequentes, pois ambos possuem a subsequencia s_b. Portanto, i_k pode ser removido tanto nos *S-step's* quanto nos *I-step's* de s conforme exibido no exemplo a seguir.

Exemplo 2.9.1. A partir do nó "$\{a\}$" na árvore pode ser obtido $S(\{a\}) = \{a, b, c, d\}$ e $I(\{a\}) = \{b, c, d\}$. As possíveis sequências a partir de um *S-step* são, $(\{a\}, \{a\})$, $(\{a\}, \{b\})$, $(\{a\}, \{c\})$ e $(\{a\}, \{d\})$. Dado que $(\{a\}, \{c\})$ e $(\{a\}, \{d\})$ não são frequentes, baseado no princípio Apriori, as sequências candidatas $(\{a\}, \{a\}, \{c\})$, $(\{a\}, \{b\}, \{c\})$, $(\{a\}, \{a, c\})$, $(\{a\}, \{b, c\})$, $(\{a\}, \{a\}, \{d\})$, $(\{a\}, \{b\}, \{d\})$, $(\{a\}, \{a, d\})$, e $(\{a\}, \{b, d\})$ também não podem ser frequentes. Portanto, a partir dos nós ({a}, {a}) ou ({a}, {b}), não é necessário gerar as sequências que contenham os itens c e d no próximo *S-step* e *I-step*.

Outra técnica seguindo o mesmo raciocínio é utilizada para podar sequências geradas a partir do *I-step*. Dado uma sequência $s = (s', \{i_1, ..., i_n\})$, desta forma, $s_a = (s', \{i_1, ..., i_n, i_j\})$ e $s_b = (s', \{i_1, ..., i_n, i_k\})$ onde $i_j < i_k$. Se s_a é frequente e s_b não é frequente, então baseando no princípio Apriori pode-se dizer que $(s', \{i_1, ..., i_n, i_j, i_k\})$ pode ser frequente. Portanto i_k pode ser removido de todas as extensões de *itemsets* de s. O algoritmo 3 exibe como funciona a busca podando os elementos que não podem ser frequêntes. O exemplo a seguir demonstra como é realizada a poda nas extensões de *itemsets*.

Exemplo 2.9.2. De forma similar ao exemplo anterior, a partir do nó $(\{a\})$ é possível obter as extensões de *itemset* $(\{a, b\})$, $(\{a, c\})$ e $(\{a, d\})$. Se $(\{a, c\})$ não é frequente então sabe-se de antemão que $(\{a, b, c\})$ não é frequente.

2.9.3 Mapeamento Vertical de *Bits*

O principal diferencial do SPAM é a estrutura de dados utilizada para gerar candidatos e verificação de suporte. A estrutura utilizada é um mapa de *bits*, conforme exibido na Figura 2.11. Um mapa de *bits* vertical é criado para cada item da base de dados. Neste trabalho, cada item é uma atividade ou um tipo de dado de contexto. Cada "linha" horizontal do mapa representa uma transação composta por uma atividade e n dados de contexto (mais detalhes no capítulo 4). Se um item ocorre na transação, então tem valor igual a 1, caso contrário o valor é 0.

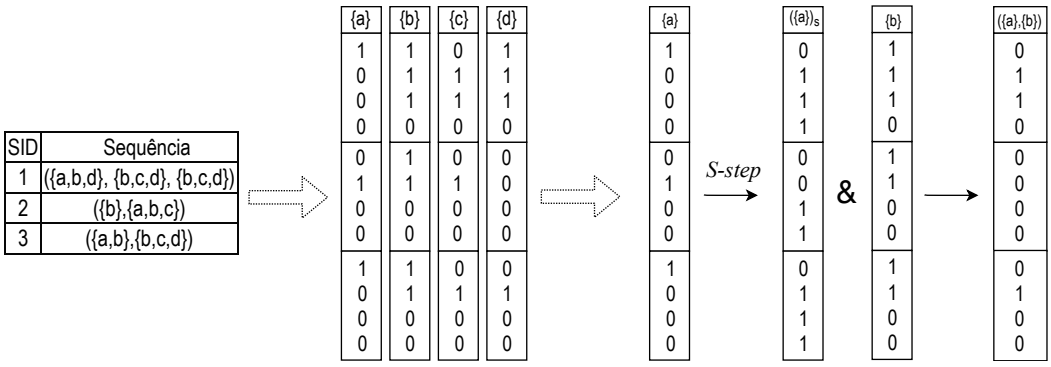


Figura 2.11: Representação em mapa de *bits*. (Adaptado de Ayres et al. [2002])

No primeiro passo apresentado na Figura 2.11, uma base de sequências é convertida em um mapa de *bits*. Tal representação torna muito mais eficiente a forma como comparar se um *itemset* está em uma transação, pois basta uma operação *AND* entre os itens envolvidos para determinar se o *itemset* está contido.

O SPAM também é eficiente na geração dos candidatos. Seja $B(s_a)$ e $B(i)$ dois mapas de *bits* para a sequência s_a e para o item i. O S-step adiciona um novo item ao final da sequência existente, ou seja, adiciona o item $\{i\}$ em s_a. O mapa de *bits* para a nova sequência, $B(s_g)$, deve ter a propriedade que se um bit tem valor 1, então a transação correspondente j deve conter i e todos os outros *itemsets* em s_g devem estar contidos nas transações antes de j. O terceiro passo da Figura 2.11 exibe um exemplo de como funciona um S-step. Seja $B((\{a\}))$ um mapa de *bits* de a, deseja-se gerar $B((\{a\}\{b\}))$. Para realizar tal operação foi selecionada a coluna correspondente ao item $\{a\}$ e a partir desta é criado a versão transformada, chamada de $\{(a)\}_s$. Devido ao primeiro bit de $\{a\}$ ser 1 todos os primeiros itens de cada transação em $\{(a)\}_s$ têm seu valor setado para 0, os demais tem seu valor invertido. Após a conversão basta uma operação AND entre os mapas, que o resultado vai ser o mapa de $(\{a, b\})$.

Este método foi testado e comparado com o SPADE e o PrefixSpan em uma base de dados sintética. O SPAM obteve resultado muito superior em bases grandes e com longas sequências. Os autores apontam que a única desvantagem em relação aos demais é que a quantidade de candidatos que é gerada ainda é maior, no entanto devido a estrutura de dados não ocupar muito espaço e existir memória suficiente nos dispositivos para tal operação, sua utilização torna-se possível. Entretanto, esta quantidade de dados utilizados ainda pode ser reduzida, melhorando a eficiência do método. No capítulo 4 é exibido como reduzir o número de candidatos gerados, tornando assim o método mais eficiente.

2.10 Modelos Estatísticos

Em termos simples, um problema de previsão de sequência pode ser caraterizado da seguinte forma: dado uma sequência de eventos $X = \{x_1, x_2, x_3, ..., x_i\}$, como determinar o evento x_{i+1}? Além dos métodos de mineração de padrões citados anteriormente existem diversas abordagens para classificar e resolver os problemas de previsão de sequência. As seções a seguir descrevem como tais abordagens funcionam. O métodos exibidos a seguir são utilizados em diversos trabalhos relacionados que são descritos no próximo capítulo.

2.10.1 Modelos Markovianos

Um processo markoviano é um tipo de processo estocástico. Um processo estocástico é definido como uma família de variáveis aleatórias $X(t)$, onde t é a variável de tempo, ou seja, $X(t)$ representa o estado em que o sistema se encontra no tempo t. Estes processos são utilizados para descrever o comportamento ao longo de um período de tempo onde a incerteza é significativa. Por exemplo $X(t)$ pode representar o nível de um estoque em um tempo t [Leite, 2008].

Um processo estocástico é dito Markoviano se a probabilidade condicional de qualquer evento futuro, dado qualquer evento passado e o estado presente, é independente do evento passado e depende apenas do estado presente. Ou seja, quando o estado futuro depende apenas do estado presente, ignorando os demais estados passados, a Equação 2.1 demonstra formalmente a definição. Este tipo de processo também é denominado como processo sem memória (do inglês, *memoryless process*) devido à sua característica de não levar em conta o passado [Kemeny et al., 1960].

$$P\{X(t_{k+1}) \leq x_{k+1} \mid X(t_k) = x_k, X(t_{k-1}) = x_{k-1}, ..., X(t_0) = x_0\} = P\{X(t_{k+1}) \leq x_{k+1} \mid X(t_k) = x_k\} \tag{2.1}$$

Um processo Markoviano é dito ser uma cadeia de Markov quando as variáveis aleatórias $X(T)$ estão definidas em um espaço de estados discreto. A Figura 2.12

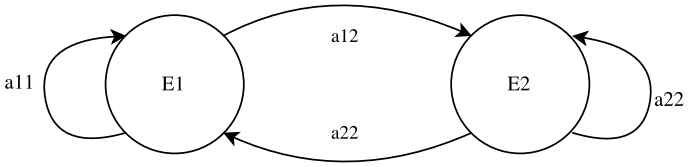


Figura 2.12: Cadeia de Markov com dois estados (E) e suas transições

exibe um exemplo de uma cadeia de Markov, E representa o estado e a_{ij} representa a probabilidade de transição do estado i para o estado j, conforme descrito na Equação 2.2. Para cada estado E o somatório das probabilidades das transições a_{ij} deve ser ser 1.

$$a_{ij} = P\{X(k+1) = j \mid X(k) = i\} \tag{2.2}$$

Com base nestes modelos, existem diversas técnicas que foram desenvolvidas para classificar e prever dados sequenciais, como por exemplo, o Modelo Oculto (ou escondido) de Markov (do inglês, *Hidden Markov Model - HMM*). Esta técnica é utilizada em diversas áreas, tais como, reconhecimento de voz [Lee et al., 1990; Nwe et al., 2003], reconhecimento de palavras manuscritas [Kundu et al., 1989; Chen et al., 1994], verificação on-line de assinatura [Yang et al., 1995], reconhecimento de ações humanas [Yang et al., 1997; San-Segundo et al., 2016], entre outros.

Diferentemente dos demais modelos de Markov, que levam em consideração apenas os eventos observáveis, o HMM é um processo duplamente estocástico, com um processo que não é observável (é oculto), mas pode ser observado por meio de outro processo estocástico que produz uma sequência de símbolos observáveis. A Figura 2.13 representa graficamente a estrutura do HMM, onde os nós sombreados representam os estados observáveis e os nós brancos representam os estados ocultos.

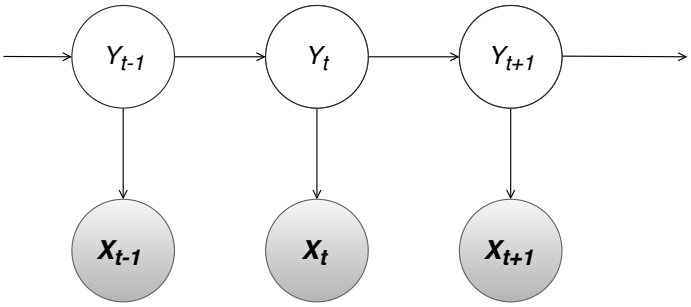


Figura 2.13: Representação gráfica do HMM. (Adaptado de Cook & Krishnan [2015])

Este modelo é popularmente utilizado para representar dados que podem ser caracterizados por um processo subjacente gerando uma sequência de observações. Um exemplo prático da aplicação do HMM para modelar dados é exibido na Figura 2.14. Neste exemplo, HMM é utilizado para modelar atividades, os estados ocultos são atividades, e os estados observáveis são os dados ou características extraídas de eventos de sensores. Dado uma sequência de dados extraídos dos sensores, não é possível determinar precisamente a atividade que produziu as observações. Isto acontece devido a múltiplas atividades gerarem eventos de sensores semelhantes.

Por exemplo, na Figura 2.14, o sensor I06 pode gerar um evento enquanto o morador prepara refeição, toma remédio ou assiste TV. Portanto, o rótulo da atividade não pode ser determinado utilizando apenas o evento I06. Entretanto, este evento disponibiliza um contexto melhor para reconhecer a atividade [Cook & Krishnan, 2015].

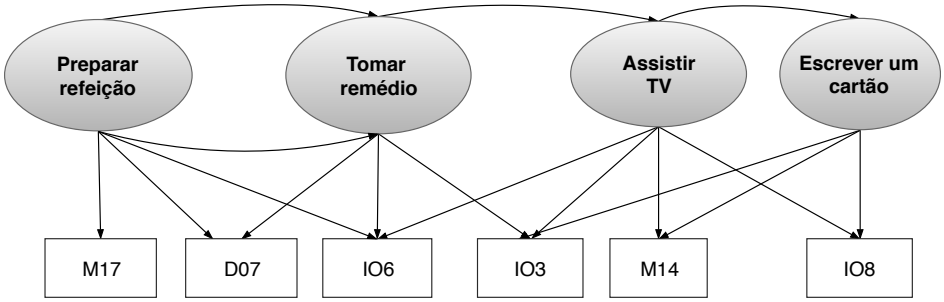


Figura 2.14: Exemplo de utilização do HMM para reconhecimento de atividades. (Adaptado de Cook & Krishnan [2015])

Antes de utilizar algum modelo markoviano para prever uma sequência, é necessário preparar os dados de entrada. Para o caso de prever uma atividade realizada por um usuário, cada atividade realizada por ele no passado pode ser considerada como um evento x_i e o evento a ser previsto (x_{i+1}) é a atividade que o usuário realizará.

$$A = \begin{Bmatrix} P(a|a) & P(l|a) & P(n|a) & P(s|a) \\ P(a|l) & P(l|l) & P(n|l) & P(s|l) \\ P(a|n) & P(l|n) & P(n|n) & P(s|n) \\ P(a|s) & P(l|s) & P(n|s) & P(s|s) \end{Bmatrix} = \begin{Bmatrix} 0.6 & 0.0 & 0.2 & 0.2 \\ 0.6 & 0.0 & 0.3 & 0.1 \\ 0.1 & 0.3 & 0.5 & 0.1 \\ 0.0 & 0.4 & 0.6 & 0.0 \end{Bmatrix} \tag{2.3}$$

Por exemplo, dada a seguinte sequência de atividade $X = \{..., l, a, a, a, a, s, n, n\}$, onde $a = andar$, $l = levantar$, $n = neutro$ e $s = sentar$ e a matriz de probabilidade de transição A representada na Equação 2.3, qual o próximo símbolo da sequência? (Neste exemplo, considera-se que é utilizado o modelo de Markov de ordem 1 para realizar a predição). Ou seja, qual a próxima ação do indivíduo dado que ele estava em estado neutro (ou parado)? Utilizando a matriz de probabilidade de transição A, é fácil ver que $P(n|n)$ é a maior, logo, dado que o indivíduo estava parado, a probabilidade de ele continuar parado é a mais alta, que é a saída da predição.

2.10.2 Previsão em Séries Temporais

Uma série temporal é um conjunto de observações medidas sequencialmente através do tempo [Chatfield, 2000]. Estas medições podem ser feitas continuamente (série

temporal contínua) através do tempo ou tomadas em um conjunto discreto de pontos de tempo (série temporal discreta). A partir da análise de uma série temporal é possível extrair conhecimento e realizar previsões utilizando diversos métodos. Um método de previsão é um procedimento para calcular valores futuros a partir de valores presentes e passados. Tal método pode identificar padrões e tendências, possibilitando desta forma, utilizar esse conhecimento para prever valores futuros na série.

Um dos métodos utilizados para previsão de séries temporais é o Média Móvel (*Moving Average* - MA). O MA é um método utilizado para suavização de dados, principalmente para dados de séries temporais que visam estimar a tendência dos dados [Hyndman, 2011]. Pode ser visto como uma janela de tempo que se move ao longo dos dados de entrada. Este método calcula uma média local dos quadros vizinhos relacionados à continuidade da sequência alvo. Seu valor é continuamente recalculado conforme novos dados são disponibilizados. Conforme avança na série, os valores mais antigos são descartados e o valor mais recente é adicionado. O limiar da janela pode ser entre curto e longo prazo dependendo da aplicação, de acordo com os parâmetros definidos.

Um modelo de média móvel é uma regressão linear do valor atual da série em relação à janela corrente e em termos do erro anteriores observados. O método visa reduzir o efeito das variações temporárias nos dados. Além disso, busca melhorar o "ajuste" dos dados com um processo chamado suavização. Desta forma é possível mostrar a tendência dos dados mais claramente e destacar qualquer valor acima ou abaixo da tendência. A equação 2.4 exibe uma exemplo de uma média móvel simples.

$$SMA = \frac{M_1 + M_2 + \cdots + M_{n-1}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} M_i \tag{2.4}$$

Além do MA outros métodos também realizam este mesmo processo de forma diferente. Como por exemplo, o ARIMA (*Autoregressive Integrating Moving Average*) que consiste de três componentes, os modelos $AR(p)$, $MA(q)$ e $ARMA(p, q)$, onde p é o número de termos autorregressivos e q é o número de erros de previsão [Zhang et al., 2018].

2.11 Considerações Finais

Este capítulo abordou as principais técnicas utilizadas para previsão sequencial. A utilização destas técnicas são demonstradas em alguns trabalhos relacionados exibidos a seguir. De acordo com o problema abordado, alguma técnica pode se sobressair em

relação a outra devido à natureza do problema em si. Além das técnicas exibidas, diversas técnicas de aprendizagem de máquina também são utilizadas para previsão. Um exemplo disso são as Redes Neurais Recorrentes (do inglês, *Recurrent Neural Network - RNN*), em especial as LSTMs (do inglês, *Long Short-Term Memory*) [Hochreiter & Schmidhuber, 1997]. Estas técnicas são capazes de obter bom desempenho para as tarefas de previsão de sequência em diversos problemas, conforme pode ser visto no capítulo a seguir.

Capítulo 3

Trabalhos Relacionados

Durante o processo de revisão da literatura foram elencados os tópicos mais próximos à este trabalho, que é o desenvolvimento de um método para previsão de atividades humanas utilizando dados obtidos por meio de sensores de *smartphones*. Os trabalhos mais relacionados com este são os trabalhos que realizam previsão de, *i)* sequências e séries temporais de modo geral; *ii)* atividades humanas, independente da forma como a atividade foi coletada; *iii)* acessos às funcionalidades do *smartphone*, como por exemplo, previsão do próximo aplicativo a ser utilizado.

As seções a seguir apresentam os principais trabalhos relacionados. Na Seção 3.1 é mostrado uma breve introdução sobre trabalhos de predição de modo geral. Na Seção 3.2 são exibidos os trabalhos principais que utilizam diversos sensores para prever atividades humanas. Boa parte destes utiliza sensores de vídeo como entrada para o método de previsão. Na Seção 3.3 são apresentados os trabalhos que realizam previsão em *smartphones*, que não são necessariamente atividades físicas, mas que mostram como os registros (*logs*) do *smartphone* pode ser utilizado para previsão. A Seção 3.4 apresenta uma breve análise sobre os trabalhos citados, bem como a diferença dos trabalhos encontrados para este.

3.1 Previsão de Sequências

Prever, antecipar e até mesmo recomendar ações futuras tem sido um tópico bem ativo nos anos recentes [Li & Fu, 2014; Bahrainian & Crestani, 2017b; Tang et al., 2018; Zong et al., 2019; Rodrigues et al., 2019]. Por motivos óbvios, a possibilidade de prever acontecimentos pode auxiliar a tomada de decisão dos seres humanos diariamente. Por esta razão, diversos métodos de previsão têm sido desenvolvidos em diversos campos. Uma aplicação bem utilizada a bastante tempo e presente nos *smartphones* é a previsão

de texto [Brown et al., 1992]. Conforme o texto é digitado, o algoritmo prevê qual será a próxima palavra e a sugere ao usuário. Em alguns casos, quando a palavra é digitada incorretamente o próprio algoritmo se encarrega de corrigir. O método utilizado para resolver este problema pode ser utilizado em outros problemas de uma área distinta e também obter resultados promissores. Mais alguns exemplos de trabalhos que utilizam técnicas de previsão são exibidos a seguir, organizados em quatro categorias.

- **Mercado Financeiro:** Prever o valor das ações no mercado financeiro foi um dos primeiros problemas a serem abordados por pesquisas de previsão em sequências temporais [Ahmadi, 1990; Kimoto et al., 1990; Kamijo & Tanigawa, 1990; Choi et al., 1995]. Devido à natureza do problema e também de sua importância, até a atualidade, diversos trabalhos ainda abordam o problema. Kim [2003] utiliza o SVM (*Support Vector Machine*) para prever o índice de preço das ações, os autores comparam o SVM com outros métodos, como BP (*back-propagation*) e CBR (*Case-Based Reasoning*), os resultados apontam que o SVM oferece uma alternativa promissora para previsão do mercado de ações.

  De forma similar Gui et al. [2014] também utilizam SVM para previsão no mercado de ações, entretanto, os autores usam lógica fuzzy no pré-processamento dos dados e demonstram a viabilidade desta abordagem. Zhang et al. [2019] optam por utilizar o HMM para prever o preço de ações. O diferencial do método é a forma na qual a ordem da cadeia de Markov utilizada, originalmente os métodos tradicionais utilizam apenas as probabilidades obtidas por meio da cadeia anterior. O método obteve melhores resultados conforme a ordem da cadeia foi aumentada.

- **Comportamental:** Diversos trabalhos tem como foco a previsão de sentimentos, a aplicação de tal previsão varia de acordo com o assunto estudado, bem como os métodos utilizados. Moreira et al. [2019] utilizam métodos de aprendizagem de máquina para prever o risco de depressão pós-parto e atingem resultados promissores com o *Random Forest*. De forma similar, Yang et al. [2018] também utilizam métodos de aprendizagem de máquina para prever usuários que possam ter depressão. Por meio de um aplicativo, a pessoa insere algumas informações pessoais e caso seja identificado que a pessoa corre risco de entrar em depressão uma notificação é enviada aos familiares cadastrados. Além disso, o próprio método recomenda ações para evitar que o usuário entre em depressão. Outro trabalho que também foca em aspecto comportamental é o de Valenza et al.

[2014], que utiliza modelos de Markov para prever o humor de pessoas que sofrem de transtorno bipolar.

- **Mobilidade:** Devido à presença de sensores de localização nos *smarthphones* e sua alta utilização, diversos trabalhos abordam como utilizar tais dados no ramo de previsão. Tang et al. [2018] utilizam HMM para prever as rotas que são traçadas por táxis durante a viagem e identificar possíveis pontos de congestionamento. Segundo os autores as formas mais comuns para resolver este tipo de problema é utilizando modelos de Markov, filtro de Kalman ou inferências Bayesianas. Também utilizando HMM Zong et al. [2019] desenvolveram um método para prever o destino (e.g.: casa, trabalho, restaurante) das pessoas baseando-se nos seus hábitos. Tal previsão pode ser útil para recomendar ao indivíduo possíveis lugares para visitar.

  Por outro lado, Rodrigues et al. [2019] argumentam que um ponto a ser melhorado nas abordagens desenvolvidas é a taxa de erro quando ocorre algum evento (e.g.: show, jogos, manifestações, desastre natural) fora do comum. Pois como os métodos são desenvolvidos baseando-se apenas nos padrões habituais, a previsão durante os dias que eventos especiais ocorre pode ter sua eficácia comprometida. Para resolver este problema os autores desenvolveram um método aplicando técnicas de aprendizagem profunda que utilizam informações sobre eventos obtidos por meio da web e dados de rotas de táxi. O foco do principal do trabalho é prever a demanda de táxi nos locais que estes eventos ocorrem.

- **Robótica:** No campo de robótica, principalmente em áreas voltadas para robôs que exercem atividades humanas ou auxiliam humanos, prever o que um ser humano faria é essencial para tomar uma decisão. Por este motivo, tal campo também tem sido motivo de interesse em várias pesquisas. Tanto para aplicações fabris [Wilcox et al., 2013] quanto para serviços pessoais [Goto et al., 2013]. Neste contexto, Hawkins et al. [2013] utilizam redes Bayesianas para inferir as próximas tarefas a serem realizadas pelo robô e citam como trabalhos futuros aprimorar a eficiência do método de desenvolvido. Com o foco similar, [Wang et al., 2018] utilizam aprendizagem profunda para inferir as tarefas que serão realizadas por um humano, de forma que com base nesta inferência um robô possa executar as atividades antecipadamente. O método desenvolvido incorpora dados de contexto para aprimorar sua eficácia e atingiu resultados promissores.

Os trabalhos citados acima exibem apenas uma minúscula quantidade de aplicações na qual a previsão de dados temporais podem ser utilizadas. No entanto, é

possível ter uma leve noção do quanto os métodos têm sido aprimorados cada vez mais e do seu grande range de áreas no qual pode ser aplicado. Além disso, é possível notar que a mesma metodologia pode ser aplicada para um problema totalmente diferente. Desta forma, também é possível aplicar métodos diferentes para o mesmo problema e obter resultados satisfatórios. Levando em consideração o problema desta dissertação, os trabalhos que mais se aproximam deste são descritos nas seções a seguir.

3.2 Previsão de Atividades Humanas

A maioria dos trabalhos que demonstra como prever atividades humanas utiliza câmera como fonte de entrada de dados. O principal motivo pelo qual isso acontece é que o campo de visão computacional tem obtido forte avanço. Tais avanços tornaram possível o reconhecimento de atividades por meio de vídeos e fez com que a previsão de novas cenas a partir de uma porção do vídeo se tornasse real [Li et al., 2012]. Utilizando a câmera é possível obter muitos dados de contexto que podem ser úteis para prever as próximas cenas, por exemplo, o cenário e os objetos ao redor do usuário. No entanto, isto aumenta a complexidade do método. A forma como os dados da câmera são interpretados e sua utilização são diversos. Em alguns trabalhos são utilizados os dados de contexto das imagens, em outros apenas os movimentos que o usuário realiza, entre outros.

O primeiro trabalho que de fato aborda o problema de predição de atividades humanas é o de Li et al. [2012]. Antes disso, o problema de predição da próxima atividade era tratado apenas como um problema de reconhecer atividades em vídeos incompletos. Li et al. [2012] utilizam uma técnica baseada no modelo de Markov de ordem variável (do inglês, *Variable order Markov Model - VMM*) para prever a próxima atividade a ser realizada por um usuário. Dado um vídeo como entrada, o método proposto divide as atividades que são realizadas em pequenas ações. Por exemplo, dado um vídeo de um pedido de casamento o método proposto identifica as pequenas ações realizadas (tais como, levantar, se aproximar, se ajoelhar, pegar aliança). Estas ações são dispostas como uma sequência de eventos. Para prever o próximo evento é utilizada uma implementação do VMM chamada Árvore de Sufixo Probabilística (do inglês, *Probabilistic Suffix Tree - PST*). A base utilizada para os experimentos foi a MHOI e uma base criada pelos autores com vídeos de jogos de tênis. O método desenvolvido foi comparado com o SVM, IBoW (*Integral Bag of Words*) e DBoW (*Dynamic Bag of Words*), e obteve a melhor acurácia. Em um trabalho mais recente os autores reconhecem que utilizar apenas as atividades para realizar uma previsão

pode não ser suficiente e que utilizando dados de contexto a acurácia é melhor [Li & Fu, 2014].

Em alguns casos, em vez de usar todas as características do vídeo, é utilizado somente a silhueta do usuário como dado de entrada, como no trabalho de Uddin [2014], que apresenta uma metodologia para reconhecer e prever atividades básicas (e.g., andar, sentar, deitar). Para reconhecer atividades foi utilizado HMM e a partir das atividades reconhecidas é criada uma base de dados. Para prever atividades foi utilizado uma estrutura simples de árvore que guarda as informações das sequências. A base de dados utilizada para realizar os testes é sintética e o método proposto obteve 91,31% de acurácia. No entanto, foram realizados poucos testes e com uma base pouco representativa.

Outro trabalho bem similar que também utiliza o HMM e utiliza imagens para o reconhecimento de atividades é o de Magnanimo et al. [2014]. O foco principal é reconhecer e prever atividades humanas (e.g., pegar faca, cortar maçã) para que os robôs possam interagir melhor com os humanos. O método foi desenvolvido com intuito de ser aplicado no ramo de robótica. Mais especificamente, em robôs ajudantes de cozinha que auxiliam o chef a preparar receitas. Como os passos para preparar uma receita são bem definidos e seguem um padrão, os autores optam por este cenário. Um dos exemplos citados é preparar um chá, que segue uma sequência de passos bem definida. Além disso, nestes cenários é possível utilizar dados de contexto para ajudar na previsão, como por exemplo, os ingredientes e utensílios. Para prever a próxima atividade foi utilizado uma rede Bayesiana dinâmica. Para realização dos experimentos foi utilizado base sintética, com poucos dados, considerando apenas a preparação de receitas. A métrica utilizada para avaliação é a acurácia, para algumas ações a probabilidade de acerto é de 20%, como no caso de "pegar, pote de chá" e "pegar, pote de café". No entanto, também pode atingir até 99% em outras, como, "colocar, açúcar".

Como ao utilizar câmeras é possível obter dados de contexto, Li & Fu [2014] propõem um *framework* para reconhecimento e predição de atividades complexas que leva em conta o contexto no qual a atividade ocorre. Por exemplo, ao identificar que determinado usuário anda pela cozinha, além de reconhecer as ações de "andar" e "parar", o método é capaz de integrar informações sobre o contexto e inferir o que o usuário faz, neste cenário poderia inferir que o usuário está cozinhando, por exemplo. Neste caso, cozinhar é uma atividade complexa, de longa duração. Os dados inicialmente são coletados por meio de sensores de vídeo. Com base nas imagens, o método além de analisar as ações do usuário também analisa os objetos ao redor. Com base na análise das ações e objetos (contexto) a previsão é realizada. Por exemplo, dado que

um usuário realiza a ação de pegar um copo e colocar água, qual a próxima ação? Provavelmente seria "beber a água".

O problema de como prever qual a próxima atividade a ser realizada foi divido em dois subproblemas. O primeiro trata de como representar as atividades ocorridas de modo sequencial. O segundo trata de como realizar a previsão utilizando tal representação. Para resolver o primeiro problema, os autores optam por dividir as imagens obtidas da câmera em ações e representar cada ação como um símbolo. A previsão das atividades foi testada em dois cenários. No primeiro cenário, o dado de entrada é uma sequência de atividades (*action-only*). No segundo cenário, além dos dados das atividades, existem também os dados de contexto como entrada (*context-aware*).

Para prever as atividades com ciência do contexto são utilizados métodos de mineração de padrões sequenciais e inferência Bayesiana, pelo fato de tais métodos se adaptarem melhor em cenários no qual é preciso inserir informação de contexto. O método desenvolvido para previsão de atividades é genérico e pode ser integrado com outros métodos de decomposição sequencial. É possível também inserir informações sobre contexto e ainda usar dados de mais sensores para melhorar a acurácia da previsão. A eficácia do método foi testada em diversos cenários, utilizando as bases MHOI(*Maryland Human-Object Interactions*) [Gupta et al., 2009], MPII-Cooking [Rohrbach et al., 2012], UCI-OPPORTUNITY [Roggen et al., 2010] e entre outras. A métrica utilizada para avaliação foi a acurácia e o método foi comparado com, IBoW, DBoW, BoW+SVM, n-grama, KNN (*K-Nearest Neighbors*) e HMM. Diversos experimentos foram realizados, inicialmente, foi testado o método desenvolvido para os casos que não é inserida informação de contexto. Logo após, os experimentos com bases que disponibilizam dados de contexto. O método desenvolvido obteve melhor resultado que todos os outros métodos, em todos os cenários experimentados atingindo até 95% de acurácia, neste mesmo cenário o n-grama atingiu 79%.

Xu et al. [2015] propõem um método para prever sequências de vídeos a partir de um trecho, os autores sugerem tratar o problema de previsão de vídeo como o de previsão de consultas em máquinas de busca (e.g. Google). A base utilizada foi a UT-Interaction e os resultados obtidos apontam que o método possui resultado superior aos métodos de referencia, que são SVM, DBoW e IBoW. Yang et al. [2015] propõem um método para prever atividades humanas do dia a dia, a partir de uma sequência de vídeo o método prevê o que acontecerá na sequência a partir das formas e dos movimentos das pessoas. O método desenvolvido utiliza matriz de auto-similaridade de BoW para prever as próximas cenas da imagem e SVM para classificar. A base utilizada foi a MHOI e obteve resultado superior ao HMM, IBoW e DBoW, na maioria dos casos.

Além disso, também foram desenvolvidos métodos para prever trajetórias humanas. Como o trabalho de Alahi et al. [2016], que desenvolveram um método utilizando LSTM (*Long Short-Term Memory*) para prever a trajetória de pedestres em locais lotados baseado em frames de vídeo. O foco dos autores é realizar a previsão em locais lotados, como em um aeroporto ou em um shopping. Os experimentos utilizam as bases de dados UCY [Lerner et al., 2007] e ETH [Pellegrini et al., 2009]. As métricas utilizadas para avaliação foram, *i)* erro de deslocamento médio [Pellegrini et al., 2009]; *ii)* erro de deslocamento final; *iii)* erro médio de deslocamento não linear. O método desenvolvido obteve taxas de erros menores comparando a LSTM desenvolvida (chamada de Social-LSTM) com LSTM tradicional. Ainda utilizando vídeos, Wang et al. [2017a] desenvolveram um algoritmo chamado TGTW (*Temporally-weighted Generalized Time Warping*), que foi baseado no GTW (*Generalized Time Warping*) [Zhou & De la Torre, 2012], para representar os vídeos e combinado com o k-NN prever a próxima atividade básica. O GTW é um algoritmo baseado no DTW (*Dynamic Time Warping*), que é um algoritmo utilizado para mensurar a similaridade entre sequências temporais. Bem utilizado para reconhecimento de fala e assinatura. O KNN atinge uma performance promissora utilizando o DTW como método para mensurar distância. Os resultados obtidos superam os métodos citados como estado da arte, que segundo os autores são IBoW, DBoW, SC (*Sparse Coding*) e MSSC (*Mixture of Segments Sparse Coding*) [Cao et al., 2013].

Fora do campo de visão computacional também existem pesquisas que abordam previsão de atividades humanas utilizando outras fontes de dados. Gu et al. [2016] utilizam CRF para reconhecer a atividade atual e prever a próxima atividade que o usuário realizará. Utilizando a base de dados AHTUS (*American Heritage Time-Use Survey*) é criada uma nova base pré processada, chamada AHDL (*Activity Human Daily Life*), que segundo os autores, foi a primeira base de conhecimento sobre atividades humanas. Com esta base é possível facilmente obter dados sobre atividades realizadas de acordo com o dia, local, atividade anterior, entre outros.

Para realizar a previsão foi desenvolvido o *ActivitySensor*, que é um modelo desenvolvido utilizando CRF (*Conditional Random Field*) de cadeia linear. Os autores argumentam que existem propriedades Markovianas no problema de predição de atividade e que os métodos que seriam o estado da arte não sintetizam essa propriedade por modelarem este problema como um problema de classificação.

Os autores realizaram experimentos com atividades denominadas de "alto nível" (viajar, praticar esportes, educação, laser...) e de baixo nível (dormir, andar, brincar, ir para o trabalho...). As métricas utilizadas para avaliação foram, acurácia, precisão, revocação e F1. Para predição de atividades de alto nível a acurácia foi de 69,10%

e para atividades de baixo nível 45,70%. Não foi citado experimento sobre eficiência, rapidez e escalabilidade. Como trabalho futuro os autores citam a utilização de tais métodos para o cenário de dispositivos móveis, como smartphones, por exemplo. Além disso, mencionam explorar a utilização de redes neurais recorrentes e também aplicar tais métodos em diversos tipos de dados, como em redes sociais.

3.3 Previsão Utilizando Dispositivos Móveis

Com a evolução da capacidade e dos sensores presentes nos *smartphones* também surgem inúmeras pesquisas utilizando dados provenientes destes dispositivos, pois existem diversos cenários no qual a previsão utilizando *smartphone* pode ser aplicada. Um bom modelo de previsão pode melhorar as recomendações realizadas por assistentes virtuais e desta forma facilitar o dia a dia do usuário. Por exemplo, prever o próximo aplicativo que o usuário utilizará e disponibilizá-lo na tela inicial melhora a usabilidade, ou prever o trajeto e informar o estado do trânsito pode auxiliar o usuário na sua tomada de decisão. Outro exemplo é a sugestão de locais próximos que podem ser visitados, como foi feito no trabalho de Ye et al. [2013], onde os autores apresentam um *framework* que é capaz de prever o próximo lugar que um usuário visitará com base nos check-ins realizados pelo próprio usuário na rede social *Gowalla* (comprada pelo Facebook em 2011 e descontinuada em 2012).

Esse *framework* foi criado para determinar qual a melhor propaganda a ser exibida de acordo com o horário e a localização do usuário. Devido ao fato de sempre existir uma grande quantidade de possíveis locais que um dado usuário pode visitar, os autores optaram por reduzir o problema de prever a próxima localização do usuário em dois sub-problemas. Em um primeiro momento, para reduzir a quantidade de dados, os autores especificam que cada lugar visitado pertence a uma categoria predefinida, que pode ser entretenimento, viagem, comida, entre outros. Com base nisto, o primeiro problema é, como prever a categoria do próximo local a ser visitado por um usuário? Neste caso, uma resposta poderia ser *entretenimento*, por exemplo.

O segundo problema é referente a como prever o lugar que o usuário visitará, ou seja, dado uma sequência de *check-ins* e uma categoria de local a ser visitado, qual a próxima localização do usuário? Uma possível resposta a este problema poderia ser um cinema, por exemplo. Para resolver estes problemas, são utilizados métodos baseados em HMM. Para prever a próxima categoria do usuário é utilizado um modelo que é chamado de *mixed HMM*. Tal método foi adotado para incorporar os dados espaciais e temporais no modelo. A métrica utilizada para avaliação foi a acurácia e o método foi

comparado com o PMM (*Periodic Mobility Model*) [Cho et al., 2011]. Para previsão da próxima localização, utilizando a base de dados do *Gowalla* o método obteve até 31,89% a mais de acurácia que o método com o qual foi comparado.

Além de prever "o quê" o usuário fará, ou "para onde" o usuário irá, também existem trabalhos que focam na frequência com que o usuário realizará determinadas ações. Dai & Ho [2014] propõem um *framework* para prever com que frequência o usuário acessará determinadas funcionalidades do seu *smartphone* no dia seguinte, utilizando as informações do uso dos dias anteriores e analisando as características comportamentais representativas. Foram selecionadas 5 funcionalidades principais (ligações, envio de SMS, acesso à mídia, aplicações e histórico de chamadas) e a frequência foi dividida em 3 níveis (não utilizou, utilizou, utilizou bastante). A base de dados utilizada no desenvolvimento foi a *Nokia Mobile Data Challenge* (MDC), que contém dados de diversos sensores do *smartphone*. Os dados foram coletados entre 2009 e 2011, com a contribuição de 185 participantes. Nesta base é possível identificar quando cada atividade foi realizada.

O método foi desenvolvido com base no PCA (*Principal Component Analysis*), utilizando os 4 componentes principais. Um dos desafios enfrentado foi o fato de que o padrão dos usuários podem variar de acordo com o dia. Por exemplo, durante a semana existe um padrão, mas nos fins de semana o padrão pode ser totalmente diferente, devido a este fato, o método pode ter acurácia até 30% menor nos dias fora do padrão analisado. Para realização dos experimentos foi utilizado o histórico dos 4 últimos dias anteriores para prever a frequência na qual determinadas ações ocorreriam nas próximas horas. A métrica utilizada para avaliação foi a acurácia, e foi comparado com o MA (*Moving Average*), ARIMA (*Autoregressive Integrated Moving Average*) e KPCA (*Kernel Principal Component Analysis*). O método teve acurácia superior ao MA e ARIMA e comparando com o KPCA obteve resultados bem similares.

De forma similar, mas com foco apenas em ligações, Cici et al. [2016] apresentam um método para prever quando um dado usuário realizará uma ligação. O método é criado com o intuito de auxiliar as operadoras a identificarem onde devem concentrar seus esforços para melhorar o tráfego de ligações. Analisando uma base de dados disponibilizada pela Telecom Itália, notou-se que o problema de previsão de tráfego poderia ser representado como um problema de classificação, que é determinar se um usuário i ligará para o usuário j em um tempo t, ou seja, $A_{ij}(t) > 0$. Um dos desafios enfrentados foi a escassez de dados, pois na base utilizada 80% dos celulares costumavam não realizar ligações. Para resolver este problema, os autores optaram por utilizar o SVD (*Singular Value Decomposition*) para identificar os componentes principais e o *Random Forest* para classificação. O método desenvolvido atingiu 85% de acurácia

e foi comparado com o *Decision Tree* que obteve 84% e com o *Naive max-class* que obteve 80%.

Outros trabalhos investigam como prever qual será o próximo aplicativo utilizado pelo usuário para exibi-lo na tela inicial, como no caso do trabalho de Baeza-Yates et al. [2015]. Para este trabalho os autores coletaram os dados por meio do aplicativo AVI-ATE, que é uma tela inicial inteligente do Yahoo[1] (descontinuada em março de 2018). Utilizando os dados coletados por cerca de 6 meses de uso do app, foi realizada uma profunda análise no comportamento dos usuários a fim de encontrar padrões como tempo de uso, funções mais utilizadas, tempo conectado no cabo, quantidade de apps executados, entre outros. Com base em tal análise, foi realizada a extração de características básicas (e.g., tempo, localização, velocidade) e de sessão (e.g., último app utilizado, última vez conectado no carregador). O método desenvolvido para realizar as predições é baseado em uma rede bayesiana, chamado de PTAN (*Parallel Tree Augmented Naive Bayesian Network*) que é uma versão paralelizada do TAN (*Tree Augmented Naive Bayes*). A versão paralelizada facilita a utilização de *frameworks* como o *Hadoop* para processamento de alta quantidade de dados.

O problema foi modelado como um problema de classificação e é comparado com algoritmos tradicionais de aprendizagem de máquina, como o Naive Bayes, SVM, Árvore de decisão (C4.5) e regressão logística. Após notar que a utilização dos apps segue uma distribuição de Pareto, foi decidido dividir os experimentos em duas etapas. No primeiro experimento utilizando todos os apps e no segundo apenas os cinco mais utilizados. Em ambos foi medido apenas a precisão. Para o primeiro experimento, utilizando apenas as características básicas, a árvore de decisão alcançou o melhor resultado com 34,90% de precisão. No entanto, ao utilizar características básicas e de sessão, o PTAN alcançou melhor o resultado, com 90,20% e em segundo lugar ficou o Naive Bayes com 76,30%. Para o experimento com os 5 mais utilizados a precisão do método proposto foi de 85,70%, seguido pela árvore de decisão, com pouco mais de 60,00%.

De forma similar, Bahrainian & Crestani [2017b] apresentam um método para prever qual será o próximo aplicativo utilizado no smartphone. O propósito é exibir os 5 apps mais prováveis de serem utilizados na tela inicial. O método proposto utiliza como base o ATM (*Author Topic Model*) e SVD/LSI (*Latent Semantic Indexing*). O método desenvolvido é baseado no SVD, mas com algumas alterações na forma como o método é aplicado. Segundo os autores, ao aplicar o método em toda a base, gera-se um modelo de recomendação global. No entanto, os dados de uso de apps podem

[1]https://yahoo.com/

ser muito diferentes de acordo o usuário, dia e horário. Portanto, em vez de aplicar o SVD em toda a base, primeiramente a base é dividida por usuários e depois em n fatias de tempo. Logo após, o método é aplicado para cada fatia individualmente. Para realização dos experimentos, foi utilizada uma base com cerca de 70 mil acessos a aplicativos pertencentes a 176 usuários. A métrica utilizada para avaliação foi a acurácia. Caso o próximo app a ser utilizado estiver entre os 5 mais prováveis, computa-se um acerto. O modelo é treinado e testado para cada usuário individualmente. No primeiro experimento foi utilizado 80% para treino e 20% para teste. O modelo proposto alcançou 43,30% de acurácia, enquanto ATM e SVD/LSI atingiram 38,78% e 37,45% respectivamente. No segundo experimento foi utilizado 67% para treino e 33% para teste. Neste caso a acurácia de ambos foi afetada, o modelo proposto alcançou 40,11% de acurácia (3,19% a menos), enquanto ATM e SVD/LSI atingiram respectivamente 38,78% (3,35% a menos) e 37,45% (2,77% a menos). Como trabalho futuro, é citado avaliar todas as interações entre usuário e *smartphone* e obter mais informações do usuário para gerar notificações úteis em tempo real.

Outros trabalhos têm como foco direto a aplicação em assistentes virtuais. No trabalho de Bahrainian & Crestani [2017a] foi desenvolvido um método que prevê as partes de uma conversa que a pessoa vai esquecer. Os autores argumentam que a memória humana pode falhar devido a diversos aspectos e que é comum que a pessoa às vezes esqueça algo, o que pode causar problemas, principalmente no ambiente de trabalho. Depois de um vasto estudo sobre como funciona a memória humana, os autores concluíram que alguns momentos são mais possíveis de esquecer e que estes momentos podem ser inferidos a partir do que a pessoa fala e sente. A base de dados utilizada é proveniente de áudios gravados em reuniões e de um sensor biofísico similar a um relógio que coleta dados para reconhecer os sentimentos. O método foi desenvolvido utilizando o HMM e prevê se uma determinada parte da conversa vai ser esquecida futuramente ou não. As métricas utilizadas para a avaliação foram F1, precisão e revocação. O modelo atingiu 68,53% de F1, 55,66% de precisão e 86,45% de revocação.

Bahrainian & Crestani [2018], em um trabalho posterior, evoluem seus métodos e propõem 4 métodos para prever os tópicos das próximas reuniões. Dessa forma, a pessoa pode se antecipar e revisar os tópicos que serão comentados e desenvolver uma reunião mais produtiva. O método com os melhores resultados foi desenvolvido utilizando filtro de Kalman. As métricas utilizadas para avaliação foram a precisão e F1. De forma similar, também usando um modelo baseado no filtro de Kalman, [Sun et al., 2016] desenvolveram um método para ser utilizado em assistentes virtuais. Este modelo foca em prever as intenções do usuário, como chamar um transporte por meio de um aplicativo, enviar mensagem, ouvir música, entre outros. O método desenvolvido,

chamado de KP2, foi comparado utilizando as métricas F1 e as taxas de satisfação do usuário com a experiencia com a assistente. Utilizando duas bases de dados do mundo real com dados coletados pelos autores, o método superou todos os métodos com os quais foi comparado, sendo eles o LambdaMART [Shokouhi & Guo, 2015], o FM (*Factorization Machine*) [Rendle et al., 2010] e o próprio Filtro de Kalman sem alterações.

3.4 Discussões e Considerações Finais

As seções anteriores reúnem os principais trabalhos relacionados à pesquisa atual. Na Tabela 3.1, é possível notar que a maioria dos trabalhos utiliza vídeo como fonte de dados. Os trabalhos que utilizam dados obtidos por meio de *smartphones* como entrada apresentam previsão de funcionalidades que serão utilizadas no dispositivo. Nenhum dos trabalhos encontrados foca especificamente na previsão de atividades físicas com base em dados que podem ser obtidos por meio de *smartphones*. Além disso, nota-se que, independente da aplicação, alguns dos métodos desenvolvidos também utilizam dados de contexto para aprimorar os resultados obtidos.

Dentre os trabalhos apresentados, o que mais se aproxima desta dissertação é o de Li & Fu [2014]. Conforme citado anteriormente, nesse trabalho os autores também utilizam dados de atividades humanas e dados de contexto para prever a próxima atividade que será realizada. No entanto, o método foi desenvolvido para prever com base em vídeos, que não se aplica à proposta desta dissertação, pois o foco é a utilização de dispositivos móveis. Além disso, esse trabalho foca especificamente na acurácia do método. Os algoritmos utilizados demandam tempo e esforço excessivo em sua execução ao serem comparados com técnicas mais recentes. Devido aos algoritmos serem executados no próprio *smartphone*, a eficiência é um ponto importante a ser observado.

Ao utilizar um dispositivo móvel sem tantos recursos computacionais é preciso analisar minuciosamente os métodos que podem ser utilizados, para que desta forma seja possível processar rapidamente todos os dados e sem um alto consumo de energia. Os algoritmos utilizados para previsão neste trabalho levam em conta todos estes aspectos e também o custo benefício envolvido.

Conforme exibido na Tabela 3.1, existem diversos métodos para previsão de sequências. Com base em tais métodos, a Tabela 3.2 exibe as principais abordagens utilizadas em relação as suas principais habilidades para prever atividades. Os pontos elencados como principais para um método de previsão de atividades humanas são: i)

Tabela 3.1: Sumarização dos trabalhos de previsão de atividades.

Referência	**Fonte de dado**	**Utiliza contexto?**	**Previsão**	**Método**
Li et al. [2012]	Vídeo	Não	Atividade simples	VMM
Ye et al. [2013]	GPS	Não	Localização	HMM
Uddin [2014]	Vídeo	Não	Atividade simples	HMM
Dai & Ho [2014]	Log do *smartphone*	Não	Frequência de utilização de aplicativo	PCA
Li & Fu [2014]	Vídeo	Sim	Atividade complexa	AprioriAll, Bayes
Magnanimo et al. [2014]	Vídeo	Não	Atividade simples	DBN
Baeza-Yates et al. [2015]	Log do *smartphone*	Sim	Próximo aplicativo utilizado	PTAN
Xu et al. [2015]	Vídeo	Não	*Frame* de vídeo	AAC
Yang et al. [2015]	Vídeo	Não	Atividade complexa	BoW, SVM
Gu et al. [2016]	Survey	Sim	Atividade complexa	CRF
Cici et al. [2016]	Log do *smartphone*	Não	Horário da próxima ligação	Random Forest
Alahi et al. [2016]	Vídeo	Não	Atividade simples e localização	LSTM
Wang et al. [2017a]	Vídeo	Sim	Atividade complexa	TGTW, KNN
Bahrainian & Crestani [2017b]	Log do *smartphone*	Sim	Próximo aplicativo utilizado	SVD
Bahrainian & Crestani [2017a]	Áudio	Sim	Falhas de memória	HMM
Bahrainian & Crestani [2018]	Áudio	Sim	Tópicos	Filtro de Kalman

classificação/predição de sequência; *ii*) utilização de dados de contexto, pois são dados essenciais para o compreendimento da situação do usuário; *iii*) geração de modelos a partir de uma baixa quantidade de dados de treino, pois neste trabalho pretende-se utilizar dispositivos móveis que não possuem uma alta capacidade de armazenamento.

Em relação às abordagens, as principais são as seguintes [Li & Fu, 2014; Tang et al., 2018], *i*) Seleção de características (*Feature Selection*) utilizando como por exemplo o n-grama; *ii*) Função de distância, comparando as distâncias entre as sequências e verificando qual é a mais próxima e utilizar algoritmos como o KNN ou o SVM como classificadores; *iii*) Redes neurais recorrentes, que podem ser usadas para problemas de predição, como LSTM, por exemplo; *iv*) Modelos markovianos, que simulam um processo generativo até gerar uma sequência, alguns exemplos são, HMM, VMM, mo-

Tabela 3.2: Comparação dos métodos de previsão de sequência. (Adaptado de Li & Fu [2014])

Modelo	**Classificação de Sequência**	**Utiliza contexto?**	**Funciona com poucos dados**
Baseados em características	X		X
Função de distância	X		X
Modelos Markovianos	X		X
LSTM	X	X	
SPM	X	X	X

delo de Markov de ordem K; *v*) Mineração de padrões sequenciais e utilizar inferência Bayesiana para inferir a próxima sequência com base nos padrões mais frequentes. Para este problema foi decidido utilizar algoritmos de mineração de padrões sequenciais. A principal razão para esta escolha foi o fato de que estes algoritmos mineram conjuntos de itens, dessa forma torna-se fácil enriquecer cada item da sequência com dados de contexto. O capítulo a seguir descreve em detalhes como tal algoritmo foi utilizado neste trabalho.

Capítulo 4

SOPHIA - Um Método para Previsão de Atividades

Este capítulo descreve um método para previsão de atividades humanas denominado SOPHIA (***SO****ftware for* ***P****rediction of* ***H****uman* ***I****nteractions and* ***A****ctivities*). Diferente dos trabalhos mencionados anteriormente, este trabalho propõe utilizar apenas dados obtidos a partir de sensores vestíveis para realizar previsão de atividades humanas. Embora o foco principal seja prever atividades humanas com base nos sensores do *smartphone*, o método desenvolvido é genérico e capaz de ser adaptado à diversas outras áreas na qual a natureza dos dados tenha um padrão temporal com alguma sazonalidade.

Antes de adentrar nos detalhes de desenvolvimento do método é necessário entender algumas questões. Primeiramente é preciso compreender o problema principal e como este foi modelado. O problema abordado é, "como prever atividades humanas utilizando o histórico dos dados (obtido por meio dos sensores) do *smartphone* de um usuário?". Tal problema foi modelado como um problema de previsão em dados representados em sequências simbólicas, pois desta forma é possível reduzir a dimensionalidade dos dados e utilizar métodos de previsão de sequências. A solução para o problema abordado envolve a resposta de duas novas questões: *i*) como representar os dados extraídos por meio do *smartphone* como sequências simbólicas? e *ii*) como realizar previsão em dados representados simbolicamente?

Logo, é possível notar que existem no mínimo duas fases principais para alcançar a solução do problema abordado. A primeira fase é chamada de fase de reconhecimento e representação dos dados. Nesta fase, os dados obtidos por meio do *smartphone* são preparados e transformados em sequências simbólicas. Esta etapa é realizada utilizando técnicas existentes na literatura e os detalhes são discutidos nas seções posteriores.

Na segunda fase, o problema de realizar previsão em dados representados simbolicamente é modelado como um problema de classificação de sequência, mas levando em consideração apenas o início da sequência. Conforme exibido no capítulo anterior, neste trabalho são utilizados algoritmos de mineração de padrões sequenciais, pois a possibilidade de usar os dados das atividades em conjunto com dados de contexto é essencial para um bom funcionamento do método. Além disso, também é interessante que o método escolhido possa ser treinado diretamente no dispositivo móvel, pois as rotinas costumam variar bastante de um indivíduo para o outro. Devido à essas limitações e requisitos, a utilização de algoritmos de mineração de padrões se encaixa perfeitamente na solução para o problema abordado.

Portanto, o problema a ser resolvido pode ser definido formalmente da seguinte forma. Seja $\mathcal{S} = \{i_1, i_2, ..., i_n\}$ onde i é um *itemset* definido como $i_k = (atividade, contexto_1, contexto_2, ..., contexto_m)$. Dado uma sequência $\mathcal{S}$ o método desenvolvido busca prever qual a atividade do *itemset* i_{n+1}. A Figura 4.1 exemplifica como uma sequência $\mathcal{S}$ pode ser formada a partir dos dados reconhecidos. Cada conjunto formado por uma atividade e seus respectivos dados de contexto são considerados um *itemset*. O conjunto de todas as atividades realizadas formam uma sequência. A partir desta sequência é realizado o processamento e previsão da próxima atividade.

Por exemplo, com base na Figura 4.1, em um primeiro momento o usuário se encontra sentado no seu local de trabalho. Cada atividade e dado de contexto é convertido para um símbolo, "a" e "b", gerando "(a,b)", que é o primeiro *itemset* da sequência. Da mesma forma, o processo se repete por todo o histórico do usuário. A seção a seguir exibe uma visão geral do processo para gerar o modelo de previsão e mais adiante cada etapa do processo é detalhada.

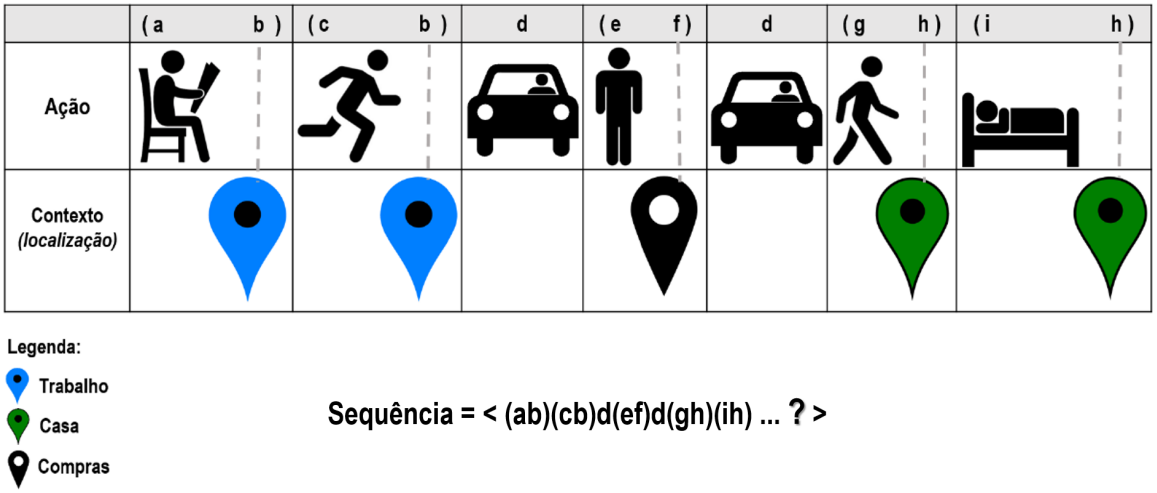


Figura 4.1: Representação sequencial a partir dos dados reconhecidos.

4.1 Visão Geral

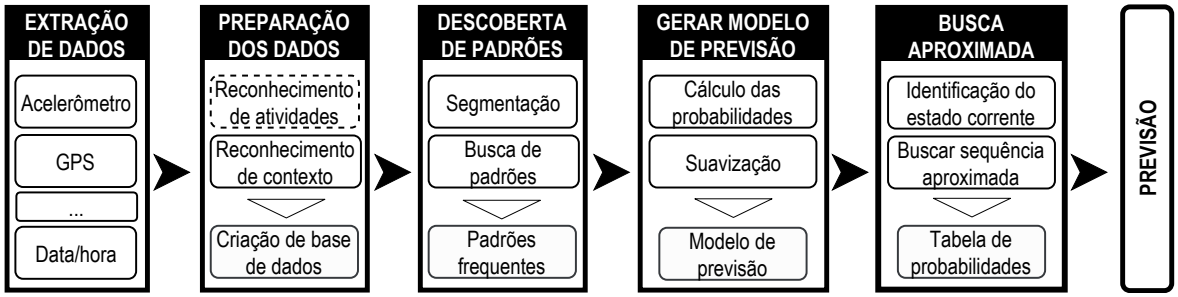


Figura 4.2: Visão geral do método de previsão.

A Figura 4.2 fornece uma visão das principais etapas necessárias para o desenvolvimento do método de previsão. Tal método foi desenvolvido utilizando algoritmos de mineração de padrões sequenciais como um de seus principais componentes. Estes algoritmos buscam padrões em sequências simbólicas. Portanto, é necessário um pré-processamento dos dados para prepará-los para o método de previsão.

Conforme pode ser notado na Figura 4.2, o processo é iniciado com a extração de dados do *smartphone*. Os dados dos sensores (e.g.: acelerômetro, GPS) são enviados para um método de reconhecimento de atividades humanas, que é descrito superficialmente na Seção 4.2.1. Este trabalho não entrará em detalhes profundos sobre como as atividades são reconhecidas a partir dos sensores do *smartphone*, mas sim em como utilizar o histórico dos dados para realizar previsões. Além disso, os dados de contexto (e.g.: GPS, data e hora) também são reconhecidos e categorizados, tal etapa é descrita em detalhes na Seção 4.2.2. Após o processamento dos dados brutos, todas as informações obtidas são armazenadas em uma base, que contém em cada tupla, as informações de quais atividades foram realizadas e o contexto na qual cada atividade ocorreu.

A Seção 4.3 descreve em detalhes como a base de dados é transformada em um banco de dados de sequências. A Seção 4.4 descreve como é realizado o descobrimento de padrões nas sequências. A partir dos padrões encontrados é montado um modelo de previsão. Na Seção 4.5 é descrito como é gerado um modelo de previsão a partir dos padrões sequenciais mais frequentes, descobertos na fase anterior.

Os métodos de reconhecimento de atividades humanas utilizando *smartphones* geralmente costumam reconhecer apenas atividades simples (conforme mencionado no capítulo 2). Caso o método de previsão utilize estas atividades, a saída do método também é uma atividade simples. Assim como nos casos no qual as atividades de entrada são atividades de mais alto nível o método também prevê atividades de alto nível. Embora prever uma atividade simples seja útil em diversos cenários como em

aplicativos que lembram o usuário de, *i*) praticar determinado esporte; *ii*) sair do trabalho e traçar a rota para o próximo destino; *iii*) de levantar pela manhã (despertador inteligente); *iv*) ir deitar para dormir, seria ainda mais interessante prever uma atividade complexa, como por exemplo, "sair para almoço" ou "ir para casa". Ainda que este não seja o foco principal do trabalho, a Seção 4.6 demonstra como tais previsões podem ser realizadas e como tratar isto, bem como algumas considerações finais sobre o método desenvolvido.

4.2 Extração de Dados

A Tabela 4.1 exibe exemplos de dados brutos extraídos do *smartphone*. As três primeiras colunas representam os dados que serão utilizados como contexto. O campo *timestamp* representa o tempo no qual o dado foi coletado. Os campos latitude e longitude referem-se a geolocalização do usuário, que pode ser obtida por meio do GPS ou até mesmo por meio da rede conectada. Os campos ACC_X, ACC_Y e ACC_Z correspondem respectivamente aos valores dos eixos X, Y e Z do acelerômetro. Diversos outros dados podem ser obtidos por outros sensores do *smartphone*. Todos estes dados auxiliares podem servir como um dado de contexto, tanto para o reconhecimento quanto para a previsão de atividades.

Tabela 4.1: Exemplos de dados brutos extraídos do *smartphone*.

Timestamp	**Latitude**	**Longitude**	**ACC_X**	**ACC_Y**	**ACC_Z**
1486727454134	-3.0892974	-59.9644671	4.3177	-2.3699	-0.43585
1486733141241	-3.0892974	-59.9644671	13.797	0.38137	2.6015
1486738565152	-3.0892974	-59.9644671	9.6841	-1.5255	-1.4438
1486743083312	-3.0892974	-59.9644671	19.463	-3.4323	-5.598

Embora seja possível aplicar métodos de previsão em dados brutos, como nos dados exibidos na Tabela 4.1, este trabalho utiliza os dados categorizados e representados simbolicamente. Os dados do acelerômetro, por exemplo, fornecem informações para reconhecer a atividade realizada, conforme descrito na Seção 4.2.1. Os dados de geolocalização e o *timestamp* fornecem informações de local, data e hora, tais dados são considerados dados de contexto e a Seção 4.2.2 descreve como podem ser reconhecidos e discretizados.

4.2.1 Reconhecimento de Atividades Humanas

O reconhecimento de atividades humanas baseado em dados de sensores é implementado seguindo uma metodologia composta por quatro principais etapas: coleta de dados, segmentação, extração de características e classificação. Em cada uma das etapas, diferentes técnicas e algoritmos são utilizados para transformar os dados brutos dos sensores em informações de alto nível, ricos em contexto, e que possibilitem a classificação da atividade humana. A Figura 4.3 apresenta esquematicamente a metodologia aplicada neste trabalho para reconhecer atividades humanas.

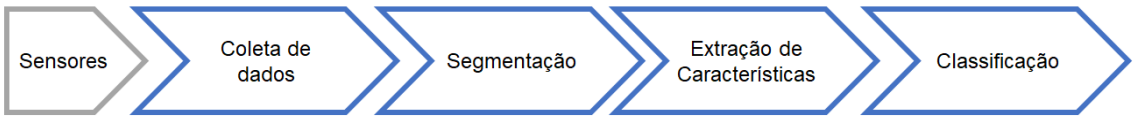


Figura 4.3: Etapas para o reconhecimento de atividades humanas.

A etapa de coleta de dados consiste em capturar as amostras do acelerômetro a uma taxa fixa de amostragem (e.g.: 50Hz) e armazenar em memória os dados em um formato padrão para as demais etapas de processamento. Os dados são formatados como séries temporais como exibido anteriormente na Tabela 4.1.

A etapa de segmentação consiste no particionamento das séries temporais coletadas em segmentos menores e de tamanhos específicos. Para isto, é utilizado a técnica de segmentação em janela deslizante (do Inglês, *sliding window*) com 50% de sobreposição entre janelas — *overlapping*. Cada segmento possui um total de 128 amostras, equivalente a uma janela de 2,56 segundos. O tamanho do segmento e a técnica de segmentação foi selecionada de acordo com trabalhos da literatura em reconhecimento de atividades, visto que o foco principal deste trabalho é no método de predição.

A etapa extração de característica é responsável pela transformação de cada segmento em uma representação de alto nível. Essa representação deve evidenciar as características relevantes para o problema de reconhecimento de atividades humanas. Para isto, são selecionados múltiplas características em dois principais domínios, tempo e frequência. Ao total são obtidas 36 características dos dados do acelerômetro tri-axial, 12 características para cada eixo (componente). As caraterísticas selecionadas são:

- Domínio do Tempo: Média, Mínimo, Máximo, Desvio Padrão, Mediana, Variância, *Zero Crossing Rate*, *Root Mean Square*.
- Domínio da frequência com aplicação da Transformada de Fourier: Energia Espectral, Entropia e a Soma dos 5 primeiros coeficientes de Fourier.

A etapa de classificação, por sua vez, consiste na aplicação de algoritmos de aprendizagem de máquina capazes de aprender padrões nos dados e inferir as atividades humanas. Para isto pode ser aplicado diferente algoritmos de classificação como Árvore de decisão, Naïve Bayes, k-Vizinhos-Próximos, Máquina de Vetores de Suporte e Redes Neurais Artificias.

4.2.2 Reconhecimento de Dados de Contexto

Conforme citado na Seção 2.2, o contexto é uma informação crítica para o entendimento de atividades humanas. Portanto, ao desenvolver métodos de previsão de atividades é essencial que seja possível extrair contexto da fonte de dados. O *smartphone* possui diversos sensores produzindo diversos dados de contexto que podem ser extraídos e reconhecidos a partir destes, tais como dados de geolocalização, tempo, clima, velocidade, rede conectada, entre outros. Embora todos estes dados sejam relevantes para identificar as diversas situações ao redor do usuário, neste trabalho será utilizado apenas a data, horário e a localização como dado de contexto, por serem mais relevantes para o problema abordado.

O dia é um importante dado de contexto. Saber se o dia da semana é um dia de trabalho, feriado ou fim de semana pode alterar totalmente a forma como o método de previsão funciona. Isto se deve ao fato de que nos feriados e fins de semana as pessoas costumam realizar atividades diferentes das que realizam em dias de trabalho [Dai & Ho, 2014]. Isto também é válido para o horário da atividade. Por exemplo, as atividades que a pessoa realiza pela manhã costumam ser diferentes das atividades realizadas à noite [Bahrainian & Crestani, 2017b]. A localização também é um dado de contexto importante, pois em alguns casos, sabendo onde o indivíduo se encontra, é mais fácil determinar o que ele fará posteriormente. Por exemplo, dado que um indivíduo se levantou e depois andou por volta das 22:00h, dependendo de onde o indivíduo se encontra a próxima atividade pode ser totalmente diferente. Se a localização for sua casa, uma provável próxima atividade é “deitar”, mas se a localização for o seu trabalho então a próxima ação mais provável seria “dirigir”. Com a utilização de um método de reconhecimento de atividades baseado em contexto, é possível inferir até mesmo a próxima atividade complexa a ser realizada [Liu et al., 2016], que neste caso, dependendo do local, pode ser “dormir” ou “voltar para casa”.

Os dados de contexto (localização, data e horário), assim como no caso das atividades, também precisam ser categorizados. Dado que um indivíduo pode visitar diferentes lugares com o mesmo propósito, utilizar a localização geográfica exata na qual o usuário se encontra pode fazer com que o método seja muito específico e tenha

sua acurácia afetada. Por exemplo, para "comprar remédio" uma pessoa pode ir em diversas farmácias com diferentes localizações, mas a ação será sempre a mesma, ou seja, ir à farmácia. Por este motivo, neste trabalho os lugares são agrupados em categorias, seguindo a mesma abordagem utilizada em outros trabalhos voltados para previsão de localização [Gambs et al., 2012; Ye et al., 2013; Cho, 2016].

Antes de representar cada dado da base por um símbolo é preciso discretizar os dados. Da mesma forma que a localização, caso o horário não seja categorizado, também pode implicar em uma baixa acurácia para o método. Por exemplo, a atividade "dirigir" ás 22:10h seria tratada como diferente da atividade de "dirigir" ás 23:11h. Entretanto, "dirigir por volta das 22:00h" é um padrão. Para o método de mineração de padrões sequenciais, 22:10 seria um símbolo x e 22:11 seria um símbolo y caso não houvesse discretização. Por tal motivo, este dado é discretizado e utiliza-se a hora corrente. Desta forma tanto 22:10 quanto 22:15 seriam representados por um símbolo x. Além de discretizar utilizando a hora, também é realizado uma discretização utilizando o turno, para fins experimentais.

Para obter informações de contexto sobre o tempo, é utilizado o *timestamp* provido pelo *smartphone*. Para este dado é preciso extrair, *i)* o horário do dia; *ii)* o dia da semana; *iii)* o turno (que pode ser, manhã, tarde, noite ou madrugada); *iv)* determinar se o dia é dia de trabalho/estudo, fim de semana ou feriado. O "turno" pode parecer redundante, pois, como existe o horário, é possível abstrair que o turno existe implicitamente no horário. No entanto, devido à possibilidade de ocorrência de regras de associação cíclicas (explicadas na Seção 2.5), este dado pode contribuir para encontrar padrões que se repetem apenas em determinado período. A Figura 4.4 mostra um exemplo de como é o dado categorizado após a conversão.

Figura 4.4: Discretização do *timestamp*.

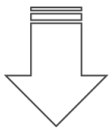
Diversas técnicas existentes na literatura podem ser utilizadas para categorizar a localização. Neste trabalho é utilizado a API *Places* do *Google Maps* [1]. Com esta API é possível reconhecer o local com base na latitude e longitude do usuário. Além disso, é possível saber o lugar e categoria do lugar no qual o usuário está. Por exemplo, dado a geolocalização latitude=-3.0944288 e longitude=-60.0225919, é possível obter a resposta de que o lugar é o Amazonas Shopping e que a categoria do lugar é shopping e/ou estabelecimento.

[1] https://cloud.google.com/maps-platform/

4.3 Preparação de Dados

Conforme explicado na Seção 2.6, os algoritmos de mineração de padrões sequenciais (SPM, *Sequencial Pattern Mining*) encontram os padrões mais frequentes dado um determinado limiar (suporte mínimo) em uma base de dados de sequências. Nas seções anteriores é descrito como "reconhecer" atividades e contexto no qual estas ocorrem dado um histórico. No entanto, apenas reconhecer os dados do acelerômetro, GSP, e tempo do *smartphone* não gera uma base de dados de sequências que é necessária para utilização de algoritmos de SPM. A Figura 4.5 exibe uma representação da base de dados antes e após o reconhecimento dos dados.

Timestamp	Latitude	Longitude	ACC_X	ACC_Y	ACC_Z
1486727454134	-3.0892974	-59.9644671	4.3177	-2.3699	-0.43585



Atividade	Categoria	Lugar	Horário	Tipo do dia	Dia	Dia da semana	Turno
Andar	Universidade	ICE	07:50:54	Trabalho	10/02/2017	Sexta	Manhã

Figura 4.5: Dados brutos após reconhecimento.

O primeiro passo para gerar a base de dados de sequências é ordenar os dados da base reconhecida em função do tempo. A partir desta nova base é preciso selecionar os dados de contexto que serão utilizados na mineração de padrões. O ideal é escolher os dados que mais contribuam para compreender o que o individuo fez e vai fazer. Além disso, a quantidade de dados a ser utilizada também tem que ser escolhida cautelosamente, pois inserir mais dados pode significar um maior tempo de processamento e de uso de memória. O capítulo 5 exibe alguns experimentos que ajudam a compreender a razão pela qual isto acontece.

A Figura 4.6 exibe a transformação da base de dados reconhecida em uma sequência. Três dados principais foram selecionados para este exemplo, os quais são: atividade, localização e horário. Cada célula da coluna corresponde a um item, e o conjunto dos itens selecionados formam um *itemset*. Conforme descrito no Capítulo 2, seja $X = \{i_1, i_2, ..., i_m\}$ um conjunto de literais (itens), um *itemset* é um conjunto não-vazio de itens e uma sequência é uma lista ordenada de *itemsets*.

Tal abordagem gera uma sequência contínua de *itemsets*. Algumas das abordagens utilizadas para representar esta sequência contínua como uma base de dados de sequências são a segmentação e janela deslizante. Neste trabalho é utilizado janela

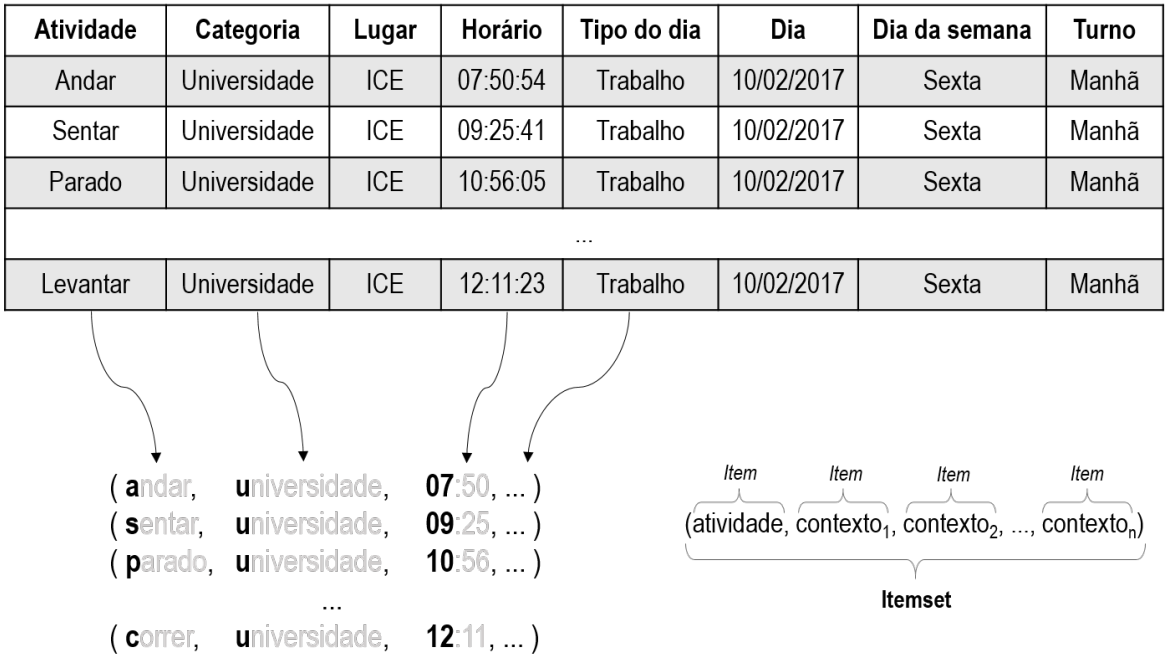


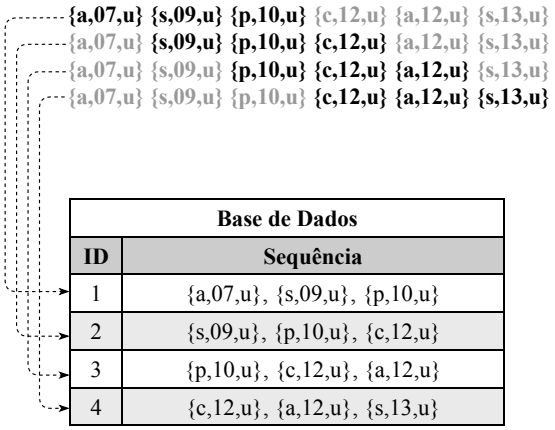
Figura 4.6: Criação da sequência a partir da base de dados.

deslizante para criar a base de sequências, pois faz mais sentido para o contexto desta dissertação. Janela deslizante (do Inglês, *Sliding Windows*) é uma técnica para dividir sequências em subsequencias com ou sem sobreposição. A definição formal desta abordagem é dada na Definição 4.3.1.

Definição 4.3.1 (Janela Deslisante)**.** Dada uma sequência de n itemsets $S = (i_1, ..., i_n)$, o janelamento de itemsets identifica um conjunto de x janelas de sequências, $P = (S_1, ..., S_x)$, com tamanhos $\{w_1, ..., w_n\}$, tal que S_i é uma subsequência ordenada de S. O conjunto das janelas é ordenado, não vazio, com possíveis sobreposições e $\bigcup_{i=1}^{i=x} S_i \supseteq S$. Uma janela S_i pode ser representada pela sequência $< e_i, e_{i+wi} >$ [Cook & Krishnan, 2015].

Neste trabalho, por padrão é usado um tamanho fixo de 4 *itemsets* para as janelas, com sobreposição de tamanho 1. No capítulo 5 são exibidos alguns experimentos que demonstram como o tamanho da janela e da sobreposição pode afetar os métodos de mineração de padrões.

A Figura 4.7 exibe o processo de transformação de uma sequência contínua em uma base de dados de sequência. Apenas para exemplificar, foi criada uma base fixando o tamanho da janela em 3, com *itemsets* de tamanho 3 e sobreposição de tamanho 2. A sobreposição refere-se a quanto de uma subsequencia estará contida na próxima sequência. Por exemplo, os dois últimos *itemsets* na linha 1 estão presentes na linha 2.



Base de Dados	
ID	**Sequência**
1	{a,07,u}, {s,09,u}, {p,10,u}
2	{s,09,u}, {p,10,u}, {c,12,u}
3	{p,10,u}, {c,12,u}, {a,12,u}
4	{c,12,u}, {a,12,u}, {s,13,u}

Figura 4.7: Representação da sequência em janelas.

4.4 Descobrimento de Padrões

A descoberta de padrões é uma das fases mais importantes do método de previsão de atividades. O trabalho utilizado como referência para este trabalho utiliza o AprioriAll (Seção 2.7) que é um algoritmo com alto custo computacional comparado às outras abordagens mais recentes, tanto em termos de tempo de processamento quanto em termos de uso de memória. Embora os *smartphones* tenham evoluído bastante, o

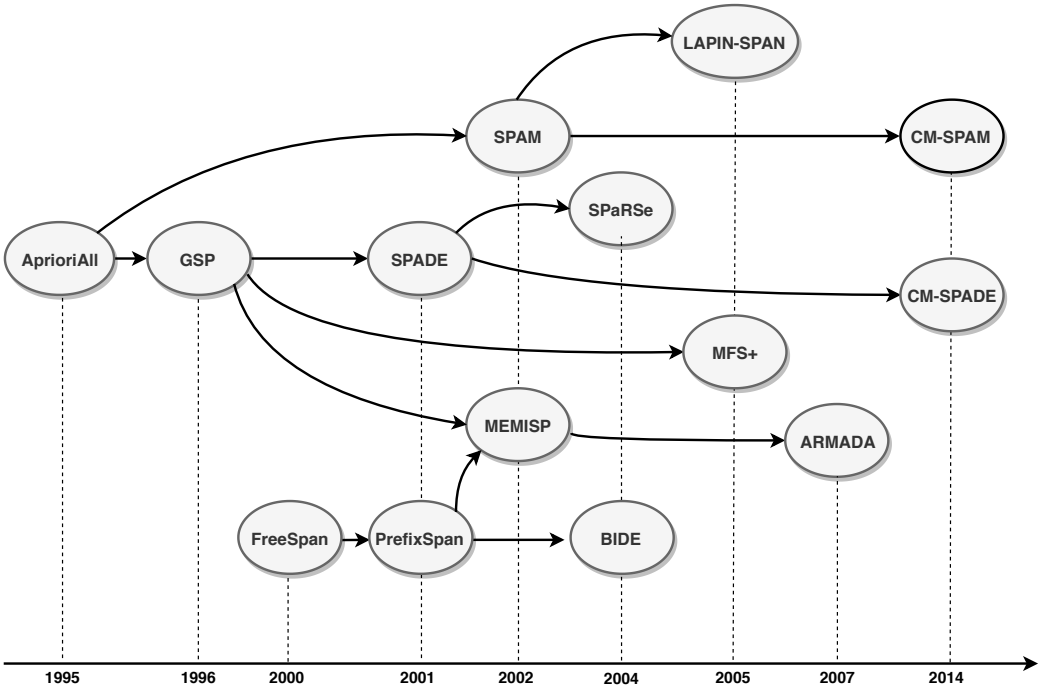


Figura 4.8: Algoritmos de SPM - Cronologia e suas principais influências. (Adaptado de João et al. [2015])

método de previsão desenvolvido precisa utilizar o mínimo de recursos possíveis. Desta forma, é possível utilizar o método de previsão tanto em *smartphones* com baixo poder de processamento, quanto poupar o consumo de energia em dispositivos mais potentes.

Portanto, é essencial que os algoritmos desenvolvidos utilizem a menor quantidade de processamento, na medida do possível. A Figura 4.8 exibe alguns algoritmos de SPM com ordem cronológica. As setas horizontais indicam a principal influência de cada algoritmo. Para este trabalho o algoritmo utilizado foi o CM-SPAM, que é uma versão do SPAM com algumas otimizações. A razão principal para esta escolha foi a sua estabilidade mesmo quando utilizado para minerar sequências longas em grande quantidade de dados [Ayres et al., 2002]. A seguir é descrito como o CM-SPAM funciona bem como algumas otimizações realizadas neste trabalho para o problema abordado.

4.4.1 Algoritmo CM-SPAM

O principal problema do SPAM, e de outros algoritmos que utilizam representação vertical da base de dados para gerar e contar candidatos, é que muitos candidatos não frequentes são gerados, principalmente nas fases iniciais. Para resolver este problema, o CM-SPAM [Fournier-Viger et al., 2014a] utiliza uma estrutura de dados chamada de mapa de coocorrência CMAP (*Co-occurence MAP*) para guardar informações de coocorrência. Desta forma, é possível reduzir a quantidade de candidatos gerados a cada fase. Para entender o funcionamento do mapa de coocorrência é preciso entender primeiramente as seguintes definições.

Definição 4.4.1. Um item k é dito ser sucedido por uma extensão de item (*i-extension*) para um item j em uma sequência $(I_1, I_2, ..., I_n)$ se j, $k \in I_x$ para um inteiro x tal que $1 \leq x \leq n$ e $k \succ_{lex} j$.

Definição 4.4.2. Um item k é dito ser sucedido por uma extensão de sequência (*s-extension*) para um item j e, em uma sequência $(I_1, I_2, ..., I_n)$ se $j \in I_v$ e $k \in I_w$ para algum inteiro v e w tal que $1 \leq v < w < n$.

Definição 4.4.3. Um mapa de coocorrência é uma estrutura que mapeia cada item $k \in I$ para um conjunto de itens que o sucedem. Como o SPAM opera com operações de I-step e S-step, também são definidos dois *CMAPs*. O $CMAP_i$, que mapeia cada item k para o conjunto $cm_i(k)$ para todos os itens $j \in I$ que sucedem k por extensão de item que não estão abaixo do suporte mínimo. Também é criado o $CMAP_s$ que mapeia cada item k para o conjunto $cm_s(k)$ para todos os itens $j \in I$ que sucedem k por uma extensão de sequência que não estão abaixo do suporte mínimo. A Figura 4.9 exibe um exemplo de como são criados os mapas para para uma base de sequências. Para

este exemplo foi considerado o suporte mínimo de $0,5$. Para o item b por exemplo, as possíveis extensões de sequências são $(\{b\},\{c\})$, $(\{b\},\{e\})$, $(\{b\},\{f\})$ e $(\{b\},\{g\})$. No entanto, se considerar apenas os itens acima do suporte estabelecido, restam apenas os itens e, f, g, que estão presentes no $CMAP_s$. O mesmo raciocínio se aplica ao $CMAP_i$.

SID	Sequência
1	({a,b}, {c}, {f,g}, {g}, {e})
2	({a,d}, {c}, {b}, {a,b,e,f})
3	({a}, {b}, {f}, {e})
4	({b},{f,g})

$CMAP_i$

item	sucedido por (i-step)
a	{b}
b	∅
c	∅
e	∅
f	{g}
g	∅

$CMAP_s$

item	sucedido por (s-step)
a	{b,c,e,f}
b	{e,f,g}
c	{e,f}
e	∅
f	{e,g}
g	∅

Figura 4.9: Mapas de coocorrência para a base de sequências. (Adaptado de Fournier-Viger et al. [2014a])

Para podar os os candidatos de uma forma mais eficiente, a poda baseada no mapa de coocorrência segue as seguintes propriedades.

- A **propriedade 1** foca especificamente em podar candidatos gerados a partir de uma extensão de item. Seja A um padrão sequencial frequente e seja k um item. Se existir um item j no último *itemset* de A, tal que k não esteja contido no conjunto de itens em $cm_i(j)$, então a extensão de item de A com k não é frequente. Prova: se um item k não aparece em $cm_i(j)$, então a extensão de item de k com j tem suporte abaixo do limiar.

- A **propriedade 2** segue uma metodologia similar, mas para extensões de sequências. Seja A um padrão sequencial frequente e k um item. Se existe um item $j \in A$ tal que k não está contido no conjunto $cm_s(j)$, então a extensão de A com k não é frequente. Prova: se um item k não aparece em $cm_s(j)$, então a sucessão de k para j por meio de uma extensão de sequência não é frequente.

- A **propriedade 3** generaliza as propriedades anteriores, mas de forma que poda todos os candidatos que iniciem com determinado prefixo. Seja A um padrão sequencial e seja k um item. Se existe um item $j \in A$ tal que k não está contido em $cm_i(j)$ e/ou $cm_s(j)$, então todas as supersequências, formadas por extensões de itens e/ou extensões de sequências, não são frequentes. Prova: se um item k não está contido em $cm_i(j)$ ou $cm_s(j)$ então a sucessão de k e j não são frequentes e por conseguinte, todas as sequências com este prefixo também não são frequentes.

O CMAP é utilizado no procedimento de busca (algoritmo 3) do SPAM. Seja *pad* um padrão sequencial considerado para ser estendido por item ($x \in I_n$) ou por sequência ($x \in S_n$). Se o último item em a do padrão *pad* não contem um $x \in cm_s(a)$ ou $x \in cm_s(a)$, então o padrão resultante de *pad* e x não é frequente (propriedade 1 e propriedade 2 do CMAP). Desta forma, x não precisa ser adicionado como um possível candidato em I_{temp} e S_{temp}, sendo podado antecipadamente. Portanto, não é necessário verificar todas as extensões de x futuras, causando assim uma economia de processamento.

Os métodos de SPM costumam encontrar todos os padrões sequenciais possíveis para as sequências existentes na base. No entanto, para o problema abordado, apenas os padrões que ocorrem exatamente após ao outro são interessantes. Por exemplo, o padrão "almoçar, dormir" não é interessante se "dormir" não tiver ocorrido imediatamente após almoçar. Logo, não há razões para manter a contagem deste tipo de padrão. Para podar tais candidatos, após a sua geração, verifica-se se este ocorre imediatamente após a sequência, caso não ocorra é podado. Tal abordagem faz com que mais candidatos sejam podados e desta forma a execução seja mais rápida.

4.5 Modelo de Previsão

Conforme exibido na Figura 4.10, a geração do modelo envolve três etapas principais. O padrões sequenciais frequentes extraídos na etapa de mineração de sequências são a entrada para esta etapa. A partir de tais padrões é realizado um cálculo de probabilidade para cada padrão sequencial encontrado. Como algumas probabilidades podem ter valor igual a 0, é necessário uma suavização, que é feito na etapa seguinte. A partir destas duas etapas, um modelo pode ser gerado. No entanto, uma melhoria que é realizada é a de busca de padrões aproximados, que é útil especialmente quando algum novo padrão surge e ainda não foi mapeado. Após tais etapas o método tem como saída a próxima atividade que será realizada.

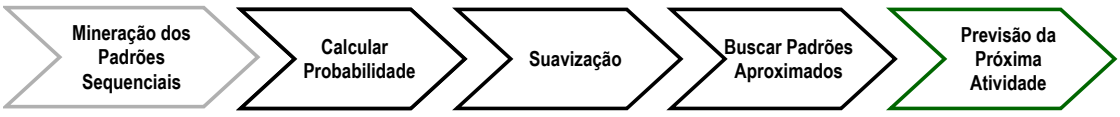


Figura 4.10: Visão geral das etapas de criação do modelo de previsão probabilístico.

4.5.1 Padrões Sequenciais Frequentes

Nesta fase, todos os padrões sequenciais considerados frequentes com base no suporte mínimo estabelecido estão prontos para utilização. O exemplo exibido na Figura 4.11 ajuda a entender melhor como os dados de entrada desta etapa estão estruturados. Dada uma base de dados de sequências D e um valor mínimo de suporte ($minsup = 0,3$), os padrões que ocorrem em uma quantidade acima do suporte mínimo são armazenados em uma tabela. A base D contém exemplos de sequências que podem ocorrer durante um dia. Para esta base, foi considerado que apenas dois dados de contexto são utilizados (*localização e tempo*). Além disso, o tempo foi categorizado como "horário de expediente" e "horário de almoço".

Um exemplo de padrão frequente na base é o padrão $\{l\},\{a\}$ que ocorre em $6/8$ vezes na base e portanto tem suporte igual a $0,75$. As supersequências obtidas a partir deste padrão também podem ser frequentes. Como o padrão $\{l,t\},\{a,t\},\{p\}$, que ocorre em $3/8$ vezes na base. Além disso, todas as subsequências do padrão $\{l,t\},\{a,t\},\{p\}$ também são frequentes de acordo com a propriedade Apriori.

4.5.2 Cálculo das Probabilidades

O problema de calcular qual a probabilidade de uma atividade ocorrer torna-se bem menos complexo quando se conhece todos os padrões. Nesta etapa, considera-se que todos os padrões frequentes e seus respectivos suportes são conhecidos. O entendimento das seguintes definições são necessárias para compreender como o cálculo é realizado.

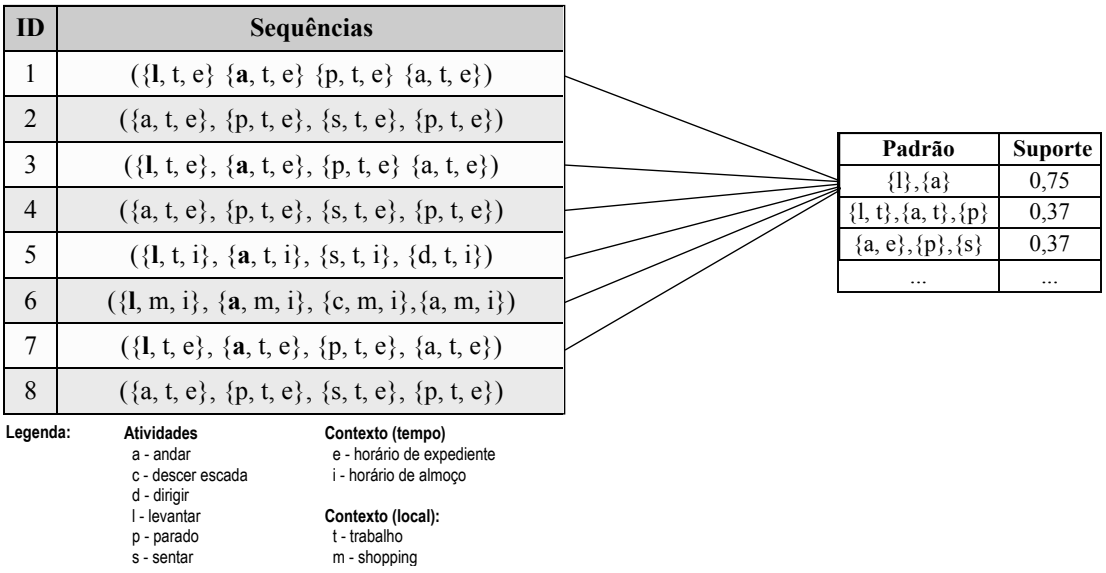


Figura 4.11: Exemplo de padrões frequentes com seu respectivo suporte.

- Cada item de uma sequência, seja ele a representação de uma atividade ou de um dado de contexto, é considerado um símbolo de um alfabeto Θ [2]. Ou seja, o alfabeto Θ é composto por todos itens que podem ocorrer em uma base de dados de sequências.
- PS representa o conjunto de todos os padrões sequenciais frequentes de uma base de dados de sequências $\mathcal{D}$.
- A próxima atividade ou a atividade a ser prevista é representada por σ onde, $\sigma \in \Theta$.
- A função que representa a probabilidade condicional de um símbolo σ ocorrer é representada por $\gamma_s(\sigma)$.
- PS_k representa o conjunto de padrões sequenciais de comprimento k. Esta definição segue o mesmo princípio de uma k-sequência, ou seja, representa a quantidade de *itemsets* contidos no padrão sequencial.
- A função *sup*() é uma função que consulta o suporte de padrão sequencial na tabela de padrões frequentes.

Para cada padrão sequencial frequente PS_k, onde $k \geq 2$, a probabilidade de uma atividade ocorrer em um itemset k pode ser calculada utilizando a regra de Bayes para cada atividade em Θ. Considerando que s é um prefixo de tamanho $|s| = n - 1$, onde, $2 \leq n \leq k$ é o tamanho do padrão sequencial. Então a probabilidade pode ser computada utilizado a equação 4.1 se $\sigma \in PS_k$.

$$P(\sigma|s) = \frac{sup(s\sigma)}{sup(s))} \tag{4.1}$$

Caso $\sigma \notin PS_k$, é utilizada a equação 4.2 marcando uma probabilidade não existente com o valor zero.

$$P(\sigma|s) = 0 \tag{4.2}$$

A Figura 4.12 exibe um exemplo de como é realizado este cálculo para um conjunto de padrões frequentes com $k = 3$. Dado que o determinado usuário realizou as atividades "andar" e "sentar", é calculado a probabilidade de cada atividade ocorrer. Para o problema abordado neste trabalho, não é calculado a probabilidade de a próxima atividade ser a mesma atividade que atual, pois o método desenvolvido tem como

[2] Normalmente um alfabeto costuma ser representado pela letra Σ [Menezes, 1998]. No entanto, neste contexto, para evitar confusão com o somatório ($\sum$), foi adotado o símbolo Θ.

foco prever qual vai ser a próxima atividade (diferente da atividade atual) que vai ocorrer, conforme exibido na Figura 4.12. Além disso, na etapa de geração das sequências todas as repetições de atividades são removidas, por exemplo, "andar, andar, sentar" é transformado em "andar, sentar". Então mesmo que nesta etapa seja considerado que a próxima atividade possa ser a mesma que a atual, tal a probabilidade sempre seria nula.

Padrão	Suporte
{andar},{sentar},{parar}	0,30
{andar},{sentar},{dirigir}	0,20
{andar},{sentar},{deitar}	0,10
{andar},{sentar},	0,50

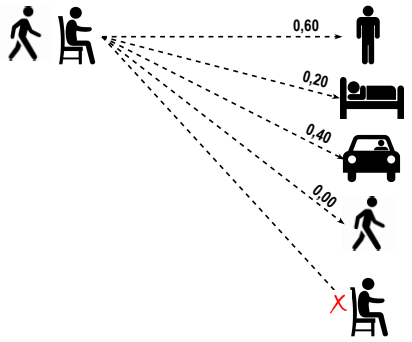


Figura 4.12: Cálculo de probabilidades utilizando apenas atividades.

No caso apresentado na Figura 4.12 a atividade com maior probabilidade de ocorrer é "ficar parado". No entanto, se houvesse mais informações sobre o contexto do usuário, esta previsão poderia ser bem diferente. A Figura 4.13 exibe como é o cálculo de probabilidades para o exemplo anterior, mas levando em conta os dados de contexto. Além de ter as informações de que as últimas atividades realizadas foram "andar" e "sentar", também é dada a informação que o local é a casa do usuário.

Padrão	Suporte
{a, m, 20},{s, m, 20},{p, m, 20}	0,10
{a, m, 22},{s, m, 22},{d, m, 22}	0,05
{a, c, 22}, {s, c, 22} ,{r, c, 22}	0,10
{a, m, 20},{s, m, 20}	0,10
{a, m, 22},{s, m, 22}	0,10
{a, c, 22}, {s, c, 22}	0,15

Legenda: **Atividades**
a - andar
d - dirigir
p - parado
s - sentar
r - deitar

Contexto (tempo)
0-23 - hora do dia

Contexto (local):
m - shopping
c - casa

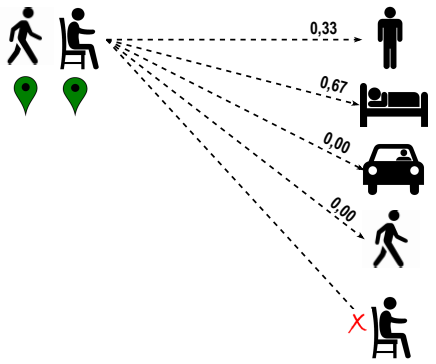


Figura 4.13: Cálculo de probabilidades com utilização de dados de contexto.

Como é de se esperar, a adição de tais dados torna a previsão mais eficaz. Com tal informação, ao utilizar os padrões frequentes, são calculadas as probabilidades de que cada atividade ocorra. Neste caso, a maior probabilidade é de "deitar", com 76% de probabilidade. Enquanto no caso anterior era de apenas 33%.

4.5.3 Suavização

Em alguns casos, $P(\sigma \mid s) = 0$, que acontece normalmente quando uma determinada atividade nunca acontece após determinada sequência (e.g.: pular após dirigir). Entretanto, probabilidades iguais a 0 (ou 1) geralmente são não realísticas, pois não se pode dizer que determinado evento é impossível de acontecer apenas pelo fato de não existir tal evento na base de treinamento. Para estes casos é necessário a utilização de métodos que estimem tal probabilidade. Tais métodos diminuem a probabilidade das sequências que possuem uma probabilidade maior que zero, para que reste uma fatia de probabilidade para as sequências que não estão contidas nos padrões sequenciais frequentes. A técnica que realiza este ajuste de probabilidades é conhecida como suavização (do inglês, *smoothing*).

Este trabalho utiliza a mesma técnica de Li & Fu [2014], descrita mais adiante, que é similar à técnicas existentes na literatura. Uma técnica de suavização bem conhecida é baseada na lei de *Laplace* [Juan & Ney, 2002], que também é chamada de *Adding One*. Esta técnica consiste em reservar uma pequena porção de probabilidade para as sequências que não existem nos dados de treinamento. Para tanto, adiciona-se 1 ao suporte de cada atividade em Θ. A equação 4.3 exibe como as probabilidades de σ podem ser suavizadas utilizando a estimativa de *Laplace*.

$$P_{LAP}(\sigma) = \frac{absSup(s\sigma)+1}{absSup(s)+|\Theta|} \tag{4.3}$$

No entanto, adicionar exatamente 1 pode não ser uma boa opção em alguns casos, pois adicionar 1 pode ser tanto um valor excessivo nos cenários no qual os suportes são baixos, quanto serem irrisórios quando os suportes são altos. Para tais casos, é possível adicionar um valor λ menor que 1. Tal abordagem é conhecida como lei de *Lidstone* e *Jeffrey-Perks*. A equação 4.4 exibe como é realizada a suavização das probabilidades utilizando esta técnica. Além disso, nos casos em que λ corresponde a exatamente 1/2, esta técnica também pode ser chamada de *Expected Likelihood Estimation* (ELE).

$$P_{LID}(\sigma) = \frac{absSup(s\sigma)+\lambda}{absSup(s)+|\Theta|\lambda} \tag{4.4}$$

A técnica adotada neste trabalho é bem similar à lei de *Lidstone* e *Jeffrey-Perks*. A equação 4.5 demonstra como é realizada a suavização das probabilidades. O valor de λ é definido empiricamente e é relacionado à uma probabilidade. O conjunto de todas atividades, exceto a atividade do último *itemset* de s em Θ, é representado por Δ. Logo, $\Delta = \mathcal{A} - s_n$, onde $\mathcal{A}$ é o conjunto de todas as atividades em Θ. Desta forma,

a probabilidade da próxima atividade ser a mesma que a atual não é calculada.

$$\gamma_s(\sigma) = (1 - \mid \Delta \mid \lambda) P(\sigma \mid s) + \lambda \tag{4.5}$$

A Tabela 4.2 exibe os valores finais das probabilidades exibidas na Figura 4.13 após a suavização considerando $\lambda = 0,05$. Desta forma, não existe nenhuma probabilidade nula.

Tabela 4.2: Suavização das probabilidades.

Atividade (σ)	**Probabilidade**	**Suavizada** $\gamma_s(\sigma)$
Parar	$0,33$	0,31
Deitar	$0,67$	0,59
Dirigir	$0,00$	0,05
Andar	$0,00$	0,05

4.5.4 Busca de Padrões Aproximados

Assim como pode existir casos nos quais alguns padrões sequenciais podem não conter determinada atividade, fazendo com que seja necessário suavizar as probabilidades, também existem casos nos quais todo o padrão pode não existir na base de treinamento. Um exemplo claro disto é quando um usuário dorme em sua casa durante todo seu histórico, mas em determinado dia ele se muda ou simplesmente decide dormir em um local diferente. O padrão sequencial com o novo local não é considerado frequente, pois existem instâncias de tal padrão na base, embora os padrões de rotina sejam os mesmos.

Visto que o método desenvolvido utiliza representação simbólica das atividades e dados de contexto, é possível caracterizar esse problema como o problema de *busca aproximada*. Uma vasta quantidade de algoritmos foi desenvolvida para resolver este problema. Normalmente, tais algoritmos são utilizados em correção de erros de digitação [Wilbur et al., 2006], reconhecimento de voz [Glass, 2003], reconhecimento óptico de texto [Ford et al., 2000], entre outros.

Damerau [1964] foi o primeiro a apresentar o problema e uma solução para corrigir palavras digitadas incorretamente. Utilizando operações de inserir, deletar, substituir e transpor caracteres, a solução proposta corrigiu 80% dos erros nos dados experimentados. Um pouco depois, Levenshtein [1966] desenvolveu uma função de similaridade de strings. Esta função, que é um algoritmo de programação dinâmica, mede quantas modificações devem ser feitas na string para que se obtenha a string correta. A equação

4.6 exibe, de forma simplificada, como funciona a função de recursão para o cálculo de distância.

$$lev_{a,b}(i,j) = min \begin{cases} 0 & \text{se } i = j = 0 \\ lev_{a,b}(i-1,j) + 1 & \text{se } i > 0 \\ lev_{a,b}(i,j-1) + 1 & \text{se } j > 0 \\ lev_{a,b}(i-1,j-1) + [a_i \neq a_j] & \text{se } i,j > 0 \end{cases} \tag{4.6}$$

O algoritmo de *Levenshtein* [3] utiliza essa função de distância para encontrar a correção de uma palavra. Além destes, existem diversos algoritmos e diversas métricas para mensurar a distância entre strings. O trabalho de Boytsov [2011] mostra uma análise completa destes algoritmos, bem como uma comparação entre os métodos. Por outro lado, Yu et al. [2016] fazem um estudo das principais métricas e algoritmos mais recentes, mostrando as vantagens e desvantagens de cada um. Estes trabalhos exibem como fazer uma busca aproximada, tanto de forma muito rápida e usando um alto processamento, quanto de forma mais lenta e utilizando uma implementação com baixo processamento. Nesta dissertação, optou-se por utilizar uma versão adaptada do algoritmo de *Levenshtein* devido ao fato de as strings possuírem um comprimento curto e fixo na maioria dos casos.

A versão simplificada do algoritmo utiliza apenas as operações de substituição e exclusão de caracteres da string. Por exemplo, com apenas uma operação de exclusão, a palavra w'=*algoritimo* pode ser corrigida, obtendo-se w=*algoritmo*. Analogamente, com apenas uma operação de substituição, a palavra w'=*previlégio* é retificada, retornando assim a palavra correta w=*privilégio*. A seção a seguir exibe com incorporar estas operações no método para previsão da próxima atividade.

4.5.5 Previsão da Próxima Atividade

Dado que todos os padrões frequentes foram categorizados, uma tabela de probabilidades foi criada relacionando os padrões a cada atividade e as probabilidades foram suavizadas. Com isso, a previsão da próxima atividade pode ser efetuada da seguinte forma: Verificam-se as n últimas atividades do histórico do usuário, com seus devidos dados de contexto, e realiza-se uma busca na tabela de probabilidades para verificação de tal padrão. Caso o padrão exista, a atividade com maior probabilidade de ocorrência

[3] http://www.levenshtein.net

é selecionada. Caso contrário, é feita uma busca aproximada na lista de padrões. A Figura 4.14 apresenta um fluxograma que descreve este processo.

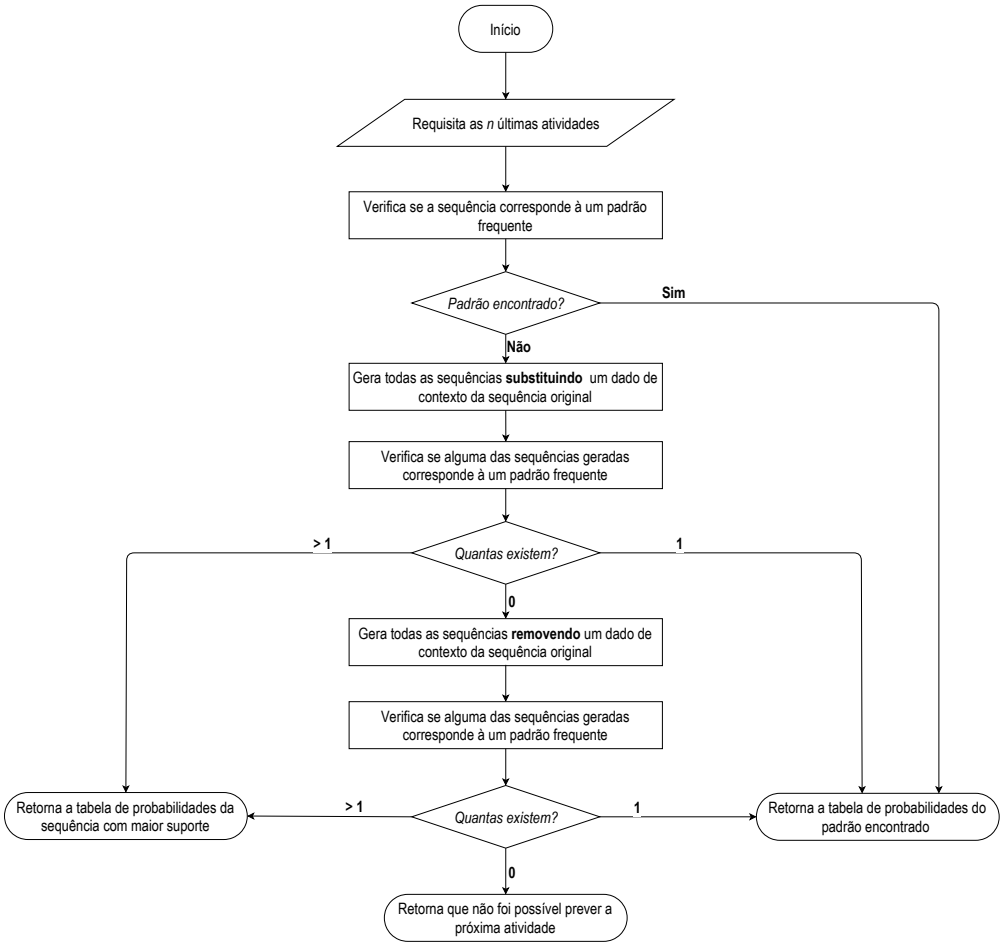


Figura 4.14: Fluxograma de funcionamento para previsão da atividade.

Como cada padrão sequencial frequente é formado por símbolos, cada *itemset* pode ser representado como uma palavra. Por exemplo, o padrão sequencial $< \{a,\ m,\ 20\}, \{s,\ m,\ 20\}, \{p,\ m,\ 20\} >$ pode ser representado como *am20, sm20, pm20*. Considerando que cada *itemset* é uma palavra ordenada lexicograficamente, é possível realizar uma busca aproximada para encontrar o *itemset* mais parecido com o padrão encontrado. Para tanto verifica-se se existe um padrão sequencial trocando apenas um dado de contexto. Caso exista, o padrão é retornado. Caso contrário, exclui-se um dado de contexto para nova verificação da existência do padrão. Em caso positivo, o padrão é retornado. Caso contrário, o mesmo processo é realizado para todos os *itemsets*. Caso após todas as trocas não exista determinado padrão sequencial, considera-se que não é possível fazer uma previsão útil a partir desta quantidade de dados. Isto pode ocorrer nos casos em que o usuário realiza uma sequência de eventos

nunca antes praticadas em um cenário totalmente desconhecido.

Utilizando a Figura 4.13 como exemplo, caso as últimas atividades fossem, "(andar, casa_dos_pais, 22), (sentar, casa_dos_pais, 22)", na etapa de substituição de dados este padrão poderia ser retornado caso fosse frequente. Caso não fosse, um padrão frequente poderia ser "(andar, 22), (sentar, 22)", removendo um dado de contexto. Esta abordagem funciona bem nos casos em que o usuário seguiu sua rotina de forma parcial. No entanto, nos casos em que o usuário realiza ações totalmente fora de sua rotina, o método desenvolvido trata tais ações como ruído.

4.6 Considerações Finais

Este capítulo apresentou os principais componentes do método de previsão de atividades proposto. Os parâmetros utilizados (e.g.: números de janelas, sobreposição, suporte mínimo) são explorados em mais detalhes no capítulo seguinte. As discussões sobre os motivos da utilização dos métodos e configuração dos parâmetros também são abordados no capítulo seguinte, juntamente com os resultados obtidos.

Conforme mencionado no início deste capítulo, a atividade alvo da previsão baseia-se na entrada de dados. Caso a entrada seja um conjunto de atividades básicas, a previsão será uma atividade básica. No entanto, um ponto importante a ser destacado é a utilização do método em cenários no qual apenas atividades básicas são reconhecidas, que é comum com a utilização de dados provenientes do *smartphone*. Neste caso, para prever atividades complexas (de alto nível), é preciso que as atividades básicas sejam transformadas em atividades complexas.

Existem técnicas para que seja possível reconhecer a atividade complexa que foi realizada pelo usuário a partir dos dados de atividades básicas. De modo geral, um método de descoberta de padrões é utilizado para descobrir os padrões frequentes e, utilizando algoritmos de classificação como o Naive Bayes, é possível descobrir a atividade complexa. Por exemplo, dado "levantar, andar, sentar, deitar" o método retorna que o usuário foi dormir [Liu et al., 2016]. Devido à este não ser o foco do trabalho, este algoritmo não foi incorporado no método desenvolvido.

No entanto, para fins experimentais, uma alternativa para o problema de prever uma atividade complexa a partir de dados de atividades básicas foi implementado no aplicativo *Sophia - SmartDay* (descrito em mais detalhes no capítulo seguinte). Utilizando os padrões frequentes encontrados na etapa de descoberta de padrões, o próprio aplicativo possui uma área de *feedback* para o usuário classificar o que cada padrão representa. Além desta área de *feedback*, o aplicativo também possui uma

área interativa que envia uma notificação solicitando o rótulo para cada sequência de padrões, conforme exibido na Figura 4.15.

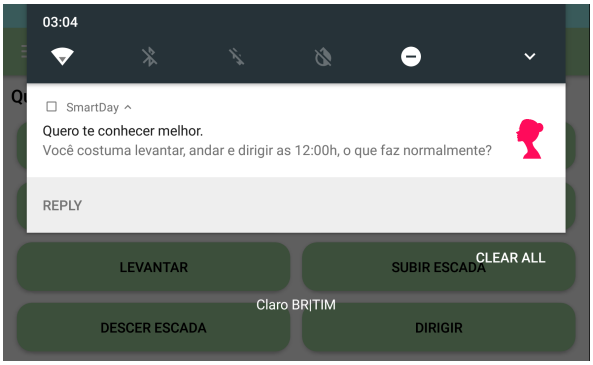


Figura 4.15: Solicitação de rotulagem por meio do aplicativo.

Outro método que também não é aprofundado neste trabalho é o de busca de sequência aproximada, por ser um tópico muito amplo. Neste trabalho, conforme citado anteriormente, foi utilizado uma versão do algoritmo de Levenshtein adaptada para o problema abordado. Os resultados obtidos neste e nos demais quesitos são apresentados no capítulo a seguir.

Capítulo 5

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados para avaliação do método SOPHIA para previsão de atividades humanas. Na seção a seguir um protocolo experimental é apresentado para avaliar o método em relação ao custo computacional e taxa de acerto. Logo após, as métricas e bases de dados são apresentadas. Mais adiante, são exibidos os resultados em relação ao tempo de processamento entre o algoritmo utilizado para minerar padrões sequenciais e as demais abordagens utilizadas na literatura. Mais adiante são exibidos os resultados relacionados à taxa de acerto do algoritmo. Logo após, é exibida uma breve discussão sobre o quanto é possível perder em termos de acurácia para ganhar em tempo de processamento.

5.1 Protocolo Experimental

Conforme descrito nos capítulos anteriores, o método de previsão desenvolvido é composto por dois componentes principais. O primeiro é responsável pelo descobrimento de padrões e o segundo pela geração do modelo de previsão a partir dos padrões frequentes encontrados. A forma como os padrões sequenciais são descobertos não tem impacto na geração do modelo de previsão. Logo, pode-se dizer que, mesmo que o segundo componente dependa da entrada obtida por meio do primeiro, ambos funcionam isoladamente.

Como os componentes funcionam de forma isolada, antes de avaliar o método como um todo, é importante que estes componentes sejam avaliados individualmente. Desta forma é possível compreender o quanto os métodos de mineração de padrões sequenciais impactam no desempenho do método, bem como o quanto a taxa de acerto do método é impactada pelos padrões encontrados.

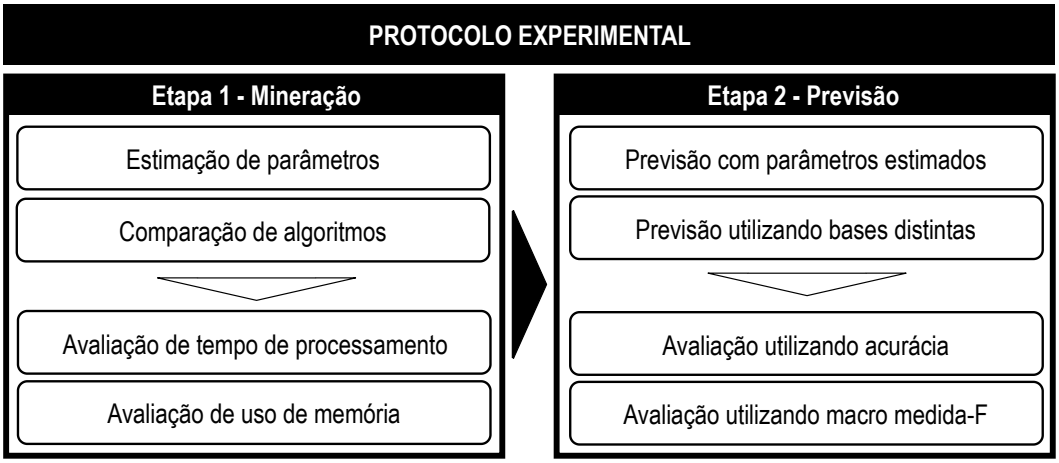


Figura 5.1: Dados utilizados na base desenvolvida.

Portanto, o protocolo experimental é dividido em duas etapas. A Figura 5.1 apresenta uma visão geral de como está organizado este protocolo. Na primeira etapa as avaliações são relacionadas apenas em relação ao custo computacional, tempo de processamento e uso de memória. Na segunda etapa, o modelo de previsão, que utiliza os padrões encontrados pelo componente anterior, é avaliado em relação à sua taxa de acerto. Os detalhes de cada etapa serão exibidos mais adiante.

Todos experimentos foram realizados utilizando uma máquina com processador *Intel* Core i7 de 3,20GHz e memória de 8GB, utilizando o sistema operacional *Windows 10*. Todos os algoritmos utilizados para realização dos testes foram implementados utilizando a linguagem de programação *Java*. Esta é uma das linguagens utilizadas na plataforma *Android*, que é a plataforma na qual o aplicativo *Sophia - SmartDay* (descrito mais adiante) foi desenvolvido.

5.1.1 Etapa 1 - Mineração

Considerando que os algoritmos de mineração de padrões sequenciais utilizam abordagens diferentes, os experimentos realizados são exibidos para: *i*) demonstrar que algoritmos como AprioriAll e GSP não são adequados para o problema em questão, devido ao seu custo computacional; *ii*) exibir em quanto o custo computacional é elevado à medida que o tamanho da sequência ou da quantidade de dias aumenta; *iii*) demonstrar que para o problema em questão, os algoritmos como CM-SPAM e SPADE são mais adequados.

Para tanto, nesta fase é avaliada a variação de parâmetros, tanto para a construção da base de sequências quanto para a mineração destas. Desta forma é possível filtrar o conjunto de parâmetros que poderão ser utilizados na segunda etapa do método

Tabela 5.1: Base com uma 3-sequência de 3-*itemsets*.

SID	Sequência
$a1$	$<(1000, 101, 802)(400, 101, 802)$**(100, 101, 802)** $>$
$a2$	$<$ **(100, 101, 802)**$(900, 101, 802)(800, 101, 802) >$
$b1$	$<(1000, 101, 802)$**(400, 101, 802) (100, 101, 802)** $>$
$b2$	$<$ **(400, 101, 802) (100, 101, 802)**$(900, 101, 802) >$

proposto. Além disto, nesta etapa é avaliado apenas o custo computacional (tempo e uso de memória) para a descoberta dos padrões frequentes. Cada um dos parâmetros avaliados nesta etapa são descritos a seguir. Os quatro primeiros parâmetros são relacionados à criação da base de sequências, enquanto os demais são relacionados exclusivamente à mineração dos padrões. A Tabela 5.1 é utilizada para exemplificar a variação dos parâmetros que são descritos adiante.

- **Tamanho da sequência/janela**: São realizados experimentos variando o tamanho da janela deslisante (de 3 à 5). O tamanho da janela corresponde ao tamanho da sequência em cada tupla da base de dados. A Tabela 5.1, por exemplo, apresenta uma base com tuplas de tamanho de janela igual à 3. O significado de cada item e como é composta a base de sequências para estes experimentos é detalhado na Seção 5.3.2.

- **Tamanho da janela de sobreposição**: De acordo com o tamanho da janela, são realizados experimentos para avaliar o impacto do tamanho da sobreposição. Por exemplo, se o tamanho da janela é 3, então durante os experimentos é avaliado o impacto de uma sobreposição de tamanho, 2, 1 e 0. A Tabela 5.1 exibe nas sequências $a1$ e $a2$ exemplos de uma sobreposição de tamanho 1 de nas sequências $b1$ e $b2$ sequências sobreposição de tamanho 2.

- **Quantidade de dados de contexto**: Experimentos são realizados variando a quantidade de dados de contexto utilizadas, sendo de no mínimo 2 (localização e tempo) e no máximo 5 (localização, tempo, tipo do dia, dia da semana e turno). Cada *itemset* da sequência é composto por uma atividade e n dados de contexto. Conforme apresentado na Tabela 5.1, que contém *itemset* de tamanho 3.

- **Dias**: No uso real, o algoritmo pode utilizar o histórico de n dias para avaliar e encontrar os padrões. O impacto que isto pode causar é avaliado em experimentos variando a quantidade de dias utilizados na criação da base de sequências.

- **Suporte mínimo**: Conforme o suporte mínimo *minsup* é decrementado, mais *itemsets* são considerados um padrão sequencial frequente. Logo, maior uso de recurso computacional. Este parâmetro é avaliado variando seu valor de 0,50 a 0,10 (ou 50% a 10%).

- **Algoritmos**: Os algoritmos avaliados em todos os quesitos acima mencionados são **SPAM** [Ayres et al., 2002], **CM-SPAM** [Fournier-Viger et al., 2014a], **SPADE** [Zaki, 2001], **CM-SPADE** [Fournier-Viger et al., 2014a] e **PrefixSpan** [Pei et al., 2004], que são considerados estado da arte. Além destes, também foi avaliado o **GSP** [Srikant & Agrawal, 1996], que é uma versão aprimorada do algoritmo usado por Li & Fu [2014].

5.1.2 Etapa 2 - Previsão

Nesta etapa do protocolo, a taxa de acerto do método é avaliada em dois cenários, cada um destes utilizando uma base de dados distinta, descritas na Seção 5.3. Devido às previsões serem realizadas utilizando dados discretos, as métricas utilizadas para avaliar o método de previsão neste trabalho seguem as mesmas utilizadas em demais trabalhos que realizam previsão em dados extraídos a partir de dispositivos móveis. As métricas utilizadas para avaliação são acurácia e macro medida-F, descritas na Seção 5.2. Para uma melhor visualização dos resultados também é gerada uma matriz de confusão para cada experimento.

A estratégia de avaliação utilizada para avaliação é a *leave-one-out*. Esta avaliação consiste em separar a base em dois conjuntos. O primeiro é o conjunto de teste, que é criado a partir de um único dia selecionado dentre os n dias da base. O segundo é o conjunto de treino que é formado a partir de todos os $n - 1$ dias, excluindo-se o alvo que compõe o primeiro conjunto. Na literatura, este tipo de avaliação também é chamada de *cross validation leave-one-out*. Além disso, nesta etapa o método também é avaliado de acordo com a variação de parâmetros obtidos na etapa 1.

5.2 Métricas

Na primeira etapa do protocolo experimental, como a avaliação é voltada apenas para custo computacional, são utilizadas apenas duas métricas. A primeira métrica é o tempo que o algoritmo leva para descobrir os padrões sequenciais frequentes, contabilizado em milissegundos (ms). A segunda métrica é o uso de memória, que neste caso é medido em Megabytes (MB).

Em relação à taxa de acerto, uma matriz de confusão é gerada a partir dos experimentos realizados, para visualização dos resultados de erro e acerto na previsão. Neste caso, a matriz de confusão é gerada a partir dos valores reais e das previsões. Os elementos necessários para construção da matriz são exemplificados na Tabela 5.2 e descritos a seguir, bem como a descrição das métricas de desempenho.

Tabela 5.2: Matriz de confusão para um problema de classificação com 3 classes.

		Valor previsto			
		1	2	3	*soma*
Valor verdadeiro	1	VP_{11}	FP_{12}	FP_{13}	NI_1
	2	FP_{21}	VP_{22}	FP_{23}	NI_2
	3	FP_{31}	FP_{32}	VP_{33}	NI_3
	soma	NT_1	NT_2	NT_3	*total*

As equações a seguir exibem como cada métrica é calculada matematicamente. Por fins de simplificação na exibição das fórmulas, as seguintes variáveis são definidas: *i*) k é quantidade de classes que ocorrem na base; *ii*) VP_{ii} (Verdadeiro-Positivo) é a quantidade de amostras de teste da classe i que foram previstas corretamente para a atividade i; *iii*) FP_{ij} (Falso-Positivo) é o total de amostras de teste da classe i que foram incorretamente previstas com a atividade j; *iv*) NI_i é a quantidade de amostras de teste da classe i; *v*) NT_j o total de amostras de testes que foram classificadas com a atividade j. As equações 5.1 e 5.2 exibem como calcular NI_i e NT_j, respectivamente.

$$NI_i = VP_{ii} + \sum_{j=1, j\neq i}^{k} FP_{ij} \ ; \tag{5.1}$$

$$NT_j = VP_{jj} + \sum_{i=1, i\neq j}^{k} FP_{ij} \ ; \tag{5.2}$$

Acurácia representa a taxa de acerto de forma geral e pode ser interpretada como a probabilidade de se prever uma atividade corretamente. Ela pode ser determinada pela quantidade de atividades que foram previstas corretamente em razão da quantidade total de atividades. A equação 5.3 exibe como a acurácia é calculada.

$$Acurácia = \frac{\sum_{i=1}^{k} VP_{ii}}{total} = \frac{\sum_{i=1}^{k} VP_{ii}}{\sum_{i=1}^{k} NI_i} = \frac{\sum_{j=1}^{k} VP_{jj}}{\sum_{j=1}^{k} NT_j} \tag{5.3}$$

A **Precisão** representa a média ponderada da fração dos rótulos previstos que estão classificados corretamente para cada classe de atividade. A equação 5.4 exibe

como pode ser calculada a precisão para a classe i.

$$Precisão_i = \frac{VP_{ii}}{NT_i} \tag{5.4}$$

Revocação representa a média ponderada da fração dos rótulos verdadeiros que foram corretamente classificados para cada classe de atividade. A equação 5.5 exibe como pode ser calculada a revocação para a classe i.

$$Revocação_i = \frac{VP_{ii}}{NI_i} \tag{5.5}$$

A **Medida-F** (do Inglês *F-score*) proporciona uma maneira de combinar precisão e revocação em uma única métrica. A Medida-F é calculada utilizando a equação 5.6.

$$Medida\text{-}F_i = \frac{2 \cdot Precisão_i \cdot Revocação_i}{Precisão_i + Revocação_i} \tag{5.6}$$

Além destas, para lidar com possíveis problemas de desbalanceamento das classes nas bases de dados, a Macro Medida-F pode ser utilizada em substituição à média da Medida-F. A equação 5.7 exibe como é possível calcular a Macro Medida-F.

$$Macro\ Medida\text{-}F = \frac{\sum_{i=1}^{k} w_i \cdot \text{Medida-F}_i}{\sum_{i=1}^{k} w_i} \tag{5.7}$$

onde Medida-F$_i$ representa a Medida-F para a classe i e w_i corresponde ao número de instâncias da classe i.

5.3 Base de Dados

Neste trabalho são utilizadas duas bases de dados, a primeira é a *Human Morning Routine Dataset* e a segunda é uma base criada durante o desenvolvimento deste trabalho, chamada de *Human Daily Activities Dataset*, ambas são descritas nas seções seguintes. Vale ressaltar que as bases públicas que contém dados extraídos por meio do *smartphone* são criadas com o propósito de reconhecer atividades, logo, tais bases não são adequadas para ser utilizadas para previsão. Pois estas bases possuem apenas dados dos voluntários executando as atividades que foram solicitadas, mas sem uma rotina natural (e.g.: sentar, levantar, correr, andar e pular) [Kwapisz et al., 2011; Reiss & Stricker, 2012; Anguita et al., 2013; Micucci et al., 2017]. No entanto, como o método desenvolvido se adapta bem à atividades de diferentes níveis, simples e complexas, o método é avaliado nos dois cenários que são descritos a seguir.

5.3.1 *Human Morning Routine Dataset - HMR*

A base *Human Morning Routine - HMR* foi disponibilizada pela Universidade Técnica de Munique. Esta base descreve as rotinas matinais de um morador. Os dados foram capturados utilizando o Kinect, de forma que a coleta foi não intrusiva. Os criadores da base também desenvolveram cenários simulados, criados a partir das informações prestadas pelo morador, como o horário e a duração da atividade realizada.

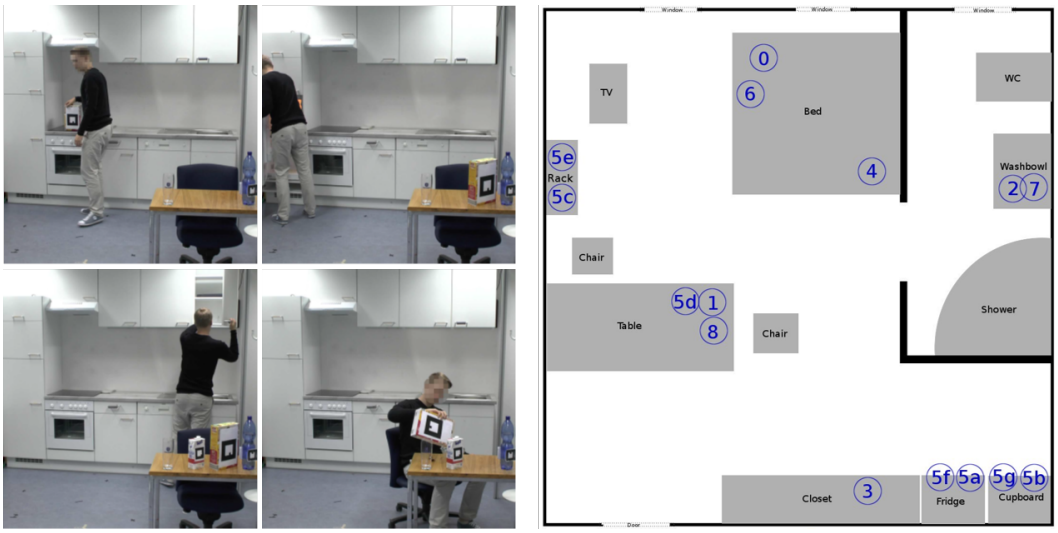


Figura 5.2: Cenário utilizado para criação da base HMR [Karg & Kirsch, 2014]. Lado esquerdo: morador realizando as tarefas matinais. Lado direito: planta do apartamento com marcação da ordem na qual as atividades são realizadas.

Tabela 5.3: Atividades e localização da base HMR.

Atividade	**Localização**
Clean table bread	*Refrigerator*
Clean table cornflakes	*Sink*
Clean table quark	*Table*
Drink water	*Oven*
Prepare bread	*Bottle place*
Prepare cornflakes	*Cupboard*
Prepare quark	*Door*
Prepare work	*Drawer*

A Figura 5.2 exibe o cenário no qual as atividades ocorrem. No cenário utilizado neste trabalho é utilizado apenas os dados que foram coletados pelo Kinect que se encontra na cozinha. A base utilizada neste trabalho contém 8 atividades e 8 dados de contexto principais relativos às localizações do morador. As atividades e localizações

disponibilizadas na base são exibidas na Tabela 5.3. Além destes dados, também é disponibilizado o tempo inicial e final para cada atividade realizada.

Embora esta base não contenha dados obtidos por meio de dispositivos móveis, que é a proposta do trabalho, boa parte das atividades que o usuário realiza no cenário são possíveis se obter por meio de sensores vestíveis.

5.3.2 *Human Daily Activities Dataset - HDA*

Como a proposta deste trabalho foca em como prever atividades humanas utilizando dispositivos móveis, além de usar base de rotinas matinais também foi criada uma base de dados com os dados de atividades realizadas durante todo o dia, chamada de *Human Daily Activities Dataset - HDA*. Como os métodos de reconhecimento de atividades atuais costumam reconhecer apenas atividades básicas, foram coletados apenas os dados de atividades básicas.

A coleta dos dados foi realizada com o auxilio de duas aplicações Android. A Figura 5.3 exibe as telas das aplicações utilizadas. A primeira aplicação foi criada durante o desenvolvimento desta dissertação. Esta é chamada de *Sophia - Smart Day* (Figuras 5.3a e 5.3b) e serve para as seguintes tarefas: *i*) coletar dados do acelerômetro; *ii*) rotular atividades; *iii*) rotular rotinas a partir das atividades reconhecidas; *iv*) testar se o método desenvolvido para prever atividades funciona em dados reais.

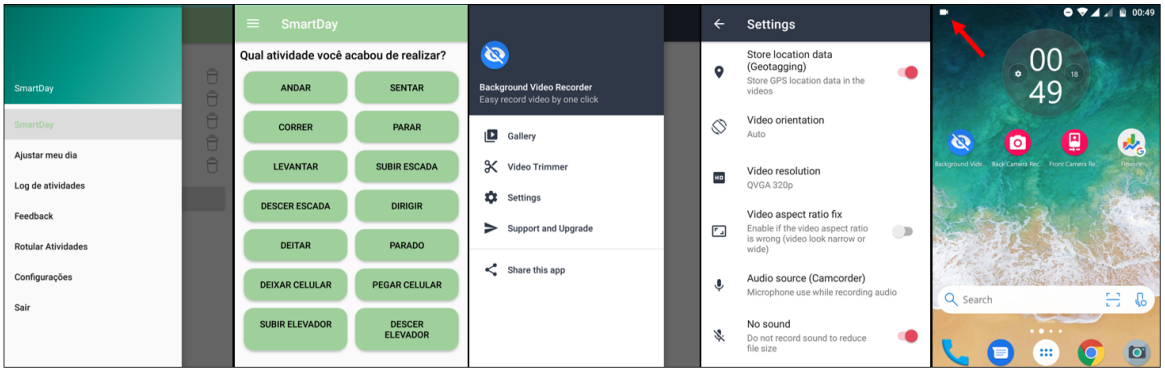


Figura 5.3: Aplicações utilizadas para coleta de dados.

A segunda aplicação é o *Background Video Recorder*[1] (figuras 5.3c, 5.3d e 5.3e), que foi utilizada para filmar o dia a dia do usuário, utilizando a câmera traseira ou até mesmo a frontal. Com este aplicativo é possível filmar em *background* tudo que o usuário faz, sem atrapalhar o uso do *smartphone*. Desta forma, é possível usar o

[1]https://play.google.com/store/apps/details?id=com.kimcy929.secretvideorecorder

mesmo dispositivo para coletar dados do acelerômetro, vídeo e também os rótulos que o usuário marcou como atividade.

Devido à possibilidade de o usuário nem sempre lembrar de rotular a atividade, o ideal é verificar se os dados rotulados estão de acordo com os vídeos coletados manualmente. Outro problema que pode acontecer é que os *smartphones* não costumam durar o dia todo sem precisar recarregar. Portanto, o usuário precisa estar atento para não deixar o dispositivo descarregar, ou salvar o vídeo antes que isto aconteça. Dependendo do *smarthpone* pode ser necessário que em alguns momentos não seja possível filmar a todo momento devido à limitação de bateria do dispositivo.

A base de dados foi criada utilizando 1 participante do sexo masculino e teve duração de três dias, sendo o início da coleta por volta das 08:00h e o fim por volta das 22:00h. O participante foi instruído a colocar e dispositivo no bolso com a parte da câmera frontal visível, conforme exibido na Figura 5.4a e deixar o dispositivo gravando. O participante também foi instruído a rotular cada atividade realizada com o aplicativo disponibilizado e após rotular devolver o dispositivo para o bolso. Os vídeos foram utilizados apenas para validar que as atividades rotuladas no aplicativo são as mesmas que foram realizadas.

As demais imagens da Figura 5.4 exibem o usuário dirigindo, saindo do carro, andando e sentado em seu local de trabalho. Como trabalho futuro pretende-se adicionar mais participantes e aumentar a duração do experimento. O dispositivo utilizado foi um *Motorola Moto G4*, os vídeos foram gravados em baixa qualidade por fins de espaço de armazenamento. Cada hora do vídeo em baixa qualidade utiliza apenas cerca de 60MB de espaço. Além disso, também com intuito de poupar espaço de armazenamento, não foi gravado o áudio durante o vídeo. A localização, embora fosse possível obter por meio do dispositivo utilizando o GPS, foi rotulada manualmente por fins de economia de energia. Durante maior parte do tempo de coleta de dados, o *smartphone* não estava conectado em nenhuma rede.

Figura 5.4: Coleta de dado utilizando dispositivo móvel.

Devido à pouca quantidade de dados coletados, foram adicionados outros dias sinteticamente, seguindo os padrões encontrados nos dias anteriores. Por exemplo, se nos dias 1, 2 e 3 o usuário realizou a ação dirigir ás 7h, 8h e 9h, respectivamente, o novo dia será criado com base na média de dois dias selecionados aleatoriamente. Para os dados não numéricos é escolhido aleatoriamente um dos dados. Devido à existência de dependências em alguns dados (e.g. se sair dirigindo para almoçar, volta para o trabalho dentro de uma hora), estas regras foram identificadas e tratadas para os dias sintéticos. Esta regra também foi aplicada para os horários: caso o horário de entrada no trabalho seja mais tarde, a saída também será mais tarde.

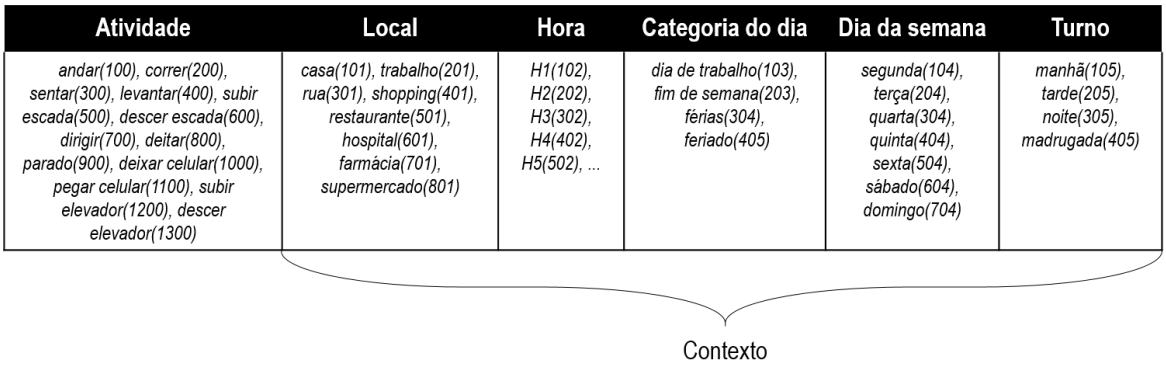


Figura 5.5: Dados utilizados na base desenvolvida.

A Figura 5.5 exibe os dados de atividades e dados de contexto presentes na base, bem como um código único associado a cada atividade, que são utilizados nos exemplos. Para realizar testes de busca de padrões entre os algoritmos de mineração de padrões foi gerada uma base com mais dados sintéticos. Esta mesma abordagem é utilizada nos demais trabalhos que realizam experimentos em mineração de padrões sequenciais [Ayres et al., 2002]. Tal abordagem é bem útil, pois desta forma é possível criar bases com sequências longas e curtas bem como variar o tamanho dos *itemsets* presentes na sequência.

Tabela 5.4: Formato da base de dados de sequências.

SID	**Sequência**
id	$< (a_1, c_{11}, ..., c_{1m})(a_2, c_{21}, ..., c_{2m})\ ...\ (a_n, c_{n1}, ..., c_{nm}) >$
1	$< (1000, 101, 802, 103)(400, 101, 802, 103)(100, 101, 802, 103) >$

A Tabela 5.4 exibe alguns exemplos de como são formadas as tuplas de uma base de dados de sequências após a etapa de preparação de dados (Seção 4.3). Por fins didáticos, tanto as atividades quanto os dados de contexto são representados por identificadores, a relação entre cada identificador e seu significado pode ser encontrado

na Figura 5.5. Na primeira linha é exibido um formato de como é cada uma tupla. Cada tupla é formada por $n > 0$ *itemsets*, onde cada *itemset* é composto por uma atividade a e m dados de contexto c. A segunda linha exibe um exemplo de uma tupla de *itemsets* de tamanho 4, com tamanho da sequência/janela igual a 3, pois existem 3 *itemsets* na sequência.

A partir da base gerada, foram geradas diversas bases menores, variando a quantidade de dados de contexto, os dias, o tamanho das sequências. De forma que seja possível analisar o comportamento dos algoritmos nestes diferentes cenários, pois existem alguns fatores que podem fazer com que o método leve mais tempo para minerar padrões. As seções a seguir exibem como cada um desses fatores pode influenciar no custo do método.

5.4 Tempo de Processamento

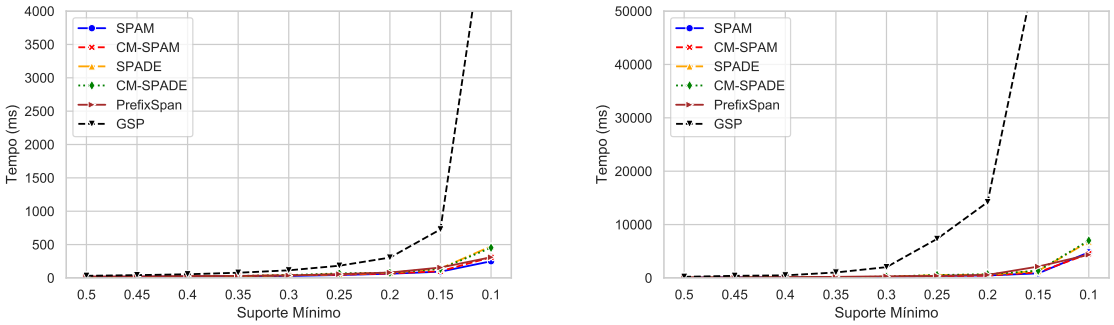
Nesta seção são exibidos os experimentos relacionados apenas ao tempo de processamento para obtenção dos padrões frequentes. Devido a alguns experimentos levarem mais tempo que outros, o valor máximo setado para o eixo y nem sempre é o mesmo em todos os gráficos. Tal abordagem é necessária para uma melhor visualização do comportamento dos algoritmos em cada cenário.

A seção 5.4.1 exibe os resultados do tempo de processamento dos algoritmos em função do suporte mínimo, variando parâmetros de tempo, quantidade de contexto, tamanho da janela e sua sobreposição. E a seção 5.4.2 exibe experimentos com variação de parâmetros similares, mas em função da quantidade de dias utilizados.

5.4.1 Tempo de Mineração em Relação ao Suporte

Os experimentos a seguir demonstram em diversos cenários como os algoritmos se comportam diante de alterações no tamanho da janela, quantidade de dados de contexto e também do tamanho de sua sobreposição. Os cenários testados levam em conta uma sequência de um dia da base, com alterações no tamanho da sequência e na quantidade de dados de contexto presentes na sequência.

A Figura 5.6 exibe uma comparação do tempo de processamento ao variar a quantidade de itens em cada *itemset*. Para este experimento foi utilizada uma base de sequências com janelas de tamanho 5 e com sobreposição de janela de tamanho 4. Incrementar o tamanho dos itens em cada *itemset* geralmente implica em maior quantidade de padrões existentes, ou seja, maior custo para encontrar os padrões. Na prática



(a) *Itemsets* de tamanho 4.

(b) *Itemsets* de tamanho 5.

Figura 5.6: Comparação do tempo de processamento com variação do tamanho dos *itemsets* em relação ao suporte.

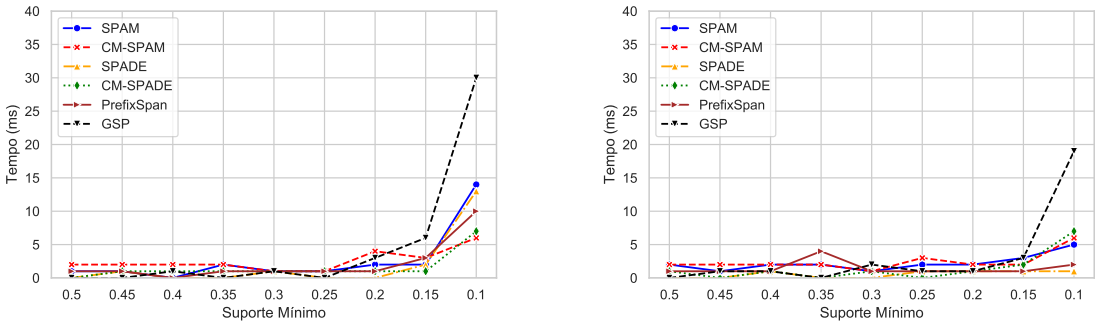
esse incremento significa aumentar a quantidade de dados de contexto incorporados na sequência.

A Figura 5.6a exibe um gráfico com o tempo de processamento para minerar padrões variando o suporte mínimo em uma base com *itemsets* de tamanho 4. Neste cenário, os algoritmos com o menor tempo de processamento foram SPAM, CM-SPAM e PrefixSpan, que ao serem avaliados com um suporte mínimo igual à 0,10, levaram 250, 310 e 310 milissegundos respectivamente para encontrar os padrões. Neste mesmo caso, o GSP durou 5906 *ms* para encontrar os padrões.

A Figura 5.6b exibe o resultado do experimento realizado com *itemsets* de tamanho 5, ou seja, uma atividade e quatro dados de contexto (atividade, local, hora, tipo do dia). Tal incremento resultou em um maior tempo de processamento para todos os algoritmos avaliados, em todos os níveis de suporte. Neste caso, os algoritmos que duraram o menor tempo para encontrar os padrões foram PrefixSpan, CM-SPAM e SPAM, com duração de 4343, 4527 e 4803 milissegundos. O GSP durou mais tempo em todos os cenários, atingindo 14202 *ms* ao ser avaliado com o suporte mínimo de 0,20 e ultrapassando 60000 *ms* ao ser avaliado com suporte mínimo de 0,15.

A Figura 5.7 exibe uma comparação do tempo de processamento ao variar o tamanho da janela de sobreposição. Para este experimento, foi utilizada uma base de sequências com janelas de tamanho 3 e *itemsets* de tamanho 3, ou seja, uma atividade e dois dados de contexto (local e hora).

A Figura 5.7a exibe um gráfico com o tempo de processamento para minerar padrões variando o suporte mínimo em uma base com janela de sobreposição igual à 1. Neste cenário, de acordo com o nível de suporte mínimo, alguns algoritmos foram mais rápidos que outros. Com suporte mínimo igual à 0,10, os algoritmos mais rápidos foram CM-SPAM e CM-SPADE, que levaram 6 e 7 milissegundos para encontrar os



(a) Sobreposição de janela = 1.

(b) Sobreposição de janela = 2.

Figura 5.7: Comparação do tempo de processamento com variação do tamanho de sobreposição de janela em relação ao suporte.

padrões, respectivamente. O que levou mais tempo foi o GSP, com duração de 30 *ms*.

Na Figura 5.7b o tamanho da sobreposição é incrementado para 2. Assim como no experimento com tamanho de sobreposição igual à 1, houve variação dos algoritmos que levaram o menor tempo de acordo com o nível de suporte. No entanto, para o caso mais demorado (suporte mínimo de 0,10) os algoritmos que levaram menos tempo foram SPADE e PrefixSpan, durando apenas 1 e 2 milissegundos para encontrar os padrões, respectivamente. O GSP foi o mais lento entre os avaliados, durando 19 *ms* e o CM-SPAM e CM-SPADE mantiveram o tempo de 6 *ms* execução. Como a base utilizada neste experimento contém poucos dados, não foi possível notar uma alta discrepância no tempo de execução.

A partir da análise dos gráficos exibidos acima é possível constatar que: *i*) os algoritmos levam um tempo maior para encontrar os padrões conforme o suporte mínimo é incrementado; *ii*) o tamanho da sobreposição não causa impacto significativo no tempo de execução dos algoritmos; *iii*) conforme o tamanho da sequência ou a quantidade de itens em cada *itemset* aumenta, o tempo de execução também aumenta; *iv*) o algoritmo GSP, que é a versão aprimorada do algoritmo AprioriAll, tem o maior tempo de execução em todos os casos.

5.4.2 Tempo de Mineração em Relação aos Dias

Os experimentos a seguir demonstram como os algoritmos de mineração se comportam conforme o histórico analisado aumenta e também ao variar parâmetros como o tamanho da sequência e dos dados de contexto. Para todos os experimentos é fixado um suporte mínimo arbitrário.

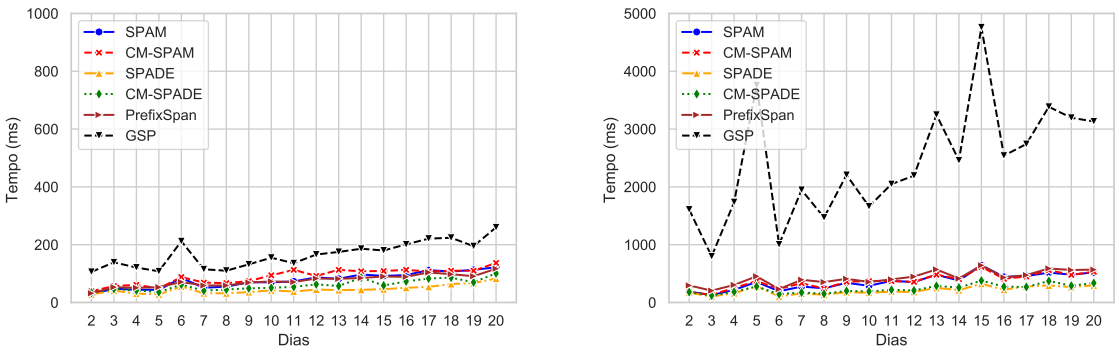
A Figura 5.8 exibe uma comparação do tempo de processamento ao variar o tama-

nho da sequência/janela. Para este experimento, foi utilizada uma base de sequências com *itemsets* de tamanho 5, ou seja, uma atividade e 4 dados de contexto (atividade, local, hora, categoria do dia) e suporte mínimo igual à 0,20. Incrementar o tamanho da sequência também implica em uma maior quantidade de padrões existentes. Logo, o custo também é maior para encontrar os padrões. Como é de se esperar, o tempo de execução aumenta à medida que mais dias são utilizados como histórico.

A Figura 5.8a exibe um gráfico com o tempo de processamento para minerar padrões, variando-se a quantidade de dias em uma base com sequências/janelas de tamanho 3 e sobreposição de tamanho 2. Os algoritmos com o menor tempo de processamento foram SPADE e CM-SPADE, que ao serem avaliados levando em consideração o dia 20, por exemplo, duraram 83 e 101 milissegundos, respectivamente. O que levou mais tempo foi o GSP, que na mesma situação durou 260 *ms*.

Similar ao exemplo anterior, a Figura 5.8b exibe um gráfico com o tempo de processamento para minerar padrões variando a quantidade de dias em uma base com sequências/janelas de tamanho 5 e sobreposição de tamanho 4. Os algoritmos com o menor tempo de processamento também foram SPADE e CM-SPADE, que ao serem avaliados levando em consideração o dia 20, por exemplo, levaram 297 e 337 milissegundos, respectivamente. Em relação à base com janelas de tamanho 3, o tempo de execução aumentou cerca de $3x$. O que levou mais tempo foi o GSP, que na mesma situação durou 3132 *ms*, com um aumento de cerca de $12x$.

Ao analisar mais a fundo o aumento elevado que ocorre em casos como os exibidos nos dias 4, 5 e 6 é possível observar que, dependendo dos dados do dia adicionado, pode ocorrer que algum padrão fique acima ou abaixo do suporte mínimo com a adição de apenas um dia. Por exemplo, considerando dois dias, D1 e D2, ambos com 100



(a) Janelas de tamanho 3.

(b) Janelas de tamanho 5.

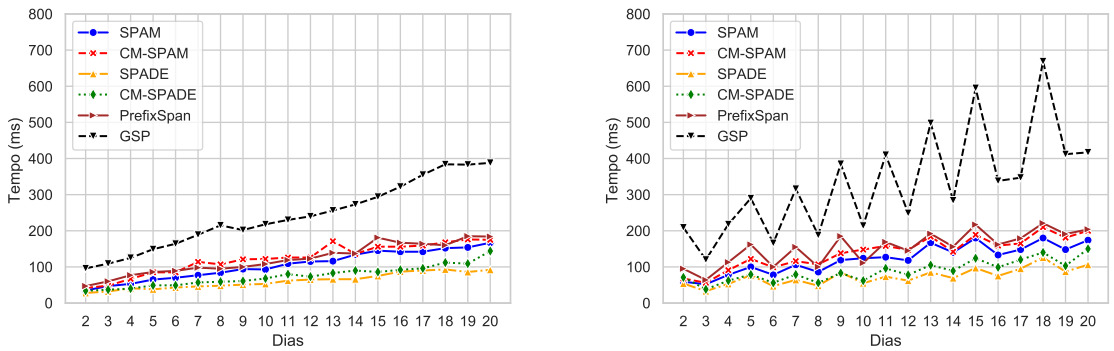
Figura 5.8: Comparação de tempo em relação aos dias variando o tamanho das sequências/janelas.

sequências. No dia D1, um padrão $P = 100, 101, 802$ tem o suporte igual a 0,19 (19% da base). No dia D2 o suporte deste mesmo padrão é 0,21 (21% da base). Ao analisar apenas o dia D1, o padrão P não será considerado frequente, pois está abaixo do suporte mínimo especificado de 0,20. No entanto, ao acrescentar o dia D2, a base passa a conter 200 sequências, das quais o padrão P pode ser encontrado em 40 destas. Logo, é considerado um padrão frequente. Caso seja adicionado um dia D3 na base, o padrão P pode deixar de ser frequente novamente caso o suporte do novo dia adicionado for inferior à 0,20.

Essa entrada e saída de sequências frequentes pode ser apontada como uma das causas para ocorrência do aumento de tempo repentino exibidos na Figura 5.8b. No dia 4, a quantidade de padrões frequentes foi de 9.613, enquanto no dia 5 aumentou consideravelmente para 16.855, ou seja, 7.242 padrões a mais. No dia 6 diminuiu 12.874 padrões, caindo para 3.981 padrões frequentes. A razão pela qual isto não ocorre com todos os algoritmos é que cada algoritmo gera os candidatos de uma forma diferente. Devido a tal fato, alguns algoritmos são mais sensíveis às sequências longas ou com muitos itens nos *itemsets*.

A Figura 5.9 exibe uma comparação do tempo de processamento ao variar o tamanho dos *itemsets*. Para este experimento, foi utilizado uma base de sequências com sequências/janelas de tamanho 5, sobreposição igual a 4 e suporte mínimo igual à 0,35. Assim como no experimento anterior, os experimentos são realizados em função da quantidade de dias utilizado como histórico.

A Figura 5.9a exibe um gráfico com o tempo de processamento para minerar padrões, variando a quantidade de dias em uma base com *itemsets* de tamanho 4. Os algoritmos com o menor tempo de processamento foram SPADE e CM-SPADE, que ao serem avaliados levando em consideração o dia 20, por exemplo, duraram 92 e 144



(a) *Itemsets* de tamanho 4. (b) *Itemsets* de tamanho 5.

Figura 5.9: Comparação de tempo em relação aos dias variando o tamanho dos *itemsets*.

milissegundos, respectivamente. O que levou mais tempo foi o GSP, que na mesma situação durou 388 *ms*.

Na Figura 5.9b é exibido o gráfico relativo à um experimento similar ao anterior, mas com incremento no tamanho dos *itemsets*. Os algoritmos com o menor tempo de processamento também foram SPADE e CM-SPADE, que ao serem avaliados levando em consideração o dia 20, por exemplo, duraram 106 e 150 milissegundos, respectivamente. Na mesma situação o GSP durou 417 *ms*. Assim como ocorreu ao incrementar o tamanho da sequência, ao aumentar o tamanho dos *itemsets* também ocorreu o efeito relacionado aos padrões que subitamente tornam-se frequentes.

Os experimentos apresentados nesta seção demonstram que os algoritmos SPADE e CM-SPADE encontram os padrões frequentes em menor tempo no cenário avaliado. O CM-SPAM se mantém estável em todos os cenários avaliados e o GSP levou mais tempo para encontrar os padrões frequentes.

5.5 Consumo de Memória

Nessa seção, de forma similar ao que acontece na seção anterior, são exibidos os experimentos avaliando os algoritmos de mineração em diversos cenários, porém com foco no uso de memória. Primeiro é avaliado o uso de memória em função do suporte, variando o tamanho da sobreposição e dos *itemsets*. Logo após, são realizados experimentos em função da quantidade de dias, variando a quantidade de *itemsets* e das sequências.

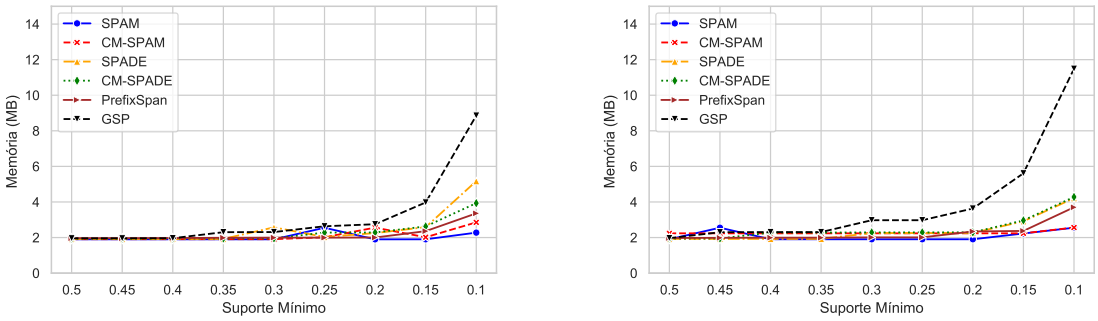
A medição do uso exato de memória de um algoritmo quando se utiliza Java não é trivial, pois o desenvolvedor não controla o uso da memória diretamente – o mecanismo utilizado para a desalocação de memória é o *Garbage Collector*(GC). Conforme o programa deixa de utilizar algumas instancias, o GC faz a desalocação destas áreas de memória. No entanto, não fazer uso destas áreas de memória não significa exatamente que o GC vai agir de imediato, mesmo fazendo chamada implícita ao GC. Além disso, esta desalocação também causa *overhead*, fazendo com que o programa em execução fique um pouco mais lento.

Portanto, devido a tais desafios, alguns ajustes precisaram ser feitos para uma avaliação mais justa dos algoritmos. O primeiro foi setar um bom valor de memória para a execução dos algoritmos. O segundo ajuste foi verificar o uso de memória em momentos que exigissem mais dos algoritmos, e não somente no início e ao final da execução. O terceiro foi utilizar apenas um programa rodando por vez na JVM e contabilizar o uso de memória apenas destes, de forma que o uso de memória externo não interfira no uso de memória pelo algoritmo.

5.5.1 Consumo de Memória de Variando o Suporte

Os experimentos exibidos nas Figura 5.10 utilizam uma base com sequências/janelas de tamanho 3 e *itemsets* de tamanho 3. Na Figura 5.10a o tamanho da sobreposição é 1. Na maioria dos casos, os algoritmos SPAM e CM-SPAM são os que consumem a menor quantidade de memória, com consumo de 2,27 e 2,85 Megabytes, respectivamente, quando o suporte mínimo é igual à 0,10. O algoritmo com maior custo foi o GSP atingindo 8,85 *MB* de consumo.

Um comportamento similar se repete no experimento apresentado na Figura 5.10b, na qual SPAM e CM-SPAM também mantiveram o menor consumo de memória, com custo de 2,55 e 2,56 Megabytes, respectivamente, quando o suporte mínimo é igual à 0,10. Da mesma forma que no experimento anterior, o GSP também obteve o maior consumo, atingindo 11,48 MB.



(a) Sobreposição de janela igual à 1.

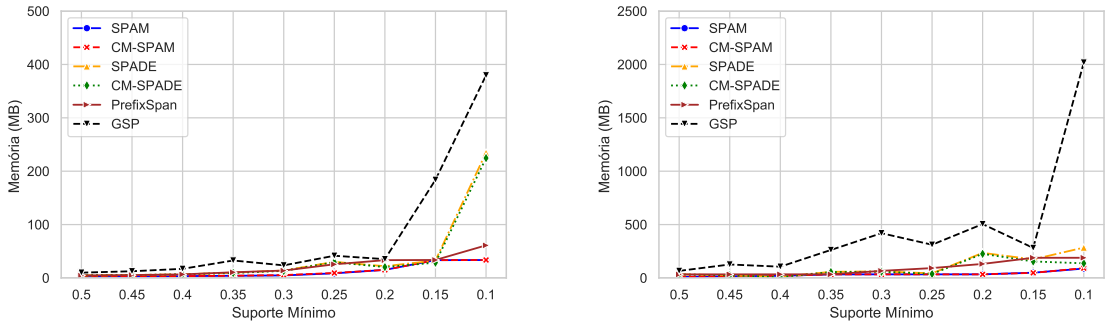
(b) Sobreposição de janela igual à 2.

Figura 5.10: Comparação de uso de memória em relação ao suporte variando o tamanho da sobreposição.

A Figura 5.11 exibe experimentos com sequências de tamanho 5 e sobreposição de tamanho 4 variando o suporte mínimo. Na Figura 5.11a, o tamanho dos *itemsets* é igual a 4, ou seja, possui o dado de uma atividade e três dados de contexto. Os algoritmos que consumiram a menor quantidade de memória foram SPAM e CM-SPAM, com consumo de 33,55 e 33,62 Megabytes, respectivamente, quando o suporte mínimo é igual a 0,10. O algoritmo que consumiu a maior quantidade de memória foi o GSP, com consumo de 379,66 MB.

Na Figura 5.11b é apresentado um experimento similar, mas com os *itemsets* com tamanho igual 5. Neste experimento, assim como no anterior, os algoritmos que menos consumiram memória foram SPAM e CM-SPAM, com consumo de 87,60 e 91,63 Megabytes, respectivamente.

De acordo com os experimentos apresentados, é possível notar claramente que



(a) *Itemsets* de tamanho 4. (b) *Itemsets* de tamanho 5.

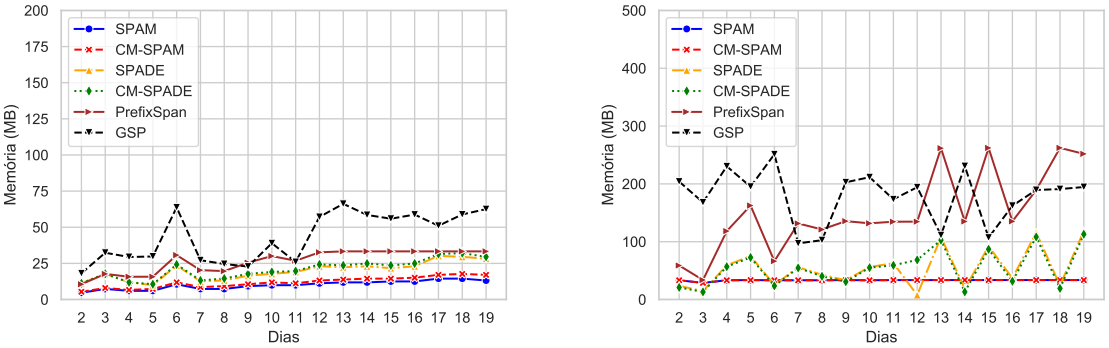
Figura 5.11: Comparação de uso de memória em relação ao suporte variando o tamanho dos *itemsets*.

os algoritmos SPAM e CM-SPAM são os mais estáveis em termos de uso de memória. Mesmo com um alto crescimento dos padrões, tais algoritmos mantiveram seus níveis de consumo de memória. Em contrapartida, o GSP obteve o maior consumo de memória entre os algoritmos avaliados.

5.5.2 Consumo de Memória de Variando os Dias

Os experimentos a seguir demonstram como ocorre o uso de memória entre os algoritmos em função da quantidade de dias utilizados como histórico. Para tanto, um valor de suporte mínimo é fixado para cada conjunto de experimentos.

A Figura 5.12 apresenta experimentos de comparação do uso de memória variando o tamanho das sequências e com o suporte mínimo fixado em 0,2 e *itemsets* de tamanho 5. Na Figura 5.12a, o tamanho da sequência é 3. Neste caso, o uso de memória é bem



(a) Sequências de tamanho 3. (b) Sequências de tamanho 5.

Figura 5.12: Comparação de uso de memória em relação aos dias variando o tamanho sequências/janelas.

estável. Os algoritmos seguem a mesma tendencia de leve crescimento, novamente os algoritmos CM-SPAM e SPAM são os que apresentam menor consumo de memória, com 13,15 e 17,05 de Megabytes em seu pior. Assim como nos casos anteriores, o GSP também foi o que mais consumiu memória, apresentando um consumo de 62,53 MB no mesmo cenário.

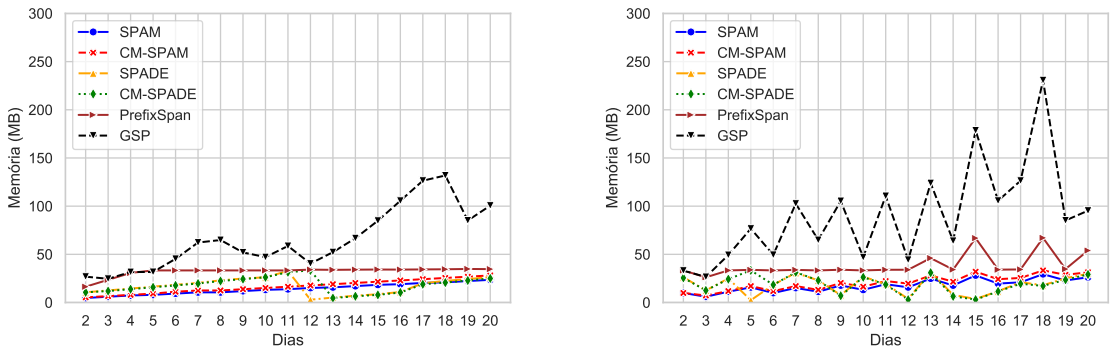
O experimento apresentado na Figura 5.12b exibe um comportamento diferente dos demais. No entanto, os mais estáveis continuam sendo CM-SPAM e SPAM com consumo de 33,64 e 33,65 em seu pior caso, respectivamente. Enquanto PrefixSpan e GSP consumem até 262,52 e 251,19 respectivamente.

É possível notar também que, assim como no experimento de tempo, os algoritmos também mudam bastante seu comportamento conforme o tamanho da sequência aumenta. Isto se deve ao dado acrescentado. De acordo com o novo dado acrescentado, novos padrões podem surgir e antigos padrões podem deixar de ser frequentes. Isto também acontece quando mais dados de contexto são inseridos.

A Figura 5.13 apresenta os experimentos comparando os algoritmos em relação à quantidade de *itemsets*. Para tanto, o valor do suporte mínimo foi fixado em 0,35 e o tamanho da sequência/janela igual a 5. A Figura 5.13a exibe os resultados do experimento realizado utilizando apenas *itemsets* de tamanho 4. Com exceção do PrefixSpan e GSP, os demais possuem em média o mesmo consumo.

O experimento apresentado na Figura 5.13b demonstra o mesmo comportamento exibido em situação anteriores, na qual a quantidade de novos padrões faz com que o consumo seja maior repentinamente e depois diminua. No entanto, tanto o SPAM quanto o CM-SPAM demonstram um comportamento mais estável que os demais.

Os experimentos exibidos nesta seção apresentam o comportamento dos algorit-



(a) *Itemsets* de tamanho 4. (b) *Itemsets* de tamanho 5.

Figura 5.13: Comparação de uso de memória em relação aos dias variando o tamanho dos *itemsets*.

mos em relação ao consumo de memória conforme os parâmetros são alterados. É possível observar nitidamente que os algoritmos PrefixSpan e GSP são os que mais consomem memória, enquanto CM-SPAM e SPAM são os que apresentam menor consumo.

5.6 Análise do Modelo de Previsão

Esta seção exibe os resultados em relação à taxa de acerto do modelo de previsão utilizando as bases HMR e HDA. As seções a seguir exibem os resultados relacionados à cada base e por fim é apresentado um experimento relacionado a quanto se perde em taxa de acerto para deixar o método mais rápido.

5.6.1 Taxa de Acerto

A Figura 5.14 exibe a matriz de confusão gerada a partir dos resultados das previsões. Para cada um dos 14 dias presentes na base, foi feita a divisão dos conjuntos para treinar com 13 dias e testar com o dia que não participou do treinamento. A acurácia geral para previsão foi de 84,40% e de 79,03% para a macro medida-F. Como as rotinas da base seguem um padrão que não muda independente dos dados do contexto (localização e

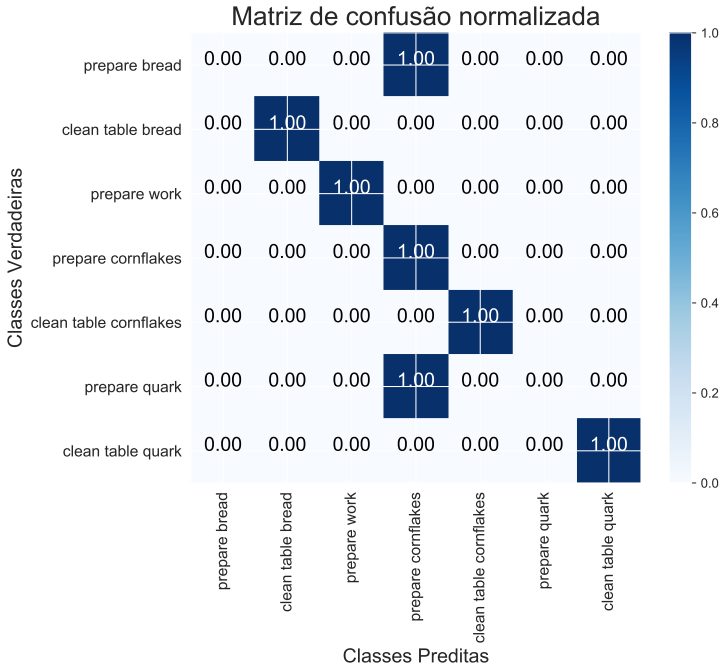


Figura 5.14: Matriz de confusão - Base HMR.

horário), o algoritmo utiliza apenas a informação da atividade com maior probabilidade de ocorrer.

A base HDA contém muito mais dados do que base de rotinas matinais, pois contém rótulos do dia quase todo. Devido a este fato, a acurácia pode variar bastante dependendo dos dados utilizados para treino e teste. Como as atividades contidas nesta base são apenas as atividades que podem ser reconhecidas por meio do *smartphone*, obter uma boa taxa de acerto torna-se ainda mais desafiador. Alguns ajustes considerados como ruídos foram feitos nesta base. Um deles é considerar que o usuário ficou "parado" apenas se o tempo de duração for superior à 10 segundos. Este ajuste se deve ao fato de que, em alguns momentos, o usuário pode interromper uma caminhada ou parar no trânsito. Estas interrupções podem ser consideradas como ruídos na previsão das próximas atividades que serão realizadas. Além disso, as ações de pegar e deixar o celular são excluídas da base para os casos em que o usuário pegou o dispositivo para registrar que realizou uma atividade.

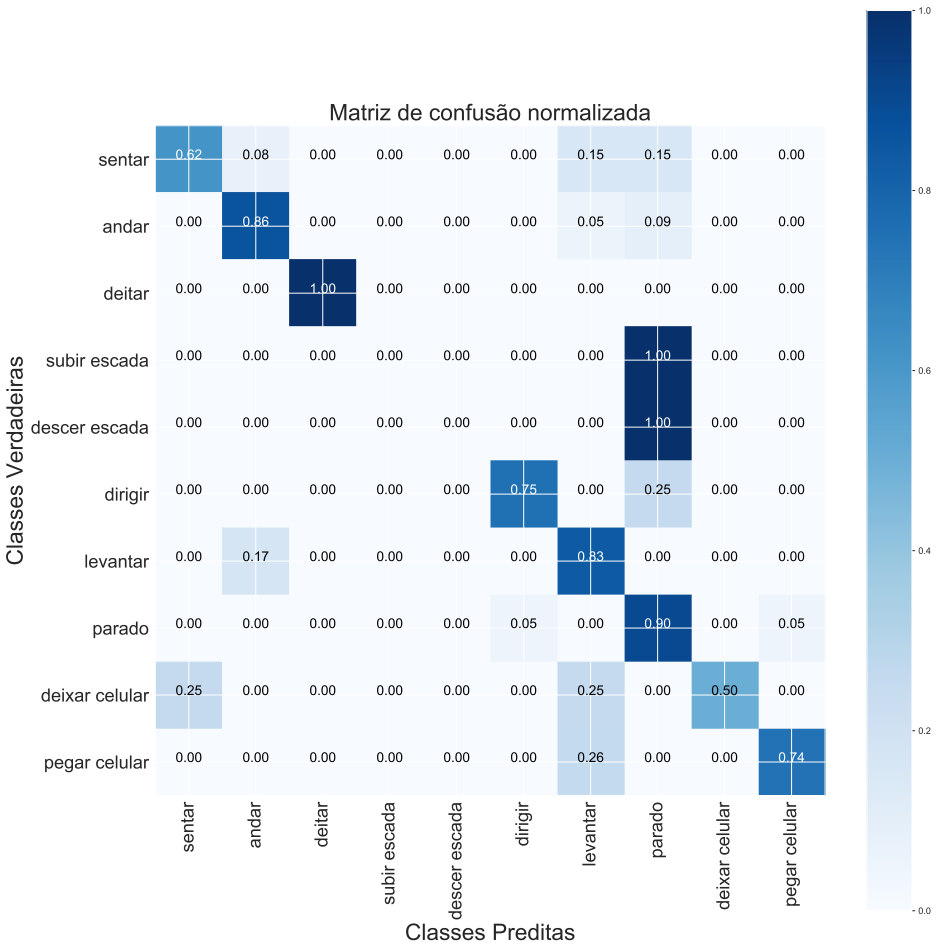


Figura 5.15: Matriz de confusão - Base HDA com dados de contexto.

A Figura 5.15 exibe a matriz de confusão gerada a partir dos resultados obtidos ao prever as atividades que ocorrem durante o dia. Para cada atividade realizada do dia, o método tenta prever qual será a próxima, independente do nível de confiança. Em alguns casos, quando o método não conhece determinado ambiente e não contém tal padrão na base, tenta-se buscar a sequência mais próxima. Ainda assim, para um conjunto de dados com muita incerteza de previsão, o método pode não encontrar uma sequência correspondente e marcar a previsão como a classe que mais ocorre.

Para o exemplo da Figura 5.15 foi utilizado o suporte mínimo de 0,05 e os dados de contexto utilizados foram a localização e a hora corrente. A acurácia obtida foi de 78,17% e de 77,15% para a macro medida-F. Além disso, utilizando o CM-SPAM com leves alterações, o tempo contabilizado para encontrar 18.696 sequências frequentes foi de 285 (ms), utilizando apenas 33 MB de memória. No entanto, algoritmo utilizado para geração das tabelas de probabilidades levou 22.084(ms) para o processamento. No total, o processo completo durou 22.369 (ms) segundos para gerar o modelo.

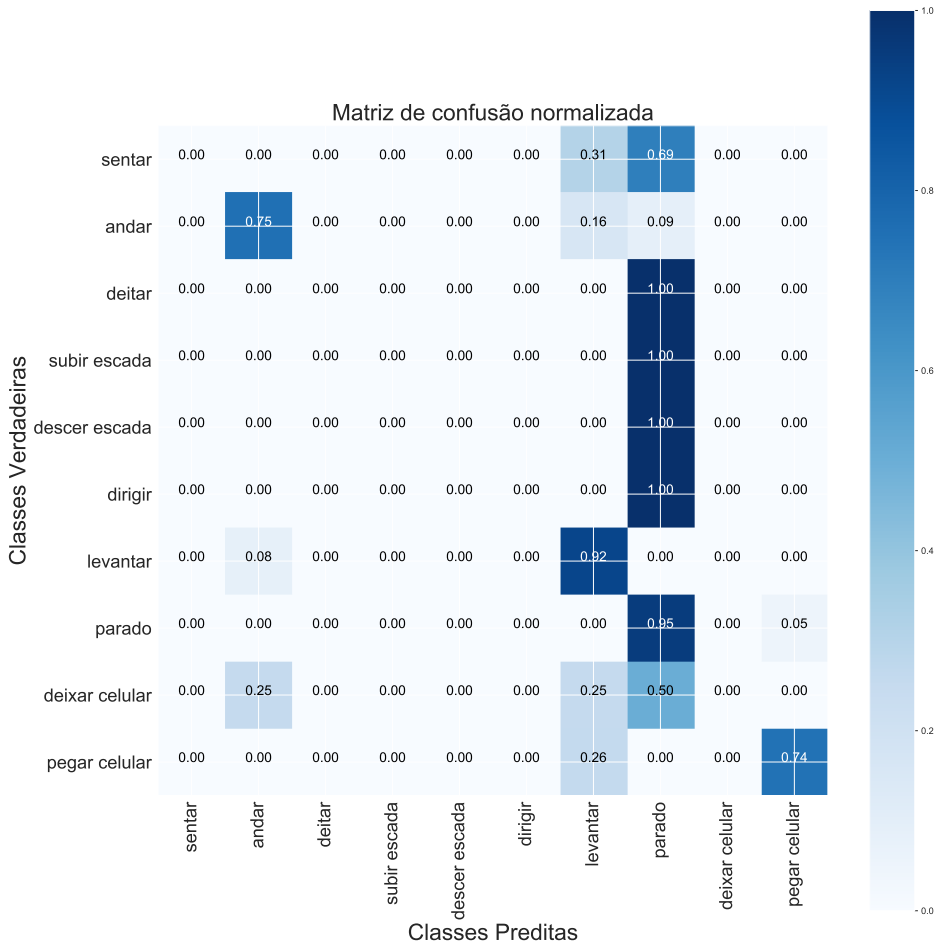


Figura 5.16: Matriz de confusão - Base HDA sem dados de contexto.

Um problema notado durante a fase de experimentos foi o tempo de busca de sequências aproximadas. Dependendo da sequência, uma simples busca chegou a durar mais de 1 segundo. No entanto, conforme abordado na seção de proposta, era esperado que a utilização de um método que não é tão eficaz para buscar strings similares pudesse ter este tipo de resultado. Outro problema notado apenas durante os experimentos foi que, devido ao algoritmo de busca aproximada não ponderar cada item que é removido ou trocado, às vezes um dado de contexto muito mais importante que outro acabava sendo removido. Esta remoção de dados de contexto mais importantes fez com que a previsão falhasse em alguns casos. Como algoritmos de busca aproximada fazem parte de um outro tópico, a melhoria desta fase será realizada em trabalhos futuros.

Para fins de comparação, utilizando os mesmos padrões sequenciais mas sem passar a sequência de testes com os dados de contexto, a acurácia caiu de 78,17% para 60,20%. Em relação à macro medida-F a diferença foi de 77,15% para 50,45% A Figura 5.16 exibe a matriz de confusão de tal base, onde é nítido que quanto menos dados de contexto são utilizados, menos o método é capaz de distinguir e prever qual será a próxima atividade. Esta é a razão principal pela qual algoritmos que se baseiam apenas em uma sequência, sem incorporar dados de contexto, não são tão adequados.

5.6.2 Trade-off entre custo e taxa de acerto

A Figura 5.17 exibe a matriz de confusão com os resultados obtidos a partir de um experimento rodado utilizando como parâmetros o suporte mínimo de 0,10 e, assim como no experimento anterior, com dois dados de contexto. Neste experimento, o algoritmo foi mais rápido que o anterior para encontrar os padrões, levando apenas 103 (ms) para encontrar 1.966 sequências frequentes, utilizando apenas 15,9 MB de memória. Quanto ao tempo para montar a tabela de probabilidades, o tempo foi de apenas 767 (ms), cerca de 30 vezes mais rápido que no caso anterior. No entanto, isto teve impacto na acurácia, que teve uma queda em relação ao resultado anterior, caindo de 78,17% para 64,88%. Em relação à macro medida-F a diferença foi de 77,15% para 61,69%. Conforme exibido na matriz de confusão, mais classes ficaram propensas a erros.

Conforme o suporte é incrementado e alguns dados de contexto são removidos, os algoritmos processam mais rápido. No entanto, como pode ser percebido no exemplo anterior, isso tem uma perda. Determinar qual o nível mais adequado de suporte para que o método não seja muito lento, nem tenha alta taxa de erro, varia de problema para problema e principalmente com a base de dados. Para os cenários avaliados, manter um nível de suporte por volta de 0,05 com dois dados de contexto não fez com que o

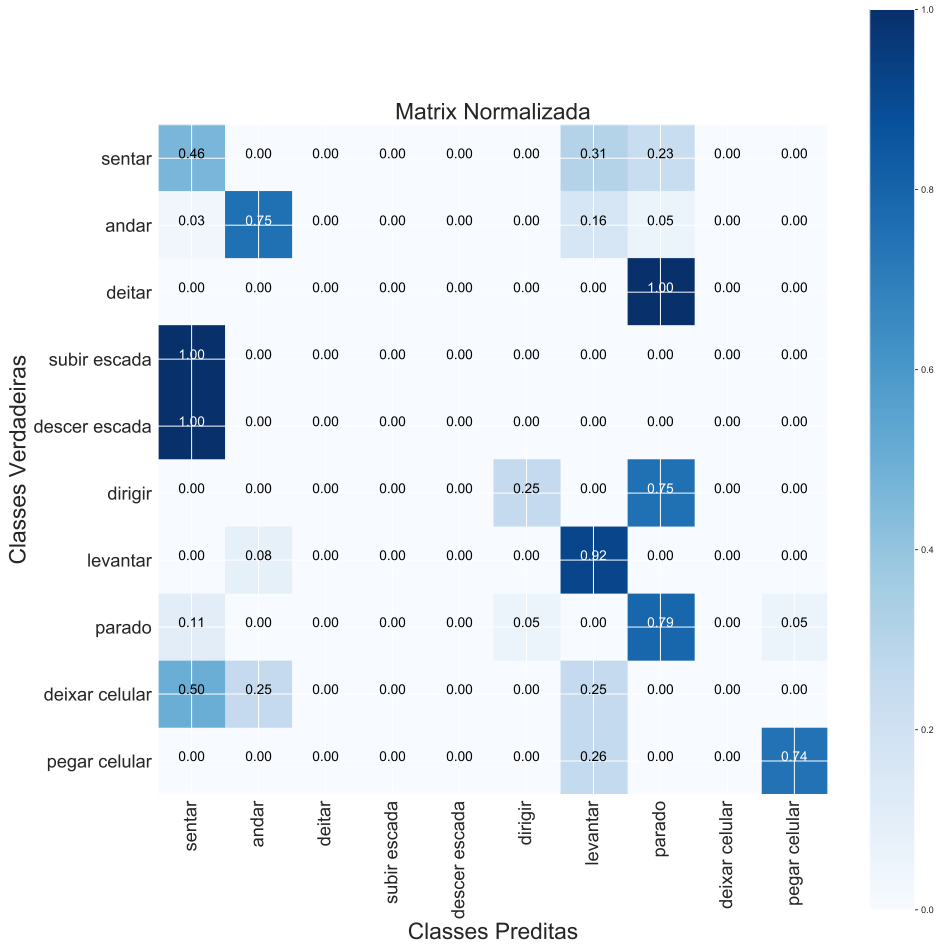


Figura 5.17: Matriz de confusão - Base HDA utilizando suporte mais elevado (0,10).

método ficasse muito lento e, ainda assim, obteve-se uma boa acurácia, considerando a complexidade dos dados avaliados.

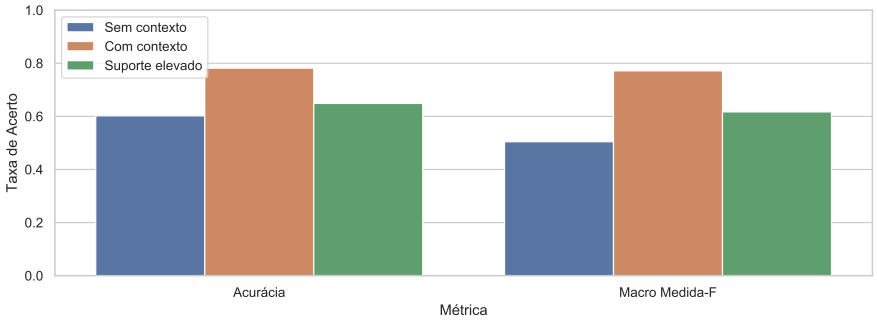


Figura 5.18: Comparação dos resultados em relação aos dados utilizados na base.

A Figura 5.18 apresenta uma comparação tanto da acurácia quanto da macro medida-F para os resultados exibidos anteriormente. Ao utilizar os dados de contexto

a taxa de acerto é maior. No entanto, quanto mais dados são utilizados, maior é o tempo de processamento. Portanto, tais parâmetros precisam ser avaliados de acordo com a aplicação e cenário utilizados.

Capítulo 6

Conclusões

Este trabalho apresentou o desenvolvimento de um método para previsão de atividades humanas, nomeado como SOPHIA. O método desenvolvido, composto por dois componentes principais para geração do modelo, foi projetado visando o melhor custo benefício em termos de taxa de acerto e custo computacional. Em cada componente do método foram realizadas adaptações com o objetivo de adequar o método à realidade de dispositivos móveis.

Este trabalho contribui para a área de pesquisa sendo o primeiro método de previsão de atividades humanas utilizando dados obtidos por meio de dispositivos móveis. Além disso, o método SOPHIA reúne algumas das melhores técnicas de SPM, também utilizadas no campo de visão computacional. Este trabalho conta ainda com aprimoramentos voltados para o problema de prever atividades humanas, tais como poda de candidatos que não serão utilizados no algoritmo de mineração de padrões e utilização de busca aproximada para tratar os casos de sequência não conhecidas pelo algoritmo. Nenhum dos trabalhos encontrados na literatura realiza tal tipo de tratamento.

Além disso, este trabalho também fornece um estudo comparativo entre os métodos de mineração de padrões com variação de cinco parâmetros, gerando resultados de uso de memória e tempo de processamento para cada um dos algoritmos avaliados. Por meio de tal análise, torna-se possível que pesquisas futuras utilizem o algoritmo mais adequado de acordo com o cenário abordado.

Os experimentos realizados demonstraram que é possível prever a maioria das atividades com taxas de acurácia superiores a 75%. No entanto, os experimentos também revelaram não ser possível prever algumas atividades, pelo fato de estas serem pouco frequentes e os dados utilizados não serem suficientes para detectar quando tais atividades podem ocorrer.

Levando em consideração os cenários avaliados, os dados disponíveis e a com-

plexidade do problema, o método SOPHIA apresentou resultados satisfatórios. Além disso, visto que o método utiliza sequencias simbólicas, pode ser facilmente adaptado para outros domínios de previsão.

6.1 Principais Desafios

O desenvolvimento de um método que se propõe à "prever o futuro", mesmo que indiretamente, possui diversos desafios. O primeiro desafio é saber se é possível resolver o problema abordado, ou seja, se é possível prever o tipo de dado especificado no problema. Para tanto, é preciso analisar os dados de entrada. No entanto, devido ao problema de previsão de atividades ser relativamente novo, as principais bases de dados disponíveis na literatura costumam utilizar apenas os dados obtidos por meio de câmeras em ambientes controlados, como em *SmartHomes*, geralmente em um cômodo da casa [Roggen et al., 2010; Ordóñez et al., 2013; Cumin et al., 2017]. Que de certa forma limita o escopo do problema abordado neste trabalho. Por outro lado, as bases que são compostas por dados provenientes de dispositivos móveis são voltadas para o reconhecimento de atividades, portanto, não possuem rotinas [Kwapisz et al., 2011; Reiss & Stricker, 2012; Anguita et al., 2013; Micucci et al., 2017].

Devido à escassez de bases compostas por dados das atividades básicas que o usuário realiza durante o dia, uma base foi criada para fins experimentais. De acordo com a análise dos dados, foi possível extrair manualmente as características que ajudam a compreender melhor as atividades humanas realizadas durante o dia. A busca exaustiva por bases com dados suficientes para prever dados significativos foi uma das etapas mais custosas em termos de tempo utilizado.

Em relação à execução dos experimentos, outro desafio encontrado foi medir o uso de memória utilizado pelo algoritmo. Em Java, o uso da memória não fica totalmente sob controle do desenvolvedor. Quem decide o momento certo de destruir os elementos da memória é o *Garbage Collector*. Desta forma, também foi preciso adaptar a medição do consumo de memória para refletir o valor real utilizado, por meio de técnicas existentes na literatura.

Além destes, devido ao método ser composto por outros módulos (mineração de padrões, criação do modelo, busca aproximada), foi necessário uma alta precaução na integração dos módulos. Além de precisar cumprir seu propósito principal, cada um dos módulos desenvolvidos possuem a limitação de tempo e espaço, para que desta forma, sejam adaptáveis ao contexto de dispositivos móveis.

6.2 Trabalhos Futuros

A base criada neste trabalho utiliza dados de apenas um usuário. Como trabalho futuro pretende-se criar uma base utilizando mais voluntários e desta forma analisar em todos os cenários a taxa de acerto do método. Devido a não existência de bases deste tipo, é de grande contribuição para esta e outras pesquisas. Além disso, pretende-se também aprimorar o método utilizado para realizar a busca aproximada, conforme citado nos capítulos anteriores, existem técnicas muito mais eficientes para tal tarefa.

Além disso, devido a inúmeras técnicas para reconhecimento e previsão de sequências serem criadas continuamente, como trabalho futuro pretende-se comparar o método desenvolvido com as abordagens mais recentes em termos de eficiência e eficácia. Pretende-se ainda calibrar o aplicativo *Sophia SmartDay* para aprender as notificações que o usuário deseja receber de acordo com o padrão de utilização do aplicativo. Além disso, também pretende-se mitigar o problema de *Cold Start* quando o usuário utilizar o aplicativo pela primeira vez.

Referências Bibliográficas

Abowd, G. D.; Dey, A. K.; Brown, P. J.; Davies, N.; Smith, M. & Steggles, P. (1999). Towards a better understanding of context and context-awareness. Em *International Symposium on Handheld and Ubiquitous Computing*, pp. 304--307. Springer.

Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. Em *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pp. 3--14. IEEE.

Agrawal, R.; Srikant, R. et al. (1994). Fast algorithms for mining association rules. Em *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pp. 487--499.

Ahmadi, H. (1990). Testability of the arbitrage pricing theory by neural network. Em *1990 IJCNN International Joint Conference on Neural Networks*, pp. 385–393 vol.1. ISSN .

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L. & Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. Em *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961--971.

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X. & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. Em *ESANN*.

Ayres, J.; Flannick, J.; Gehrke, J. & Yiu, T. (2002). Sequential pattern mining using a bitmap representation. Em *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pp. 429--435, New York, NY, USA. ACM.

Baeza-Yates, R.; Jiang, D.; Silvestri, F. & Harrison, B. (2015). Predicting the next app that you are going to use. Em *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pp. 285--294, New York, NY, USA. ACM.

Bahrainian, S. & Crestani, F. (2017a). Towards the next generation of personal assistants: Systems that know when you forget. Em *ICTIR 2017 - Proceedings of the 2017 ACM SIGIR International Conference on the Theory of Information Retrieval*, pp. 169–176. cited By 4.

Bahrainian, S. A. & Crestani, F. (2017b). Tracking smartphone app usage for time-aware recommendation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10647 LNCS:161–172. cited By 0.

Bahrainian, S. A. & Crestani, F. (2018). Augmentation of human memory: Anticipating topics that continue in the next meeting. Em *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval*, pp. 150--159. ACM.

Bao, L. & Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. Em *International Conference on Pervasive Computing*, pp. 1--17. Springer.

Boytsov, L. (2011). Indexing methods for approximate dictionary searching: Comparative analysis. *Journal of Experimental Algorithmics (JEA)*, 16:1--1.

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D. & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467--479.

Cao, Y.; Barrett, D.; Barbu, A.; Narayanaswamy, S.; Yu, H.; Michaux, A.; Lin, Y.; Dickinson, S.; Siskind, J. M. & Wang, S. (2013). Recognize human activities from partially observed videos. Em *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2658–2665. ISSN 1063-6919.

Chatfield, C. (2000). *Time-series forecasting*. CRC Press.

Chen, M.-Y.; Kundu, A. & Zhou, J. (1994). Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE transactions on Pattern analysis and Machine Intelligence*, 16(5):481--496.

Cho, E.; Myers, S. A. & Leskovec, J. (2011). Friendship and mobility: user movement in location-based social networks. Em *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1082--1090. ACM.

Cho, S.-B. (2016). Exploiting machine learning techniques for location recognition and prediction with smartphone logs. *Neurocomputing*, 176:98--106.

Choi, J.; Lee, M. & Rhee, M. (1995). Trading s&p 500 stock index futures using a neural network. Em *Proceedings of the third annual international conference on artificial intelligence applications on wall street*, pp. 63--72.

Choudhury, T.; Philipose, M.; Wyatt, D. & Lester, J. (2006). Towards activity databases: Using sensors and statistical models to summarize people's lives. *IEEE Data Eng. Bull.*, 29(1):49--58.

Cici, B.; Alimpertis, E.; Ihler, A. & Markopoulou, A. (2016). Cell-to-cell activity prediction for smart cities. Em *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pp. 903--908. IEEE.

Cook, D. J. & Krishnan, N. C. (2015). *Activity learning: discovering, recognizing, and predicting human behavior from sensor data*. John Wiley & Sons.

Cumin, J.; Lefebvre, G.; Ramparany, F. & Crowley, J. L. (2017). A dataset of routine daily activities in an instrumented home. Em *International Conference on Ubiquitous Computing and Ambient Intelligence*, pp. 413--425. Springer.

Dai, P. & Ho, S. S. (2014). A smartphone user activity prediction framework utilizing partial repetitive and landmark behaviors. Em *2014 IEEE 15th International Conference on Mobile Data Management*, volume 1, pp. 205--210. IEEE. ISSN 1551-6245.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171--176. ISSN 0001-0782.

De Amo, S. (2004). *Técnicas de mineração de dados*. Universidade Federal de Uberlândia.

Deng, Z.-H. & Lv, S.-L. (2014). Fast mining frequent itemsets using nodesets. *Expert Systems with Applications*, 41(10):4505--4512.

Deng, Z.-H. & Lv, S.-L. (2015). Prepost+: An efficient n-lists-based algorithm for mining frequent itemsets via children–parent equivalence pruning. *Expert Systems with Applications*, 42(13):5424--5432.

Dernbach, S.; Das, B.; Krishnan, N. C.; Thomas, B. L. & Cook, D. J. (2012). Simple and complex activity recognition through smart phones. Em *Intelligent Environments (IE), 2012 8th International Conference on*, pp. 214--221. IEEE.

Dong, X.; Gong, Y. & Cao, L. (2018). F-nsp+: A fast negative sequential patterns mining method with self-adaptive data storage. *Pattern Recognition*, 84:13 – 27. ISSN 0031-3203.

Fan, L.; Wang, Z. & Wang, H. (2013). Human activity recognition model based on decision tree. Em *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, pp. 64--68. IEEE.

Fayyad, U.; Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37.

Ferreira, A. B. d. H. (2004). Novo dicionário aurélio da língua portuguesa. Em *Novo dicionário Aurélio da língua portuguesa*. Editora Positivo.

Ford, G.; Hauser, S. E.; Le, D. X. & Thoma, G. R. (2000). Pattern matching techniques for correcting low-confidence ocr words in a known context. Em *Document Recognition and Retrieval VIII*, volume 4307, pp. 241--250. International Society for Optics and Photonics.

Fournier-Viger, P.; Gomariz, A.; Campos, M. & Thomas, R. (2014a). Fast vertical mining of sequential patterns using co-occurrence information. Em Tseng, V. S.; Ho, T. B.; Zhou, Z.-H.; Chen, A. L. P. & Kao, H.-Y., editores, *Advances in Knowledge Discovery and Data Mining*, pp. 40--52, Cham. Springer International Publishing.

Fournier-Viger, P.; Gueniche, T. & Tseng, V. S. (2012). Using partially-ordered sequential rules to generate more accurate sequence prediction. Em *International Conference on Advanced Data Mining and Applications*, pp. 431--442. Springer.

Fournier-Viger, P.; Lin, J. C.-W.; Kiran, R. U.; Koh, Y. S. & Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54--77.

Fournier-Viger, P.; Wu, C.-W.; Gomariz, A. & Tseng, V. S. (2014b). Vmsp: Efficient vertical mining of maximal sequential patterns. Em Sokolova, M. & van Beek, P., editores, *Advances in Artificial Intelligence*, pp. 83--94, Cham. Springer International Publishing.

Furtado, D. A. et al. (2005). *Mineração de padrões seqüênciais múltiplos*. Universidade Federal de Uberlândia.

Gambs, S.; Killijian, M.-O. & del Prado Cortez, M. N. (2012). Next place prediction using mobility markov chains. Em *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, p. 3. ACM.

Glass, J. R. (2003). A probabilistic framework for segment-based speech recognition. *Computer Speech & Language*, 17(2-3):137--152.

Goto, H.; Miura, J. & Sugiyama, J. (2013). Human-robot collaborative assembly by on-line human action recognition based on an fsm task model. Em *Human-Robot Interaction 2013 Workshop on Collaborative Manipulation*.

Gu, Y.; Feng, M.; Yao, Y.; Liu, W. & Song, J. (2016). We know what you are doing or going to do: Towards accurate human activities sensing. Em *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9. cited By 1.

Gui, B.; Wei, X.; Shen, Q.; Qi, J. & Guo, L. (2014). Financial time series forecasting using support vector machine. Em *Computational Intelligence and Security (CIS), 2014 Tenth International Conference on*, pp. 39--43. IEEE.

Gupta, A.; Kembhavi, A. & Davis, L. S. (2009). Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1775--1789.

Han, D.; Bo, L. & Sminchisescu, C. (2009). Selection and context for action recognition. Em *2009 IEEE 12th International Conference on Computer Vision*, pp. 1933–1940. ISSN 2380-7504.

Han, J.; Pei, J. & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Han, J.; Pei, J.; Mortazavi-Asl, B.; Chen, Q.; Dayal, U. & Hsu, M.-C. (2000). Freespan: Frequent pattern-projected sequential pattern mining. Em *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pp. 355--359, New York, NY, USA. ACM.

Han, J.; Pei, J.; Yin, Y. & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53--87.

Han, M. (2013). *An Integrative Human Activity Recognition Framework based on Smartphone Multimodal Sensors*. Tese de doutorado, Kyung Hee University Seoul, Korea.

Hang, A.; von Zezschwitz, E.; De Luca, A. & Hussmann, H. (2012). Too much information!: User attitudes towards smartphone sharing. Em *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pp. 284--287, New York, NY, USA. ACM.

Harris, T. (2015). Credit scoring using the clustered support vector machine. *Expert Systems with Applications*, 42(2):741--750.

Hawkins, K. P.; Vo, N.; Bansal, S. & Bobick, A. F. (2013). Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. Em *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 499–506. ISSN 2164-0572.

Henpraserttae, A.; Thiemjarus, S. & Marukatat, S. (2011). Accurate activity recognition using a mobile phone regardless of device orientation and location. Em *Body Sensor Networks (BSN), 2011 International Conference on*, pp. 41--46. IEEE.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735--1780.

Hull, R.; Neaves, P. & Bedford-Roberts, J. (1997). Towards situated computing. Em *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pp. 146--153. IEEE.

Hyndman, R. J. (2011). Moving averages. Em *International encyclopedia of statistical science*, pp. 866--869. Springer.

Ignatov, A. (2018). Real-time human activity recognition from accelerometer data using convolutional neural networks. *Applied Soft Computing*, 62:915 – 922. ISSN 1568-4946.

João, R. S. et al. (2015). *Mineração de padrões sequenciais e geração de regras de associação envolvendo temporalidade*. Universidade Federal de São Carlos.

Juan, A. & Ney, H. (2002). Reversing and smoothing the multinomial naive bayes text classifier. Em *PRIS*, pp. 200--212.

Kamijo, K. & Tanigawa, T. (1990). Stock price pattern recognition-a recurrent neural network approach. Em *1990 IJCNN International Joint Conference on Neural Networks*, pp. 215–221 vol.1. ISSN .

Karantonis, D. M.; Narayanan, M. R.; Mathie, M.; Lovell, N. H. & Celler, B. G. (2006). Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE transactions on information technology in biomedicine*, 10(1):156--167.

Karg, M. & Kirsch, A. (2014). A human morning routine dataset. Em *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1351--1352. International Foundation for Autonomous Agents and Multiagent Systems.

Kemeny, J. G.; Snell, J. L. et al. (1960). *Finite markov chains*, volume 356. van Nostrand Princeton, NJ.

Kern, N.; Schiele, B. & Schmidt, A. (2007). Recognizing context for annotating a live life recording. *Personal Ubiquitous Comput.*, 11(4):251--263. ISSN 1617-4909.

Khan, A. M.; Tufail, A.; Khattak, A. M. & Laine, T. H. (2014). Activity recognition on smartphones via sensor-fusion and kda-based svms. *International Journal of Distributed Sensor Networks*.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307--319.

Kimoto, T.; Asakawa, K.; Yoda, M. & Takeoka, M. (1990). Stock market prediction system with modular neural networks. Em *1990 IJCNN International Joint Conference on Neural Networks*, pp. 1–6 vol.1. ISSN .

Kundu, A.; He, Y. & Bahl, P. (1989). Recognition of handwritten word: first and second order hidden markov model based approach. *Pattern recognition*, 22(3):283--297.

Kwapisz, J. R.; Weiss, G. M. & Moore, S. A. (2011). Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74--82.

Lau, S. L. & David, K. (2010). Movement recognition using the accelerometer in smartphones. Em *Future Network and Mobile Summit, 2010*, pp. 1--9. IEEE.

Lee, K.-F.; Hon, H.-W.; Hwang, M.-Y. & Huang, X. (1990). Speech recognition using hidden markov models: a cmu perspective. *Speech Communication*, 9(5-6):497--508.

Leite, P. B. C. (2008). *Identificação de Tipos de Culturas Agrícolas a partir de Seqüências de Imagens Multitemporais Utilizando Modelos de Markov Ocultos*. Tese de doutorado, PUC-Rio.

Lerner, A.; Chrysanthou, Y. & Lischinski, D. (2007). Crowds by example. Em *Computer Graphics Forum*, volume 26, pp. 655--664. Wiley Online Library.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. Em *Soviet physics doklady*, volume 10, pp. 707--710.

Li, K. & Fu, Y. (2014). Prediction of human activity by discovering temporal sequence patterns. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1644--1657.

Li, K. & Fu, Y. (2016). Actionlets and activity prediction. Em *Human Activity Recognition and Prediction*, pp. 123--151. Springer.

Li, K.; Hu, J. & Fu, Y. (2012). *Modeling Complex Temporal Composition of Actionlets for Activity Prediction*, pp. 286--299. Springer Berlin Heidelberg, Berlin, Heidelberg.

Liu, Y.; Nie, L.; Liu, L. & Rosenblum, D. S. (2016). From action to activity: Sensor-based activity recognition. *Neurocomputing*, 181:108 – 115. ISSN 0925-2312. Big Data Driven Intelligent Transportation Systems.

Lopes, A.; Mendes-Moreira, J. & Gama, J. (2012). Semi-supervised learning: predicting activities in android environment. Em *Workshop on Ubiquitous Data Mining*, p. 38. Citeseer.

Magnanimo, V.; Saveriano, M.; Rossi, S. & Lee, D. (2014). A bayesian approach for task recognition and future human activity prediction. Em *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pp. 726–731. IEEE. ISSN 1944-9445.

Marszalek, M.; Laptev, I. & Schmid, C. (2009). Actions in context. Em *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2929–2936. ISSN 1063-6919.

Masseglia, F.; Cathala, F. & Poncelet, P. (1998). The psp approach for mining sequential patterns. Em *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, PKDD '98, pp. 176--184, Berlin, Heidelberg. Springer-Verlag.

Mathie, M.; Celler, B. G.; Lovell, N. H. & Coster, A. (2004). Classification of basic daily movements using a triaxial accelerometer. *Medical and Biological Engineering and Computing*, 42(5):679--687.

Mayrhofer, R.; Radi, H. & Ferscha, A. (2003). *Recognizing and predicting context by learning from user behavior*. na.

Menezes, P. B. (1998). *Linguagens formais e autômatos*. Sagra-Dcluzzato.

Micucci, D.; Mobilio, M. & Napoletano, P. (2017). Unimib shar: A dataset for human activity recognition using acceleration data from smartphones. *Applied Sciences*, 7(10):1101.

Minor, B.; Doppa, J. R. & Cook, D. J. (2015). Data-driven activity prediction: Algorithms, evaluation methodology, and applications. Em *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 805--814. ACM.

Moreira, M. W.; Rodrigues, J. J.; Kumar, N.; Saleem, K. & Illin, I. V. (2019). Postpartum depression prediction through pregnancy data analysis for emotion-aware smart systems. *Information Fusion*, 47:23 – 31. ISSN 1566-2535.

Moutacalli, M. T.; Bouzouane, A. & Bouchard, B. (2015). Sensors activation time predictions in smart home. Em *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, p. 37. ACM.

Murre, J. M. & Dros, J. (2015). Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644.

Nazerfard, E. & Cook, D. J. (2015). Crafft: an activity prediction model based on bayesian networks. *Journal of ambient intelligence and humanized computing*, 6(2):193--205.

Nwe, T. L.; Foo, S. W. & De Silva, L. C. (2003). Speech emotion recognition using hidden markov models. *Speech communication*, 41(4):603--623.

Nweke, H. F.; Teh, Y. W.; Al-garadi, M. A. & Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233 – 261. ISSN 0957-4174.

Ordóñez, F.; de Toledo, P.; Sanchis, A. et al. (2013). Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. *Sensors*, 13(5):5460--5477.

Pei, J.; Han, J.; Mortazavi-Asl, B.; Wang, J.; Pinto, H.; Chen, Q.; Dayal, U. & Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440. ISSN 1041-4347.

Pellegrini, S.; Ess, A.; Schindler, K. & van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. Em *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268. ISSN 2380-7504.

Pokou, Y. J. M.; Fournier-Viger, P. & Moghrabi, C. (2016). Authorship attribution using small sets of frequent part-of-speech skip-grams. Em *FLAIRS Conference*, pp. 86--91.

Quispe, K. G. M.; Lima, W. S. & Souto, E. J. P. (2018). Human activity recognition on smartphones using symbolic data representation. Em *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pp. 93--100. ACM.

Ravi, N.; Dandekar, N.; Mysore, P. & Littman, M. L. (2005). Activity recognition from accelerometer data. Em *Aaai*, volume 5, pp. 1541--1546.

Reiss, A. & Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. Em *2012 16th International Symposium on Wearable Computers*, pp. 108–109. ISSN 2376-8541.

Rendle, S.; Freudenthaler, C. & Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. Em *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pp. 811--820, New York, NY, USA. ACM.

Reyes-Ortiz, J.-L.; Oneto, L.; Sama, A.; Parra, X. & Anguita, D. (2016). Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754--767.

Rodrigues, F.; Markou, I. & Pereira, F. C. (2019). Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. *Information Fusion*, 49:120 – 129. ISSN 1566-2535.

Roggen, D.; Calatroni, A.; Rossi, M.; Holleczek, T.; Förster, K.; Tröster, G.; Lukowicz, P.; Bannach, D.; Pirkl, G.; Ferscha, A. et al. (2010). Collecting complex activity datasets in highly rich networked sensor environments. Em *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pp. 233--240. IEEE.

Rohrbach, M.; Amin, S.; Andriluka, M. & Schiele, B. (2012). A database for fine grained activity detection of cooking activities. Em *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1194--1201. IEEE.

Ryan, N.; Pascoe, J. & Morse, D. (1999). Enhanced reality fieldwork: the context aware archaeological assistant. *Bar International Series*, 750:269--274.

Ryder, J.; Longstaff, B.; Reddy, S. & Estrin, D. (2009). Ambulation: A tool for monitoring mobility patterns over time using mobile phones. Em *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pp. 927--931. IEEE.

San-Segundo, R.; Echeverry-Correa, J. D.; Salamea, C. & Pardo, J. M. (2016). Human activity monitoring based on hidden markov models using a smartphone. *IEEE Instrumentation & Measurement Magazine*.

Schilit, B.; Adams, N. & Want, R. (1994). Context-aware computing applications. Em *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pp. 85--90. IEEE.

Schweizer, D.; Zehnder, M.; Wache, H.; Witschel, H.-F.; Zanatta, D. & Rodriguez, M. (2015). Using consumer behavior data to reduce energy consumption in smart homes: Applying machine learning to save energy without lowering comfort of inhabitants. Em *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pp. 1123--1129. IEEE.

Shokouhi, M. & Guo, Q. (2015). From queries to cards: Re-ranking proactive card recommendations based on reactive search history. Em *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pp. 695--704, New York, NY, USA. ACM.

Sigg, S. (2008). *Development of a novel context prediction algorithm and analysis of context prediction schemes*. kassel university press GmbH.

Srikant, R. & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. Em *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '96, pp. 3--17, London, UK, UK. Springer-Verlag.

Sun, Y.; Yuan, N. J.; Wang, Y.; Xie, X.; McDonald, K. & Zhang, R. (2016). Contextual intent tracking for personal assistants. Em *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 273--282, New York, NY, USA. ACM.

Tang, J.; Liang, J.; Zhang, S.; Huang, H. & Liu, F. (2018). Inferring driving trajectories based on probabilistic model from large scale taxi gps data. *Physica A: Statistical Mechanics and its Applications*, 506:566 – 577. ISSN 0378-4371.

Terzi, O. (2012). Monthly rainfall estimation using data-mining process. *Applied Computational Intelligence and Soft Computing*, 2012.

Uddin, M. Z. (2014). A robust daily human activity recognition and prediction system. Em *Proceedings of the 2014 International C* Conference on Computer Science & Software Engineering*, C3S2E '14, pp. 12:1--12:8, New York, NY, USA. ACM.

Valenza, G.; Nardelli, M.; Lanatà, A.; Gentili, C.; Bertschy, G.; Paradiso, R. & Scilingo, E. P. (2014). Wearable monitoring for mood recognition in bipolar disorder based on history-dependent long-term heart rate variability analysis. *IEEE Journal of Biomedical and Health Informatics*, 18(5):1625–1635. ISSN 2168-2194.

Vavoulas, G.; Pediaditis, M.; Spanakis, E. G. & Tsiknakis, M. (2013). The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones. Em *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, pp. 1--4. IEEE.

Vijayarani, S.; Dhayanand, S. & Phil, M. (2015). Kidney disease prediction using svm and ann algorithms. *International Journal of Computing and Business Research (IJCBR)*, 6(2).

Wang, H.; Yang, W.; Yuan, C.; Ling, H. & Hu, W. (2017a). Human activity prediction using temporally-weighted generalized time warping. *Neurocomputing*, 225:139--147.

Wang, K.; Xu, Y. & Yu, J. X. (2004). Scalable sequential pattern mining for biological sequences. Em *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pp. 178--187, New York, NY, USA. ACM.

Wang, L.; Zhao, X.; Si, Y.; Cao, L. & Liu, Y. (2017b). Context-associative hierarchical memory model for human activity recognition and prediction. *IEEE Transactions on Multimedia*, 19(3):646--659.

Wang, P.; Liu, H.; Wang, L. & Gao, R. X. (2018). Deep learning-based human motion recognition for predictive context-aware human-robot collaboration. *CIRP Annals*, 67(1):17 – 20. ISSN 0007-8506.

Wilbur, W. J.; Kim, W. & Xie, N. (2006). Spelling correction in the pubmed search engine. *Inf. Retr.*, 9(5):543--564. ISSN 1386-4564.

Wilcox, R.; Nikolaidis, S. & Shah, J. (2013). Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. *Robotics*, p. 441.

Xu, Z.; Qing, L. & Miao, J. (2015). Activity auto-completion: Predicting human activities from partial videos. Em *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3191--3199.

Yan, X.; Han, J. & Afshar, R. (2003). Clospan: Mining closed sequential patterns in large datasets. Em *In SDM*, pp. 166--177.

Yan, Z.; Chakraborty, D.; Misra, A.; Jeung, H. & Aberer, K. (2012). Semantic Activity Classification Using Locomotive Signatures from Mobile Phones. Relatório técnico, Various institutions.

Yang, H.-L.; Liu, A.-S. & Fu, L.-C. (2015). Daily activity prediction based on spatial-temporal matrix for ongoing videos. Em *Society of Instrument and Control Engineers of Japan (SICE), 2015 54th Annual Conference of the*, pp. 258--263. IEEE.

Yang, J. (2009). Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. Em *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, pp. 1--10. ACM.

Yang, J.; Xu, Y. & Chen, C. S. (1997). Human action learning via hidden markov model. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(1):34--44.

Yang, L.; Widjaja, B. & Prasad, R. (1995). Application of hidden markov models for signature verification. *Pattern recognition*, 28(2):161--170.

Yang, S.; Zhou, P.; Duan, K.; Hossain, M. S. & Alhamid, M. F. (2018). emhealth: Towards emotion health through depression prediction and intelligent health recommender system. *Mobile Networks and Applications*, 23(2):216--226. ISSN 1572-8153.

Yang, Z. & Kitsuregawa, M. (2005). Lapin-spam: An improved algorithm for mining sequential pattern. Em *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pp. 1222–1222. ISSN .

Ye, J.; Zhu, Z. & Cheng, H. (2013). What's your next move: User activity prediction in location-based social networks. Em *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 171--179. SIAM.

Yu, M.; Li, G.; Deng, D. & Feng, J. (2016). String similarity search and join: a survey. *Frontiers of Computer Science*, 10(3):399--417. ISSN 2095-2236.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31--60. ISSN 1573-0565.

Zhang, M.; Jiang, X.; Fang, Z.; Zeng, Y. & Xu, K. (2019). High-order hidden markov model for trend prediction in financial time series. *Physica A: Statistical Mechanics and its Applications*, 517:1 – 12. ISSN 0378-4371.

Zhang, W.; Du, Y.; Yoshida, T.; Wang, Q. & Li, X. (2018). Samen-svr: using sample entropy and support vector regression for bug number prediction. *IET Software*, 12(3):183--189.

Zhong, M.; Wen, J.; Hu, P. & Indulska, J. (2015). Advancing android activity recognition service with markov smoother. Em *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pp. 38--43. IEEE.

Zhou, F. & De la Torre, F. (2012). Generalized time warping for multi-modal alignment of human motion. Em *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1282--1289. IEEE.

Zong, F.; Tian, Y.; He, Y.; Tang, J. & Lv, J. (2019). Trip destination prediction based on multi-day gps data. *Physica A: Statistical Mechanics and its Applications*, 515:258 – 269. ISSN 0378-4371.