Capítulo 4

Detectando Padrões de Sociabilidade de Seres Humanos Através do Processamento de Eventos Complexos

Ivan Rodrigues (UFMA), Francisco Silva (UFMA), Luciano Coutinho (UFMA), Jean Marques (UFMA), Ariel Teles (IFMA/UFMA/UFDPar)

Abstract

Currently, ubiquitous computing is explored to develop new approaches to detect human social behaviors that can indicate the well-being state of individuals. In particular, Complex Event Processing (CEP) can detect social behavior patterns using data from sensors data of ubiquitous technologies. This knowledge is used to provide promising tools to automatically generate information about well-being states. This short course aims to introduce complex event processing techniques and their application for recognizing sociability patterns of human beings. Additionally, a tool capable of detecting sociability patterns through CEP is presented.

Resumo

Atualmente, a computação ubíqua é explorada para desenvolver novas abordagens para detectar comportamentos sociais humanos que podem indicar o estado de bem-estar dos indivíduos. Em particular, o Processamento de Eventos Complexos (Complex Event Processing - CEP) pode detectar comportamentos sociais por meio de dados de sensores de tecnologias onipresentes. Esse conhecimento é usado para fornecer ferramentas promissoras para gerar informações sobre estados de bem estar de maneira automática. Este minicurso objetiva introduzir técnicas de processamento de eventos complexos e sua aplicação na tarefa de reconhecimento de padrões de sociabilidade dos seres humanos. Adicionalmente, uma ferramenta capaz de detectar padrões de sociabilidade através do CEP é apresentada.

4.1. Introdução

A sociabilidade pode ser definida como "a tendência de se afiliar com os outros e preferir estar com os outros a permanecer sozinho" [Cheek and Buss 1981]. Este comportamento

humano apresenta relações diretas com o estado de saúde e com o convívio em grupo. A partir da sociabilidade dos indivíduos é possível encontrar padrões de comportamentos, que apresentam aplicabilidade em diversos domínios, como monitoramento da saúde mental, saúde e análise comportamental. Para os especialista em saúde mental, monitorar e avaliar o comportamento social de seus pacientes é essencial para realizar diagnósticos de transtornos mentais e realizar intervenções adequadas, visto que o aspecto social está relacionado ao estado de bem-estar dos indivíduos [Umberson and Montez 2010]. A sociabilidade também é um comportamento de interesse nas tarefas de análise comportamental, que visam entender o convívio em grupo de indivíduos. Na área da saúde, um exemplo recente é o impacto da necessidade do isolamento social imposto pela pandemia do novo coronavírus (COVID-19), no qual o entendimento deste comportamento pode ser utilizado por especialistas para realizar intervenções na saúde dos indivíduos. Portanto, monitorar e avaliar o comportamento social é uma tarefa importante para que profissionais especializados realizem avaliações e intervenções sobre bem-estar de maneira eficiente.

Tradicionalmente, o monitoramento e avaliação de comportamentos sociais relacionados à saúde mental é baseado em autorrelatos subjetivos, ou seja, os indivíduos relatam retrospectivamente suas experiências sociais vivenciadas e seus sentimentos quanto a elas. Entretanto, esta abordagem é limitada por um conjunto de vieses cognitivos (e.g., viés de memória e viés de desejabilidade) [Van de Mortel et al. 2008], que implicam em relatos incoerentes das atividades sociais vivenciadas pelos indivíduos. Por exemplo, o viés de memória implica em relatos inconsistentes, uma vez que os indivíduos não são capazes de lembrar fielmente o que sentiram e vivenciaram em sua rotina passada. Outro motivo que limita a abordagem tradicional de monitoramento da sociabilidade é que ela ocorre em ambientes clínicos, que são significativamente diferentes do contexto natural dos indivíduos [Trull and Ebner-Priemer 2013].

Atualmente, as tecnologias pervasivas (e.g., *smartphones*, *smartwatches*) representam um meio promissor de mitigar essas limitações, pois métodos computacionais (e.g., aprendizado de máquina e mineração de dados) podem processar dados de contexto coletados passivamente de sensores incorporados nesses dispositivos para identificar situações sociais automaticamente [Grav et al. 2012]. Por exemplo, algoritmos de aprendizado de máquina podem ser treinados para reconhecer conversações contidas em áudios provenientes do microfone [Wang et al. 2017]. Estas inferências de atividades sociais são responsáveis por criar um grande fluxo de eventos sociais, que podem ser processados para identificar padrões sociais. Dentre estas técnicas, é possível ressaltar o Processamento de Eventos Complexos (do inglês, *Complex Event Processing* - CEP) [Etzion et al. 2011], que fornece um conjunto de ferramentas para processar fluxos de dados de maneira eficiente, realizando tarefas como agregação de dados, derivação de informações de alto nível e reconhecimento de padrões. No cenário de detecção de padrões sociais, o CEP pode processar eventos primitivos para detectar relacionamentos de temporalidade, causalidade e semânticos entre estes, derivando eventos complexos (i.e., eventos compostos) para criar um novo fluxo ou emitir um alerta para o usuário quando padrões forem reconhecidos.

Este capítulo apresenta as técnicas de CEP para realizar a detecção de padrões de sociabilidade a partir do fluxo de eventos sociais derivados dos dispositivos pervasivos. Será detalhando os principais componentes de um motor CEP, assim como suas

principais funcionalidades. Especificamente, são apresentados os primeiros passos com a tecnologia, os conceitos de agentes e redes de processamento de eventos, partições de contexto, janelamento de dados e padrões. Também é detalhado o processo de utilização de uma ferramenta capaz de abstrair a complexidade de implementação de regras CEP para detectar padrões de sociabilidade e mudanças de comportamentos sociais. Para isso, será explicado todo o processo de configuração dos parâmetros da ferramenta e implementação das estratégias de detecção de padrões de sociabilidade através da *Application Programming Interface* (API) disponibilizada pela ferramenta.

Este capítulo está organizado como segue. A Seção 4.2 uma fundamentação teórica relacionada a técnicas de detecção de interações sociais. A Seção 4.3 descreve os conceitos fundamentais sobre CEP. A Seção 4.4 apresenta a ferramenta capaz de detectar padrões de sociabilidade através do CEP. A Seção 4.5 descreve domínios de aplicações da detecção de padrões de sociabilidade. No final, conduzimos as considerações finais na Seção 4.6.

4.2. Fundamentação Teórica

Os sensores embutidos nos dispositivos ubíquos (e.g., acelerômetro, microfone, e sensores de localização) fornecem dados capazes de caracterizar situações sociais, permitindo assim o desenvolvimento de sistemas capazes de automatizar o processo de monitoramento do comportamento social [Moura et al. 2020]. Esses sistemas usam dados coletados dos sensores para identificar situações sociais, nas quais são utilizadas técnicas estatísticas, mineração de dados, aprendizado de máquina, dentre outros métodos de análise e processamento de dados. Portanto, a partir do processamento de dados de contexto obtidos a partir de dispositivos pervasivos é possível inferir situações sociais (i.e., encontros face a face e comunicação mediada por dispositivos) de maneira automatizada, sem interação ativa do indivíduo.

4.2.1. Interações Face a Face

Encontros face a face denotam interações físicas, nas quais não há presença de uma tecnologia mediadora [Moura et al. 2020]. Esses encontros são identificados quando os indivíduos envolvidos estão fisicamente no mesmo espaço (i.e., em distâncias curtas) e interagindo entre si. Os dispositivos que compõem a computação ubíqua (e.g., dispositivos móveis e vestíveis) possuem sensores capazes de caracterizar essas situações, fornecendo meios de quantificar as relações sociais. A maioria desses dispositivos possuem microfones e tecnologias de comunicação sem fio (e.g., *WiFi*, *Bluetooth* e NFC), que podem ser usadas para identificar proximidade e conversação [Wang et al. 2017].

As interfaces de comunicação sem fio são comumente usadas para reconhecer interações físicas experimentadas por indivíduos [Onnela et al. 2014]. Estas interfaces possuem a capacidade de identificar dispositivos próximos, vindo a funcionar como um indicador de interações sociais. Portanto, pesquisadores têm utilizado esta tecnologia para caracterizar a sociabilidade do indivíduo monitorado através do registro dos identificadores dos dispositivos encontrados (e.g., IDs de *Bluetooth*) [Wang et al. 2017], permitindo realizar análises adicionais para identificar correlações com o bem-estar mental e extrair características sociais apropriadas para classificar e prever estados mentais.

Os microfones embutidos nos dispositivos ubíquos também tem sido explorados para reconhecer interações sociais face a face [Wang et al. 2017]. Pesquisadores têm utilizado algoritmos de aprendizado de máquina (e.g., Modelo Oculto de Markov e Modelo de Mistura de Gaussianas) para identificar a voz humana em amostras de áudio. Normalmente, os fluxos de áudio são segmentados em quadros, os quais são usados para conceber características adequadas para desenvolver modelos de aprendizado de máquina capazes de reconhecer a fala de um indivíduo. Essa conscientização da situação social tem sido usada para inferir o engajamento social, bem como para identificar evidências de isolamento social [Wang et al. 2017].

4.2.2. Interações Mediada por Dispositivo

A comunicação mediada por dispositivo é uma interação social que ocorre por meio da tecnologia [Moura et al. 2020], como dispositivos móveis e vestíveis ou através das redes sociais online. Os eventos sociais que ocorrem nessas mídias permitem o registro de informações valiosas para monitorar o comportamento social. Por exemplo, registros de chamadas telefônicas e aplicativos sociais (e.g., SMS, E-mail, *WhatsApp*, *Facebook*, *Twitter*) podem identificar características como vínculos sociais, suporte social, frequência de interações sociais, entre outros aspectos [Wongkoblap et al. 2017]. Assim, as abordagens computacionais podem usar essa rica fonte de dados comportamentais para identificar características da sociabilidade dos indivíduos, representando uma ferramenta valiosa para o processo de monitoramento do bem-estar mental.

O alto fluxo de atividades sociais derivadas das comunicações mediadas por dispositivo apresenta um ambiente adequado para identificar comportamentos sociais significativos. Por esse motivo, soluções tem explorado os registros de interações sociais vividas através dos dispositivos ubíquos para detectar a sociabilidade dos indivíduos. A maioria dos estudos desenvolveu maneiras de quantificar a sociabilidade do usuário a partir de registros de chamadas e mensagens de texto. As soluções presentes na literatura geralmente projetam características (e.g., número de chamadas, duração das chamadas, contatos exclusivos e número de mensagens enviadas) para caracterizar o comportamento social dos indivíduos [Barnett et al. 2018]. Além disso, as soluções usam essa caracterização para desenvolver modelos capazes de classificar e prever estados mentais [Servia-Rodríguez et al. 2017], associar o comportamento social ao estado mental [Wang et al. 2017] e quantificar a sociabilidade [Eskes et al. 2016].

As redes sociais online, como *Twitter*, *Facebook* e *Instagram*, são cada vez mais indispensáveis para alcançar a conexão social. Nessas plataformas online, as pessoas realizam diferentes tipos de atividades, como criar e compartilhar status, publicar fotos e vídeos, estabelecer novas amizades, trocar mensagens com amigos, dentre outros eventos sociais. A partir dessas ações, os usuários expressam seus sentimentos e pensamentos, além de expor informações sobre sua rotina diária. Dessa forma, a dinâmica dos usuários nas redes sociais online produz um fluxo de dados comportamental, o que representa um perfil da prática social desse indivíduo.

4.3. Processamento de Eventos Complexos

Atualmente, os dispositivos ubíquos e redes de sensores são responsáveis por coletar continuamente um grande volume de dados, vindo a evidenciar a necessidade de analisar e reagir em tempo real a este crescente fluxo de dados. Para mitigar esse desafio, é possível utilizar o CEP [Etzion et al. 2011], que fornece um conjunto de ferramentas para processar fluxos de dados de maneira eficiente, realizando tarefas como agregação de dados, derivação de informações de alto nível e reconhecimento de padrões. O CEP utiliza o paradigma de processamento orientado a evento, no qual cada evento modela uma observação em um domínio específico, que pode ser resultante da ação ou mudança de estado de objetos físicos ou virtuais. Neste cenário, o CEP processa eventos primitivos para detectar relacionamentos de temporalidade, causalidade e semânticos entre estes, derivando eventos complexos (i.e., compostos) para criar um novo fluxo ou emitir um alerta para o usuário. A arquitetura geral de um mecanismo CEP é mostrada na Figura 4.1, o qual segue as seguintes etapas básicas [Etzion et al. 2011]:

1. Eventos são criados por diferentes fontes (e.g., sensores, web e aplicações clientes), que são responsáveis por gerar um fluxo de entrada para o mecanismo CEP;

2. Especialistas da área de aplicação do sistema especificam as regras e padrões que serão instanciados no mecanismo CEP. O mecanismo processa o fluxo de eventos, emitindo eventos derivados sempre que as regras e padrões forem satisfeitos;

3. Consumidores recebem os eventos derivados, vindo a engatilhar alertas e executar ações de maneira automática.

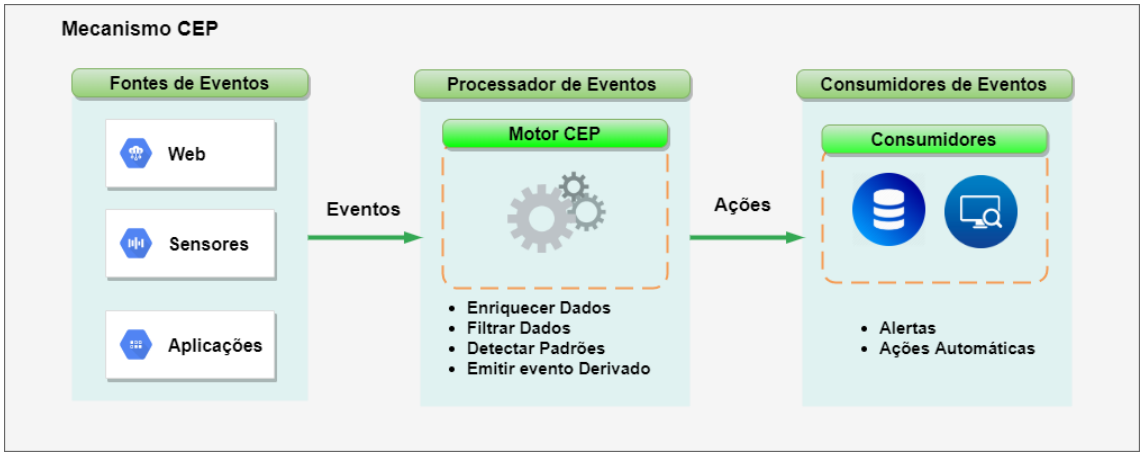

Figura 4.1. Visão geral de um mecanismo CEP.

4.3.1. Motor CEP Esper

Esper [EsperTech][1] é um motor CEP implementado em Java que atende a necessidade de processamento online de fluxos de dados, sendo um projeto de código aberto. Este

[1] http://www.espertech.com/

motor CEP é baseado em cláusulas do tipo *Event Condition Action* (ECA), a qual especifica que o motor CEP analisa condições no fluxo de dados e, caso satisfeita, uma ação é acionada [Wang 2014]. A instanciação dessas condições é realizada através de consultas CEP, as quais são descritas através de uma linguagem declarativa denominada *Event Processing Language* (EPL), que implementa e estende os operadores do *Structured Query Language* (SQL). Através desta linguagem de consulta, é possível extrair padrões de relacionamentos entre eventos, aplicar filtros, efetuar agregações e realizar cálculos de funções sobre conjuntos de eventos tais como: médias, máximos, mínimos e contagem. Assim, a expressividade desta linguagem permite realizar consultas complexas no fluxo de eventos, tornando-a adequada para tarefas que requisitem respostas em tempo real e processamento incremental. No decorrer deste capítulo, serão apresentados configurações e implementações do *Esper* através da linguagem de programação Java, assim como definições de consultas contínuas por meio de EPLs.

4.3.2. Primeiros Passos

Esta seção apresenta os passos básicos da utilização do motor CEP *Esper*, descrevendo as configurações e implementações necessárias para instanciar um fluxo de trabalho CEP. O primeiro passo necessário é criar uma instância da classe *EPServiceProviderManager*, que é obtida chamando seu método denominado *getDefaultPovider()*. O Código 4.1 apresenta a implementação da obtenção de uma instância da classe *EPServiceProviderManager*.

Código 4.1. Obter instância do mecanismo.

```
EPServiceProvider engine = EPServiceProviderManager.getDefaultProvider
    ();
```

No próximo passo, é necessário adicionar ao motor *Esper* informações sobre os eventos que serão processados. Esta etapa é necessária para que o *Esper* consiga reconhecer a estrutura básica dos eventos, permitindo assim realizar validações no decorrer do processamento. Desta forma, projeta-se as classes que serão responsáveis por estruturar os eventos, isto é, cada instância da classe representa um evento. Por exemplo, o Código 4.2 implementa a classe que estrutura os eventos do tipo *SocialEvent*, contendo os atributos *userId*, *eventType* e *duration*.

Código 4.2. Obter instância do mecanismo.

```
package com.lsdi.social.mhealth.model;

 public class SocialEvent {
    private long userId;
    private String typeEvent;
    private Double duration;

 public SocialEvente(long userId, String typeEvent, Double duration) {
    this.userId = userId;
    this.typeEvent = typeEvent;
    this.duration = duration
  }

  public long getUserId() {
```

```
    return id;
  }
  public String getTipeEvent() {
    return startTime;
  }
   public Double getDuration() {
    return endTime;
  }
}
```

Para informar ao *Esper* que a classe *SocialEvent* apresenta uma estrutura de um evento válido é necessário chamar o método *addEventType()*, que é derivado da interface de configuração. O Código 4.3 informa ao *Esper* que o evento *SocialEvent* apresenta uma estrutura de evento válida a ser processada. Após informar os tipos válidos de eventos, é possível implementar regras CEP que serão aplicadas ao fluxo de eventos.

Código 4.3. Adicionar informações dos eventos válidos.

```
engine.getEPAdministrator().getConfiguration().addEventType(SocialEvent
    .class);
}
```

O próximo passo é especificar as EPLs que irão detectar padrões e realizar modificações no fluxo de eventos gerado. Nesta etapa, os especialistas de domínio utilizam seu conhecimento para implementar regras complexas para detectar situações de interesse. Similar ao SQL, os especialistas podem utilizar regras de seleção, agregação (e.g., count, sum, max e avg), agrupamento (group by), filtro (e.g., where, having), dentre outras cláusulas. A EPL projetada deve ser inserida no *Esper* através do método *createEPL()*, que é derivado da interface *Administrator*. O Código 4.4 apresenta uma EPL para selecionar todos os eventos do tipo *SocialEvent* que possuem duração superior a 50 segundos e inserir esta EPL no Esper.

Código 4.4. EPL para selecionar *SocialEvent* com duração superior a 50 segundos.

```
String epl = "select * from SocialEvent where duration > 50";
EPStatement statement = engine.getEPAdministrator().createEPL(epl);
```

Até esse ponto, o *Esper* já está configurado e possui EPLs instanciadas. Entretanto, ainda é necessário implementar classes ouvintes que são responsáveis por receber resultados sempre que o fluxo de evento satisfazer as regras definidas nas EPLs. Para tanto, utiliza-se uma classe de retorno denominada *EPStatement* que receberá os resultados das EPLs. O Código 4.5 apresenta a implementação de um ouvinte para uma *statement*, que recebe o resultado da EPL e imprimi os dados do evento no console.

Código 4.5. Ouvinte de uma EPL.

```
statement.addListener( (newData, oldData) -> {
  long userId = (long) newData[0].get("userId");
  String eventType = (String) newData[0].get("typeEvent");
  Double duration = (Double) newData[0].get("duration");
 System.out.println(" Id: "+userId+"; TypeEvent: "+typeEvent+"; Duracao
    : "+duration+"  ");
});
```

Por fim, é necessário inserir os eventos no mecanismo, para assim começar a identificar padrões. Para tanto, deve-se utilizar o método *sendEvent()*, que é derivado da interface *Runtime*. O Código 4.6 envia eventos para o *Esper* através do método *sendEvent()*.

Código 4.6. Enviando eventos para o mecanismo.

```
1 engine.getEPRuntime().sendEvent(new SocialEvent(1, "Conversacao", 120))
    ;
```

4.3.3. Agentes de Processamento de Eventos e Rede de Processamento de Eventos

Na área de sistemas distribuídos, os fluxos de eventos gerados podem apresentar estruturas (sintaxe) e/ou significados (semântica) incompatíveis [Luckham 2008] com os consumidores. Em outros casos, o processamento de apenas um evento primitivo não é suficiente para desencadear uma ação no consumidor de eventos, necessitando de uma combinação complexa de eventos [Luckham 2008]. Assim, é necessário utilizar os *Event Processing Agents* (EPAs), que são módulos de software responsáveis por detectar padrões e realizar modificações no fluxo de dados. O fluxo básico de trabalho de um EPA é: (i) receber eventos de entrada; (ii) realizar processamento sobre o fluxo de entrada; e (iii) encaminhar ou derivar novos eventos. O tipo de um EPA é definido através de suas regras instanciadas, sendo classificado da seguinte forma [Etzion et al. 2011]:

- **Agentes de filtragem:** eliminam eventos que não satisfazem uma determinada condição;
- **Agentes de detecção de padrões:** examinam o fluxo para detectar a ocorrência de padrões específicos;
- **Agentes de transformação:** modificam o conteúdo dos eventos processados.

Geralmente, uma arquitetura de sistemas CEP é composta por uma *Event Processing Network* (EPN) [Luckham 2008], que é uma representação conceitual da interconexão de um conjunto de EPAs, e são responsáveis pela análise e manipulação do fluxo de eventos. Assim, uma EPN pode ser definida como a representação do comportamento do mecanismo de processamento, definindo conceitualmente a conexão entre o conjunto de agentes de processamento, produtores e consumidores. Portanto, em uma EPN, os eventos derivados produzido por uma EPA podem criar um novo fluxo de eventos de entrada para outras EPAs, que por sua vez, poderão realizar processamentos adicionais. A Figura 4.2 apresenta a estrutura de uma EPN.

4.3.4. Contexto e Partições de Contexto

O *Esper* também fornece o conceito de contextos [EsperTech], os quais permitem agrupar um conjunto de eventos que estão relacionados a um determinado grupo, vindo a segmentar o fluxo em uma ou mais partições de contexto. Portanto, as operações de processamento de eventos são associadas a contextos específicos, operando em cada uma dessas partições independentemente [Cugola and Margara 2012]. As partições de contexto podem ser definidas como um subconjunto de eventos relacionados, em que a atribuição de eventos a partições de contextos é realizada com base em suas propriedades (e.g., lógico,

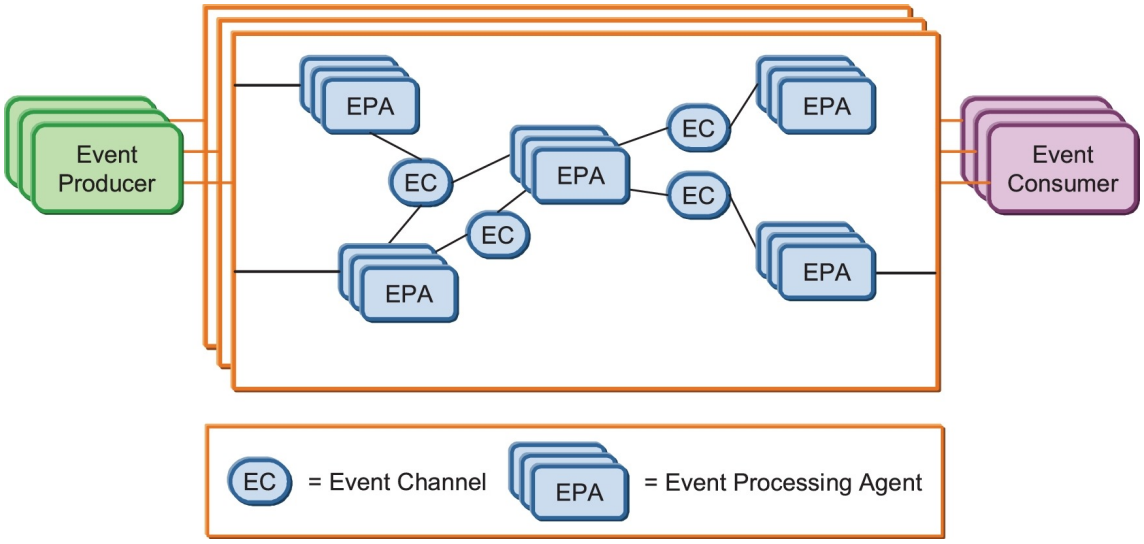

Figura 4.2. Rede de Processamento de Eventos [Etzion et al. 2011].

espacial e temporal). A Figura 4.3 apresenta as dimensões de contexto nas quais o fluxo de dados pode ser agrupado: temporal, espacial, estado e atributo.

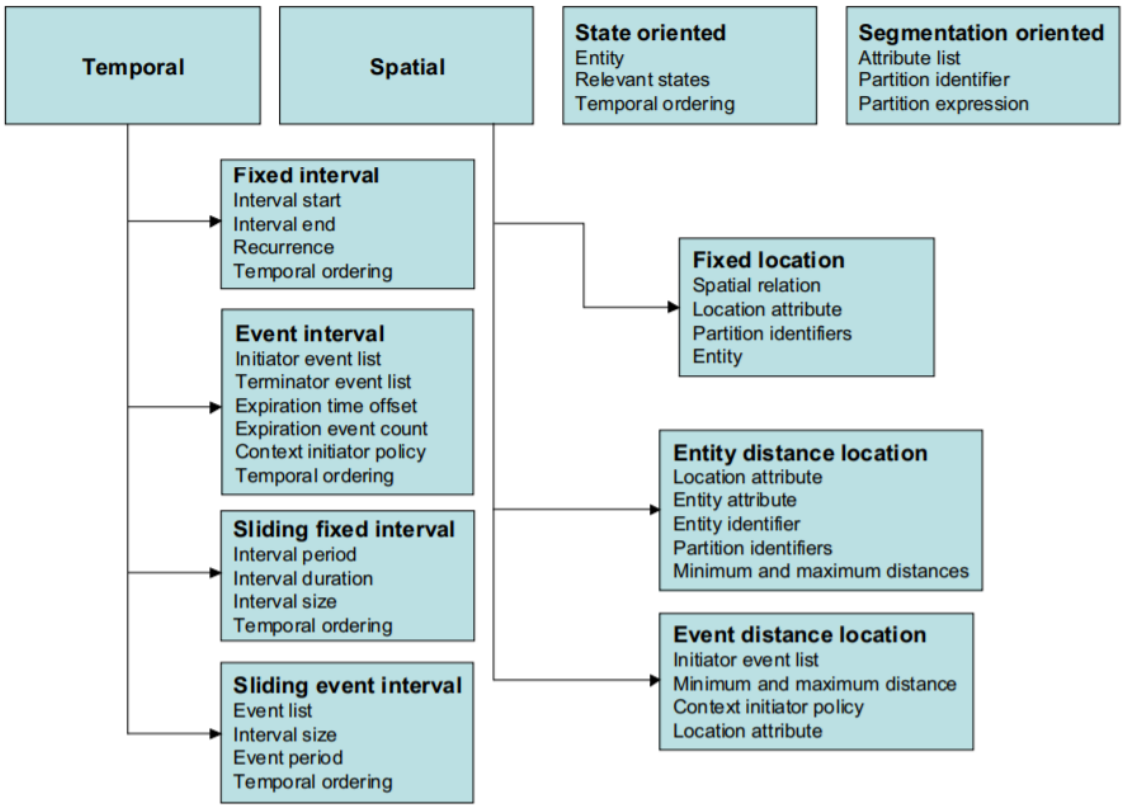

Figura 4.3. Dimensões de contexto [Etzion et al. 2011].

Por exemplo, em um cenário de monitoramento do comportamento social humano, cada pessoa pode ser considerada um contexto, no qual as regras de agregações, padrões

ou janelas de dados são específicas do indivíduo. Portanto, os *SocialEvent* devem ser processados de maneira individualizada para detectar padrões personalizados. Neste exemplo, o contexto detectado será o indivíduo monitorado e a análise individual dos eventos sociais pode ser feita utilizando agregações (e.g., *count*, *sum*, *max* e *min*), consultas (i.e., *select*), padrões (i.e., *pattern*), dentre outras operações. O Código 4.7 cria um contexto *SegmentedByUser* que considera o valor da propriedade *userId* como a chave de identificação do indivíduo dentre todos os eventos do tipo *SocialEvent*. Essa chave será a propriedade que indicará para qual partição de contexto um evento será encaminhado.

Código 4.7. Criação de contexto para o fluxo *SocialEvent*.

```
1 CREATE CONTEXT SegmentedByUser PARTITION BY userId FROM SocialEvent
```

O Código 4.8 apresenta um exemplo da utilização do contexto *SegmentedByUser* (Código 4.7), onde uma consulta é realizada para contabilizar a duração total (i.e., *SUM duration*) dos eventos sociais para cada indivíduo.

Código 4.8. Consulta e agregação realizada sobre o fluxo *SocialEvent*.

```
1 CONTEXT SegmentedByUser
2 SELECT userId, SUM(duration) FROM SocialEvent GROUP BY duration
```

O contexto segmentado por categoria é um conceito fornecido pelo *Esper* que permite agrupar instâncias de eventos em partições de contexto com base no valor de um atributo ou coleção de atributos [EsperTech]. Isto é, utiliza uma expressão de predicado para definir a associação do evento a partição de contexto. Portanto, cada evento pode pertencer a nenhuma, uma ou várias partições de contexto, de acordo com o resultado das expressões de predicado. Por exemplo, considere um EPA que recebe um fluxo de eventos sociais de um usuário, no qual cada evento contém o atributo dia da semana (e.g., domingo, segunda-feira, terça-feira, etc). O valor desse atributo pode ser usado como predicado para agrupar eventos de tal forma que haja uma partição separada para cada dia. Cada partição de contexto contém eventos relacionados a um dia da semana, para que o comportamento do indivíduo seja modelado independente dos outros dias. Considerando o exemplo apresentado, o Código 4.9 expressa uma EPL que define as categorias de contexto referentes aos dias: segunda, terça, quarta, sábado e domingo. Essa EPL permite usar o atributo *ctxDay* contido no fluxo *SocialEvent* como um predicado para segmentar o fluxo em partições de contexto.

Código 4.9. Contexto segmentado por categoria.

```
1 CREATE CONTEXT CategoryDayContext
2 GROUP ctxDay=Segunda AS SEGUNDA,
3 GROUP ctxDay=Terca AS TERCA,
4 GROUP ctxDay=Quarta AS QUARTA,
5 GROUP ctxDay=Sabado AS SABADO,
6 GROUP ctxDay=DOMINGO AS DOMINGO
7 FROM SocialEvent
```

4.3.5. Janela de Tempo

O motor CEP *Esper* também fornece o conceito de janelas de tempo, que é definido como um contexto temporal [Etzion et al. 2011] que subdivide o fluxo de eventos em

intervalos, aplicando as regras e operadores apenas a eventos contidos nesse espaço de tempo. Por exemplo, se definirmos uma regra com janela de tempo de 20 segundos, é criada automaticamente uma partição de contexto que agrupará os últimos eventos que ocorreram nesse intervalo, aplicando os operadores definidos a este grupo de eventos.

Existem alguns tipos de janelas de tempo que podem ser utilizadas, como as janelas *Landmark* e *Sliding* [EsperTech]. A Figura 4.4 apresenta um exemplo de janelas de dados do tipo *Landmark* e *Sliding* [Roriz 2017]. Especificamente, estas janelas diferem no modo de manter os eventos mais atuais agrupados. A janela do tipo *Landmark* utiliza uma estratégia de processamento em lote, mantendo agrupado todos os eventos que ocorreram durante um intervalo de tempo t, aplicando as regras e operadores neste conjunto de eventos. Outro tipo de janela é a *Sliding*, que move uma janela de referência no decorrer do processamento do fluxo, ou seja, permite especificar regras que serão aplicadas apenas aos eventos que ocorreram nas últimas unidades de tempo t. Este conceito de janelamento de dados é importante para processamento de fluxos de dados, uma vez que oferece suporte ao processamento contínuo do fluxo de dados mais recente.

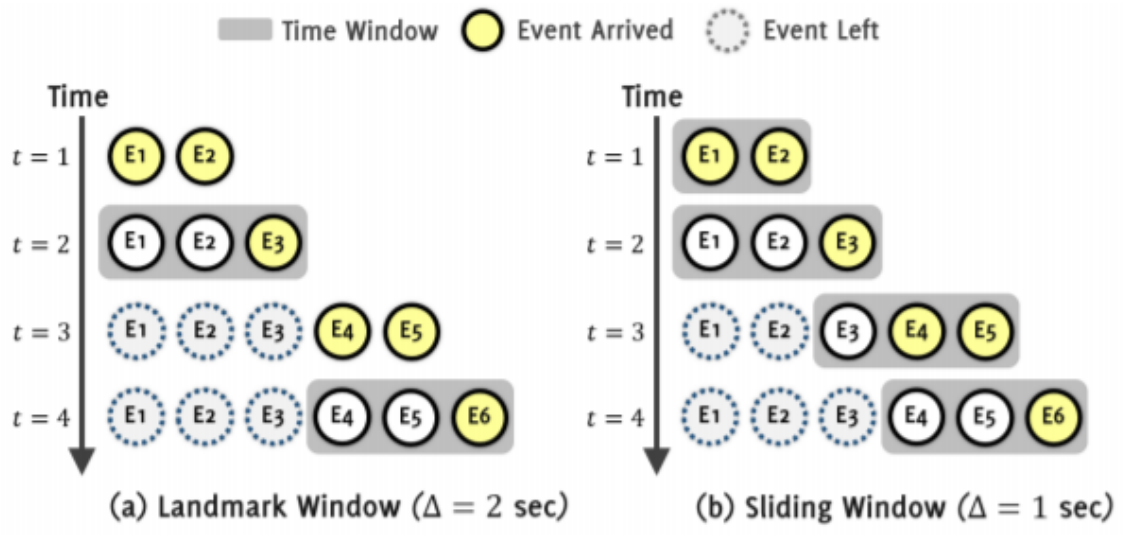

Figura 4.4. Exemplo de janelas de dados *Landmark* e *Sliding* [Roriz 2017].

O Código 4.10 apresenta uma EPL que especifica uma janela de dados. Neste exemplo, será realizado a contagem de eventos do tipo *SocialEvent* em uma janela de dados de 60 segundos.

Código 4.10. Especificando janela de dados.

```
1 select count(*) from SocialEvent.win:time(60 sec)
```

4.3.6. Padrões

Os padrões são utilizados para identificar quando a ocorrência de um ou vários eventos evento satisfazem uma determinada situação especificada pelo padrão projetado pelo especialista. Para especificar padrões, podem ser utilizados 4 tipos de operadores, descritos abaixo [Kyriakopoulos 2018].

- *and*, *or* e *not*: operadores lógicos;
- *every*, *every-distinct*, *[num]* e *until*: operadores que controlam subexpressões de repetição do padrão;
- -> (seguido por): operador temporal que controla a ordem dos eventos;
- *timer:within*, *timer:withinmax* e *while*: operadores de guarda ou *where-conditions* que controlam o ciclo de vida das sub-expressões.

O Código 4.11 demonstra um exemplo de uso de padrões em regras CEP. Neste exemplo, uma EPL é expressada para corresponder a todos os eventos do tipo *SocialEvent* que apresentam duração superior a 60 segundos ou são derivados de uma conversação (typeEvent). Logo, para que o fluxo de eventos corresponda ao padrão, pelo menos uma das duas condições devem ser verdadeiras. A especificação de padrões pode ser incluída em qualquer lugar de uma regra CEP, como nas cláusulas *from*, *joins* e *subqueries*, como exemplificado abaixo.

Código 4.11. Exemplo de uso de padrão.

```
1 EVERY (
2    dur = SocialEvent(duration > 60)
3 OR
4    type = SocialEvent(typeEvent = 'Coversation')
5   )
```

4.4. Ferramenta de Detecção de Padrões de Sociabilidade

Com base na necessidade de desenvolver soluções capazes de monitorar objetivamente o comportamento social, pesquisadores do Laboratório de Sistemas Distribuídos Inteligentes (LSDi), situado na Universidade Federal do Maranhão (UFMA), desenvolveram uma ferramenta capaz de processar inferências de atividades sociais derivadas de dispositivos pervasivos para detectar padrões de sociabilidade [Rodrigues et al. 2020] sensíveis ao contexto. A ferramenta é uma biblioteca com uma API bem definida em linguagem Java. O reconhecimento dos padrões de sociabilidade é realizado para contextos específicos (e.g., dias úteis, dias chuvosos e fins de semana), permitindo a identificação da variabilidade do comportamento em diferentes condições de contexto. A solução desenvolvida também é capaz de identificar mudanças nos padrões de sociabilidade que refletem comportamentos sociais anormais e variações nas rotinas sociais. Esta solução foi implementada com base na combinação da abordagem de Mineração de Padrões Frequentes (FPM) [Aggarwal et al. 2014] com o CEP [Etzion et al. 2011]. Em resumo, as funcionalidades da ferramenta a ser apresentada são:

- Realiza o aprendizado incremental de padrões de sociabilidade enriquecidos por contexto através da combinação de FPM e CEP;
- Identifica mudanças nos padrões de sociabilidade que refletem comportamentos sociais anormais e variações nas rotinas sociais;

- Utiliza a lógica *fuzzy* para modelar o conhecimento do especialista na tarefa de detecção de comportamentos anormais e mudanças de rotinas sociais;
- Fornece uma API para permitir a rápida implementação de estratégias para detectar padrões de sociabilidade enriquecidos por contexto e configurar o reconhecimento de mudanças comportamentais.

4.4.1. Solução Arquitetônica e Principais Características

O CEP permite a ferramenta reagir em tempo real ao fluxo de dados por meio de uma linguagem de consulta contínua. Este método processa dados como uma sequência de eventos [Etzion et al. 2011], na qual cada evento modela uma observação em um domínio específico. Por exemplo, nesta solução, um evento representa uma atividade social em um horário específico do dia. Esses eventos são gerados por dispositivos ubíquos, que podem fazer inferências de situações sociais processando dados de contexto. Para implementar o algoritmo embarcado na ferramenta, foi utilizado o Esper, visto que este possibilita realizar a análise contínua de fluxos. Assim, o fluxo de processamento da ferramenta (Figura 4.5) consiste nas seguintes etapas:

- **(a) *Enrich Social Event*:** injeta o *slot* (extraído de seu *timestamp*) e os *Context Atibutes* (CAs - e.g., dia da semana, dia chuvoso) nos eventos sociais. O resultado desse processo é a emissão de um evento enriquecido intitulado *SocialUpdate*;
- **(b) *Context Partition EPA*:** segmenta o fluxo *SocialUpdate* com base nos CAs. Portanto, um evento derivado chamado *ContextEvent*, que possui o *slot* e o rótulo do contexto, é emitido para cada CA do evento;
- **(c) *Count Table*:** é uma tabela nomeada que é responsável por manter a contagem dos eventos que ocorreram em determinados contextos em cada *slot*. Assim, esta contagem é atualizada a cada evento derivado *ContextEvent* emitido;
- **(d) *Candidate Slot EPA*:** verifica quais *slots* alcançaram um número adequado de eventos para se tornarem candidatos a formar um intervalo de sociabilidade, enviando-os para a fase de extração de padrões;
- **(e) *Extract Pattern*:** Identifica quais conjuntos de *slots* candidatos formam um intervalo de tempo no qual o indivíduo habitualmente socializa.

4.4.2. Detecção de Mudança de Rotina Social e Comportamentos Sociais Anormais

A detecção de comportamentos sociais anormais é implementada através da abordagem de processamento de janelas de dados em combinação com uma métrica de similaridade, que permite verificar a mudança de padrão de um instante de tempo t1 para t2. Portanto, a ferramenta compara um padrão de sociabilidade e uma observação de comportamento social. A Figura 4.6 apresenta a estratégia de detecção de comportamentos anormais da ferramenta. O especialista deve definir o tamanho da janela de dados de uma observação social (e.g., um dia, cinco dias ou uma semana), assim como a quantidade de observações necessárias para extrair um padrão de sociabilidade.

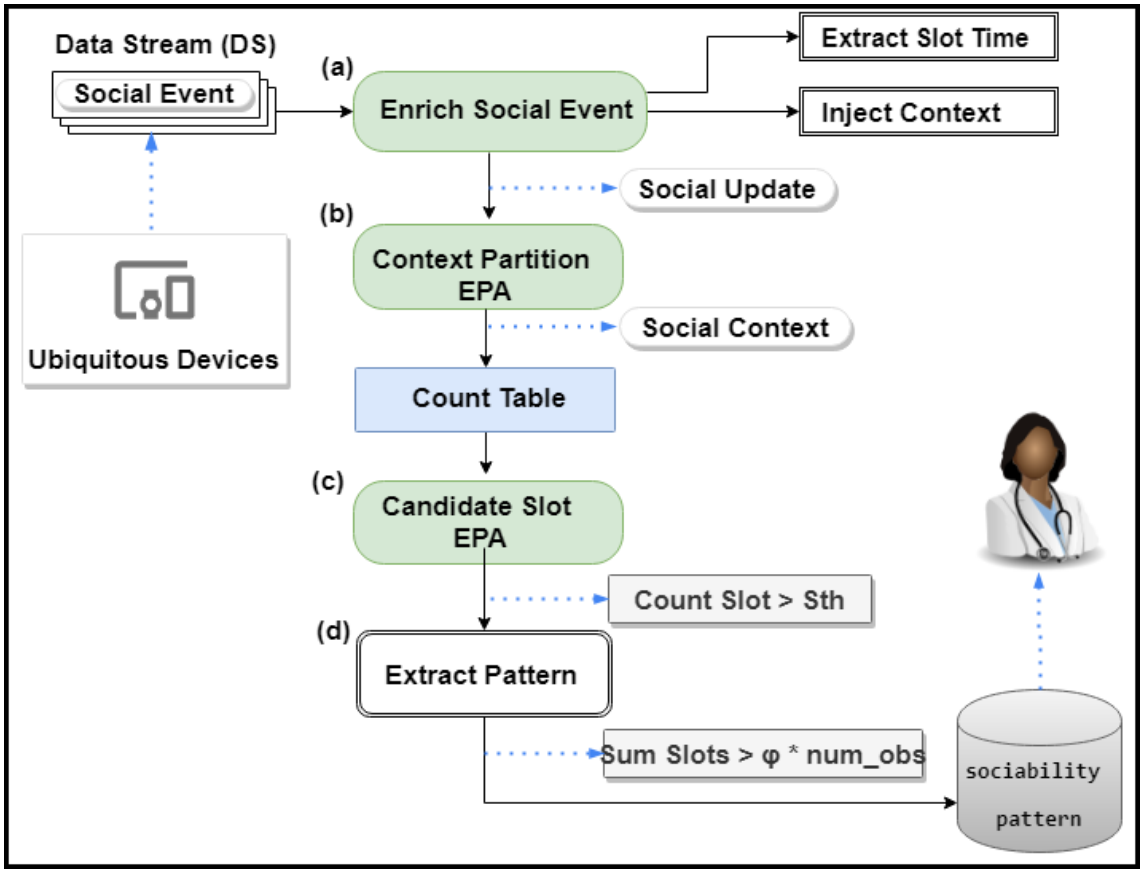

Figura 4.5. Fluxo de processamento da ferramenta desenvolvida.

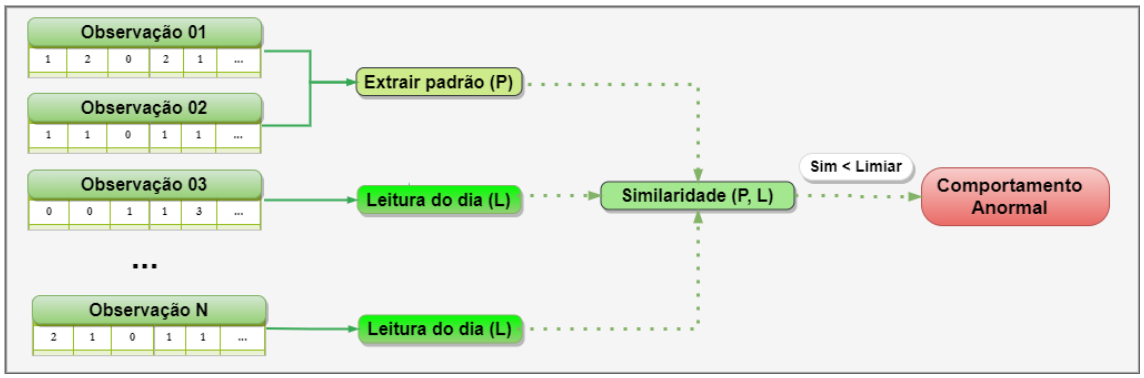

Figura 4.6. Estratégia de detecção de comportamentos sociais anormais.

A detecção de mudança de rotina social é baseada na comparação entre padrões de sociabilidade consecutivos. Portanto, a ferramenta utiliza a métrica de similaridade de *Jaccard* ($J(A,B) = \frac{A \cap B}{A \cup B}$) para comparar padrões de sociabilidade. Quando é identificado uma similaridade inferior a um limite definido pelo especialista, duas tarefas são desencadeadas, a saber: (i) atualização do padrão de sociabilidade atual com a rotina mais recente; (ii) emissão de um evento para notificar as partes interessadas sobre a detecção de mudança de rotina social.

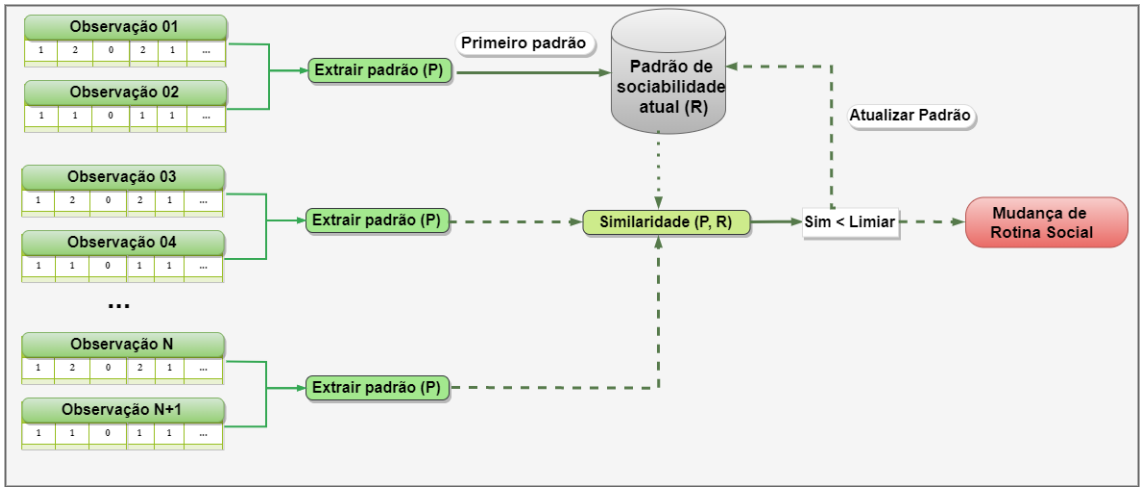

Figura 4.7. Estratégia de detecção de mudança de rotina social.

4.4.3. Modelagem do Conhecimento do Especialista

A modelagem do conhecimento especialista é implementada através dos conceitos da lógica *fuzzy*, que permite o mecanismo de detecção de mudanças comportamentais emitir notificações considerando a natureza imprecisa desta tarefa. Especificamente, é utilizada a biblioteca de código aberto *jFuzzyLogic* [2], que viabiliza o desenvolvimento de controladores *fuzzy*. Esta biblioteca implementa a *Fuzzy Control Language* (FCL), que é uma linguagem específica de domínio para utilizar os conceitos da lógica *fuzzy*. Portanto, a *jFuzzyLogic* permite integrar na ferramenta um Sistema de Inferência *Fuzzy* (SIF), possibilitando o especialista especificar as variáveis, os conjuntos *fuzzy* e as regras.

Em princípio, o especialista deve determinar os conjuntos *fuzzy* que compõem o universo de discurso. Especificamente, são definidos três conjuntos *fuzzy*, a saber, *sensibilidade*, *similaridade* e *drift*. O conjunto *sensibilidade* é responsável por definir o nível de discrepância entre padrões que representam uma mudança, isto é, modelam o conhecimento da sensibilidade de detecção de mudanças. O conjunto *similaridade* é responsável por definir os níveis de correspondência entre os padrões avaliados. O conjunto *drift* representa a saída do SIF (i.e., *defuzzifier*), responsável por modelar os níveis de mudanças de comportamentos sociais. Por fim, o especialista deve especificar as preposições lógicas que devem ser estruturadas do seguinte modo: <condição> *AND* <condição> *THEN* <consequência>, que orientaram a decisão do mecanismo.

4.4.4. Orientações para Utilizar a Ferramenta

A ferramenta de detecção de padrões de sociabilidade, através do CEP, fornece uma API de programação de fácil implementação. Entretanto, algumas configurações são necessárias antes de executar a ferramenta. Especificamente, são necessárias as seguintes configurações: (i) configurar os parâmetros dos algoritmos utilizados pela ferramenta; (ii) configurar os atributos de contexto considerados pela rede de processamento CEP; (iii) implementar os conjuntos *fuzzy* através da FCL; e (vi) utilizar a API de programação para

[2] jfuzzylogic.sourceforge.net

definir as estratégias de detecção de padrões de sociabilidade e mudanças de comportamentos sociais.

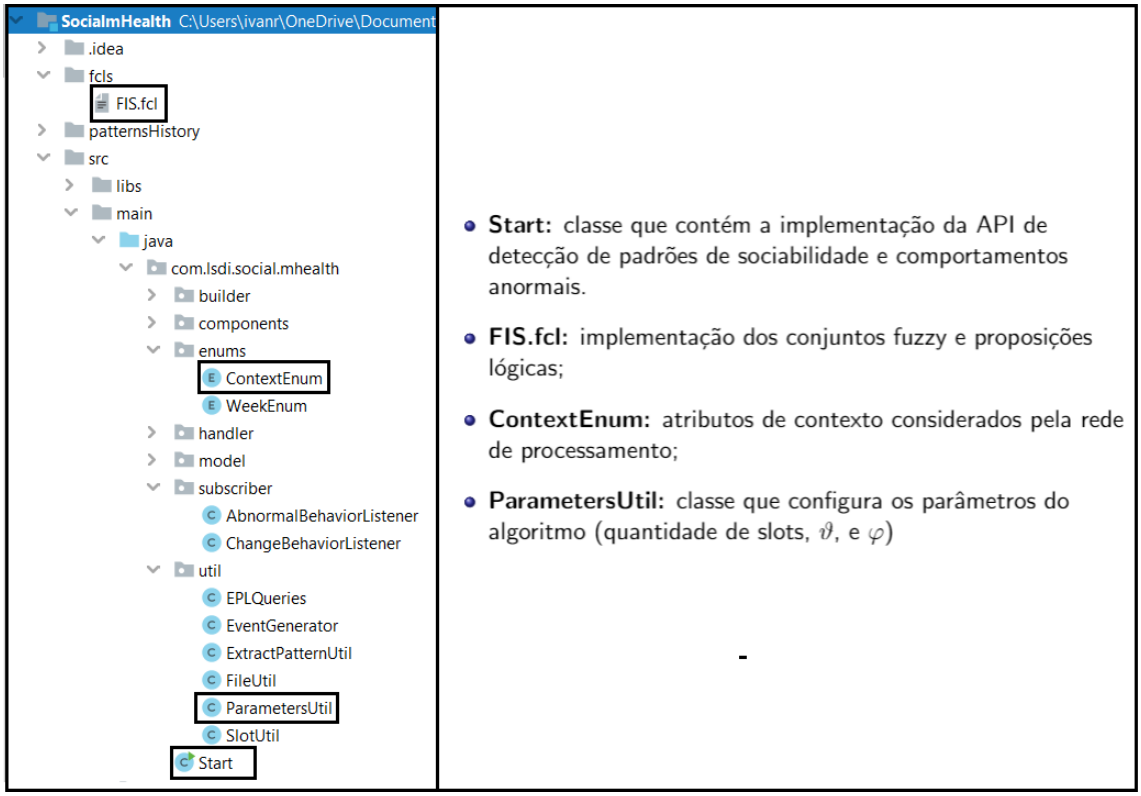

Figura 4.8. Estrutura do código fonte da ferramenta.

4.4.4.1. Implementando o Sistema de Inferência *Fuzzy*

A primeira configuração necessária é representar o conhecimento especialista através do SIF embutido na ferramenta. Nesta etapa, através da FCL, o desenvolvedor especifica os conjuntos de fuzzificação (i.e., similaridade e sensibilidade) e de defuzzificação (i.e., *drift*). O Código 4.12 apresenta um exemplo de implementação dos conjuntos similaridade, sensibilidade e *drift* através da FCL.

Código 4.12. Especificação dos conjuntos *fuzzy*.

```
1
2  FUZZIFY sensibility
3    TERM low := (0, 1) (25, 1) (50, 0);
4    TERM moderate := (25, 0) (50,1) (75, 0);
5    TERM high := (50, 0) (75, 1) (100, 1);
6  END_FUZZIFY
7
8  FUZZIFY similarity
9    TERM low := (0, 1) (40, 1) (50, 0);
10   TERM moderate := (40, 0) (50,1) (60, 0);
11   TERM high := (50, 0) (60, 1) (100, 1);
12 END_FUZZIFY
```

```
13
14 DEFUZZIFY drift
15   TERM no_change := (0, 1) (50, 1) (65, 0);
16     TERM moderate_change := (50, 0) (65, 1) (80, 0);
17     TERM change := (65, 0) (80, 1) (100,1);
18   METHOD : COG; // Use 'Center Of Gravity' defuzzification method
19 END_DEFUZZIFY
```

Por fim, deve-se especificar as preposições lógicas que nortearam a decisão do FIS. O Código 4.13 apresenta um exemplo de implementação de preposições lógicas baseadas nos conjuntos *fuzzy* especificados no Código 4.12. Através dessas regras, a ferramenta é capaz de detectar mudanças de comportamento social com graus de pertinência no conjunto *drift*, modelando assim o conhecimento especialista necessário para esta tarefa.

Código 4.13. Especificação de Proposições Lógicas

```
1 RULEBLOCK No1
2     RULE 1 : IF sensibility IS low AND similarity IS low THEN drift IS
      change;
3   RULE 2 : IF sensibility IS low AND similarity IS moderate THEN drift
      IS no_change;
4   RULE 3 : IF sensibility IS low AND similarity IS high THEN drift IS
      no_change;
5     RULE 4 : IF sensibility IS moderate AND similarity IS low THEN
      drift IS change;
6   RULE 5 : IF sensibility IS moderate AND similarity IS moderate THEN
      drift IS moderate_change;
7   RULE 6 : IF sensibility IS moderate AND similarity IS high THEN drift
       IS no_change;
8     RULE 7 : IF sensibility IS high AND similarity IS low THEN drift IS
       change;
9   RULE 8 : IF sensibility IS high AND similarity IS moderate THEN drift
       IS change;
10  RULE 9 : IF sensibility IS high AND similarity IS high THEN drift IS
      no_change;
11 END_RULEBLOCK
```

4.4.4.2. Definição dos Atributos de Contexto

O usuário deve especificar os CAs na classe *ContextEnum*. O Código 4.14 apresenta um exemplo de implementação de CAs dos dias da semana (segunda a sexta) e semana (dia útil e final de semana). Desta forma, a ferramenta será capaz de detectar padrões de sociabilidade baseados nos CAs registrados, permitindo modelar a sociabilidade para cada situação de interesse.

Código 4.14. Definição dos Atributos de Contexto.

```
1 package com.lsdi.social.mhealth.enums;
2
3 public enum ContextEnum {
4     ALL_, WEEK_,WEEKEND_,SUNDAY_, MONDAY_,TUESDAY_,WEDNESDAY_,THURSDAY_
      ,FRIDAY_,SATURDAY_;
5 }
```

4.4.4.3. Configurando os Parâmetros da Ferramenta

Nesta etapa, necessita-se configurar os valores dos parâmetros da ferramenta. Para tanto, na classe denominada *ParametersUtil*, o usuário deve atribuir valores aos atributos que são responsáveis por configurar a quantidade de *slots* de tempo (NUM_SLOTS), e limiares de suporte para a detecção de padrões de sociabilidade (THETA e PHI). O Código 4.15 apresenta um exemplo de definição de valores para os parâmetros da ferramenta.

Código 4.15. Configuração dos Parâmetros da Ferramenta.

```
1 package com.lsdi.social.mhealth.util;
2
3 public class ParametersUtil {
4     public static double T = 0.5;
5     public static double NUM\_SLOTS =  24/T; //48 slots
6     public static double THETA = 0.02;
7     public static double PHI = 0.7;
8 }
```

4.4.4.4. Configurando o *Broker MQTT*

A ferramenta de detecção de padrões de sociabilidade utiliza o *Message Queuing Telemetry Transport* (MQTT) [3] como protocolo de distribuição de dados, vindo utilizar esse meio de comunicação para enviar os padrões de sociabilidade e notificações de mudanças de comportamentos sociais para as aplicações clientes. Portanto, faz-se necessário instanciar um *broker MQTT* responsável por intermediar a comunicação entre as fontes de eventos, ferramenta e aplicações clientes. A seguir, apresenta-se os passos básicos para configurar um *broker MQTT*:

- O usuário pode aproveitar o *broker MQTT* do Eclipse instanciado em iot.eclipse.org ou configurar localmente em sua máquina.
- Passos da instalação local:
  1. Baixar o executável do *broker* Mosquitto nesta URL: https://mosquitto.org/download/;
  2. Executar o instalador do *broker* Mosquitto;
  3. Executar o *prompt* de comando como administrador e entrar no diretório onde o *broker* Mosquitto está instalado (e.g., “cd C:\mosquitto”);
  4. Iniciar o serviço do *broker* através do comando: “net start mosquitto”.

4.4.4.5. API de Programação

O Código 4.16 apresenta um exemplo da utilização da API fornecida pela ferramenta. Especificamente, o usuário deve criar dois objetos, a saber, *StreamReceiver* e *SociabilityPattern*. O objeto *StreamReceiver* é responsável por configurar a conexão entre a ferramenta

[3] http://mqtt.org/

e o *broker* MQTT instanciado, vindo a atribuir valores para a URL do *broker* e para o tópico no qual a ferramenta irá se subscrever para receber eventos sociais. O objeto *SociabilityPattern* é responsável por configurar e habilitar todo o funcionamento da solução. No construtor, dois parâmetros são especificados, sendo estes o nome do contexto a ser considerado e o nível de sensibilidade de detecção. Logo após, o tópico raiz é especificado, o qual será utilizado para enviar as notificações. Esta informação é necessária pois o mecanismo publica os novos padrões de sociabilidade e notificações de mudanças de comportamentos no *broker* MQTT, permitindo aplicações clientes interessadas se subscreverem neste tópico para receber atualizações. Por fim, o usuário poderá habilitar as estratégias de detecção de comportamentos anormais e mudanças de rotina social.

Código 4.16. API de Programação.

```
1
2  StreamReceiver receiver = new StreamReceiver();
3         receiver.setBroker("tcp://127.0.0.1:1883");
4         receiver.setTopic("social");
5         receiver.receiverStream();
6
7 SociabilityPattern sociabilityPattern = new SociabilityPattern
8             .Builder(ContextEnum.MONDAY.toString(),
    sensitivityOfChange:50.0)
9             .setRootTopic("com/lsdi/sociability")
10            .setAbnormalBehavior(true)
11            .setChangeBehavior(true)
12            .build();
```

Ao executar a classe *Start* a ferramenta estará pronta para processar os eventos sociais. As notificações serão publicadas em tópicos no *broker* MQTT configurado, os quais serão subscrito por aplicações clientes interessadas em receber notificações sobre padrões de sociabilidade encontrados e mudanças de comportamentos sociais. As estruturas dos tópicos são:

- Estrutura básica: tópico raiz/contexto/tipo do evento:
  - 'com/lsdi/sociability/MONDAY/newPattern'
  - 'com/lsdi/sociability/FRIDAY/AbnormalBehavior'
  - 'com/lsdi/sociability/SUNDAY/ChangeBehavior'

4.5. Aplicações da Detecção de Padrões de Sociabilidade

Atualmente, o aspecto social dos indivíduos tem implicações diretas com o seu estado de bem-estar, podendo realizar um papel protetor ou contribuir para a degradação da saúde mental. Desta forma, existe um interesse crescente em monitorar o comportamento social dos indivíduos para realizar intervenções e detectar situações de interesse para profissionais da saúde em tempo real. Como discutido neste capítulo, a detecção de padrões de sociabilidade pode ser realizada através do processamento de fluxos de dados gerados pelos dispositivos pervasivos, onde se aplica técnicas computacionais como CEP e aprendizado de máquina. Estas ferramentas possuem aplicações em diversos domínios, os quais são apresentados a seguir.

4.5.1. Análise Comportamental

Um domínio que as aplicações de detecção de comportamento social podem ser aplicadas é a análise comportamental, que objetiva monitorar e entender o comportamento de um indivíduo ou conjunto de indivíduos (e.g., famílias e comunidades) [Olguin et al. 2009]. Neste campo, uma aplicação da detecção de padrões de sociabilidade é na utilização por uma empresa para identificar se seus funcionários estão socializando. A ausência de sociabilidade pode ser um problema decorrente do estresse gerado pela altas cargas de trabalho [Moura et al. 2020]. Portanto, esta aplicação de detecção de comportamento social pode ajudar a entender o comportamento social de seus funcionários.

Monitorar o comportamento social de indivíduos em seu ambiente natural (e.g., em casa, no trabalho) através de dispositivos pervasivos, tais como *smartphones*, *smartwatches* e *smart bands*, pode fornecer parâmetros psicométricos e traçar seus padrões de sociabilidade [Markowetz et al. 2014]. Esses dispositivos são difundidos na rotina diária, o que permite que as metodologias para monitorar objetivamente em tempo real o comportamento social do indivíduo seja mais fidedigno. Uma das vantagens em se utilizar padrões de sociabilidade gerados a partir dos dispositivos pervasivos para a análise comportamental é a capacidade desses dispositivos serem onipresentes, ou seja, estão dispostos no cotidiano do indivíduo e realizando coleta de dados instantaneamente [Jun et al. 2010].

4.5.2. Saúde Mental

Outro domínio no qual aplicações de detecção de padrões de sociabilidade são bastante úteis é o monitoramento da saúde mental. Especificamente, neste domínio, os métodos tradicionais de avaliar o estado mental dos indivíduos é limitado por vieses cognitivos, como viés de memória e desejabilidade social. O viés de memória implica em usar relatórios retrospectivos dos pacientes sobre suas atividades sociais realizadas na rotina diária, nas quais o tempo de lembrança pode ser dias, semanas e até meses, embora muitas vezes confiável, a memória humana também é falível [Schacter 1999]. Já a desejabilidade social implica em um dos tipos de enviesamento de resposta, ou seja, o indivíduo é levado a dar respostas que satisfaçam suas necessidades para atribuírem a si próprios atitudes ou comportamentos com valores socialmente desejáveis e para rejeitarem em si mesmos a presença de atitudes ou comportamentos com valores socialmente indesejáveis [Edwards 1982]. Logo, os métodos tradicionais de avaliação psicológica que analisam os aspectos sociais dos indivíduos estão aos poucos sendo substituída pelas aplicações da detecção de padrões de sociabilidade.

4.5.2.1. Avaliação Momentânea Ecológica e Fenotipagem Digital

A Avaliação Momentânea Ecológica (do inglês, *Ecological Momentary Assessment* - EMA) é um importante método de pesquisa usado para coletar, em momentos fixos ou aleatórios, relatórios de indivíduos sobre percepções de seus comportamentos e sentimentos [Shiffman et al. 2008]. Geralmente, os indivíduos relatam suas percepções sobre informações de interesse (e.g, sono e sociabilidade), sentimentos (e.g., humor, ansiedade e estresse), atividades que estão realizando (ou realizadas no passado), entre outras informações relevantes para a avaliação do bem estar. O objetivo principal do uso da EMA

na avaliação do bem estar mental é identificar o fluxo de experiências e comportamentos experimentados pelos indivíduos ao longo do tempo, permitindo minimizar o viés de memória e aumentar a validade ecológica.

A definição de EMA não remete a um único método ou a uma tecnologia específica [Shiffman et al. 2008], mas refere-se a um conjunto de métodos que compartilham o objetivo de coletar dados repetidamente, em tempo real e no ambiente natural do indivíduo. Assim, é possível implementar a EMA através de diários tradicionais (em papel), telefonemas, mensagens de texto e, mais recentemente, aplicativos móveis. Os aplicativos móveis permitem o monitoramento em tempo real, além de fornecer ferramentas mais poderosas para realizar intervenções, como objetos multimídia e acesso a redes sociais online. No entanto, ainda existem vieses resultantes do relato subjetivo dos indivíduos que são fatores limitantes da EMA. Portanto, é necessário utilizar abordagens de monitoramento que possam coletar passivamente marcadores comportamentais sociais, removendo as limitações apresentadas por possíveis autorrelatos tendenciosos.

A ampla adoção de dispositivos pervasivos, incluindo *smartphones* e *tablets*, possibilita desenvolver ferramentas mais eficientes para rastrear o estado de bem estar, como a Fenotipagem Digital, um termo definido por Torous et al. [Torous et al. 2016]. Refere-se à "quantificação momento a momento do fenótipo humano em nível individual in situ usando dados de *smartphones* e outros dispositivos digitais pessoais". A fenotipagem digital visa usar tecnologias difundidas para coletar um grande volume de dados comportamentais de indivíduos, o que permite usar métodos computacionais (e.g., aprendizado de máquina, mineração de dados e processamento de eventos complexos) para transformar essas informações em conscientização sobre situações como humor, estresse, interações sociais e atividades físicas. Portanto, é possível usar essa abordagem para adicionar consciência da situação social aos aplicativos móveis que implementam EMAs. Esses aplicativos permitem que os profissionais da saúde realizem avaliações com base em informações objetivas, em vez de confiar apenas em autorrelatos subjetivos.

4.5.2.2. Computação Positiva

O termo Computação Positiva foi criado por Calvo e Peters [Calvo and Peters 2014]. É uma mudança de paradigma que defende o "design e desenvolvimento de tecnologia para apoiar o bem-estar psicológico e o potencial humano" [Calvo and Peters 2014]. A noção de computação positiva surgiu da necessidade de enfrentar os efeitos negativos do ônus do uso de alguns tipos de tecnologia, que incluem, por exemplo, o estresse causado por notificações excessivas e o sentimento de perda de privacidade. Portanto, soluções computacionais positivas são desenvolvidas para promover o bem-estar mental e ajudar a tornar realidade todas as potencialidades humanas, respeitando as necessidades psicológicas dos indivíduos. Além disso, eles são concebidos para melhorar a eficiência e a eficácia dos trabalhadores do conhecimento.

A computação ubíqua e as tecnologias da Internet das Coisas (IoT) contribuíram para promover a computação positiva inteligente [Lee et al. 2019]. Alguns benefícios de soluções que usam esse paradigma incluem a facilitação de novas maneiras de detectar mudanças no comportamento humano que podem indicar problemas de bem-estar ou o

início de transtornos mentais, fornecer intervenções terapêuticas oportunas e possibilitar o rastreamento de respostas para avaliar a eficácia das intervenções [Lee et al. 2019].

4.5.3. O Cenário da Pandemia do Novo Coronavírus (COVID-19)

O surto de infecção pelo novo coronavírus (COVID-19) entre os humanos pelo mundo está impactando drasticamente a saúde global e saúde mental [Torales et al. 2020]. Um dos efeitos dessa pandemia é o isolamento social, com mudanças no padrão de sociabilidade dos indivíduos, uma vez que as pessoas devem se manter afastadas fisicamente na tentativa de reduzir o avanço da contaminação do vírus. Este surto está levando a problemas como estresse e sintomas depressivos causados pelo isolamento social [Torales et al. 2020]. Com o isolamento social, houveram mudanças na maneira das pessoas se socializarem, com poucas interações físicas entre elas. Por outro lado, aumentaram as interações virtuais (e.g., videoconferência e uso de redes sociais online).

As aplicações pervasivas, tais como as desenvolvidas a partir da solução apresentada neste capítulo, podem ser capazes de monitorar e identificar padrões de sociabilidade das pessoas também nesse novo contexto de pandemia. Dessa forma, elas podem ajudar a monitorar as pessoas para evitarem contaminação com o vírus e também os profissionais na linha de frente [Sear et al. 2020], que enfrentam uma alta carga de estresse em decorrência do excesso de trabalho.

4.6. Considerações Finais

Este capítulo apresentou algumas técnicas de detecção de padrões de sociabilidade de seres humanos através do CEP. Inicialmente, foi introduzido uma fundamentação teórica sobre metodologias de inferência de situações sociais a partir do processamento de dados contidos em dispositivos ubíquos. Em seguida, apresentou-se técnicas de CEP para processar o fluxo de detecções de interações sociais. Especificamente, demonstrou-se os passos iniciais para configurar e utilizar o motor CEP *Esper* e apresentou-se os principais conceitos de CEP, a saber, agentes de processamento, redes de processamento, partições de contexto, janelamento de dados e padrões. Posteriormente, utilizou-se uma ferramenta que implementa os conceitos CEP para detectar padrões de sociabilidade de humanos e a ocorrência de mudanças comportamentais, vindo a introduzir a solução de arquitetura desta ferramenta, orientações básicas para sua utilização, o processo de implementação do FIS, definição dos atributos de contexto, inserção dos parâmetros, configuração do *broker* MQTT e a utilização da API de programação. Por fim, delineou-se os cenários de aplicação da metodologia apresentada de detecção de padrões de sociabilidade de humanos. Portanto, através dos padrões de sociabilidade identificados, profissionais especializados podem realizar avaliações da dimensão social dos indivíduos, vindo a basear suas ações em informações objetivas.

Referências

[Aggarwal et al. 2014] Aggarwal, C. C., Bhuiyan, M. A., and Hasan, M. A. (2014). *Frequent Pattern Mining Algorithms: A Survey*, pages 19–64. Springer International Publishing, Cham.

[Barnett et al. 2018] Barnett, I., Torous, J., Staples, P., Sandoval, L., Keshavan, M., and

Onnela, J.-P. (2018). Relapse prediction in schizophrenia through digital phenotyping: a pilot study. *Neuropsychopharmacology*, 43(8):1660.

[Calvo and Peters 2014] Calvo, R. A. and Peters, D. (2014). *Positive Computing: Technology for Well-Being and Human Potential*. The MIT Press.

[Cheek and Buss 1981] Cheek, J. M. and Buss, A. H. (1981). Shyness and sociability. *Journal of personality and social psychology*, 41(2):330.

[Cugola and Margara 2012] Cugola, G. and Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR)*, 44(3):1–62.

[Edwards 1982] Edwards, A. (1982). *The Social Desirability Variable in Personality Assessment and Research*. Greenwood Press.

[Eskes et al. 2016] Eskes, P., Spruit, M., Brinkkemper, S., Vorstman, J., and Kas, M. J. (2016). The sociability score: App-based social profiling from a healthcare perspective. *Computers in Human Behavior*, 59:39–48.

[EsperTech] EsperTech. Espertech, esper – complex event processing. http://www.espertech.com/esper/. Online; Accessed: Jul 7, 2019.

[Etzion et al. 2011] Etzion, O., Niblett, P., and Luckham, D. C. (2011). *Event processing in action*. Manning Greenwich.

[Grav et al. 2012] Grav, S., Hellzèn, O., Romild, U., and Stordal, E. (2012). Association between social support and depression in the general population: the hunt study, a cross-sectional survey. *Journal of Clinical Nursing*, 21(1-2):111–120.

[Jun et al. 2010] Jun, Z., Wenjie, D., Chunxia, Z., Chunmei, Z., Jinyong, L., and Shuangxi, L. (2010). Design of simulation software for mission change real-time control based on pervasive computing. In *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, pages 432–435.

[Kyriakopoulos 2018] Kyriakopoulos, K. (2018). Distributed complex event processing (cep) system based on the esper engine. *Master's thesis. School of Electrical and Computer Engineering, Technical University of Crete*.

[Lee et al. 2019] Lee, U., Han, K., Cho, H., Chung, K.-M., Hong, H., Lee, S.-J., Noh, Y., Park, S., and Carroll, J. M. (2019). Intelligent positive computing with mobile, wearable, and iot devices: Literature review and research directions. *Ad Hoc Networks*, 83:8–24.

[Luckham 2008] Luckham, D. (2008). The power of events: An introduction to complex event processing in distributed enterprise systems. In Bassiliades, N., Governatori, G., and Paschke, A., editors, *Rule Representation, Interchange and Reasoning on the Web*, pages 3–3, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Markowetz et al. 2014] Markowetz, A., Błaszkiewicz, K., Montag, C., Switala, C., and Schlaepfer, T. E. (2014). Psycho-informatics: Big data shaping modern psychometrics. *Medical Hypotheses*, 82(4):405 – 411.

[Moura et al. 2020] Moura, I., Teles, A., Silva, F., Viana, D., Coutinho, L., Barros, F., and Endler, M. (2020). Mental health ubiquitous monitoring supported by social situation awareness: A systematic review. *Journal of Biomedical Informatics*, 107:103454.

[Olguin et al. 2009] Olguin, D., Waber, B., Kim, T., Mohan, A., Ara, K., and Pentland, A. (2009). Sensible organizations: Technology and methodology for automatically measuring organizational behavior. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 39:43–55.

[Onnela et al. 2014] Onnela, J.-P., Waber, B. N., Pentland, A., Schnorf, S., and Lazer, D. (2014). Using sociometers to quantify social interaction patterns. *Scientific reports*, 4:5604.

[Rodrigues et al. 2020] Rodrigues, I., Silva, F., Coutinho, L., and Teles, A. S. (2020). Mental health ubiquitous monitoring: Detecting context-enriched sociability patterns through complex event processing. In *IEEE 33rd International Symposium on Computer Based Medical Systems (CBMS)*.

[Roriz 2017] Roriz, M. (2017). *DG2CEP : An On-line Algorithm for Real-time Detection of Spatial Clusters from Large Data Streams through Complex Event Processing Marcos*. PhD thesis, Potífica Universidade Católica do Rio de Janeiro, https://www.maxwell.vrac.puc-rio.br/30249/30249.PDF.

[Schacter 1999] Schacter, D. L. (1999). The seven sins of memory: Insights from psychology and cognitive neuroscience. *American Psychologist*, 54(3):182–203.

[Sear et al. 2020] Sear, R. F., Velásquez, N., Leahy, R., Restrepo, N. J., Oud, S. E., Gabriel, N., Lupu, Y., and Johnson, N. F. (2020). Quantifying covid-19 content in the online health opinion war using machine learning. *IEEE Access*, 8:91886–91893.

[Servia-Rodríguez et al. 2017] Servia-Rodríguez, S., Rachuri, K. K., Mascolo, C., Rentfrow, P. J., Lathia, N., and Sandstrom, G. M. (2017). Mobile sensing at the service of mental well-being: A large-scale longitudinal study. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 103–112, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Shiffman et al. 2008] Shiffman, S., Stone, A. A., and Hufford, M. R. (2008). Ecological momentary assessment. *Annual Review of Clinical Psychology*, 4(1):1–32.

[Torales et al. 2020] Torales, J., O'Higgins, M., Castaldelli-Maia, J. M., and Ventriglio, A. (2020). The outbreak of covid-19 coronavirus and its impact on global mental health. *International Journal of Social Psychiatry*.

[Torous et al. 2016] Torous, J., Kiang, M. V., Lorme, J., and Onnela, J.-P. (2016). New tools for new research in psychiatry: A scalable and customizable platform to empower data driven smartphone research. *JMIR Mental Health*, 3(2).

[Trull and Ebner-Priemer 2013] Trull, T. J. and Ebner-Priemer, U. (2013). Ambulatory assessment. *Annual Review of Clinical Psychology*, 9(1):151–176.

[Umberson and Montez 2010] Umberson, D. and Montez, J. K. (2010). Social relationships and health: A flashpoint for health policy. *Journal of Health and Social Behavior*, 51:S54–S66.

[Van de Mortel et al. 2008] Van de Mortel, T. F. et al. (2008). Faking it: social desirability response bias in self-report research. *Australian Journal of Advanced Nursing, The*, 25(4):40.

[Wang 2014] Wang, J. (2014). *Encyclopedia of business analytics and optimization*. IGI Global.

[Wang et al. 2017] Wang, R., Chen, F., Chen, Z., Li, T., Harari, G., Tignor, S., Zhou, X., Ben-Zeev, D., and Campbell, A. T. (2017). *StudentLife: Using Smartphones to Assess Mental Health and Academic Performance of College Students*, pages 7–33. Springer International Publishing, Cham.

[Wongkoblap et al. 2017] Wongkoblap, A., Vadillo, M. A., and Curcin, V. (2017). Researching mental health disorders in the era of social media: Systematic review. *Journal of Medical Internet Research*, 19(6):e228.