ERCEMAPI2020.ENUCOMPI.COM.BR | 10 A 12 DE SETEMBRO DE 2020

ERCEMAPI

Escola Regional de Computação
do Ceará, Maranhão e Piauí

LIVRO DE MINICURSOS ERCEMAPI 2020

REALIZAÇÃO

SBC
Sociedade Brasileira de Computação

ORGANIZAÇÃO

PATROCÍNIO

Loggi

G Bank
Feito para o hoje

Google

CARREIRA RH

VIII Escola Regional de Computação Ceará, Maranhão e Piauí

10 a 12 de setembro de 2020

LIVRO DE MINICURSOS

ERCEMAPI 2020

Organização do Livro

ARIEL SOARES TELES (IFMA)

DARIO BRITO CALÇADA (UESPI)

NECIO DE LIMA VERAS (IFCE)

Coordenação Geral

RODRIGO AUGUSTO ROCHA SOUZA BALUZ (UESPI)

CLODOALDO BRASILINO LEITE NETO (IFPI)

Sociedade Brasileira da Computação

Porto Alegre

2020

Capa: Agência WOW
Editoração: Secretaria Regional SBC Piauí

Ficha Catalográfica elaborada pela Bibliotecária
Christiane Maria Montenegro Sá Lins CRB/3 – 952

E74

Escola Regional de Computação Ceará, Maranhão, Piauí (8.: 2020: Parnaíba, PI)

Livro de minicursos / 8ª Escola Regional de Computação Ceará, Maranhão, Piauí ERCEMAPI. Parnaíba, PI, 10 a 12 de setembro de 2020; organização Ariel Soares Teles, Dario Brito Calçada, Nécio Lima Veras; coordenação geral Rodrigo Augusto Rocha Souza Baluz, Clodoaldo Brasilino Leite Neto. - Porto Alegre: Sociedade Brasileira da Computação, 2020.

207 p. il. col.

Disponível em: https://sol.sbc.org.br/

ISBN.: 978-65-87003-11-5

1. Ciência de Dados. 2. Inteligência Artificial. 3. Desenvolvimento Web. 4. Métricas de Software. I. Teles, Ariel Soares. II. Calçada, Dario Brito. III. Veras, Nécio Lima. IV. Baluz, Rodrigo Augusto Rocha Souza. V. Leite Neto, Clodoaldo Brasilino. VI. Título.

CDD 004

Sociedade Brasileira de Computação – SBC

Contato

Av. Bento Gonçalves, 9500

Setor 4 - Prédio 43.412 - Sala 219

Bairro Agronomia

91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78

http://www.sbrc.org.br

Mensagem da Coordenação Geral

É uma grande satisfação poder reunir as comunidades dos estados do Ceará, Maranhão e Piauí durante a ERCEMAPI 2020, entre 10 e 12 de setembro de 2020, com o objetivo de disseminar o conhecimento técnico e científico sobre temas e assuntos de vanguarda na área de Computação. Originalmente planejada para acontecer na cidade de Parnaíba, litoral do estado do Piauí, foi necessário mudanças para o formato virtual em virtude do atual cenário em que nos encontramos. A VIII Escola Regional de Computação Ceará, Maranhão e Piauí, promovida pela Sociedade Brasileira de Computação (SBC), por meio da secretária regional no Piauí, teve como evento satélite o XIII Encontro Unificado de Computação do Piauí (ENUCOMPI 2020). O evento que tem como tema central “Computação: ciência, inovação e empreendedorismo”, dispõe em sua programação de palestras, workshops, minicursos, sessões técnicas de artigos científicos e muito networking e conhecimento sendo gerado. Gostaríamos de agradecer o apoio institucional da Universidade Estadual do Piauí (UESPI), da Universidade Federal do Piauí (UFPI), do Instituto Federal do Piauí (IFPI) e do Centro Universitário Maurício de Nassau (UNINASSAU). Nosso agradecimento a todos os membros do comitê de programa e, em especial, aos coordenadores do comitê de programa científico, Rodrigo Veras (UFPI), Robson Siqueira (IFCE) e Francisco Glaubos (UFMA); e, aos coordenadores da trilha de minicursos, Ariel Soares (IFMA), Dario Calçada (UESPI) e Nécio Veras (IFCE). Nosso muito obrigado ao apoio financeiro prestado pelas grandes empresas Loggi, Google, GBank e CarreiraRH. Agradecer o apoio da Sociedade Brasileira de Computação (SBC) e das secretarias regionais do Ceará (Danielo Gomes) e do Maranhão (Davi Viana). Sem o apoio de todos os colaboradores, seria inviável a realização deste evento. Em nome da comissão organizadora da ERCEMAPI 2020, desejamos a todos os participantes dias enriquecedores em conhecimento e troca de experiências, e de agradável convívio virtual.

Rodrigo Augusto Rocha Souza Baluz (UESPI)

Clodoaldo Brasilino Leite Neto (IFPI)

Coordenação Geral da ERCEMAPI 2020

Mensagem da Coordenação de Minicursos

O Livro de Minicursos da ERCEMAPI 2020 colabora com o objetivo da Escola Regional de Computação Ceará, Maranhão e Piauí em disseminar o conhecimento técnico e científico sobre temas e assuntos de vanguarda na área de Computação. Em sua 8ª edição, os minicursos abordam conteúdos relacionados à ciência de dados, inteligência artificial, desenvolvimento web, métricas de software e revisão sistemática da literatura, como forma de atualizar os conhecimentos da comunidade acadêmica e profissional, de uma forma didática e de amplo acesso ao público. Os nove minicursos foram selecionados através de um processo de revisão por pares do tipo "*blind*", de um total de 26 propostas de minicursos submetidas, o que implicou uma taxa de aceitação de 34,5%.

No **capítulo 1**, "Descoberta Automática de Conhecimento por meio de Redes de Regras de Associação Filtradas", os autores apresentam os conceitos fundamentais sobre as redes de regras de associação filtradas, as métricas utilizadas no processamento de análise destas regras, além de discutirem a geração automática de hipóteses pela descoberta de padrões de interesse.

O **capítulo 2**, "Covid-19: Aquisição, tratamento e visualizações interativas de dados do Ministério da Saúde", apresenta as ferramentas que estão posicionadas no estado da arte, das áreas de visualização e ciência de dados, como as bibliotecas *Pandas*, *Selenium*, *Leaflet* e *Crossfilter*, para a construção de uma visualização interativa dos dados oficiais da Covid-19 no Brasil, obtidos no painel oficial do Ministério da Saúde.

O **terceiro capítulo**, "Utilização de Técnicas de *Data Augmentation* em Imagens: Teoria e Prática", aborda de forma teórica e prática, as principais técnicas de *Data Augmentation* em imagens, desde as mais tradicionais até as que envolvem *Deep Learning*, com o intuito de gerar novos dados virtuais, a partir de exemplos reais, para prover mais volume e dar mais suporte ao desenvolvimento de aplicações na área de inteligência artificial.

No **capítulo 4**, "Detectando Padrões de Sociabilidade de Seres Humanos Através do Processamento de Eventos Complexos", introduzir técnicas de processamento de eventos complexos e sua aplicação na tarefa de reconhecimento de padrões de sociabilidade de seres humanos para detectar comportamentos sociais por meio de dados de sensores de tecnologias onipresentes.

O **capítulo quinto**, "Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap", os autores apresentam a engenharia de software orientada a reúso e os *frameworks Django* e *Bootstrap*, que proporcionam reuso com a intenção de reduzir o tempo de desenvolvimento e a complexidade no desenvolvimento web.

No **sexto capítulo**, "Soluções de Aprendizado de Máquina para Estimar em Tempo Real a Pose Humana em Aplicações de Saúde", os autores apresentam a biblioteca *TensorFlow* e suas extensões *PoseNet* e *MediaPipe*, úteis no desenvolvimento de aplicações para o reconhecimento

em tempo real de estruturas e movimento do corpo humano, as quais contribuem para as pesquisas na área de informática em saúde.

O **capítulo 7**, "Aprendizagem Profunda em Unity com ML-Agents", direciona para a prática da criação de ambientes com agentes virtuais, utilizando a *Unity*, e no uso de algoritmos de aprendizagem profunda, utilizando o pacote *ML-Agents, o qual* se comunica com o *TensorFlow*.

O **capítulo oitavo**, "Como Estimar um Software? Métricas para a Aferição de Esforço, Prazo e Custo de um Produto de Software", apresenta os conceitos básicos sobre as métricas de software e um processo de precificação baseado no modelo CEK, utilizando *Canvas*, EAP e *Kanban*, para a estimativa adequada, com base no esforço necessário, do prazo e custo para o desenvolvimento de sistemas.

Por fim, o **capítulo 9**, "Levantamento bibliográfico utilizando a ferramenta The End", apresenta aos acadêmicos e pesquisadores um método de realizar um levantamento bibliográfico por meio de Mapeamento Sistemático da Literatura utilizando os recursos da ferramenta *The End*.

Os nove capítulos deste livro possuem metodologias e ferramentas para a área de Tecnologia da Informação e Comunicação, sendo uma excelente oportunidade para familiarização dos interessados com novos temas de pesquisa que podem vir a ser úteis em suas vidas profissionais.

Agradecemos a todos os professores que compuseram o Comitê de Programa de Minicursos, por suas valiosas contribuições para os trabalhos e dedicação no processo de revisão, em especial aos coordenadores gerais da ERCEMAPI 2020, Prof. Rodrigo Baluz (UESPI) e Prof. Clodoaldo Brasilino (IFPI), por todo o suporte durante a seleção e elaboração deste livro.

Finalmente, desejamos que todos os participantes dos minicursos da ERCEMAPI 2020 aproveitem as apresentações, que este ano serão realizadas remotamente.

Ariel Soares Teles (IFMA)

Dario Brito Calçada (UESPI)

Nécio de Lima Veras (IFCE)

Coordenadores de Minicursos da ERCEMAPI 2020

Comitê de Programa de Minicursos

Antônio Oséas de Carvalho Filho (UFPI)

Ariel Soares Teles (IFMA)

Dario Brito Calçada (UESPI)

Davi Viana dos Santos (UFMA)

Iális Cavalcante de Paula Júnior (UFC)

Necio de Lima Veras (IFCE)

Omar Andrés Carmona Cortés (IFMA)

Pedro de Alcântara dos Santos Neto (UFPI)

Pedro Porfírio Muniz Farias (UNIFOR)

Ricardo de Andrade Lira Rabêlo (UFPI)

Secretarias Regionais SBC

Danielo Gonçalves Gomes (UFC / SR Ceará)

Davi Viana (UFMA / SR Maranhão)

Rodrigo Augusto Rocha Souza Baluz (UESPI / SR Piauí)

Sumário

Capítulo 1

Descoberta Automática de Conhecimento por meio de Redes de Regras de Associação Filtradas

Matheus William Gomes dos Santos, Andreiver Mateus Ferreira Silva e Dario Brito Calçada

Abstract

Association Rules Mining is an excellent technique for finding patterns between elements within a data set. There is a considerable variation in applications of this knowledge extraction technique in several areas, ranging from direct applications in the scientific field or even in the financial market. However, the process for viewing patterns identified by using Association Rules mining should be improved, as the volume of data is very high. Within this context, the filtered Association Rules Networks (Filtered-ARNs) appear, which take into account mathematical factors to prove the influence between the elements of a rule, to facilitate the identification of related patterns of interest to a particular study.

Resumo

A Mineração de Regras de Associação é uma excelente técnica para encontrar padrões entre elementos dentro de um determinado conjunto de dados. Existe uma variação considerável de aplicações desta técnica de extração de conhecimento em várias áreas, estendendo-se desde aplicações diretas no campo científico ou até no mercado financeiro. No entanto, o processo para visualização dis padrões identificados pela uso da mineração de Regras de Associação deve ser melhorado, já que o volume de dados é muito elevado. Dentro desse contexto, aparecem as Redes de Regras de Associação Filtradas (Filtered-ARNs), que levam em consideração fatores matemáticos para comprovação da influência entre os elementos de uma regra, a fim de viabilizar a identificação de padrões de interesse relacionados a um determinado estudo.

1.1. Considerações Iniciais

A mineração de dados por meio da descoberta de Regras de Associação é uma tarefa que visa a identificação de padrões em bases de dados e, posteriormente, alcançar conhecimento acerca do tema e do problema em questão [Le and Vo 2016]. Além disso, esse tipo

de mineração pode ser utilizado para descobrir hipóteses em determinado domínio de conhecimento, possibilitando avanços em processos de pesquisa em conjunto com métodos estatísticos [Vinaya and Shah 2016].

A Mineração de Regras de Associação inicia-se a partir de observações dos eventos para formar uma estrutura na qual o processo que está gerando os eventos possa ser evidenciado. Uma Regra de Associação [Agrawal and Shafer 1996, Agrawal and Srikant. 1994, Agrawal et al. 1994] possui uma forma padrão A => B, na qual A e B podem ser atributos, itens ou "objetos de dados". A Mineração de Regras de Associação corresponde a uma considerável gama de áreas de pesquisa. Tornando-se necessário limitar o escopo de cada trabalho que tiver a sua utilização.

Neste capítulo são apresentados os conceitos fundamentais relacionados a Redes de Regras de Associação Filtradas. Na seção 2, são descritos os conceitos e a definição de Regras de Associação e sobre o seu processo de mineração. Na seção 3, são mostrados a definição e os conceitos a cerca de Redes de Regras de Associação bem como conceitos básicos sobre Redes. Na seção 4 são introduzidas as Redes de Regras de Associação Filtradas bem como as métricas utilizadas no processamento de análise dessas regras. E, por fim , na seção 5 é discutida a geração automática de hipóteses pela descoberta de padrões de interesse em *datasets*.

1.2. Mineração de Regras de Associação

Nos últimos anos, cada vez mais empresas, organizações e usuários manipulam e armazenam quantidades cada vez maiores de dados. Este fato, tornou a compreensão dos dados uma atividade de relevância considerável no suporte à tomada de decisões [Zaki and Meira 2013].

Como também descrito na seção anterior, a mineração de dados tem como definição a descoberta de padrões em bases de dados [Aggarwal 2015]. Além disso, a mineração aparece como uma alternativa na solução à dificuldade em resumir e encontrar padrões não óbvios em quantidades de dados cada vez maiores. Em relação à Mineração de Regras de Associação, pode-se dividi-la em 3 etapas:

- **Pré-processamento:** preparação da base e domínio dos dados para a etapa de extração de padrões, podendo ocorrer a remoção de itens não interessantes.
- **Extração de padrões:** são efetuados os cálculos de medidas, construção dos *itemsets* frequentes e formulação das Regras de Associação.
- **Pós-processamento:** nessa etapa ocorre a remoção de regras não interessantes, causando uma redução do número de regras a serem exploradas pelo usuário.

Em decorrência de quantidades cada vez maiores de dados, o número de Regras de Associação descobertas pode tornar-se mais um problema para a sua interpretação, gerando assim, uma nova necessidade de mineração. Portanto, é necessária a utilização de melhores maneiras de interpretar o conhecimento trazido pelas Regras de Associação, como o uso de Redes [Pandey et al. 2009].

A fim de elucidar o conceito e objetivo das Regras de Associação, pode-se citar duas definições:

- **Definição 1:** seja $I = \{i_1; i_2; ...; i_n\}$ um conjunto de objetos denominados itens que podem assumir valores binários 0 ou 1 , que representam a presença ou não de um objeto em particular. Seja T um conjunto de transações, em que cada transação D corresponde a um conjunto de itens tal que $D \subseteq I$. Considera-se ainda que um conjunto de itens A está contido numa transação D, se todos os itens do conjunto tiverem valor "verdadeiro" na transação, ou seja, fizerem parte dessa mesma transação. Uma Regra de Associação R pode ser representada por uma expressão no formato: $A \Rightarrow B$, com $A \subseteq I; B \subseteq I$ e $A \cap B = \emptyset$. É ainda possível tratar as variáveis quantitativas ou qualitativas, criando intervalos de valores e utilizando-as, posteriormente, como variáveis binárias. A é denominado de antecedente (LHS – Left Hand Side) da regra e B o consequente (RHS - Right Hand Side).

- **Definição 2:** para cada regra $(LHS \Rightarrow RHS)$, extraída de um conjunto de transações T, é calculado um valor de suporte (sup), apresentado na Equação 1, que verifica a força de associação entre LHS e RHS (probabilidade de ocorrência da transação $LHS \cup RHS$); e um valor de confiança (conf), Equação 2, que mede a força da implicação lógica da regra (probabilidade condicional de RHS dado LHS) [AGRAWAL, IMIELINSKI and SWAMI 1994].

$$sup(LHS \Rightarrow RHS) = P(LHS \cup RHS) \tag{1}$$

$$conf(LHS \Rightarrow RHS) = P(RHS|LHS) \tag{2}$$

O suporte de uma regra tem como definição a probabilidade de que uma transação qualquer satisfaça LHS e RHS simultaneamente, enquanto que a confiança é a chance de uma transação satisfazer RHS, dado que ela satisfaz LHS.

1.2.1. Extração das Regras de Associação

O problema de se descobrir todas as Regras de Associação de um problema divide-se em duas partes [de Vasconcelos and de Carvalho 2004]:

- Encontrar todos os conjuntos de itens que possuam um suporte de transações acima de um limite mínimo informado.

- Selecionar, dentre as regras geradas, apenas as que possuam o grau de confiança mínimo, correspondente à confiança mínima estabelecida.

Dado um conjunto de transações, o problema de mineração de Regras de Associação está em gerar todas as regras que contenham o suporte e confiança iguais ou maiores do que os valores mínimos determinados pelo usuário, referenciados, respectivamente, como suporte mínimo (minsup) e confiança mínima (minconf). Por exemplo, considerando a base de dados "Compra" apresentada na Tabela 1, a qual possui dados referentes

a compras diárias de um indivíduo dentre um período de 10 (dez) dias. Neste caso, define-se que minsup é 0,3 (30%) e que minconf é de 0,8 (80%).

Considerando-se que LHS = CAFÉ e RHS = PÃO, pode-se calcular o suporte e a confiança para a regra (CAFÉ $\Rightarrow$ PÃO) tendo como resultados sup(CAFÉ $\cup$ PÃO) = 0,3 ou 30% e conf(CAFÉ $\Rightarrow$ PÃO) = 1 ou 100%. Este resultado implica duas afirmações:

- "em 30% das compras (em 10 dias) desse indivíduo ele comprou café e pão"
- "sempre que esse indivíduo comprou café, ele comprou pão"

Essas afirmações podem servir como base para a elaboração de hipóteses que podem conduzir estudos futuros sobre o comportamento padrão de compras do sujeito deste caso.

1.2.1.1. Algoritmo Apriori

O *Apriori* é considerado o algoritmo mais conhecido para extração de Regras de Associação [Agrawal and Srikant. 1994]. Por ele, é possível obter todos os conjuntos de *itemsets* frequentes (L_k). Em seu funcionamento, o algoritmo gera conjuntos de itens candidatos (padrões) por meio da busca em profundidade. Posteriormente, os padrões não frequentes são eliminados. Todos os conjuntos de *itemsets* frequentes são obtidos a partir dos conjuntos de itens candidatos.

O funcionamento do algoritmo principal (Figura 1.1) pode ser explicado pelo uso de duas funções: *Apriori_gen*, para gerar os itens candidatos e eliminar os não frequentes, e a *Gen_rules* para extração das Regras de Associação.

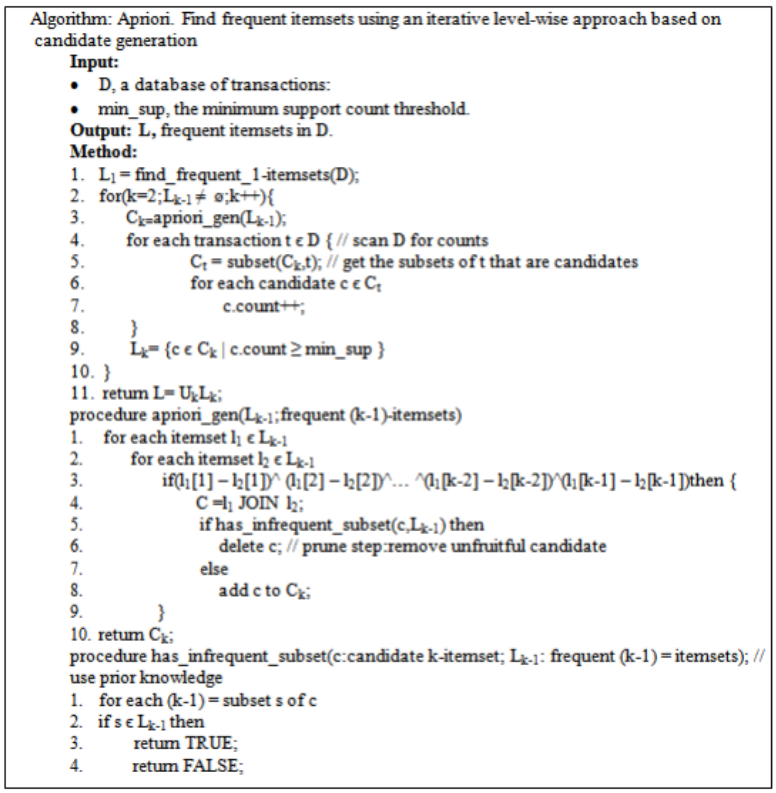
A descoberta de Regras de Associação pelo *Apriori* pode ser dividida em:

- Descoberta de todos os conjuntos de *itemsets* frequentes e;
- Geração de Regras de Associação (a partir dos conjuntos de *itemsets* frequentes descobertos).

O primeiro passo, dentre os dois citados anteriormente, é considerado o de maior custo computacional, causando maior atenção nos estudos e pesquisas em mineração de dados. Tal custo motivou o surgimento de diferentes algoritmos com objetivo de maximizar a eficiência computacional do *Apriori*.

As maiores desvantagens do *Apriori* decorrem do fato de não possuir suporte para uma quantidade elevada de dados brutos. Essas desvantagens são significativas devido ao fato de a quantidade de dados produzidas nos últimos anos ter aumentado consideravelmente. Assim, é necessário o uso de técnicas eficientes que analisem e gerenciem quantidades massivas de dados. Mesmo com essa necessidade, os algoritmos da família do Apriori (gerados a partir do original) são o mais utilizados para Mineração de Regras de Associação.

Além do *Apriori*, existem outros algoritmos com finalidade de geração de Regras de Associação. Esses algoritmos diferem-se do *Apriori* pela atenção na correção da



Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation

Input:

- D, a database of transactions:
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
1.  L1 = find_frequent_1-itemsets(D);
2.  for(k=2;Lk-1 ≠ ø;k++){
3.      Ck=apriori_gen(Lk-1);
4.      for each transaction t ϵ D { // scan D for counts
5.              Ct = subset(Ck,t); // get the subsets of t that are candidates
6.              for each candidate c ϵ Ct
7.                  c.count++;
8.      }
9.      Lk= {c ϵ Ck | c.count ≥ min_sup }
10. }
11. return L= UkLk;
procedure apriori_gen(Lk-1;frequent (k-1)-itemsets)
1.   for each itemset l1 ϵ Lk-1
2.       for each itemset l2 ϵ Lk-1
3.           if(l1[1] – l2[1])^ (l1[2] – l2[2])^... ^(l1[k-2] – l2[k-2])^(l1[k-1] – l2[k-1])then {
4.               C =l1 JOIN l2;
5.               if has_infrequent_subset(c,Lk-1) then
6.                   delete c; // prune step:remove unfruitful candidate
7.               else
8.                   add c to Ck;
9.           }
10. return Ck;
procedure has_infrequent_subset(c:candidate k-itemset; Lk-1: frequent (k-1) = itemsets); //
use prior knowledge
1.  for each (k-1) = subset s of c
2.  if s ϵ Lk-1 then
3.       return TRUE;
4.       return FALSE;
```

Figura 1.1. Algoritmo Apriori [Han and Kamber 2006]

necessidade de percorrer os conjuntos de elementos várias vezes, consequentemente, possuem melhor desempenho computacional com o mesmo resultado. Dentre esses outros algoritmos, pode-se citar, *FP-Growth* e o *Apriori-TID*.

1.2.2. Critérios para Seleção e Classificação das Regras de Associação

Após a descoberta das Regras de Associação de determinado problema, é necessário medir o nível de interesse dos padrões descobertos [Usman and Usman 2016]. No total, nove critérios específicos são usados mensurar se determinado padrão é ou não interessante, sendo eles concisão, cobertura, confiabilidade, peculiaridade, diversidade, novidade, surpreendimento, utilidade e capacidade de ação.

Além disso, esses critérios podem ser divididos em três classificações: medidas objetivas, medidas subjetivas e medidas baseadas em semântica. Que são definidas da seguinte forma:

- **Medidas objetivas:** baseiam-se apenas nos dados originais. Nenhum conhecimento sobre o usuário ou aplicativo é necessário. Em geral, baseiam-se em probabilidade ou teoria da informação. Algumas dessas medidas consideram que uma Regra de Associação é interessante apenas quando o valor de suporte dessa regra é maior que o valor de suporte esperado. Outra função das medidas objetivas é demonstrar como os itens influenciam uns aos outros, podendo esta influência ser de modo direto, quando os itens variam de modo proporcional, ou inverso, quando o

aumento da incidência de um item leva à diminuição de outro.

- **Medidas de interesse subjetivas:** Consideram principalmente a opinião de um analista para determinar a força da regra [Gonçalves 2005].
- **Medidas baseadas em semântica:** Utilizam as estruturas semânticas das regras para estabelecimento de sua importância [Han et al. 2012].

Considerando esses critérios, pode-se utilizar três métodos para determinar o interesse de um padrão. Primeiro, pode-se classificar cada padrão em interessante ou não. Em segundo lugar, pode-se determinar uma relação de preferência para representar padrões mais interessantes que outros. E por fim, pode-se também elencar os tipos de padrões. Assim, o uso de medidas de interesse facilita a identificação automática de padrões interessantes em Regras de Associação.

Além das medidas supracitadas de suporte e confiança, outras medidas para as regras podem ser calculadas. A seguir segue a definição de duas medidas assimétricas, *Added Value* e *Gain*, que são utilizadas nas Redes de Regras de Associação Filtradas, que serão explicitadas nas Seções seguintes.

- ***Added Value* [-1..0..1]:** a medida *Added Value* (AV) (Equação 3) indica o quanto a frequência do consequente aumenta na presença do antecedente, ou seja, mede o ganho de *RHS* na presença de *LHS* [Sahar 2003]. Se AV for positivo, então a frequência de RHS aumenta na presença de LHS. Sendo AV negativo, a frequência de RHS diminui na presença de LHS. Se AV possuir valor nulo (zero) , tem-se uma coincidência aleatória, ou seja, a frequência de LHS não altera em nada a frequência de RHS.

$$AV = Conf(LHS \Rightarrow RHS) - sup(RHS) \tag{3}$$

- ***Gain* [0..1]:** é uma medida proposta por Fukuda et al. (1996) que dá um trade-off entre suporte e confiança (Equação 4), auxiliando na seleção das regras de acordo com a frequências da mesma em relação à confiança mínima.

$$Gain = sup(LHS \cap RHS) - minconf.sup(LHS) \tag{4}$$

Considerando-se a Equação 2, pode-se inferir que:

$$sup(LHS \cap RHS) = sup(LHS).conf(LHS \Rightarrow RHS) \tag{5}$$

portanto, fazendo a substituição na Equação 4, e colocando-se em evidência o fator sup(LHS) obtém-se a Equação 6.

$$Gain = [conf(LHS \Rightarrow RHS) - minconf].sup(LHS) \tag{6}$$

Por meio da Equação 6, percebe-se que a medida objetiva *Gain* funciona como uma normalização da medida de Confiança. Quando o valor de *Gain* = 0 a confiança da

regra é igual a confiança mínima ($conf(LHS \Rightarrow RHS) = minconf$), e a partir daí todos os outros valores são calculados, sendo que *Gain* = 1 evidencia um valor de dependência estatística total, pois seria possível apenas quando o valor de minconf é insignificante (minconf = 0, i.e. sem limitação de confiança), a $conf(LHS \Rightarrow RHS)$ = 1 e o sup(LHS) = 1, ou seja, em todos os momentos que ocorresse LHS, também ocorreria RHS, com LHS em todas as instâncias da base de dados analisada.

A vantagem desta medida sobre a confiança é que pode-se calcular com mais exatidão a influência do elemento antecedente sobre o consequente, provocando assim uma maior credibilidade à medida, podendo também, ser utilizada para selecionar regras.

1.2.3. Pós-processamento das Regras de Associação

Na etapa de Pós-Processamento, o conhecimento, obtido por meio das Regras de Associação, pode ser simplificado, avaliado, visualizado ou simplesmente documentado [Piri et al. 2018]. A etapa de extração de Regras pode ser considerada simples, mas a compreensão de um conjunto significativo de regras torna a etapa de pós-processamento um desafio. Pode-se dividir o Pós-Processamento em três etapas [Namaki et al. 2017, Simard et al. 2016, Hendrickx et al. 2015, Zhao et al. 2009], sendo elas:

- **Avaliação:** etapa na qual é realizada a avaliação do conhecimento extraído do conjunto de dados por meio de critérios, tais como confiabilidade, aplicabilidade.
- **Interpretação e explanação:** etapa na qual é realizada a documentação, visualização, modificação e/ou comparação (ao conhecimento pré-existente) do conhecimento extraído do conjunto de dados de forma a torná-lo compreensível.
- **Filtragem:** etapa na qual é realizada a filtragem do conhecimento que foi extraído do conjunto de dados, podendo ser realizada por vários mecanismos que variam de acordo com a técnica utilizada.

Na busca por facilitar a interpretação de um número elevado de Regras de Associação, foram realizados diversos estudos, nos quais foram propostas diferentes abordagens tais como: Avaliação por consulta, Técnicas de visualização, etc.

- **Avaliação por consulta:** permite que o usuário explore o conjunto de regras por meio do uso de uma linguagem de consulta (tipicamente inspirada no SQL - Structured Query Language) [Meo et al. 1996].
- **Poda de regras:** Tem o objetivo de eliminar (excluir) regras redundantes ou que não são interessantes para o usuário [Toivonen 1996].
- **Técnicas de visualização:** utilizam de recursos gráficos, como Redes, para uma estruturação visual do conhecimento, possibilitando o uso de técnicas gráficas, bem como a observação direta das relações obtidas no processo de mineração de dados [Klemettinen et al. 1994] e, consequentemente, auxiliando na interpretação de Regras de Associação.

1.3. Redes de Regras de Associação

Para compreender o funcionamento e a estrutura das Redes de Regras de Associação, é importante ter o conhecimento de alguns conceitos básicos de Redes. Portanto, é imprescindível o esboço de alguns elementos associados às Redes para se poder introduzir as Redes de Regras de Associação

1.3.1. Redes

Uma rede é uma representação simplificada de uma estrutura de padrões e conexões, ou elementos e suas relações, que podem ser demonstradas por um Grafo. O termo "Rede" é usado para referenciar todo sistema que possa ser retratado por um Grafo, que é um termo usado para caracterizar um conjunto de ferramentas conceituais específicas com o objetivo de descrever a Rede.

Uma Rede, em seu modelo mais simples, tem como componentes um conjunto de pontos unidos em pares por linhas. Os pontos também podem ser referidos como "nós" ou "vértices", e as linhas chamadas de "arestas" (Figura 1.2).

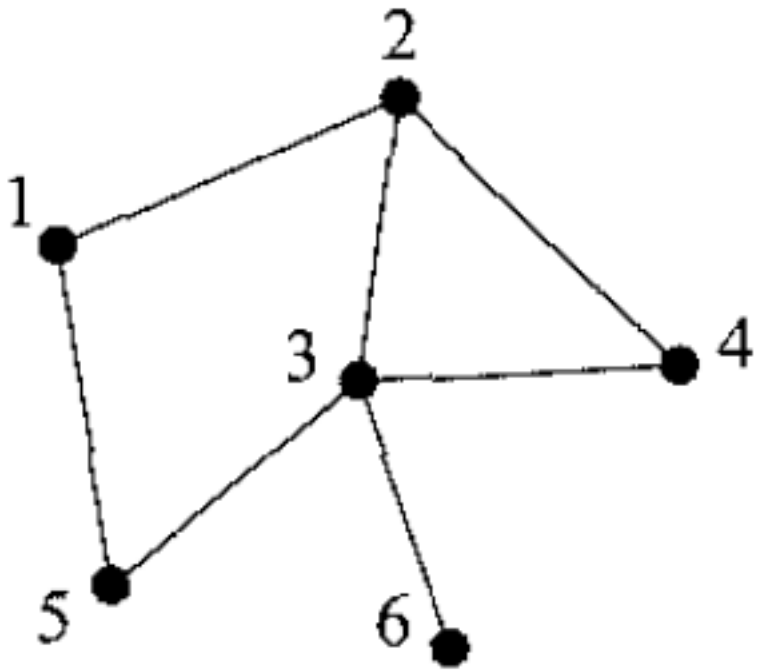


Figura 1.2. Representação simplificada de uma Rede

Normalmente, uma Rede R é representada com $R = (V, E)$, em que V é o conjunto de vértices e E é um conjunto de arestas, que podem estar ligados a alguns pares de vértices em V. Estatisticamente, um Grafo pode ser caracterizado por valores derivados, tais como o grau médio dos nós e o comprimento médio (caminho) entre os nós. Características adicionais como: diâmetro da rede, número de triângulos, número de isomorfismos e o coeficiente de agrupamento também podem ser analisados [Nettleton 2013].

Em uma Rede R = (V, E), vários links e auto-conexões não são permitidas de-

pendendo do tipo de Rede que está sendo implementada. Se G é uma Rede Dirigida, considera-se o conjunto universal, denotado por U, contendo todas as $|V| * (|V| - 1)$ potenciais ligações dirigidas entre um par de nós em V, o qual |V| denota o número de elementos em V. Se R é uma Rede Sem Direção, o conjunto universal U contém $|V| * (|V| - 1)$ links. Deste modo, a representação da Rede está relacionada diretamente ao tipo de dado que ela representa.

O primeiro passo na análise da estrutura de uma Rede é muitas vezes a visualização da mesma, existem diversas ferramentas que podem gerar a imagem que representa uma Rede e suas características. Esta etapa é extremamente útil para análise de dados de Rede, pois permite que se possa identificar informações importantes que de outra forma seria difícil com dados brutos. O olho humano é extremamente talentoso em escolher padrões, e visualizações permitem ao ser humano colocar esta cognição para trabalhar em problemas de Rede [Newman 2010].

O estudo de um grande volume de Redes do mundo real, revelou que algumas dessas estruturas, estão presentes em muitos sistemas naturais e artificias, constituindo a base de classificação de sistemas reais. Uma Rede, também chamada de Grafo em literatura matemática, é uma coleção de vértices unidos por aros. Vértices e aros são também chamados de nós e arestas em ciência da computação, sites e títulos em física e atores e os laços em sociologia [Newman 2010].

As Redes, em sua maioria, possuem no máximo uma única aresta entre qualquer par de vértices. Em casos raros, em que está condição não é atendida, e um par de vértice está ligado por mais de uma aresta, essas são referenciadas como multiarestas. Outra ocasião não muito comum na maioria das Redes, são as auto-arestas ou *auto-loops*. Estas estruturas são chamadas dessa forma pois conectam um vértice a se mesmo. Uma Rede que não tem nem auto-arestas nem multiarestas é chamada de Rede Simples ou Grafo Simples. Uma Rede com multiarestas e/ou auto-arestas é chamada um Multigrafo Figura 1.3.

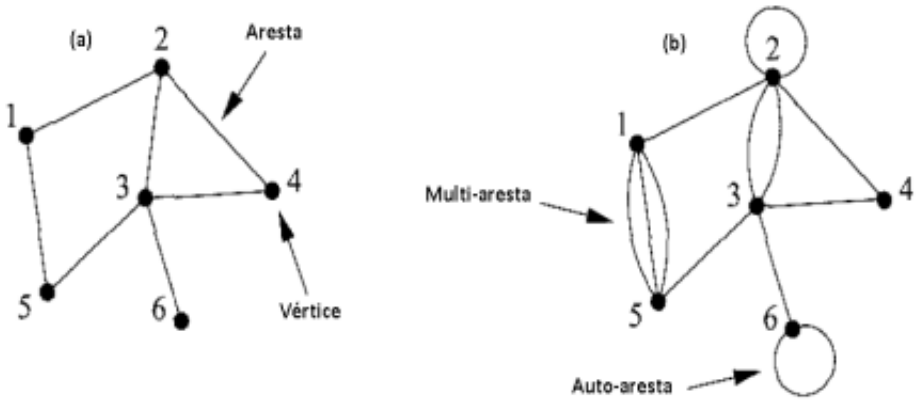


Figura 1.3. Duas pequenas Redes. (a) Rede Simples (b) Rede Multiarestas com 1 (uma) multiaresta e 2 (duas) auto-arestas

Existem diferentes maneiras de representar matematicamente uma Rede, uma delas, que se apresenta com bons resultados, é a Matriz de Adjacência que representa as

ligações existente entre os vértices de uma Rede.

As Matrizes de Adjacência das Redes indicam a quantidade de arestas entre cada um dos vértices que são identificados pela linha i e coluna j da matriz.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 & 0 & 3 & 0 \\ 1 & 2 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \end{pmatrix}$$

Figura 1.4. Matrizes de Adjacência

A matriz da Figura 1.4 por exemplo, representa os grafos da Figura 1.3.

Segundo Newman (2010), os valores em uma Matriz podem representar o peso, ou seja, o grau de aproximação entre os vértices de uma Rede formando uma Rede Ponderada. As matrizes também podem indicar algum tipo de direcionamento da Rede formando, nesse caso, uma Rede Direcionada que pode adotar, por exemplo, o valor "1"se for de i para j e o valor "0"se for de j para i.

Juntar dois vértices de uma só vez pode ser fazer necessário em alguns tipos de ligações. Uma Rede que represente relações familiares por exemplo, na qual uma família possui mais de um membro, deve usar uma hiper-aresta, que é um tipo generalizado de aresta que junta mais de dois vértices, a fim de demonstrar da melhor maneira uma conexão entre seus membros. Os Grafos em que as hiper-arestas aparecem são chamados de Hipergrafos Figura 1.5.

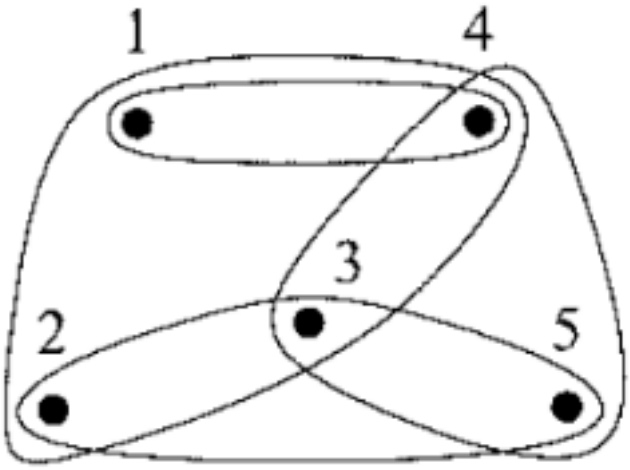


Figura 1.5. Hipergrafo - em hipergrafos as ligações são simbolizadas por loops que circulam os vértices

Uma associação de nós representado por um Hipergrafo, pode ser representado por uma Rede Bipartida.

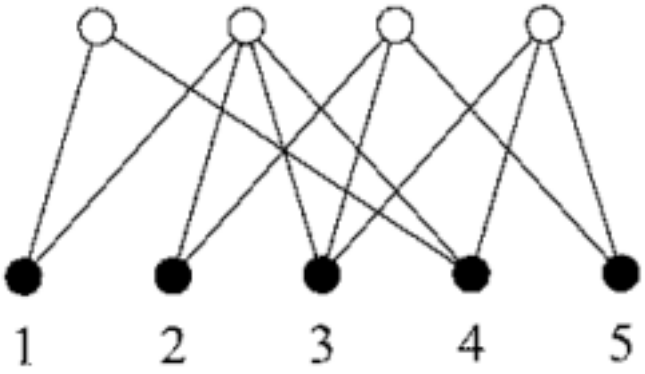


Figura 1.6. Rede Bipartida

Em uma Rede Bipartida (Figura 1.6), existe dois tipos de vértices: Um representando os vértices originais e o outro representando os grupos a que pertencem.

Uma outra forma de representar uma Rede é por meio de árvores. Esta estrutura é uma Rede Conectada, não direcionado e que não contém circuitos fechados. Os nós na Rede são acessíveis por todos os outros através de algum caminho. Uma Rede pode também consistir em duas ou mais partes desconexas uma da outra, e também é chamado de árvore um nó individual sem conexão. A Rede é chamada de floresta quando todos os seus componentes são árvores. A representação gráfica de uma Árvore consiste de um nó raiz no topo da estrutura, e ramificações abaixo, nas quais os vértices na parte inferior que só possuem uma ligação são chamados de folha Figura 1.7.

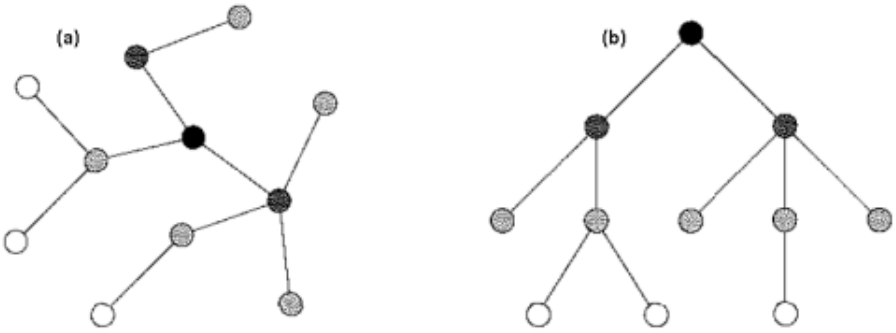


Figura 1.7. Dois esquemas de uma mesma árvore. Na árvore (a) os vértices estão posicionados conforme uma conveniência e na árvore (b) segue-se a estrutura com nó raiz

1.3.1.1. Medidas de Centralidade

Se a estrutura de uma Rede é conhecida, pode-se calcular, a partir de uma variedade de medidas quantitativas quais são aquelas características que conseguem se adaptar melhor a cada tipo de Rede. Algumas métricas são oportunas e selecionadas de acordo com o conhecimento que se deseja extrair [Newman 2010].

Estas medidas são importantes métricas que podem ser aplicadas em diversos tipos de situações. Redes Sociais, Redes de transportes e mercado financeiro, são alguns exemplos. Ela podem ser utilizadas para medir o quanto um vértice de uma Rede é mais ou menos importante do que os demais.

A Centralidade de grau ou informação, é uma medida que diz respeito ao número de ligações diretas que um certo vértice possui, ou seja, a centralidade de grau é uma contagem das adjacências de um vértice.

Outra medida de centralidade é a Centralidade de Auto-vetor, que define a importância de um nó verificando se este está ligado a outros vértices que se encontram em uma posição central na Rede. Se for esse for o caso, este nó tem uma ta Centralidade de Auto-vetor.

A Centralidade de Intermediação, também chamada de *Betweenness*, avalia a importância de um vértice considerando a quantidade de menores caminhos que passa por tal vértice.O vértice com maior centralidade de intermediação é aquele que participa de maneira mais ativa em um processo de interação, no qual os caminhos mais curtos são percorridos [Silva 2010].

O *Page Rank*, considera a importância de um *Website* levando em conta a quantidade de links que cada página possui. Na internet, os sites e páginas estão ligados por links, formando uma Rede de Informação.

Pode-se citar mais uma série de outras métricas importantes no estudo das Redes. Com os valores da centralidade de todos os vértices, ou de todas as arestas, define-se o Comprimento Médio do Caminho [Brandes 2001] e o Diâmetro da Rede [Coleman and Moré 1983].

1.3.2. Construindo uma Rede de Regras de Associação

Redes de Regras de Associação (ARN, do inglês Association Rules Network) consistem de uma estrutura para sumarizar, podar, e analisar um conjunto de Regras de Associação extraídas, para a concepção de hipóteses. Pode-se utilizar as Redes com o intuito de facilitar o processo e alcançar uma estrutura capaz de obter Regras automaticamente.

De uma perspectiva de descoberta de conhecimento, a ARN facilita o entendimento das relações e da utilidade das características dos dados. Matematicamente, as ARNs são Hipergrafos Direcionados.

A ideia central das Redes de Regras de Associação é que as Regras descobertas no processo de mineração passem por uma espécie de poda, afim de identificar e informações que atendam o objetivo específico da pesquisa. Se a pesquisa tiver uma variável de interesse, esta se torna o "alvo" ou "objetivo" e uma Rede formada por variáveis que estão relacionadas ao objetivo pode ser formada. Em seguida, uma estrutura pode ser elaborada e testada por meio de métodos estatísticos.

Normalmente a tarefa de Mineração de Dados envolve centena de variáveis que podem apresentar inconsistências. A ARN pode ser utilizada, para podar essas informações. Resumindo as ARNs, oferecem os seguintes recursos: poda de informações, estrutura de rede e geração e avaliação de hipóteses.

Para a criação da ARN, são realizadas quatro etapas:

- **Passo A:** Tendo uma base de dados e o um suporte e confiança mínima, todas as Regras de Associação devem ser extraídas utilizando algum algoritmo padrão como *Apriori*, *Apriori-TID* ou *FP-Growth*.
- **Passo B:** Escolher um alvo. Uma variável de interesse, que será representada no grafo como o nó objetivo.
- **Passo C:** Realizar a poda. Retirando informações inconsistentes e não relevantes. O resultado disso é uma ARN.
- **Passo D:** Identificar caminhos mais curtos entre o nó objetivo e os demais na ARN. O conjunto destes caminhos representa a Rede exploratória para o nó objetivo.

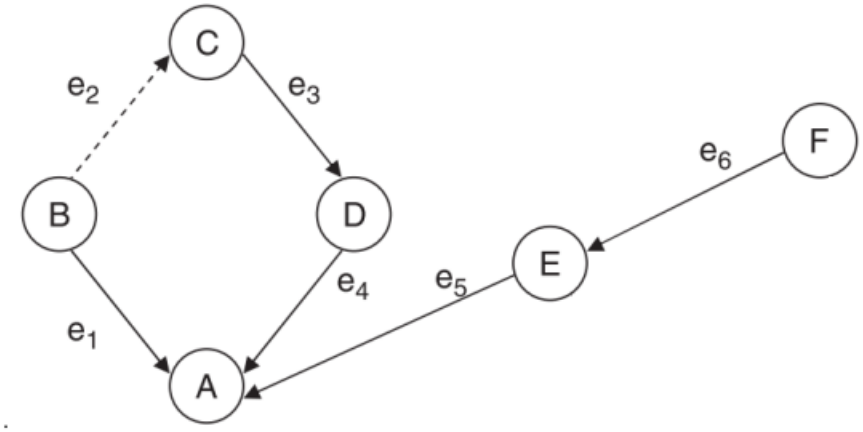


Figura 1.8. Exemplo de ARN

Na Figura 1.8, a ARN tem como alvo o vértice "A". Logo, todas as Regras extraídas que possuem A como consequência são selecionadas. Neste caso apenas as regras $(B \Rightarrow A)$, $(D \Rightarrow A)$ e $(E \Rightarrow A)$ serão selecionadas.

A ARN oferece o benefício de organização de regras em um contexto, de tal forma que um raciocínio objetivo pode ser explicado usando as regras mais relevantes no conjunto. A poda local é uma outra vantagem da ARN, já que uma regra redundante para um nó objetivo em particular pode tornar-se relevante para outro alvo, tornando essa abordagem mais flexível do que a poda com base em medidas estatísticas.

As Redes de Regras de Associação fornecem um mecanismo para sintetizar as Regras de Associação de maneira estruturada. Elas implementam uma metodologia que integra a pesquisa de mineração de dados com métodos estatísticos. A pesquisa em mineração está relacionada ao aperfeiçoamento de uma abordagem teórica existente, já os métodos estatísticos estão ligados a validação das teorias sobre os dados da pesquisa. Estas características demonstram a amplitude do uso da ARN e as diferentes áreas em que podem ser aplicada.

1.4. Redes de Regras de Associação Filtradas

Com o aumento dos conjuntos de dados, enumerar todas as possíveis combinações dos elementos, e depois verificar sua a correlação se torna uma tarefa computacional inviável. A utilização de Mineração de Regras se faz importante pois oferece mecanismos que podem ser utilizadas na validação de hipóteses.

Por meio da Mineração de dados é possível filtrar grandes quantidades de dados e gerar padrões que são interessantes, e até surpreendentes em certos casos. A desvantagem dessa metodologia é grande quantidade de padrões e hipóteses que são geradas a partir de um conjunto de dados, tantas que se torna difícil decidir quais são confiáveis e quais valem a pena serem analisadas.

Algoritmos utilizados na descoberta de Regras usam medidas capazes de avaliar a qualidade de uma regra. O supor mínimo e a confiança se destacam, embora *Lift*, *Gain*, *Certainty Factor*, *Added Value* ou *Leverage* também sejam medidas que fornecem informações sobre a regra. Os algoritmos, normalmente, extraem todas as Regras de Associação de um conjunto de dados de acordo com o suporte mínimo e valor mínimo de confiança, fazendo com que o número de Regras extraídas seja alto, dificultando a capacidade de exploração do usuário.

A Rede de Regras de Associação Filtrada (*Filtered*-ARN) alia medidas objetivas com estrutura de Rede para modelar e auxiliar a visualização e análise das regras extraídas de um *dataset*. A estrutura da *Filtered*-ARN é semelhante a da ARN, o que muda é a maneira como as regras são selecionadas, já que considera influência estatística comprovada para promover a identificação de hipóteses com maior probabilidade de serem verdadeiras.

O resultado desse processo são regras que possuam maior chance de serem interessantes, e o objetivo central das Redes de Regras de Associação Filtradas é apresentar um Grafo com estas regras. As hipóteses geradas podem ser validadas por diversos métodos, um deles é a a avaliação realizada por um especialista na área que está sendo pesquisada. Na *Filtered*-ARN o usuário pode visualizar um conjunto de itens que têm influência estatística em vez de elementos que apenas se relacionam com o item objetivo.

1.4.1. Modelo Proposto para Geração de Hipóteses de Maior Confiabilidade

A Tabela 1.1 contém o *dataset Lenses*. Nela é possível identificar algumas informações sobre um paciente e que tipo de lente é recomendado para ele.

Usando o *dataset Lenses*, se o usuário construir uma ARN com nó alvo o atributo "[lenses]=hard contact lense", o atributo "[prescription]=myope" aparece diretamente conectado ao nó objetivo.

A hipótese "um paciente com miopia tem maior probabilidade de usar uma lente rígida" pode ser formulada a partir do conhecimento extraído da Rede. No entanto a ARN, pode apresentar relações que não possuem influência entre os elementos da regra.

Ao calcular a medida *Added Value* da regra "[prescription]=myope $\Rightarrow$[lenses]=hard contact lense", encontra-se AV = 0, o que indica que a variável "prescription]=myope" não apresenta relevância para a ocorrência da variável "[lenses]=hard contact lense". Portanto

Tabela 1.1. Trecho do *Dataset Lenses*

Age	Prescription	Astigmatic	Tear	Lenses
young	myope	no	reduced	none
young	myope	no	normal	soft contact lenses
young	myope	yes	reduced	none
young	myope	yes	normal	hard contact lense
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft contact lenses
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard contact lense
Pre-presbyopic	myope	no	reduced	none
Pre-presbyopic	myope	no	normal	soft contact lenses
Pre-presbyopic	myope	yes	reduced	none
Pre-presbyopic	myope	yes	normal	hard contact lense
Pre-presbyopic	hypermetrope	no	reduced	none
Pre-presbyopic	hypermetrope	no	normal	soft contact lenses
Pre-presbyopic	hypermetrope	yes	reduced	none
Pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard contact lense
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft contact lenses
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

a hipótese levantada possui caráter equivocado, pois os dados não possuem influência direta.

A *Filtered*-ARN permite a exploração de um alvo com análise de dependência entre os elementos das regras utilizando filtros de medidas objetivas na etapa da seleção das regras.

O Algoritmo para geração da *Filtered*-ARN pode ser descrito em 3 passos:

- **Passo A:** Extração das Regras com corte por suporte e confiança mínimos.
- **Passo B:** Efetuar o calculo das medidas *Added Value* e *Gain*, cortando todas as regras que tiverem AV = 0 e valore inferiores ao ganho mínimo.
- **Passo C:** Escolher uma variável de interesse, ou seja, um nó alvo.

Em relação a etapa de Mineração de Regras, a única restrição adicionada se comparada a uma Mineração de Regras de Associação convencional é que as regras devem possuir conjuntos unitários no antecedente e consequente. Esse formato facilita a modelagem da *Filtered*-ARN.

O segundo passo, utiliza medidas objetivas para filtrar as regras. Para a seleção, considera-se apenas as regras que possuem elementos com dependência estatística e definição do ganho mínimo de influência.

O ultimo, passo o usuário deve decidir o item que deseja entender do conjunto de dados. Escolhido um alvo, é efetuada a construção da *Filtered*-ARN. Primeiro, o item alvo é colocado no gráfico (Nível 0). Então todas as regras em que seu *LHS* ainda não consta no gráfico e que possuem o item alvo como *RHS* são modelados na rede (Nível 1). O processo é repetido tendo agora como alvo os itens do Nível 1, em seguida o Nível 2 e assim por diante, e só termina quando não há mais regras a serem modeladas.

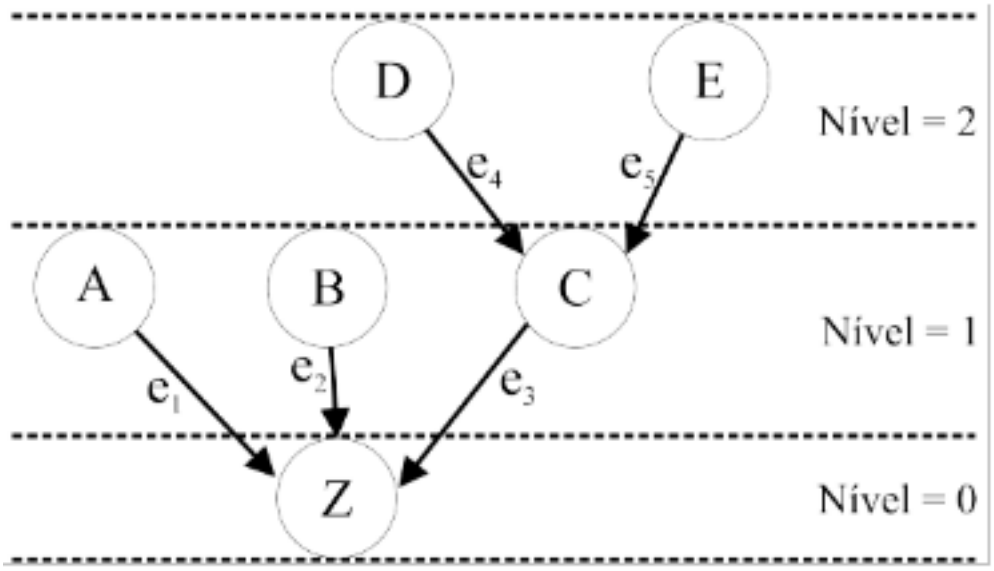


Figura 1.9. Níveis dos nós de uma *Filtered*-ARN

Para avaliar a *Filtered*-ARN dois métodos podem ser utilizados: i) faz-se uma comparação com uma ARN convencional, analisando as diferenças, as vantagens e desvantagens de usar cada uma dessas abordagens; ii) faz-se uma comparação com uma árvore de decisão com o objetivo de confrontar os resultados gráficos e decidir qual a melhor estrutura para análise do *dataset*.

1.4.2. Procedimento Metodológico

Para a aplicação do minicurso foram explanados os conceitos básicos descritos nas Seções anteriores e realizada a demonstração de experimentos práticos com alguns datasets. Foi selecionado a base de dados "*Lenses*" para exemplificação dos resultados.

Os objetivos desejados com a aplicação desse minicurso foram:

- Apresentar fundamentos sobre Redes de Regras de Associação Filtradas e suas características.
- Abordar conceitos que se fazem necessários para a construção de uma *Filtred*-ARN, como por exemplo, Regras de Associação, Redes e Medidas Objetivas.
- Realizar, por meio de um experimento, a construção de uma *Filtreddataset*.
- Comparar a *Filtered*-ARN com outras estruturas, para visualizar as diferença na geração de regras ou saídas, e demonstrar como o levantamento de hipóteses pode ser impactado.

1.4.2.1. *Dataset*

Para a abordagem prática da *Filtered*-ARN, o *dataset Lenses* foi utilizado. Neste conjunto de dados, cada linha representa os atributos de um paciente e a lente de contato que foi prescrita para ele. Por meio dessas informações é possível descrever quais características influenciam na prescrição de cada tipo de lente de contato. Vale a pena destacar que este *dataset* é uma simplificação do problema, logo as hipóteses levantadas podem não corresponder ao cenário real.

Por possuir um pequeno número de atributos, todos os valores foram considerados (suporte mínimo definido como 0), já a confiança mínima foi definida para 0,25, dessa forma classes que ocorreram pelo menos 1 em 4 vezes podem serem estudadas. O tamanho da classe foi definido com 2, ou seja, um item no LHS e um item no RHS.

Foram extraídas 99 Regras de Associação desta configuração. Após a etapa de filtragem 60 regras restaram.

1.4.2.2. *Filtred*-ARN e comparações

A Rede Filtrada foi gerada tendo como um alvo o item "[lenses] = hard". A Rede possui 4 níveis, onde no nível 0 se encontra o item objetivo. Apenas 2 itens estão ligados ao alvo: "[tear]=normal" e "[astigmatic]=yes". Essas regras podem ser capazes de implicar no item "[lenses] = hard", tornando interessante o estudo, já que são os únicos parâmetros que geram influencia no item objetivo. Por este motivo a geração de hipóteses verdadeiras apresenta um alto grau de probabilidade.

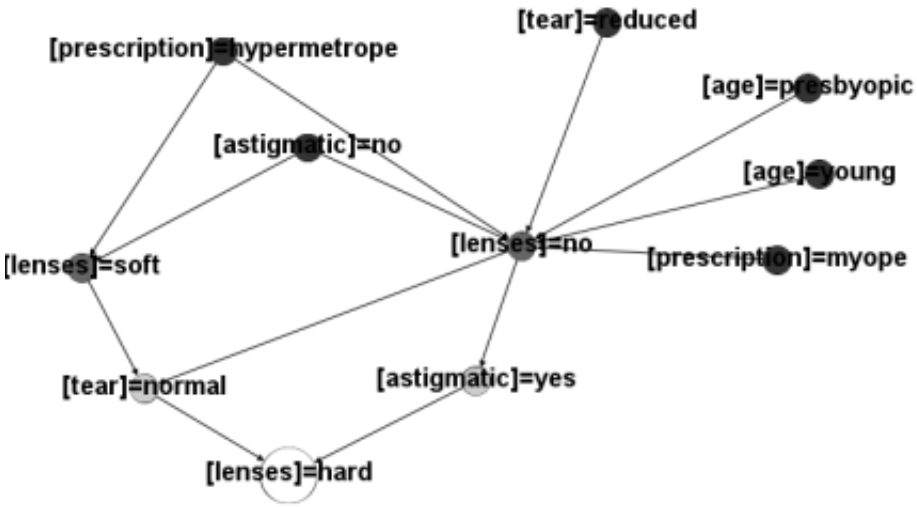


Figura 1.10. *Filtered*-ARN com "[lenses]=hard" como item alvo

Na Figura 1.10 é possível observar que os itens do nível 2 são as outras classes que indicam o tipo de lente "[lenses]=soft" e "[lenses]=no" e que estas implicam na ocorrência dos itens no nível 1 ("[astigmatic]=yes" e "[tear]=normal").

Nota-se no nível 3 a existência de diferentes regras que possuem conexão apenas

com a classe “[lenses]=no”. O que gera hipóteses para a construção de uma nova *Filtered*-ARN com “[lenses]=no” como objetivo, provocando um novo direcionamento na exploração do conhecimento.

A Figura 1.11 apresenta uma Rede de Regra de Associação comum. O item objetivo é o mesmo (“[lenses]=hard), no entanto esta estrutura possui 3 níveis, e é completamente diferente da Rede Filtrada.

Nota-se, no nível 1, que diversos itens implicam no nó alvo. O motivo desta ocorrência está no fato de que essa rede considera regras sem influencia comprovada (*Added Value* = 0), como “[prescription]=myope”=>“[lenses]=hard” e “[age]=young”=>“[lenses] =hard”, o que leva a geração de hipóteses equivocadas a respeito do uso de lentes rígidas.

Vale salientar outra diferença notória: O aumento de número de nós no nível 3, sem a distinção de dependência entre os mesmos. Isto dificulta a realização de estudos, pois a estrutura demonstra que todos os itens de um nível possuem o mesmo grau de importância.

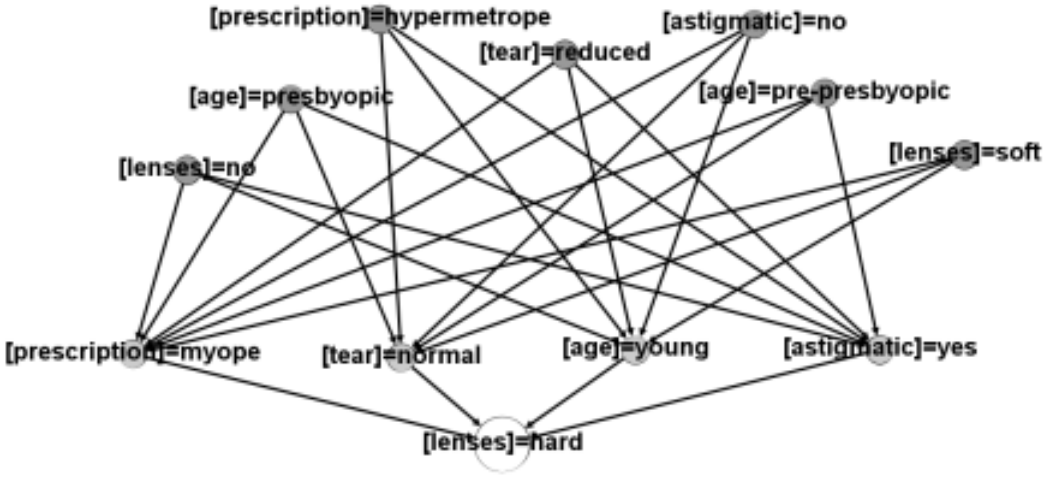


Figura 1.11. ARN com “[lenses]=hard” como item alvo

Ao comparar a *Filtered*-ARN com uma estrutura da árvore (Figura 1.12) de decisão, é possível identificar diferenças na explicação dos itens objetivos. Nas duas estruturas os itens “[tear]=reduced” e “[prescription]=hypermetrope” estão conectados diretamente a “[lenses]=no”, porém na Rede Filtrada, outras regras foram geradas.

A árvore indica que “[prescription]=myope” se liga diretamente à “[lenses]=hard”, porém, levando em consideração que esta condição não tem influência (AV = 0), esta saída pode ocasionar na geração de hipóteses equivocadas. Para evitar este tipo de equívoco se utiliza a *Filtred*-ARN.

1.5. Considerações Finais

As Regras de Associação possuem, como principal, tarefa de auxiliar a encontrar elementos que implicam na presença de outros em uma mesma transação. A partir do encontro

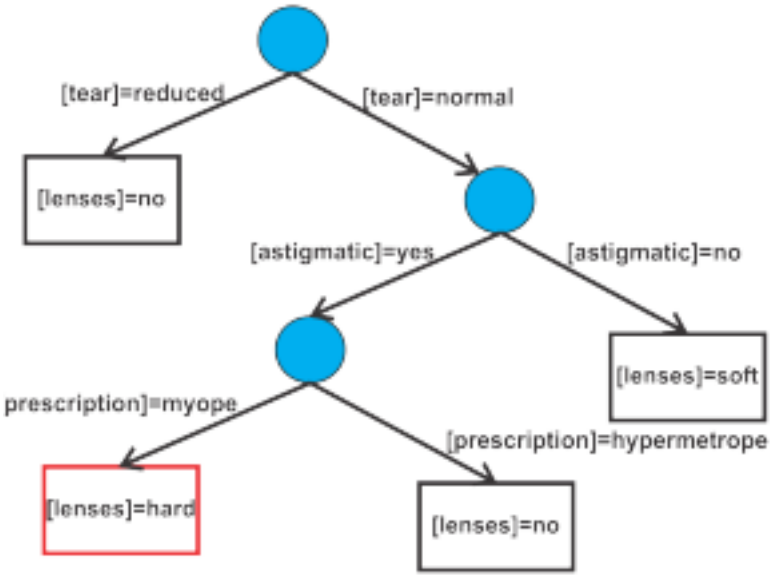


Figura 1.12. Árvore de Decisão para o *dataset Lenses*

desses elementos, é possível identificar relacionamentos e padrões frequentes em conjuntos de dados.

A utilização da descoberta de Regras de Associação possui grande aplicabilidade em diferentes contextos, permitindo a obtenção de conhecimento e resolução de problemas. Porém, há a necessidade de adequações nas regras, principalmente na fase de pós-processamento, a fim de evitar problemas relacionados às suas análises. A aplicabilidade das Regras se estende para várias tarefas da mineração de dados, sendo necessária a adequação do processo de extração das regras levando em conta fatores como custo computacional, tempo de resposta, entre outros.

Nesse capítulo foram esclarecidos temas relacionados à obtenção de Regras de Associação, bem como conceitos básicos relacionados à Redes voltadas a Mineração de Regras de Associação para extração de conhecimento. E, finalmente, foi explicado o funcionamento das Redes de Regras de Associação Filtradas (*Filtered*-ARNs), que foram o maior enfoque deste trabalho.

Por fim, acredita-se que o uso das *Filtered*-ARNs, podem auxiliar na extração de padrões de interesse ao usuário, e também, em diversas fases da Mineração de Regras de Associação, conforme demonstrado na avaliação experimental.

Referências

[Aggarwal 2015] Aggarwal, C. C. (2015). *Data Mining: The Textbook*. Springer, New York, USA, 1st edition.

[Agrawal et al. 1994] Agrawal, R., Imielinski, T., and Swami, A. (1994). Mining Association Rules between Sets of Items in Large Databases. *Special Interest Group on Management of Data*, 22(2):207–216.

[Agrawal and Shafer 1996] Agrawal, R. and Shafer, J. C. (1996). Parallel mining of asso-

ciation rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969.

[Agrawal and Srikant. 1994] Agrawal, R. and Srikant., R. (1994). Fast algorithms for mining association rules. *Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, Proceedings of Twentieth International Conference on Very Large Data Bases (VLDB)*, pages 487–499.

[Brandes 2001] Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177.

[Coleman and Moré 1983] Coleman, T. F. and Moré, J. J. (1983). Estimation of Sparse Jacobian Matrices and Graph Coloring Blems. *SIAM Journal on Numerical Analysis*, 20(1):187–209.

[de Vasconcelos and de Carvalho 2004] de Vasconcelos, L. M. R. and de Carvalho, C. L. (2004). Aplicação de Regras de Associação para Mineração de Dados na Web. *Instituto de Informática da Universidade Federal de Goiás*, page 20.

[Gonçalves 2005] Gonçalves, E. C. (2005). Regras de Associação e suas Medidas de Interesse Objetivas e Subjetivas Objective and Subjective Measures for Association Rules. *INFOCOMP Journal of Computer Science*, 4(1):26–35.

[Han and Kamber 2006] Han, J. and Kamber, M. (2006). *Data Mining Concepts and Techniques*. Morgan Kaufmann, San Francisco, CA, second edition.

[Han et al. 2012] Han, J., Kamber, M., and Pei, J. (2012). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Waltham, MA, USA, 3rd edition.

[Hendrickx et al. 2015] Hendrickx, T., Cule, B., Meysman, P., Naulaerts, S., Laukens, K., and Goethals, B. (2015). Mining association rules in graphs based on frequent cohesive itemsets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9078, pages 637–648.

[Klemettinen et al. 1994] Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., and Verkamo, A. (1994). Finding interesting rules from large sets of discovered association rules". In Nabil, R., editor, *Proceedings of 3rd International Conference on Information and Knowledge Management*, pages 401–407.

[Le and Vo 2016] Le, T. and Vo, B. (2016). The lattice-based approaches for mining association rules: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(4):140–151.

[Meo et al. 1996] Meo, R., Psaila, G., and Ceri, S. (1996). A new SQL-like operator for mining association rules. In Vijayaraman, T., editor, *Proceedings of the 22nd International Conference on very Large Data Bases*, pages 122–123.

[Namaki et al. 2017] Namaki, M. H., Wu, Y., Song, Q., Lin, P., and Ge, T. (2017). Discovering Graph Temporal Association Rules. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17*, pages 1697–1706.

[Nettleton 2013] Nettleton, D. F. (2013). Data mining of social networks represented as graphs. *Computer Science Review*, 7(1):1–34.

[Newman 2010] Newman, M. (2010). *Networks: An introduction*, volume 55. Oxford University Press, New York, USA, 1st edition.

[Pandey et al. 2009] Pandey, G., Chawla, S., Poon, S., Arunasalam, B., and Davis, J. G. (2009). Association Rules Network: Definition and Applications. *Statistical Analysis and Data Mining*, 1(4):260–179.

[Piri et al. 2018] Piri, S., Delen, D., Liu, T., and Paiva, W. (2018). Development of a new metric to identify rare patterns in association analysis: The case of analyzing diabetes complications. *Expert Systems with Applications*, 94:112–125.

[Sahar 2003] Sahar, S. (2003). *What Is Interesting: Studies on Interestingness in Knowledge Discovery*. PhD thesis, Tel-Aviv University.

[Silva 2010] Silva, T. S. A. (2010). *Um Estudo de Medidas de Centralidade e Confiabilidade em Redes*. PhD thesis, Centro Federal de Educação Tecnológica do Rio de Janeiro.

[Simard et al. 2016] Simard, F., St-Pierre, J., and Biskri, I. (2016). Mining and visualizing robust maximal association rules on highly variable textual data in entrepreneurship. In *Proceedings of the 8th International Conference on Management of Digital EcoSystems - MEDES*, pages 215–222.

[Toivonen 1996] Toivonen, H. (1996). Sampling large databases for association rules. *The VLDB Journal*, pages 134–145.

[Usman and Usman 2016] Usman, M. and Usman, M. (2016). Multi-level mining and visualization of informative association rules. *Journal of Information Science and Engineering*, 32(4):1061–1078.

[Vinaya and Shah 2016] Vinaya, M. and Shah, K. (2016). Performance Evaluation of Distributed Association Rule Mining Algorithms. *Procedia - Procedia Computer Science*, 79:127–134.

[Zaki and Meira 2013] Zaki, M. J. and Meira, M. J. (2013). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.

[Zhao et al. 2009] Zhao, Y., Zhang, C., and Cao, L. (2009). *Post-Mining of Association Rules : Techniques for Effective Knowledge Extraction*.

Capítulo 2

COVID-19: Aquisição, tratamento e visualizações interativas de dados do Ministério da Saúde

Alexandre R. C. Ramos, Jonnison L. Ferreira, Moisés Laurence de F. Lima Junior, Aristofánes Corrêa Silva

Abstract

Given the COVID-19 pandemic, declared on March 11, there was an exponential increase in the available data. The data science area, in this context, can produce accurate and objective computational solutions to reduce the impact of the pandemic on some aspects of our life in society. In this perspective, this chapter presents the workflow for building interactive visualizations of COVID-19 in Brazil, involving the acquisition and treatment of data obtained from the official panel of the Ministry of Health. The techniques and tools presented enabled direct and accurate visualizations, which can help decision making by the public authorities and guarantee the communication of pandemic information in Brazilian states and regions.

Resumo

Diante da pandemia da COVID-19, declarada em 11 de março, houve um aumento exponencial na disponibilização de dados relacionados ao tema. O campo da ciência de dados, neste contexto, pode produzir soluções computacionais tempestivas e diretas para contribuir com a redução do impacto da pandemia em nossa sociedade. Nesta seara, este capítulo apresenta o fluxo de trabalho para construção de visualizações interativas da COVID-19 no Brasil, envolvendo a aquisição e tratamento de dados obtidos no painel oficial do Ministério da Saúde. As técnicas e ferramentas apresentadas possibilitaram visualizações diretas e precisas, que podem auxiliar a tomada de decisão pelo poder público, bem como garantir a comunicação das informações da pandemia nos estados e regiões brasileiras.

2.1. Introdução

O surto de COVID-19 foi relatado pela primeira vez em Wuhan (China) e se espalhou para mais de 50 países de forma assustadoramente rápida, assim, a Organização Mundial

da Saúde (OMS) declarou a COVID-19 como Emergência de Saúde Pública de Interesse Internacional (PHEIC) em 30 de Janeiro de 2020. Diante do aumento exponencial no número de casos registrados, o que fatalmente acarretaria em um colapso de sistemas de saúde pelo mundo, em 11 de março a infecção causada pela SARS-CoV-2 foi caracterizada como uma pandemia. No Brasil, o crescimento acelerado da doença também apresentou destaque e preocupação, pois rapidamente a situação evoluiu de casos importados (primeiro caso registrado em 26 de fevereiro) à transmissão comunitária (confirmada em 20 de março) [Datasus 2020].

Com o estado de pandemia declarado pela OMS, houve um aumento exponencial de dados relacionados ao tema sendo disponibilizados por diferentes órgãos oficiais [Datasus 2020, Government 2020]. Neste cenário, as técnicas e ferramentas para análise e visualização destes dados mostram-se fundamentais para o planejamento de ações governamentais contundentes e tempestivas, como também constituem ferramentas para a comunicação efetiva da população sobre os problemas acarretados pela pandemia.

Em [Doyle 2020] é destacada a necessidade crítica de dados epidemiológicos oportunos, precisos e acessíveis no acompanhamento da COVID-19. Dados de casos com detalhamento geográfico são particularmente valiosos durante surtos, mas raramente disponibilizados em tempo real e/ou apresentados de forma simples. Desta maneira, se torna fundamental e necessário o desenvolvimento de ferramentas para visualizações interativas de dados, possibilitando a comunicação e contextualização das informações.

Esse capítulo discorre sobre os aspectos teóricos e práticos de etapas relevantes no campo da ciência de dados aplicada à COVID-19, bem como suas possíveis contribuições no enfrentamento da pandemia. Assim, apresentaremos técnicas e ferramentas utilizadas para a aquisição e tratamento de dados (Seção 2.2) e na construção de visualizações interativas (Seção 2.3), tendo como base o conjunto de dados fornecidos pelo portal da COVID-19 do Ministério da Saúde.

2.2. Aquisição e Tratamento de Dados

2.2.1. Aquisição de Dados

Esta seção apresenta um fluxo de trabalho para a aquisição e tratamento de dados do Ministério da Saúde sobre o Covid-19. Essa etapa consiste em acessar as fontes de dados. No nosso caso, os dados estão disponíveis no sítio[1] do ministério da saude dedicado ao acompanhando da disseminação do vírus no território nacional.

A Figura 2.1 aprensenta a página inicial do site do ministério da saúde dedicado ao monitoramento do Covid-19. Onde podemos verificar alguns gráficos e interações. Para obter os dados em um formato ideal a ser processado, devemos clicar no botão ”Arquivo CSV”, o qual destacamos na Figura 2.1 com uma caixa vermelha. Ao clicar no botão somos então redirecionados para o *download* de um arquivo *CSV (Comma Separeted Values)* com as informações do Covid-19 organizadas por data e estado.

Realizar essa tarefa manualmente torna o processo aquisição dos dados muito lento, e deixando dependente de uma iteração manual, onde o administrador do sistema

[1] https://covid.saude.gov.br/

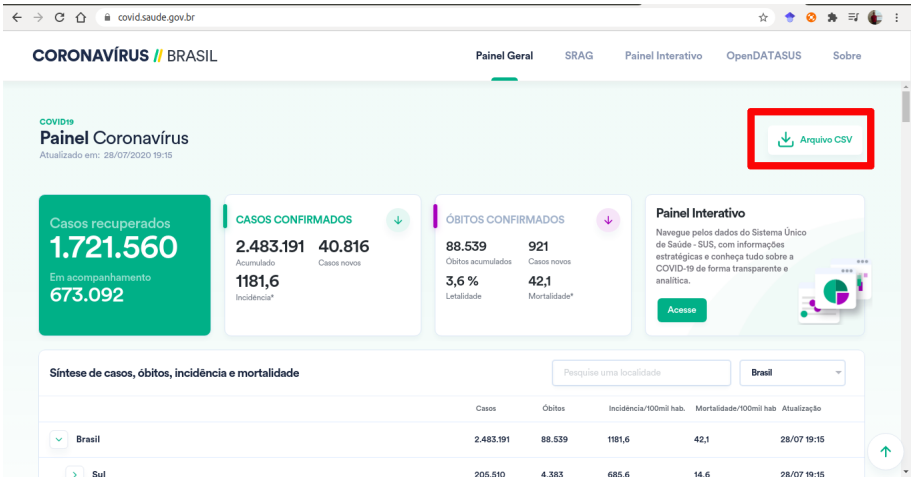


Figura 2.1. Página inicial do site do ministério da saúde dedicado ao monitoramento do Covid-19

deve realizar o processo manualmente. Existem diversas técnicas que permitem automatizar esse processo, esse tipo de aplicação é chamada de *Web Scraping*. Mitchell (2019) define *Web Scraping* como um conjunto de técnicas que permitem a mineração de dados de diversas fontes na *web* de maneira automatizada, onde determinados *scripts* acessam o conteúdo do site e extraem as informações importantes para o contexto analisado.

Esse processo tem funcionamento baseado em requisições *HTTP* automáticas ao sítios onde as informações relevantes são armazenadas, após a requisição o *HTML* é processado e a informação relevante é extraída. Existem diversas bibliotecas que oferecem uma *API* amigável para a execução dessas requisições, como o *python-requests* para execução da requisição [Reitz 2020] e *BeatifulSoap*[2], que são alternativas de código aberto bem populares.

O site do Ministério da saúde altera diariamente o a *URL* do arquivo *CSV*, além disso o link para *download* do arquivo é gerado automaticamente por meio do controle de eventos de interação com o usuário através da execução de *JavaScript*. Portanto além de automatizar a requisição é necessário realizar o *scraping* do *JavaScript*, permitindo então a simulação da execução da ação de clicar no botão de acesso ao arquivo, para então capturar o link do arquivo.

Utilizaremos o *Selenium*[3] para realizar esse *Scraping de Javascript*. O *Selenium* é um *framework* de código aberto que permite a execução automatizada de diversas tarefas pelo browser, proposto inicialmente como ferramenta de testes automatizados, também é muito utilizado para essa captura de informações na *web* [Mitchell 2019].

2.2.1.1. Utilizando o *Selenium* para capturar os dados

Primeiro passo para utilização do *Selenium* juntamente com o *Python* é a instalação da biblioteca, para isso é utilizado o Código Fonte 2.1.

[2]http://www.crummy.com/software/BeautifulSoup/bs4/
[3]http://docs.seleniumhq.org/

```
1 pip install selenium
```

Código Fonte 2.1. Instalando o *Selenium*.

Após a instalação, iniciaremos nosso *script* de *download* automático do arquivo *CSV*, o Código Fonte 2.2, responsável adicionar as bibliotecas que utilizaremos no *script*. A primeira linha é responsável por importar o *webdriver*. O *webdriver* é o navegador onde o *Selenium* irá executar a requisição automática, o nosso caso utilizaremos o navegador *Mozilla Firefox*, além deste o *Selenium* suporta outros navegadores, como o *Google Chrome* e *PhantomJS*. A segunda linha importa da classe *Python* que utilizaremos para configurar o firefox. Por fim importaremos as bibliotecas *glob* e *os* que utilizaremos para mantermos apenas a versão mais atual do *CSV* disponibilizado pelo ministério da saúde.

```
1 from selenium import webdriver
2 from selenium.webdriver.firefox.options import Options
3 import glob,os
```

Código Fonte 2.2. Importando bibliotecas necessárias.

O passo seguinte será apagar as versões anteriores dos arquivos e configurar o navegador para *download* do arquivo da atualizado do sítio do ministério da saúde. O Código Fonte 2.3 é o código que utilizaremos para realizar essa configuração. A linha 1 e 2 definem onde ficaram os arquivos do *CSV* original e o *CSV* após o tratamento para evitar inconsistências, respectivamente. As linhas 4 e 5 percorrem o diretório de *download* e apagam as versões anteriores dos arquivos a serem baixados.

O código das linhas 7 a 11 servem para configurar o navegador para realizar o *download* de qualquer arquivo selecionado pelo *script* e salvar no diretório correto sem a necessidade de confirmação do usuário caso o arquivo seja *CSV*. As linhas 13 e 14 configuram o navegador que o mesmo seja executado em plano fundo, sem a necessidade de abrir a interface gráfica.

```
1  download_path = '/tmp/'
2  output_path = '/webapps/covid/static/'
3
4  for file in glob.glob(download_path+"*.csv"):
5    os.remove(file)
6
7  profile = webdriver.FirefoxProfile()
8  profile.set_preference("browser.download.folderList", 2)
9  profile.set_preference("browser.download.manager.showWhenStarting",
       False)
10 profile.set_preference("browser.download.dir", download_path)
11 profile.set_preference("browser.helperApps.neverAsk.saveToDisk","text/
       csv,application/csv,text/plan,text/comma-separated-values")
12
13 options = Options()
14 options.headless = True
```

Código Fonte 2.3. Apagando versões anteriores e configurando o navegador.

Com as configurações realizadas, basta abrir o navegador e acessar o site, as linhas 1 e 2 do Código Fonte 2.4 abrem o navegador com as configurações definidas e

acessa o sítio do ministério da saúde. Após acessar o site precisamos encontrar o botão de *download*, para isso devemos identificá-lo de maneira única na página, ao analisar o sitio percebemos que ele é o botão a possuir a classe "*btn-outline*", com isso, nas linhas 4 e 5 encontramos o botão e realizamos *download*. A linha 7 é utilizada para que o navegador seja encerrado após o download do arquivo. Com isso encerramos o *script* de *download* e podemos tratar os dados.

```
1 driver=webdriver.Firefox(firefox_profile=profile,options=options)
2 driver.get("https://covid.saude.gov.br/")
3
4 btn = driver.find_element_by_class_name("btn-outline")
5 btn.click()
6
7 driver.close()
```

Código Fonte 2.4. Abrindo o navegador e acessando o sítio.

2.2.2. Tratamento de Dados

Em aplicações onde buscamos dados de diversas fontes, é necessário o tratamento dos dados, para que não hajam inconsistências [Grus 2019]. Mesmo utilizando uma única fonte de dados precisaremos fazer alguns tratamentos nos dados, para isso utilizaremos a biblioteca pandas[4].

O Pandas fornece uma sério de estruturas e funções de dados avançadas, projetadas para torna o trabalho com dados estruturados mais rápido e simples, fornecendo um ambiente de análise de dados poderoso e produtivo [McKinney 2012].

O arquivo fornecido pelo ministério da saúde possui algumas inconsistências, como datas em formatos diferentes e informações que mudam de um dia para o outro. Para resolver esses problemas utilizaremos a biblioteca *pandas*.

O Código Fonte 2.5 mostra o tratamento dos dados, na linha 1 importamos as bibliotecas necessárias, usaremos *glob* para encontrar o arquivo no diretório de download, como pode ser visto na linha 3. Na linha 4 realizaremos a leitura do arquivo.

As linhas 5 e 6 configuramos as inconsistências que podem ocorrer no campo que define a data da coleta e convertemos os dados da coluna para o tipo *datetime64*. Na linha salvamos o arquivo *CSV* processado no local definido no Código Fonte 2.3, selecionando apenas os campos que queremos e formatando a data corretamente, para um melhor tratamento na interface *web*.

```
1 import pandas,glob
2
3 csv_name = glob.glob(download_path+"*.csv")[0]
4 pd_df = pd.read_csv(csv_name,sep=";")
5 pd_df = pd_df.rename(columns=lambda x: "data" if "data" in x.lower()
      else x)
6 pd_df['data'] = pd_df['data'].astype('datetime64[ns]')
7 pd_df.to_csv(output_path+"minSaude.csv",header=['regiao','estado','data
      ','casosNovos','casosAcumulados','obitosNovos','obitosAcumulados'],
      index=False,date_format='%d/%m/%Y')
```

[4] https://pandas.pydata.org/

Código Fonte 2.5. Tratando os dados com o *pandas*.

2.3. Visualização de Dados

Esta seção apresenta um fluxo de trabalho para a construção de visualizações da *COVID-19* após a aquisição e tratamento de dados do Ministério da Saúde. Assim, apresentaremos inicialmente as visualizações propostas e seu contexto (Subseção 2.3.1) e discutiremos a codificação do projeto e a utilização de um conjunto de bibliotecas relacionadas à construção de gráficos e mapas (Subseção 2.3.2). Vale ressaltar: para a devida compreensão e possibilidade de adaptações nas visualizações produzidas nesse trabalho, são necessários conhecimentos introdutórios relacionados à programação Web (HTML, CSS, *Javascript*, DOM, dentre outros).

O fluxo de trabalho e as informações discutidas nessa seção partem da estruturação dos dados fornecidos pelo ministério da saúde em um arquivo de dados tabelados (CSV), cuja construção é descrita na Seção 2.2. Este arquivo[5] é constituído pelos dados da COVID-19 no Brasil organizados por região, estado, data, número de novos casos, número de casos acumulados, número de novos óbitos e número de óbitos acumulados.

2.3.1. Apresentação

As visualizações apresentadas neste trabalho tem como objetivo tornar simples e interativas as informações relacionadas à COVID-19 presentes no painel do Ministério da Saúde[6]. Portanto, nos concentramos em questões simples e diretas que poderão ser respondidas rapidamente na leitura das visualizações propostas. Por exemplo:

1. ”Qual o número de casos confirmados e de óbitos do meu estado?”
2. ”Como está a situação do meu estado em relação à sua região ou ao Brasil?”
3. ”Como aconteceu a evolução temporal de novos casos e óbitos em meu estado ou Região?”

As Figuras 2.2 e 2.3 ilustram as visualizações propostas, formadas por um painel, com mapas temáticos dos estados e regiões brasileiras (Figura 2.2), e um *DashBoard* interativos (Figura 2.3). Ao refletir sobre as questões apontadas inicialmente, podemos observar que as variáveis região, estado, casos novos e acumulados, óbitos novos e acumulados e data se repetem em diferentes combinações. Neste contexto se insere a interatividade proposta em nossas visualizações, buscando possibilitar ao público alvo uma navegação simples com dados consistentes e diretos.

No painel de mapas temáticos três controles foram possibilitados ao usuário: estado ou região; incidência ou letalidade e data. Além disso as informações detalhadas de cada estado podem ser acessadas pela legenda do canto superior direito, quando o

[5] Disponível aqui: https://drive.google.com/file/d/1cc6T4fLSixunlt-AcndJY7q6DP7lD6bV/view?usp=sharing

[6] https://covid.saude.gov.br/

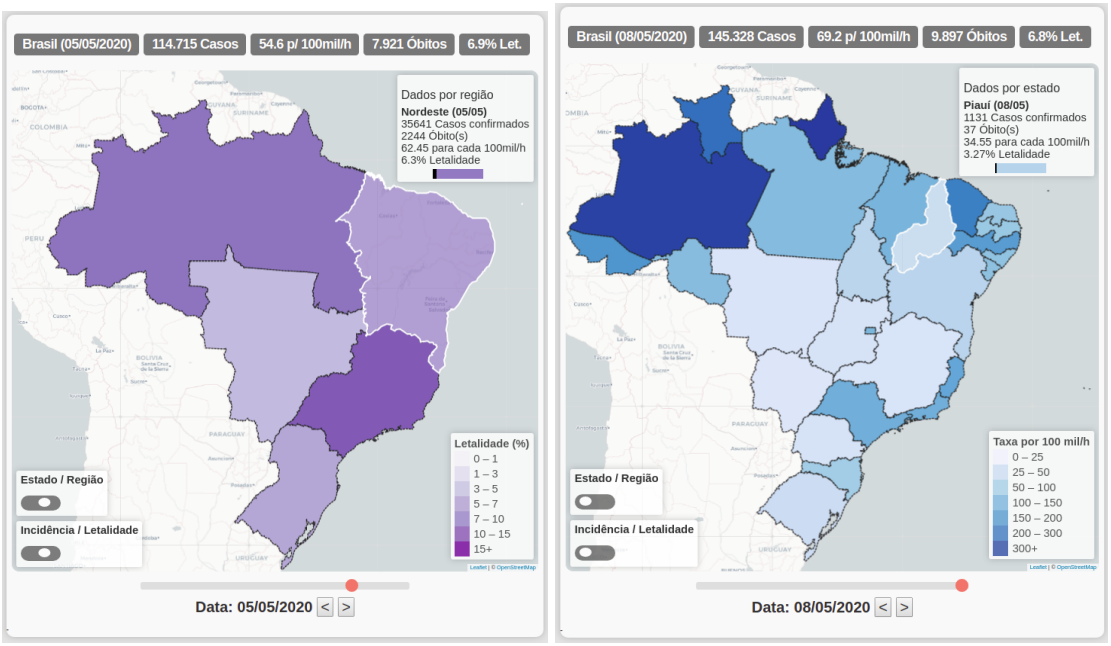


Figura 2.2. Mapas interativos do Coronavírus no Brasil. (A) Letalidade por Região e (B) Incidência por estado. Também disponível em vídeo, no link: `https://youtu.be/rOKTCQCueCc`

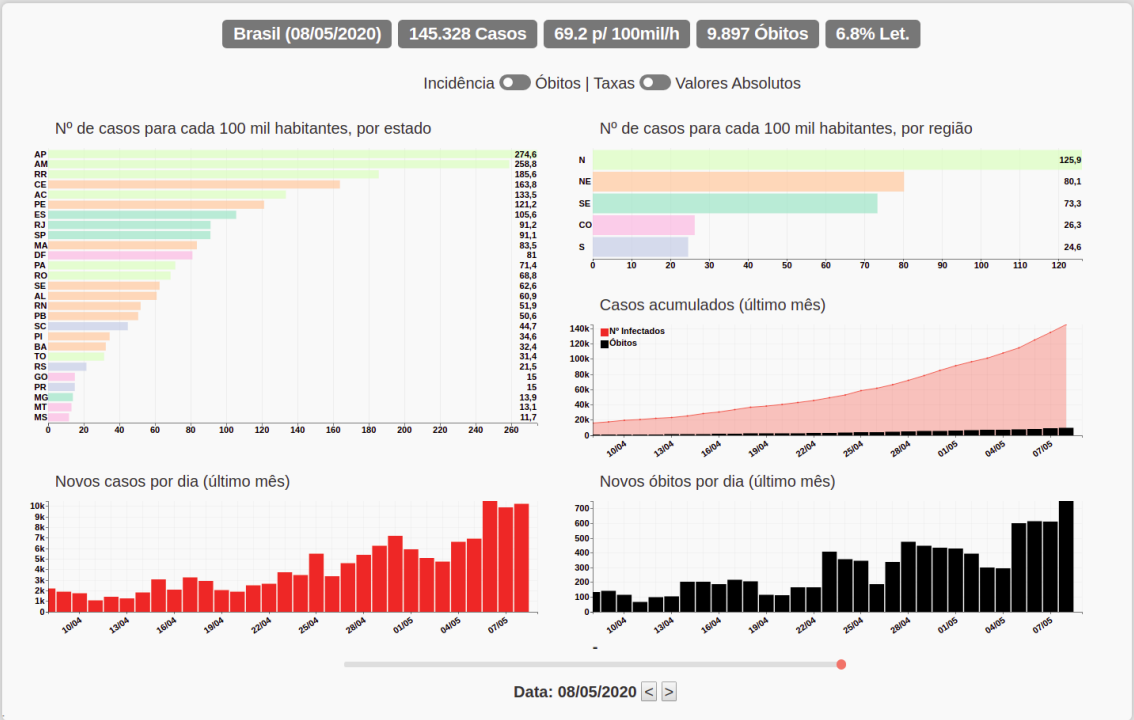


Figura 2.3. DashBoard Interativo do Coronavírus no Brasil. Também disponível em vídeo, no link: `https://youtu.be/yBpQFCCEFuA`

mouse se encontra sobre o respectivo limite geográfico. Já no *Dashboard* os próprios gráficos estão interligados como filtros. É possível observar a evolução temporal das variáveis de uma região ou estado clicando em sua respectiva barra horizontal. Além disso, também foram possibilitados três controles: incidência ou óbitos; taxas ou valores absolutos e data. Assim, a interatividade proposta possibilita múltiplas combinações e evita uma visualização extensa e cansativa, reduzindo também o espaço em tela necessário à comunicação.

Para definição dos elementos visuais presentes nestas visualizações, tais como a seleção de escala de cores, tipos de gráficos, conjunto de dados, experiência do usuário, dentre outros, utilizamos como base as informações apresentadas nos livros *Storytelling with Data: A Data Visualization Guide for Business Professionals* [Knaflic 2015] e *The Truthful Art: Data, Charts, and Maps for Communication* [Cairo 2015].

A partir da disponibilização de visualizações como estas em um ambiente Web, um grande número de pessoas podem ter acesso e navegar por informações da pandemia em seu estado ou região, reduzindo a gigantesca carga de informação disponibilizada em outros veículos e tratando do assunto de maneira direta, simples e interativa. Projetos como este podem, portanto, gerar contribuições significativas em diferentes contextos e problemáticas. Ademais, o domínio de técnicas de visualização, como as utilizadas neste trabalho, capacitam profissionais para a construção de comunicações robustas e eficazes.

2.3.2. Codificação e Bibliotecas utilizadas

Esta subseção apresenta um conjunto de bibliotecas e os principais trechos de códigos utilizados na construção das visualizações elaboradas neste trabalho. Com o desenvolvimento em uma plataforma Web, utilizando a linguagem *Javascript*, destacamos a utilização das bibliotecas D3js (Subsubseção 2.3.2.1), Leaflet (Subsubseção 2.3.2.2), DC e Crossfilter (Subsubseção 2.3.2.3). Ademais, a Subsubseção 2.3.2.4 demonstra os principais trechos de código desenvolvidos.

2.3.2.1. D3js

D3js[7] é uma biblioteca JavaScript que utiliza os padrões HTLM, SVG e CSS para gerar visualizações de dados interativas e dinâmicas. Sua ênfase nos padrões da web possibilita a utilização de todos os recursos dos navegadores modernos sem a necessidade de se vincular a estruturas proprietárias [Meeks 2015].

A D3js permite a manipulação de documentos com base em dados. Possui agilidade, comportando grandes conjuntos de dados e funcionamento dinâmico para interação e animação, ademais seu estilo funcional possibilita a reutilização de códigos através de uma coleção diversificada de módulos oficiais e desenvolvidos pela comunidade[8]. Neste trabalho, a D3js é responsável pela leitura da base de dados e pela manipulação de elementos do DOM, como demonstrado na Subsubseção 2.3.2.4.

[7] https://d3js.org/

[8] Disponíveis em https://observablehq.com/@d3/gallery

2.3.2.2. Leaflet

Leaflet[9] é uma biblioteca *JavaScript* de código aberto utilizada para criar mapas interativos, como ilustra a Figura 2.4. É projetada para funcionar de maneira eficiente nas principais plataformas desktop e mobiles e pode ser estendida através de plugins ou apresentar uma API de fácil utilização e código-fonte simples e legível, com diversos exemplos e tutoriais oficiais disponibilizados.

Neste trabalho, a *Leaflet* foi fundamental para a construção do painel com mapas interativos, ilustrado na Figura 2.2, o que pode ser observado nos respectivos códigos fonte comentados (Subsubseção 2.3.2.4). Além disso, quatro tutoriais oficiais são recomendados para a devida compreensão deste painel: *Leaflet Quick Start Guide*[10]; *Using GeoJSON with Leaflet*[11]; *Interactive Choropleth Map* [12]; E *Working with map panes*[13].

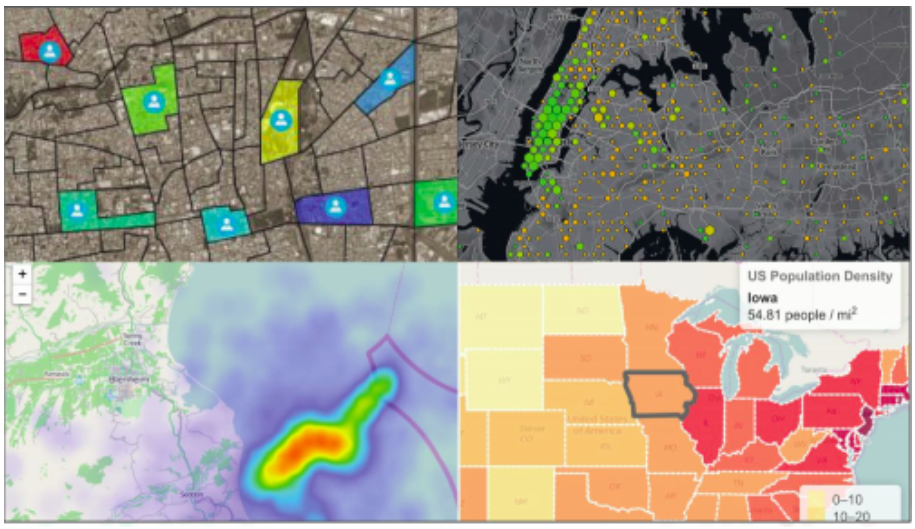


Figura 2.4. Exemplos de mapas construídos a partir da leaflet. Fonte: https://leafletjs.com/examples.html

2.3.2.3. DCjs e *Crossfilter*

Especificamente para a construção de *Dashboards* na Web, duas bibliotecas utilizadas se destacam: DCjs[14] e *Crossfilter*[15]. DCjs consiste em uma biblioteca *JavaScript* utilizada para a renderização de gráficos, com suporte nativo *Crossfilter*, que, por sua vez, é utilizada para análise e exploração extremamente rápida de grandes conjuntos de dados. Juntas, estas bibliotecas possibilitam a construção simplificada e eficaz de *Dashboards* interativos, como ilustrado na Figura 2.5.

[9] https://leafletjs.com/
[10] https://leafletjs.com/examples/quick-start/
[11] https://leafletjs.com/examples/geojson/
[12] https://leafletjs.com/examples/choropleth/
[13] https://leafletjs.com/examples/map-panes/
[14] https://dc-js.github.io/dc.js/
[15] https://square.github.io/crossfilter/

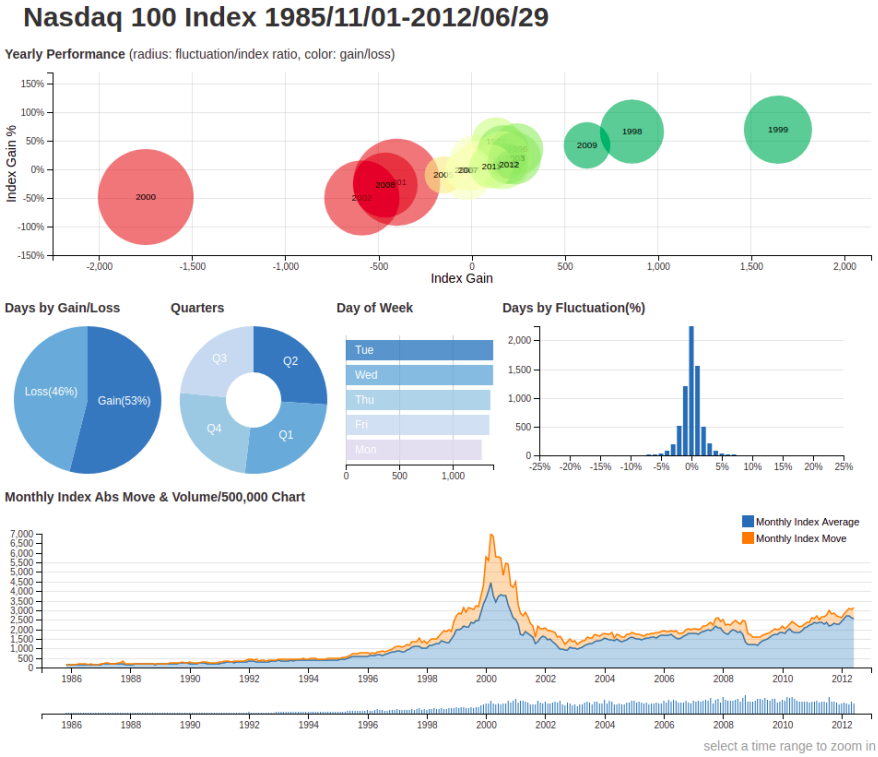


Figura 2.5. Exemplos de *DashBoard*, com diferentes tipos de gráficos, construído a partir da DCjs. Disponível em https://dc-js.github.io/dc.js/

Destaca-se, inicialmente, que a compreensão dos conceitos de dimensão e agrupamento *Crossfilter* são fundamentais [16]. Neste trabalho, o *Dashboard* interativo ilustrado na Figura 2.3 tem como base quatro exemplos DCjs oficiais simples: *Ordinal Line*[17]; *Ordinal Bar*[18]; *Row*[19]; E *Composite*[20]. As adaptações e alterações necessárias, como a formatação de números, rotação de legendas e manipulação de outros elementos visuais, foram possibilitadas a partir das suas respectivas documentações[21] e podem ser observadas nos códigos fonte (Subsubseção 2.3.2.4).

2.3.2.4. Principais trechos de códigos

Nesta seção apresentaremos os principais trechos de código relacionados à construção das visualizações produzidas. Considerando que a estilização (CSS) e marcação (HTML) de elementos não são temáticas centrais deste trabalho, bem como a demonstração do projeto completo seria extensiva e fora de contexto, destacamos três trechos de código fundamentais, com comentários explicativos, para possibilitar a compreensão e construção

[16] Um breve e conciso tutorial está disponível em: `https://www.tutorialspoint.com/dcjs/dcjs_introduction_to_crossfilter.htm`

[17] https://dc-js.github.io/dc.js/examples/ordinal-line.html

[18] https://dc-js.github.io/dc.js/examples/ordinal-bar.html

[19] https://dc-js.github.io/dc.js/examples/row.html

[20] https://dc-js.github.io/dc.js/examples/composite.html

[21] Disponíveis em: https://dc-js.github.io/dc.js/docs/html/

das visualizações apresentadas.

O Código Fonte 2.6 demonstra a leitura e manipulação dos dados da COVID-19 no Brasil utilizando as bibliotecas *D3js* e *Crossfilter*. Inicialmente os dados são tratados e armazenados em variáveis com a *D3js* (entre as linhas 1 e 11) e, em seguida, duas dimensões com chaves simples e composta (linhas 15 e 37) e grupos (linhas 20 e 39) *Crossfilter* são definidos e manipulados. Este código ilustra a leitura e acesso aos dados no ambiente de desenvolvimento Web com as respectivas bibliotecas.

```
1  d3.csv("../minSaude.csv", function(data){
2      dtgFormat = d3.time.format("%d/%m/%Y");
3      data.forEach(function(d) {
4          d.regiao = d.regiao;
5          d.uf = d.estado;
6          d.data = dtgFormat.parse(d.data);
7          d.casosNovos = +d.casosNovos;
8          d.casosAcumulados = +d.casosAcumulados;
9          d.obitosNovos = +d.obitosNovos;
10         d.obitosAcumulados = +d.obitosAcumulados;
11     });
12     //Leitura de dados pela CrossFilter:
13     var facts = crossfilter(data);
14     // Dimensao CrossFilter com chave composta (Estado + Data):
15     var Data_UF = facts.dimension(
16         function(d){
17             return 'UF:'+d.uf +', data:'+d.data
18         }),
19     // Agrupando o numero de casos por data e estado:
20     groupCasos_DataUF =
21         Data_UF.group().reduceSum(function(d){return d.casosAcumulados
       ;});
22         // Acessando elementos do grupo CrossFilter:
23     groupCasos_DataUF.all()
24             .forEach(function(d){
25                     console.log(d.key) // UF: x, data: y;
26                     console.log(d.value) // Total de casos acumulados.
27             });
28         // Acessando um elemento a partir de uma chave ('SP',
       '10/04/2020'):
29         var key = 'UF:'+'SP'+', data:'+dtgFormat.parse('10/04/2020'),
30         resposta = groupCasos_DataUF.all()
31         .filter(
32             function(i){
33                 return i.key == key
34         })[0].value;
35         console.log('total de casos de SP em 10/04/2020: '+resposta);
36     // Dimensao CrossFilter com chave simples (Regiao):
37         var Regiao = facts.dimension(function(d){return d.regiao;});
38     // Agrupando obitos por regiao (Todos os registros de obitos novos
       por regiao serao somados):
39         var groupObitos = Regiao.group()
40         .reduceSum(function(d){
41                     return d.obitosNovos;
42         });
43     // Acessando um elemento a partir de uma chave ('Nordeste'):
```

```
44          var resposta = groupObitos.all().filter(function(i){ return i.
    key == 'Nordeste' })[0].value;
45          console.log(resposta);
46
47  }
```

Código Fonte 2.6. Leitura e manipulação dos dados com D3js e *Crossfilter*

O Código Fonte 2.7 ilustra a construção de um mapa temático dos casos de COVID-19 no Brasil em 10 de maio de 2020 com *Leaflet*. Neste código, o painel é inicializado (linhas 1 a 7) com mapas *OpenStreetMap* [22]. Em seguida, definimos uma data de referência (linha 9) e uma escala linear (linhas 10 a 12) que será utilizada para mapear e colorir os estados a partir do número de casos [23]. Entre as linhas 14 e 19 acrescentamos os limites dos estados [24], com a chamada de duas funções para estilização e iteração de cada polígono (Ou limite geográfico de cada estado). Na função **estilização** (linhas 21-32), acessamos o número de casos por estado como já demonstrado no Código Fonte 2.6 e atribuímos a respectiva cor utilizando o mapa linear previamente definido. Já a função **ParaCadaEstado** (linhas 33-38) acrescenta o tratamento de eventos *Leaflet* [25] aos polígonos, neste caso utilizamos o evento *click* para informar ao usuário o valor exato do número de casos de um estado (linhas 40-50). Por fim, elaboramos um *Leaflet Control* [26] para representar a legenda do mapa (linhas 52-76).

```
1       let centro = [-14, -54],zoom = 4,
2       Mapa = L.map('divMapa', { zoomControl:false }).setView(centro, zoom
        );
3
4       L.tileLayer(
5       'https://cartodb-basemaps-{s}.global.ssl.fastly.net/light_all/{z}/{
        x}/{y}.png',
6       {maxZoom: 18, attribution: '&copy; <a href="http://www.
        openstreetmap.org/copyright">OpenStreetMap</a>'}
7       ).addTo(Mapa);
8
9       let date = dtgFormat.parse('10/05/2020'),
10      escalaDeCores = d3.scale.linear()
11      .domain([0,1000,5000,10000,15000,25000,40000])
12      .range(['#edf8fb','#ccece6','#99d8c9','#66c2a4','#41ae76','#238b45'
        ,'#005824']);
13
14      L.geoJson(
15          Estados, {
16              style: estilizacao,
17              onEachFeature: ParaCadaEstado,
```

[22] https://www.openstreetmap.org/

[23] Escala de cores disponível em: https://colorbrewer2.org/#type=sequential&scheme=BuGn&n=7

[24] **Estados** é uma variável previamente definida, contendo os limites geográficos dos estados brasileiros em *GeoJson*, elaborada a partir do projeto Geodata BR - Brasil, disponível em: https://github.com/tbrugz/geodata-br

[25] Outros eventos são descritos na documentação oficial, disponível em: https://leafletjs.com/reference-1.6.0.html#map-event

[26] Conforme descrito em: https://leafletjs.com/reference-1.6.0.html#control

```
18          }
19      ).addTo(Mapa);
20
21      function estilizacao(feature){
22          var key = 'UF:'+feature.properties.UF+', data:'+date,
23          Ncasos = groupCasos_DataUF.all().filter(function(i){ return i.
    key == key })[0].value;
24          return {
25              weight: 2,
26              opacity: 1,
27              color: '#242424',
28              dashArray: '3',
29              fillOpacity: 1,
30              fillColor: escalaDeCores(Ncasos)
31          };
32      }
33      function ParaCadaEstado(feature, layer) {
34          layer._leaflet_id = feature.properties.UF;
35              layer.on({
36                  click: clickEstado
37                  // Outros Eventos_Leaflet: ...,
38          });}
39
40      function clickEstado(e) {
41          let layer =  e.target;
42          var cod = layer.feature.properties.UF;
43          var key = 'UF:'+cod+', data:'+date,
44          Ncasos = groupCasos_DataUF.all()
45              .filter(
46                  function(i){
47                      return i.key == key
48                  })[0].value;
49          alert('Estado: '+cod+';\nN de Casos em '+date+': '+Ncasos);
50      }
51
52      let legenda = L.control({position: 'bottomright'});
53
54      legenda.onAdd = function (map) {
55        var title = 'Num de Casos';
56          var format = d3.format("s");
57          var cores = escalaDeCores.range();
58          let div = L.DomUtil.create('div', 'legenda'),
59          labels = [],
60          n = cores.length,
61          from, to;
62          labels.push(title);
63          for (let i = 0; i < n; i++) {
64              let c = cores[i];
65              let fromto = escalaDeCores.domain();
66              var v1 = format(d3.round(fromto[i],1)),
67              v2 = d3.round(fromto[i+1],1);
68              labels.push(
69              '<i style="background:' + cores[i] + '"></i> ' +
70              v1 + (v2 ? ' &ndash; ' + format(v2) : '+'));
71          }
```

```
72          div.innerHTML = labels.join('<br>');
73          return div;
74      }
75
76      legenda.addTo(Mapa);
```

Código Fonte 2.7. Construção de um mapa temático com o número de casos da COVID19 por estado em 10/05/2020.

O Código Fonte 2.8 demonstra a construção de dois gráficos interativos e interconectados com *D3*, *DC* e *Crossfilter*. O gráfico 1 representa o número total de casos por estado em barras horizontais e o gráfico 2 o número de novos casos por dia em barras verticais. Inicialmente definimos as dimensões e grupos *Crossfilter* que serão utilizados nestes gráficos (linhas 1 a 8), que foram inicializados e atribuídos às respectivas *divs HTML* (linhas 13 e 14). A construção destes gráficos, presente nas linhas 16 a 33 e 34 a 47, seguem os exemplos e documentações descritos na Subsubseção 2.3.2.3, onde é possível compreender cada elemento estabelecido. Destacam-se, portanto, as adaptações realizadas para o público brasileiro: Formatação dos números e data entre as linhas 54 e 72. Além disso, entre as linhas 49 e 52 realizamos um filtro na dimensão data para que apenas os valores até o dia 10 de maio sejam considerados e entre as linhas 74 e 78 demonstramos como rotacionar a legenda para o eixo X do gráfico 2. Com este trecho simples de código, as bibliotecas *DC* e *Crossfilter* se encarregam dos filtros e demais interações entre gráficos[27].

```
1   // Dimensao e grupo para o grafico 1:
2   var dim_UF = facts.dimension(function(d){return d.uf;}),
3   Casos_UF = dim_UF.group()
4       .reduceSum( function(d){return d.casosNovos;});
5   // Dimensao e grupo para o grafico 2:
6   var dim_Data =  facts.dimension(function(d){return d.data;}),
7   Casos_Data = dim_Data.group()
8       .reduceSum(function(d){return d.casosNovos;});
9   // Janela temporal para grafico 2:
10  var maxDate = dtgFormat.parse("10/05/2020"),
11  minDate = d3.time.day.offset(maxDate, -20);
12  // Inicializacao dos graficos:
13  var grafico1 = new dc.rowChart("#divGrafico1");
14  var grafico2 = new dc.barChart("#divGrafico2");
15  // Contrucao dos graficos:
16  grafico1
17      .width(900)
18      .height(800)
19        .margins({left: 100, top: 10, right: 50, bottom: 60})
20      .renderLabel(true)
21          .renderTitleLabel(true)
22      .labelOffsetX(-35)
23      .elasticX(true)
24      .dimension(dim_UF)
25          .group(Casos_UF)
26      .title(function(d){
27        if(d.value == 0)
```

[27] Outras informações detalhadas podem ser encontradas em https://github.com/square/crossfilter/wiki/API-Reference

```
                return '-';
            return d.value.toLocaleString("pt-BR");
              })
              .colorAccessor(function (d){return d.value;})
              .colors(function(d){
                  return escalaDeCores(d);});
    grafico2
        .width(900)
        .height(500)
        .elasticY(true)
            .margins({left: 60, top: 30, right: 80, bottom: 60})
            .x(d3.time.scale().domain([minDate, maxDate]))
        .xUnits(d3.time.days)
        .centerBar(true)
        .brushOn(false)
        .renderHorizontalGridLines(true)
            .renderVerticalGridLines(true)
        .colors(['#e34a33'])
        .dimension(dim_Data)
        .group(Casos_Data);
    // Exemplo de filtro com dimensao crossfilter
    dim_Data.filter(function(d){
            if(d<=dtgFormat.parse("10/05/2020"))
            return d;
        });
    // Formatacao de numeros e data no grafico 2:
    var formatDay = d3.time.format("%d"),
    formatMonth = d3.time.format("%m");
    grafico2
        .title(function(d){
            var dia   = formatDay(d.key),
            mes = formatMonth(d.key);
            return ('('+dia+'/'+mes+'): '+d.value+'');
        })
        .xAxis()
            .ticks(d3.time.days, 3)
            .tickFormat(function(d){
                var dia = formatDay(d),
                mes = formatMonth(d);
                return (dia+'/'+mes);
            })
        .yAxis()
                .tickFormat(function(d){
                    return d.toLocaleString("pt-BR");
                });
    // Rotacao da legenda inferior - Eixo X - no grafico 2:
    grafico2
        .on('renderlet', function (chart) {
            chart.selectAll('g.x text')
            .attr('transform', 'translate(-20,10) rotate(-35)')
        });
    // Renderizacao dos graficos:
    dc.renderAll();
```

Código Fonte 2.8. Construção de dois gráficos interligados e interativos com *D3 DC* e *Crossfilter*.

Os três trechos de código apresentados são subsequentes e inter-relacionados, exemplificando a utilização de cada biblioteca no fluxo de trabalho para construção de visualizações interativas da COVID-19. Estes trechos, associados aos respectivos elementos HTML e sua estilização[28], representam versões simplificadas das visualizações construídas neste trabalho, seus resultados são ilustrados na Figura 2.6.

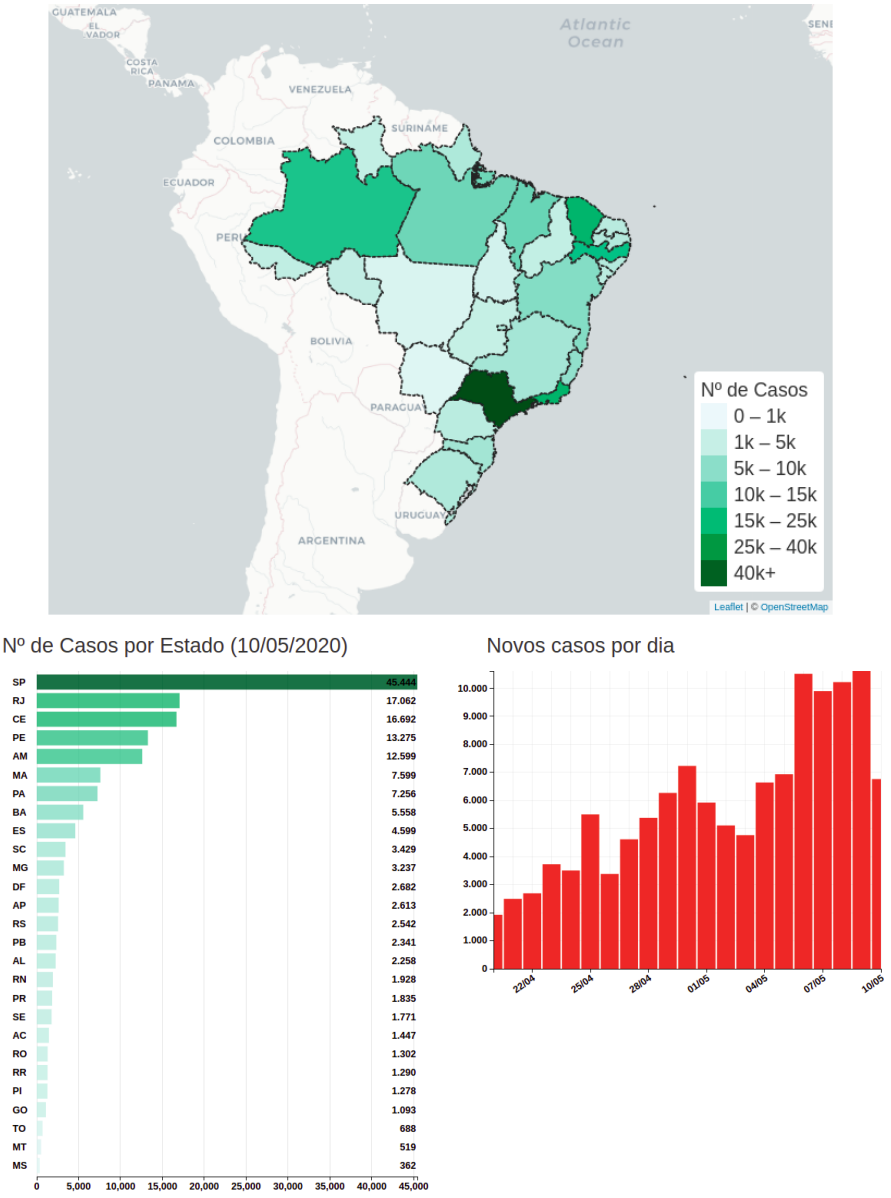


Figura 2.6. Ilustração das visualizações simplificadas da COVID-19 construída a partir dos Códigos 2.6, 2.7 e 2.8, para demonstração das bibliotecas utilizadas e suas contribuições.

[28] Disponíveis aqui: https://drive.google.com/drive/folders/11a--lk-HBLvhf5dtkczDNZBzfxbXWCn5?usp=sharing

2.4. Conclusão

Este trabalho apresentou técnicas eficientes para aquisição, tratamento e visualizações interativas de dados, com ênfase nos dados disponibilizados pelo portal da COVID-19 do Ministério da Saúde. Assim, fornecemos subsídios para a utilização e construção de ferramentas práticas e eficazes que podem auxiliar a sociedade e seus governantes em problemáticas como as geradas pela pandemia da COVID-19.

Epidemias são diferentes de doenças isoladas, sua abrangência envolve diversos fatores e diferentes atores, uma vez que não se relacionada apenas à saúde das pessoas, mas também à política, economia, cultura, educação e outros aspectos de vida. Atualmente, métodos e técnicas como as apresentadas neste trabalho tornam-se cada vez mais importantes, tendo em vista que a quantidade de dados disponíveis é gigantesca. O conhecimento e domínio de tais técnicas podem auxiliar diferentes profissionais no desenvolvimento de soluções computacionais tempestivas e satisfatórias aos mais diversos problemas enfrentados no cotidiano da vida em sociedade.

Referências

[Cairo 2015] Cairo, A. (2015). *The Truthful Art: Data, Charts, and Maps for Communication*. New Riders.

[Datasus 2020] Datasus (2020). Corona virus brasil. `https://covid.saude.gov.br`. Acessado em 2020-05-09.

[Doyle 2020] Doyle, S. (2020). Migrant workers falling through cracks in health care coverage. *CMAJ*, 192(28):E819–E820.

[Government 2020] Government, N. Y. S. (2020). New york state covid-19 data. `https://data.ny.gov`. Acessado em 2020-05-09.

[Grus 2019] Grus, J. (2019). *Data Science do zero: Primeiras regras com o Python*. Alta Books.

[Knaflic 2015] Knaflic, C. N. (2015). *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley.

[McKinney 2012] McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. "O'Reilly Media, Inc.".

[Meeks 2015] Meeks, E. (2015). *D3.js in Action*. Manning Publications.

[Mitchell 2019] Mitchell, R. (2019). *Web Scraping com Python – 2ª edição: Coletando mais dados da web moderna*. NOVATEC.

[Reitz 2020] Reitz, K. (2020). requests: Python http for humans. *URL: http://python-requests. org*.

Capítulo 3

Utilização de Técnicas de *Data Augmentation* em Imagens: Teoria e Prática

Maíla Claro, Luis Vogado, Justino Santos and Rodrigo Veras

Abstract

Recent advances in the acquisition and processing of digital images have made it possible to use the data provided from medical images in new and revolutionary ways, especially when associated with artificial intelligence. However, there is still a certain deficiency in obtaining these image databases and certain computational techniques require a large volume of data to provide reliable solutions and the ability to generalize different inputs. It is in this context that the Data Augmentation techniques are inserted into images. These techniques are intended to generate new virtual data from real examples to provide more volume and support the development of more robust applications. In this mini-course, the main techniques of Data Augmentation in images will be approached in a theoretical and practical way, from the most traditional to those involving Deep Learning.

Resumo

Recentes avanços na aquisição e processamento de imagens digitais permitiram o uso eficiente e inovador do dados fornecidos por meio de bases de dados em aplicações de inteligência artificial. Entretanto, ainda há uma certa deficiência na obtenção dessas bases de imagens e determinadas técnicas computacionais necessitam de um grande volume de dados para fornecer soluções confiáveis e com a capacidade de generalizar diferentes entradas. É nesse contexto que se inserem as técnicas de Data Augmentation em imagens. Essas técnicas tem como intuito gerar novos dados virtuais a partir de exemplos reais para prover mais volume e dar mais suporte ao desenvolvimento de aplicações mais robustas. Neste minicurso serão abordadas de forma teórica e prática, as principais técnicas de Data Augmentation em imagens, desde as mais tradicionais até as que envolvem Deep Learning.

3.1. Introdução

O Processamento Digital de Imagens (PDI) não é uma tarefa simples, na realidade envolve um conjunto de tarefas interconectadas. Tudo se inicia com a captura de uma imagem, a qual, normalmente, corresponde à iluminação que é refletida na superfície dos objetos, realizada através e um sistema de aquisição. Após a captura por um processo de digitalização, uma imagem precisa ser representada de forma apropriada para tratamento computacional. Imagens podem ser representadas em duas ou mais dimensões [Acharya and Ray 2005].

A classificação de imagens é uma das grandes áreas de aplicação do Aprendizado de Máquina, a quantidade de exemplos rotulados disponíveis e os resultados estão relacionados, onde quanto maior a quantidade de exemplos, maior a capacidade de predição dos classificadores gerados. No entanto, em situações da vida real, possuir bases de dados com uma grande quantidade de exemplos rotulados nem sempre é uma tarefa fácil, e muitas vezes, custosa. Desse modo, podemos utilizar abordagens de *Data Augmentation* (DA), que é um conceito fundado por técnicas computacionais com o objetivo de aumentar a quantidade de exemplos rotulados em um conjunto de dados e assim, melhorar os resultados obtidos [Taylor and Nitschke 2018].

A classificação de imagens possui diversas aplicações, por exemplo, a análise de dados de satélites [Penatti et al. 2015], reconhecimento facial [Vargas et al. 2016], detecção de doenças ([Claro et al. 2020]), dentre outras funções. Porém, em aplicações como imagens aéreas e médicas, os dados são limitados e, dessa forma é evidente a necessidade de utilização de técnicas de DA, já que a classificação de imagens oferece uma boa quantidade de aplicações e utilidade ao ser humano.

3.1.1. Representação Computacional de Imagens

Para representar e manipular imagens em um computador é necessário definir um modelo matemático apropriado [Gomes and Velho 1997]. Como uma imagem é resultado de um estímulo de luz, é possível estabelecer um universo matemático no qual se possa definir modelos abstratos de imagens de forma que permitam sua representação discreta com o propósito de possibilitar uma codificação da mesma em um computador [Gomes and Velho 1997].

Considere um escala de cinza, tal como mostrado na Figura 3.1, para representar uma imagem monocromática, i e j são dois números inteiros tais que $1 \leq \mathrm{i} \leq m$ e $1 \leq \mathrm{j} \leq n$. *f(i, j)* é a função inteira tal que $1 \leq f(i, j) \leq X$, onde X indica o valor branco em uma escala de cinza. Nesta situação uma matriz F (Figura 3.1) é chamada de imagem digital [Batchelor and Waltz 2012].

Uma imagem digital é considerada como uma matriz no qual os índices de linhas e colunas identificam um ponto na imagem, sendo que o respectivo valor do elemento dessa matriz identifica a intensidade (brilho) da imagem naquele ponto. A esses elementos de uma matriz digital dar-se o nome de “pixels". Na Figura 3.2 vemos um exemplo de uma imagem digital [Gonzalez and Woods 2000].

Em PDI trabalhamos basicamente com três tipos de imagens: Imagem Colorida (três bandas de cores), Imagem em Escala de Cinza (uma banda de cor) e Imagem Binária

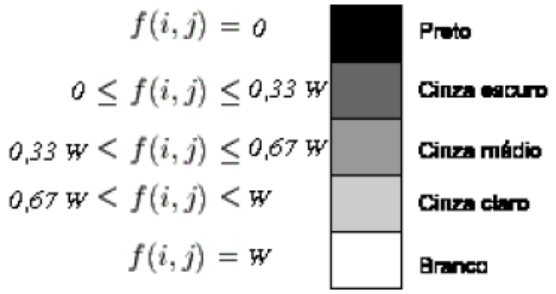


Figura 3.1. Demonstração de um mapeamento entre os valores armazenados na matriz F e as suas respectivas representações em tons de cinza [Batchelor and Waltz 2012]

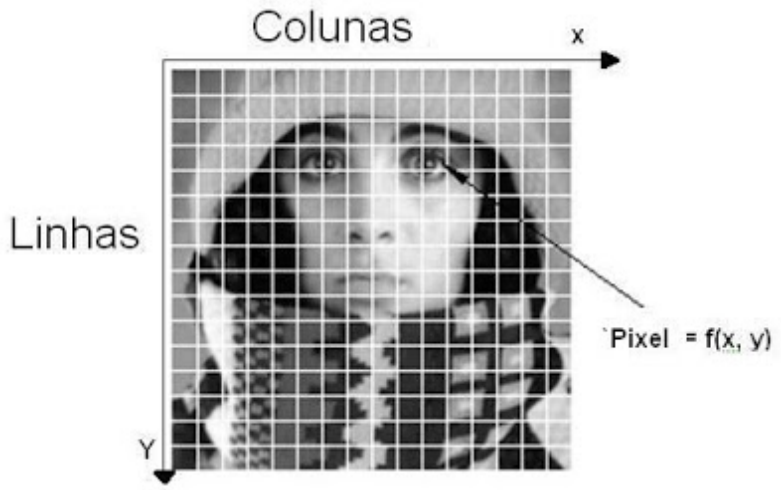


Figura 3.2. Exemplo de uma representação de Imagem Digital.

(Preto e Branco). No caso, da imagem em escala de cinza, a intensidade dos pontos da imagem, ou seja o valor do pixel, varia numa escala entre 0 e 255, onde o 0 (zero) indica a cor preta e o 255 a cor branca, tendo nesse intervalo o intensidade do cinza, como mostra a Figura 3.3.

Figura 3.3. Níveis de intensidade de Cinza.

Existe outro padrão para se trabalhar a intensidade do pixel, que é o intervalo entre 0 e 1, por exemplo: 0,15, 0,52, onde o 0 (zero) pode ser considerado como branco e o 1 (um) preto. É o chamado padrão com valores contínuos (fracionários), e no caso do intervalo 0 à 255 valores discretos (inteiros). A Figura 3.4 demonstra como seria a representação de uma imagem monocromática e os valores de seus respectivos pixels [Marques Filho and Neto 1999].

Já para imagens coloridas, ou seja, que possuem mais de uma banda de frequên-

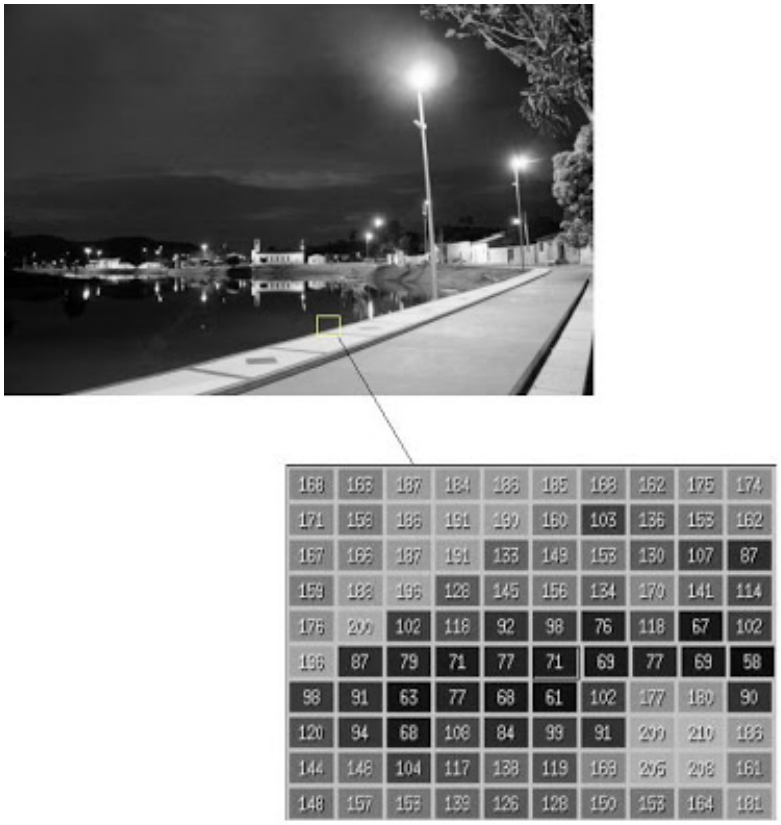


Figura 3.4. Imagem em monocromática e seus pixels.

cias, cada banda possui sua própria função de intensidade de brilho. Um exemplo de imagens coloridas, e é a mais utilizada, é o padrão RGB, apesar de existirem outros como por exemplo o HSI (*hue, saturation, intensity*). No modelo RGB a imagem possui três bandas que compõem as três cores primárias, Vermelho (*Red*), Verde (*Green*) e Azul (*Blue*).

Pode-se considerar, portanto, que uma imagem colorida é a composição de três imagens monocromáticas, denominadas, respectivamente, de banda vermelha (ou banda R), banda verde (ou banda G), e banda azul (ou banda B) e que a sobreposição destas três imagens, que individualmente são monocromáticas, compõe uma imagem colorida (ver Figura 3.5). E com a sobreposição/mistura de cada uma dessas cores e suas respectivas intensidades obtêm-se a formação de outras cores. Já no caso das imagens binárias, as opções de intensidade do pixel são apenas o preto ou branco, onde dependendo da imagem e do problema o preto pode representar o fundo da imagem e o branco os objetos ou vice-versa

Existem várias maneiras de lidar com complicações associadas a dados limitados no aprendizado de máquina. O aumento da imagem é uma técnica útil na construção de redes neurais convolucionais que podem aumentar o tamanho do conjunto de treinamento sem a aquisição de novas imagens. A ideia é simples, será preciso duplicar as imagens com algum tipo de variação para que o modelo possa aprender com mais exemplos. Idealmente, podemos aumentar a imagem de maneira a preservar os recursos essenciais para

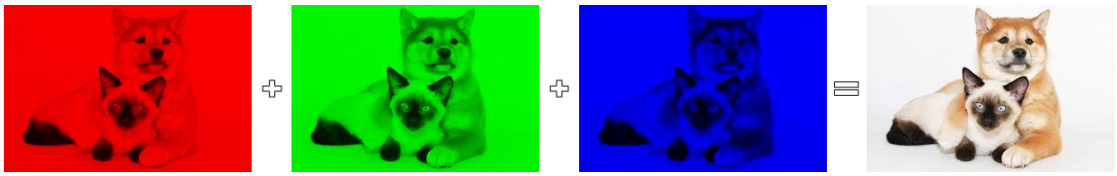


Figura 3.5. Imagens das Bandas R, G, B e RGB.

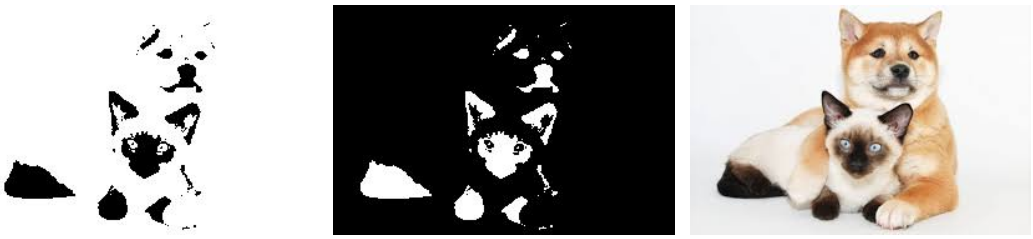


Figura 3.6. Exemplos de imagens em Binário com o plano de fundo branco, Binário com o plano de fundo preto e uma Imagem colorida.

fazer previsões, mas reorganiza os pixels o suficiente para adicionar algum ruído. O aumento será contraproducente se produzir imagens muito diferentes daquilo em que o modelo será testado, portanto, esse processo deve ser executado com cuidado.

3.2. Técnicas de *Data Augmentation*

Dados limitados são um grande obstáculo na aplicação de modelos de aprendizagem profunda, como redes neurais convolucionais. Frequentemente, classes desequilibradas podem ser um obstáculo adicional, embora possa haver dados suficientes para algumas classes, igualmente importantes, mas classes abaixo da amostra sofrerão com baixa precisão específica da classe. Esse fenômeno é intuitivo. Se o modelo aprender com alguns exemplos de uma determinada classe, é menos provável prever a invalidação da classe e os aplicativos de teste [Shorten and Khoshgoftaar 2019].

As primeiras demonstrações que mostram a eficácia das técnicas de *Data augmentation* vêm de transformações simples, como inversão horizontal, aumento de espaço de cores e corte aleatório. Essas transformações codificam muitas das invariâncias discutidas anteriormente que apresentam desafios para as tarefas de reconhecimento de imagem [Shorten and Khoshgoftaar 2019]. As técnicas abordadas nesta pesquisa são transformações geométricas, transformações de espaço de cores, filtros de núcleo, imagens misturadas, *random erasing*, aumento de espaço de recursos, treinamento *adversarial*, aprimoramento baseado em *Generative Adversarial Networks* (GAN) e transferência de estilo neural. Esta seção irá explicar como cada algoritmo de aumento funciona.

3.2.1. Transformações Geométricas

Esta seção descreve diferentes técnicas baseadas em transformações geométricas e muitas outras funções de processamento de imagem. Estes métodos de transformações geométricas são caracterizadas por sua facilidade de implementação. A compreensão dessas transformações fornecerá uma boa base para uma investigação mais aprofundada das téc-

nicas de aumento de dados.

Flipping

Flipping é uma técnica em que é feito um inversão na imagem original, onde essa inversão pode ser na horizontal ou na vertical. Inverter o eixo horizontal é muito mais comum do que inverter o eixo vertical. Esse aumento é um dos mais fáceis de implementar e provou ser útil em conjuntos de dados como CIFAR-10 e ImageNet. Em conjuntos de dados que envolvem reconhecimento de texto como MNIST [Deng 2012] ou SVHN, essa não é uma transformação de preservação de rótulo [Shorten and Khoshgoftaar 2019]. Como por exemplo na base MNIST em que é uma identificação dos números de 0 (zero) a 9 (nove), o número 6 (seis) poderia ser confundido com o número 9, depois de ser realizada uma inversão de horizontal e depois uma vertical. As Figuras 3.7 e 3.8 apresentam exemplos de imagens invertidas.

Figura 3.7. Exemplos da utilização da técnica Flip Horizontal.[1]

Figura 3.8. Exemplos da utilização da técnica Flip Vertical.[1]

Rotação

Os aumentos de rotação são feitos girando a imagem para a direita ou esquerda em um eixo entre 1º e 359º. A segurança dos aumentos de rotação é fortemente determinada pelo parâmetro do grau de rotação. Rotações leves, como entre 1 e 20 ou -1 a -20, podem ser úteis em tarefas de reconhecimento de dígitos, como MNIST, mas à medida que o grau de rotação aumenta, o rótulo dos dados não é mais preservado após a transformação [Shorten and Khoshgoftaar 2019]. A Figura 3.9 demonstra exemplos de imagens quadradas giradas em ângulo reto.

Translação

Mudar as imagens para a esquerda, direita, para cima ou para baixo pode ser uma transformação muito útil para evitar distorções posicionais nos dados. Por exemplo, se todas as imagens em um conjunto de dados estiverem centralizadas, o que é comum em conjuntos de dados de reconhecimento de face, isso exigiria que o modelo também fosse testado em imagens perfeitamente centralizadas. Como a imagem original é traduzida em

[1]Fonte: https://towardsdatascience.com/exploring-image-data-augmentation-with-keras-and-tensorflow-a8162d89b844

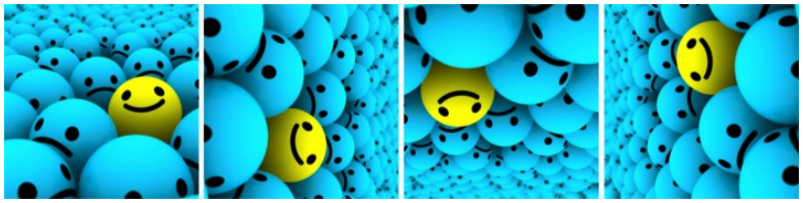


Figura 3.9. Da esquerda para à direita as imagens são giradas 90º graus no sentido horário em relação à anterior.[2]

uma direção, o espaço restante pode ser preenchido com um valor constante, como 0 ou 255, ou pode ser preenchido com ruído aleatório ou gaussiano. Esse preenchimento preserva as dimensões espaciais da imagem pós-aumento [Shorten and Khoshgoftaar 2019]. Esse método de aprimoramento é muito útil, pois a maioria dos objetos pode ser localizada em praticamente qualquer lugar da imagem. Isso força a rede neural convolucional a procurar em todos os lugares da imagem como é apresentado um exemplo na Figura 3.10.

Figura 3.10. Da esquerda, temos a imagem original, a imagem traduzida para a direita e a imagem traduzida para cima.[2]

Cropping ou Corte

Cortar imagens pode ser usado como uma etapa prática de processamento para dados de imagem com dimensões métricas de altura e largura cortando uma área central de cada imagem. Além disso, o corte aleatório também pode ser usado para fornecer um efeito muito semelhante às traduções. O contraste entre o corte aleatório e as traduções é que o corte reduzirá o tamanho da entrada, como (256, 256) para (224, 224), enquanto as traduções preservam as dimensões espaciais da imagem. Dependendo do limite de redução escolhido para o corte, isso pode não ser uma transformação de preservação do rótulo. A Figura 3.11 apresenta exemplos de cortes nas imagens.

Zoom / escala

Um zoom aleatório é obtido pelo argumento *zoom_range*. Um zoom menor que 1.0 amplia a imagem, enquanto um zoom maior que 1.0 diminui o zoom da imagem, como podemos observar na Figura 3.12.

Shear / cisalhamento

A transformação de cisalhamento inclina a forma da imagem. Isso é diferente da rotação no sentido de que, na transformação de cisalhamento, fixamos um eixo e esticamos a imagem em um determinado ângulo conhecido como ângulo de cisalhamento.

[2]Fonte: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Figura 3.11. Da esquerda para a direita, temos a imagem original, uma seção quadrada cortada da parte superior esquerda e, em seguida, uma seção quadrada cortada da parte inferior direita. As seções cortadas foram redimensionadas para o tamanho da imagem original.[2]

Figura 3.12. Exemplos da utilização da técnica Zoom.[1]

Isso cria um tipo de “alongamento"na imagem, que não é visto em rotação, como podemos visualizar na Figura 3.13. Temos a função *Shear_range* que especifica o ângulo da inclinação em graus.

Figura 3.13. Exemplos da utilização da técnica de Shear.[1]

3.2.2. Transformações Fotométricas

Além de transformações geométricas capazes de mudar a forma, tamanho e posição de elementos dentro imagem, outros tipos de transformações atuam sobre a composição pictórica delas. Mudanças aplicadas sobre o valor armazenado nos pixels e em alguns casos a substituição parcial de conteúdo são encontradas em abordagens como adição de ruído, mudanças no espaço de cor e processamentos de imagem, filtragem com kernel, combinação de imagens e *random erasing*.

Adição de ruído

Ruídos em imagens são caracterizados por valores aleatórios em pixels. Geralmente são provocados por fatores relacionados ao processo de obtenção da imagem. Em imagens analógicas, ruídos podem ser causados pelo próprio filme (aspereza da superfície), já em imagens digitais, variações de temperatura, falhas no sensor, entre outros podem causar esse efeito.

Inserir ruído em imagens digitais como técnica de DA foi testado por [Moreno-Barea et al. 2018] em nove base de dados. A adição de ruído em imagens pode ajudar

CNNs a aprender características mais robustas. A Figura 3.14 exemplifica dois tipos de ruídos comuns o gaussiano e o impulsivo.

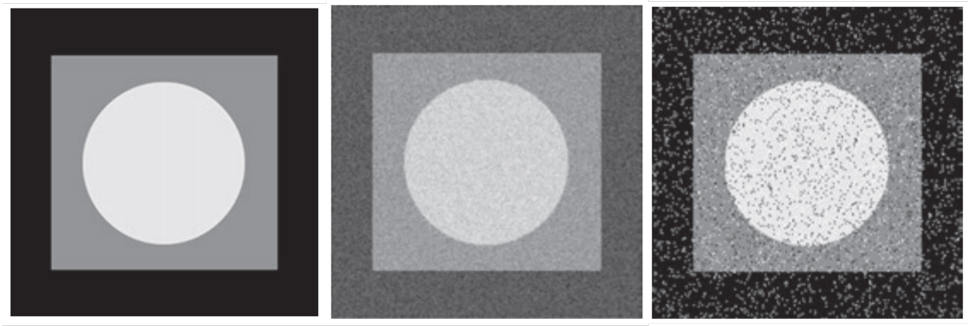


Figura 3.14. Ruídos gaussiano (b) e impulsivo (c) aplicados sobre imagem modelo (a). [Gonzalez and Woods 2000]

Transformações no espaço de cores

Os dados de uma imagem digital podem ser representados como um array de três dimensões (altura $\times$ largura $\times$ canais de cores). O aprimoramento no âmbito das cores é outra estratégia prática para implementação de *Data Augmentation*.

Os aprimoramentos de cores mais simples incluem o isolamento de um único canal de cor, como R, G ou B do modelo de cor RGB. Uma imagem pode ser rapidamente convertida em sua representação em um dos canais de cores, isolando essa matriz e adicionando duas matrizes povoadas com o valor zero nos outros canais de cores. Além disso, os valores RGB podem ser facilmente manipulados com operações simples de matriz para aumentar ou diminuir o brilho da imagem [Shorten and Khoshgoftaar 2019]. A Figura 3.15 ilustra essas técnicas.

Outros tipos de aumento podem derivar de ajustes de histograma de cores que descreve a imagem. A alteração dos valores de intensidade nesses histogramas resulta em alterações de iluminação e contraste, como as usadas nos aplicativos de edição de fotos.

É possível pensar em muitas formas de implementar transformações aplicáveis sobre o espaço de cores. Aumentar ou reduzir valor de todos os pixel em um valor fixo - simulando um ambiente com mais ou menos luz, iluminar cores específicas - mudança na temperatura de cor, a utilização de escala de cinza são só alguns exemplos.

Há de se observar que algumas aplicações são sensíveis à transformações no espaço de cores podendo gerar efeitos negativos sobre a tarefa. A utilização de escala de cinza, por exemplo, é uma opção capaz de gerar ganhos no processamento, mas que pode causar perdas na acurácia [Chatfield et al. 2014]. Outro caso, o escurecimento pode tornar objetos da imagem indetectáveis ou suas bordas se fundirem com a cena, dificultando uma possível tarefa de segmentação.

Filtragem com *kernel*

Também conhecida por filtragem espacial, consiste em deslisar uma janela sobre a imagem original realizando operações em cada posição ocupada, similar ao mecanismo

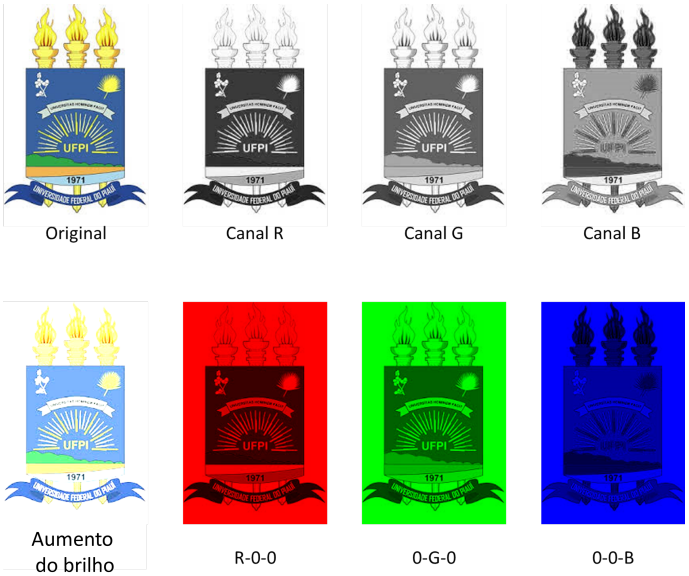


Figura 3.15. Transformações simples baseadas no espaço de cores.

de convolução. Estruturas internas de uma CNN já são capazes de realizar tais operações, portanto para alguns casos, não compensa aplicar esse tipo de aumento.

Duas aplicações de kernels muito comuns no processamento de imagens, são os filtros de borramento e de aguçamento (veja a Figura 3.16). Intuitivamente, realizar DA por meio de borramento das imagens de treino pode aumentar a resistência do algoritmo sobre imagens borradas por movimento no conjunto de testes. Adicionalmente aguçar imagens pode destacar detalhes sobre o objeto de interesse [Shorten and Khoshgoftaar 2019].

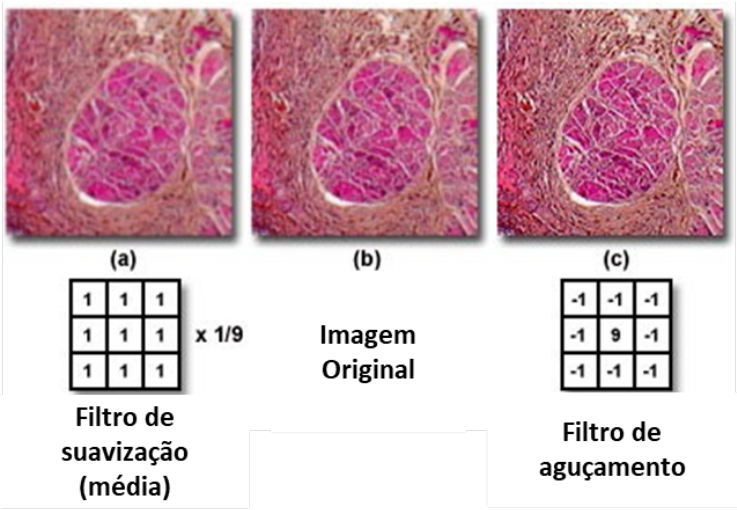


Figura 3.16. Filtros de borramento e aguçamento.[2]

[2]Fonte: https://static3.olympus-lifescience.com/data/olympusmicro/primer/digitalimaging/images/processintro/processintrofigure7.jpg

Uma técnica de DA avaliada por [Kang et al. 2017], denominada de *PatchShuffle Regularization* utiliza um mecanismo semelhante, no entanto em vez de operar sobre o nível de intensidade, em sua abordagem o filtro realiza trocas aleatórias nas posições dos pixels, com uma determinada taxa de probabilidade de mudança p. A Figura 3.17 ilustra o seu funcionamento.

Figura 3.17. ***PatchShuffle Regularization.*** ***a*****: aplicação do um filtro 2×2 sobre uma imagem 4×4.** ***b*****: Exemplos da aplicação sobre imagens reais com diferentes tamanhos de filtros. [Kang et al. 2017]**

Combinação de imagens

O aumento de dados por meio da combinação consiste em gerar imagens cujo conteúdo venha de um conjunto de outras imagens da base. O conteúdo gerado pode não fazer muito sentido com a realidade (Figura 3.18), divergindo dos exemplares que serão naturalmente fornecidos na etapa de teste.

Métodos lineares foram avaliados por [Inoue 2018]. Sua abordagem combinou pares de imagens por meio da média das intensidades dos pixels correspondentes (semelhante a Figura 3.18-b). Apesar de contraintuitivo, a inclusão de imagens de classes diferentes para serem combinadas (mesmo preservando apenas um dos rótulos) promoveu resultados melhores no conjunto de validação quando comparados com as técnicas tradicionais. Há de se observar que sua metodologia alternou épocas de treinamento com e sem imagens combinadas, e um ajuste fino com imagens puras finalizou o treinamento.

A (Figura 3.18-c) ilustra métodos não lineares de combinar imagens, esses métodos foram avaliados por [Summers and Dinneen 2019]. Utilizando estas abordagens eles obtiveram resultados de com performance melhor quando comparados aos métodos tradicionais.

Outra forma de combinar imagens também de maneira não linear, encontradas no trabalho de [Takahashi et al. 2019], consiste em recortes aleatórios de imagens que serão

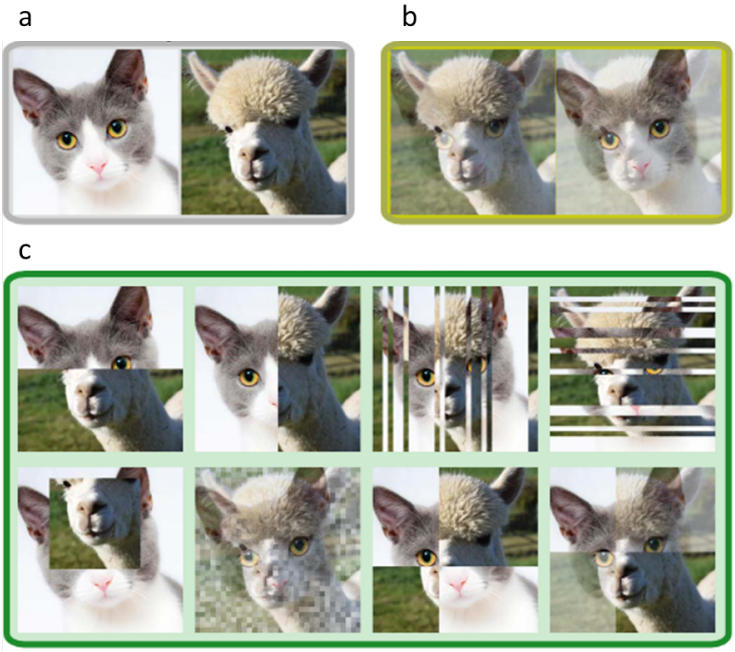


Figura 3.18. Combinação de imagens. *a*: imagens de entrada, *b*: combinação com método linear, *c*: combinação por métodos não lineares. [Summers and Dinneen 2019]

concatenados uns aos outros, a montagem decorrente desse processo forma uma nova imagem (Figura 3.19).

Random erasing

Essa técnica introduzida por [Zhong et al. 2020] consiste em “apagar"regiões aleatórias da imagem. Na prática uma parte da imagem tem seus pixels substituídos por um valor fixo ou um ruído conforme ilustrado na Figura 3.20.

A abordagem de remover uma parte da imagem ataca um problema conhecido com oclusão. Com essa técnica algoritmos de reconhecimento passam a se ater a detalhes gerais da imagem se tornando mais robustos sobre situações comuns onde algum objeto

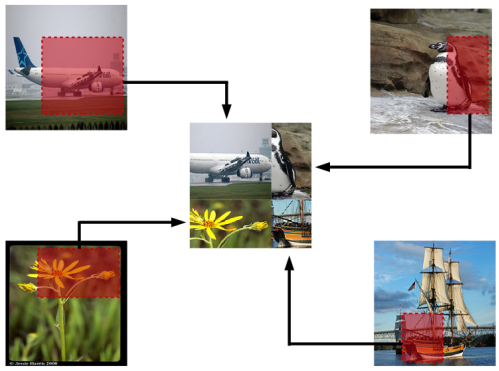


Figura 3.19. combinação com métodos não lineares. [Takahashi et al. 2019]

toma parcialmente a frente de outro a ser reconhecido. Por exemplo, uma CNN para reconhecimento de pessoas pode se ater apenas aos olhos ou ao rosto, treinar a rede com partes excluídas vai forçar a rede a dar importância a outras características.

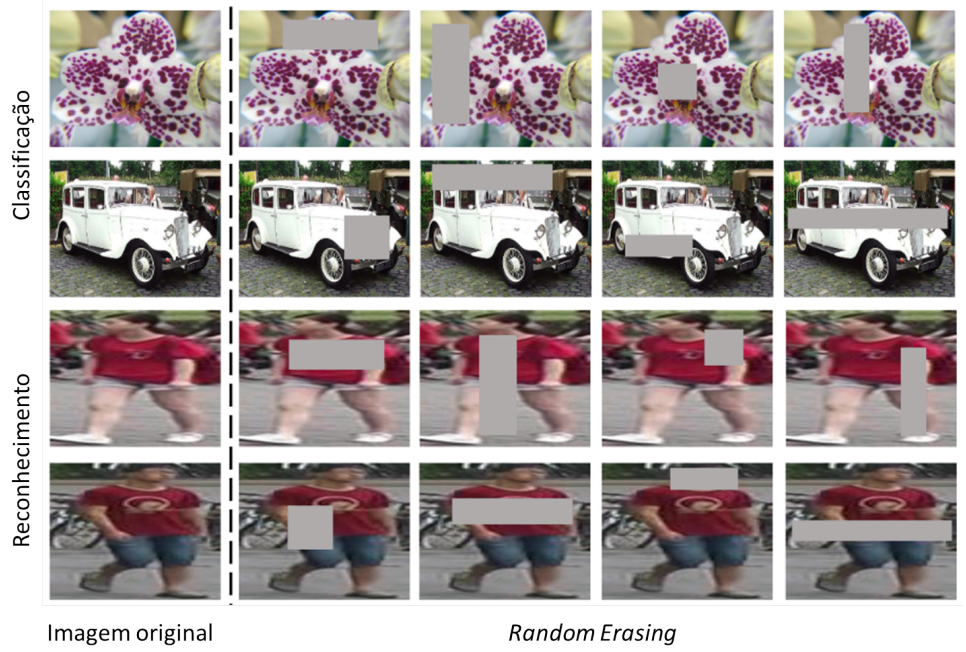


Figura 3.20. *Random erasing* [Zhong et al. 2020]

3.2.3. Transformações baseadas em *Deep Learning*

Feature Space Augmentation

As técnicas mais tradicionais utilizadas no aumento de dados são baseadas em imagens como entrada para as redes neurais convolucionais. No entanto, também é possível aplicar técnicas de aumento em um espaço de características gerado por camadas totalmente conectadas de uma CNN [DeVries and Taylor 2017].

Dentre as principais técnicas desse tipo de aumento, temos: *Synthetic Minority Oversampling Technique* (SMOTE) [Nitesh VC 2002] e *Autoencoders* [Pascal and Hugo 2010]. A SMOTE é uma técnica utilizada para gerenciar o desbalanceamento de classes em um determinado problema, aumentando a classe desbalanceada por meio da sintetização de novos exemplos a partir do algoritmo de agrupamento *K-nearest neighbor* (KNN) [Aha and Kibler 1991]. O funcionamento do SMOTE é iniciado com a escolha aleatória de um exemplo da classe desbalanceada. A partir do exemplo, os k vizinhos mais próximos são selecionados. Dentre eles, um dos vizinhos é escolhido também de forma aleatória e o novo exemplo é gerado a partir de um novo ponto entre o exemplo inicial e o seu vizinho selecionado. Esse procedimento é executado até que a classe esteja balanceada.

Os *Autoencoders* também são utilizados no aumento de dados para o espaço de características. Essa estrutura é dividida em duas partes, a primeira é o *encoder* que irá receber uma imagem como entrada e retorna um vetor de baixa dimensionalidade. Esse

vetor serve de entrada para o *decoder*, que irá reconstruir a imagem original a partir do vetor compactado pelo *encoder*. Sendo assim, o exemplo gerado não possuirá exatamente as mesmas características da original, formando uma imagem artificial. Na Figura 3.21 apresentamos um exemplo simples de *Autoencoder*.

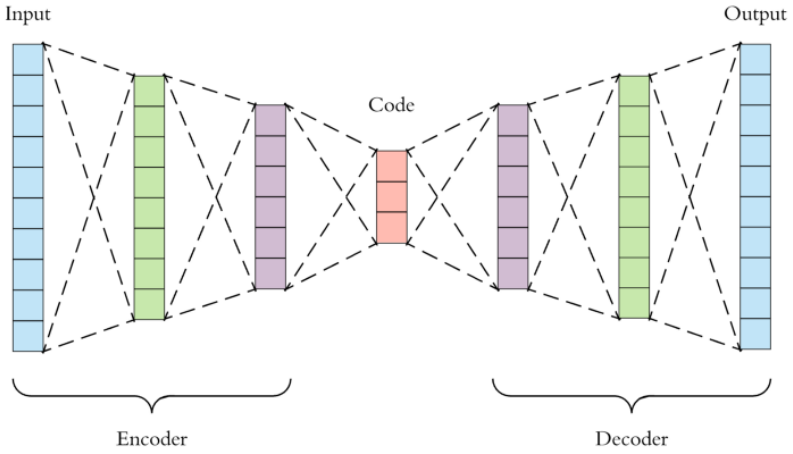


Figura 3.21. Exemplo de *Autoencoder*.[3]

Outra forma de aumento do espaço de características é a inserção ruídos, interpolações e extrapolação. No entanto, assim como as outras técnicas, a principal desvantagem desse tipo de aumento é a dificuldade na interpretação dos vetores. Já os *Autoencoders* para CNNs demandam um tempo excessivo para treinar devido a presença de inúmeros dados.

Treinamento Adversarial

Uma das soluções para buscas no espaço de características e para auxiliar no desenvolvimento de técnicas de aumento é o treinamento *Adversarial*. Essa técnica consiste na utilização de duas ou mais redes neurais que possuem diferentes objetivos. Um dos objetivos é realizar ataques como a inserção de ruídos nas imagens no intuito de desafiar completamente a intuição sobre como esses modelos representam imagens. Diante disso, esses ataques são realizados ao treinar uma rede rival com aumentos provenientes de outra rede [Moosavi-Dezfooli et al. 2015].

Um dos principais objetivos para utilizar esse tipo de aumento é a melhora proporcionada nos pontos fracos no limite de decisão aprendido. Sendo assim, esse tipo de treinamento também auxilia na descobertas de pontos fracos ou pertubações em modelos de *deep learning*. A eficácia do treinamento *Adversarial* na forma de busca por ruído ou aumento ainda é um conceito relativamente novo que não foi amplamente testado e compreendido.

No entanto não fica claro na literatura se essa técnica auxilia na redução do *overfitting*. Trabalhos futuros buscam expandir a relação entre resistência a ataques adversários e desempenho real em conjuntos de dados de teste.

[3]Fonte: `https://towardsdatascience.com/applied-deep-learning-part-3\-autoencoders-1c083af4d798`

Aumento com GANs

Outra técnica utilizada nos últimos anos para o aumento de dados são os modelos generativos. Esses modelos utilizam os dados da própria base de dados para gerar novas imagens com características similares à base de dados. Levando em consideração os ataques adversariais e a estrutura dos *Autoencoders*, foram propostas as *Generative Adversarial Networks* (GANs). Essas arquiteturas possuem duas redes adversárias, o gerador que gera novos exemplos a partir da base de dados e o discriminador que prediz se o exemplo gerado pode ser considerado real ou falso. O intuito do gerador é enganar o discriminador, ou seja, fazer com que os exemplos gerados estejam mais próximos possíveis de exemplos reais. A utilização dessas arquiteturas vem ganhando destaque ao longo dos anos, uma vez que seus resultados se mostraram surpreendentes.

No domínio das imagens, a GAN mais conhecida para aplicações é a *Deep Convolutional Generative Adversarial Network* (DCGAN) [Radford et al. 2015]. Essa arquitetura utiliza CNNs no lugar de redes neurais simples como em GANs mais simples. A partir de um ruído aleatório, as camadas deconvolucionais da DCGAN tendem a gerar exemplos correspondentes a base de dados e classe das imagens de entrada. O exemplo artificial servirá de entrada para o discriminador, que é nada mais que uma CNN comum com uma função de ativação em sua última camada.

Diferente dos *Autoencoders*, as DCGANs não precisam ser simétricas. Sendo assim, apenas a saída do gerador e do discriminador devem possuir tamanhos equivalentes. Na Figura 3.22 apresentamos um exemplo de DCGAN em uma aplicação da literatura.

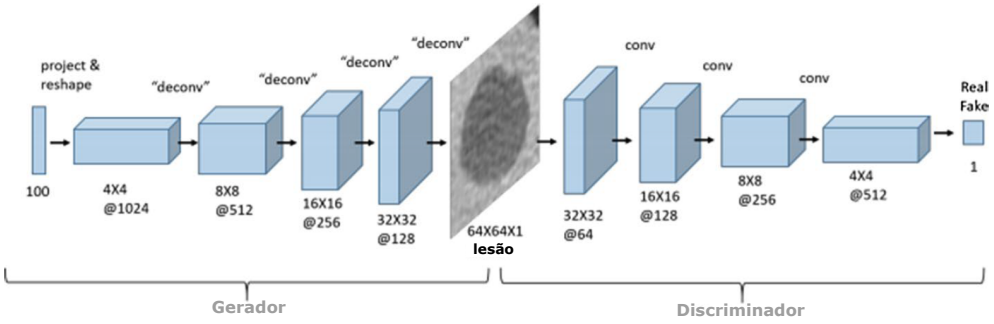


Figura 3.22. Exemplo de DCGAN aplicada a geração de imagens de lesão de fígado. [Frid-Adar et al. 2018]

Outros tipos de GANs também são empregados na literatura como diferentes finalidades, como as *Progressively Growing GANs* [Heljakka et al. 2018]. Essa arquitetura tem como principal objetivo trabalhar com imagens em alta resolução, partindo baixas resoluções para as mais altas. Outra arquitetura bastante utilizada é a CycleGAN [Zhu et al. 2017]. Essa GAN consegue transferir o domínio de uma base de imagens para outro. Isso permite que uma entrada de uma determinada classe seja remapeada para outra, fazendo com que o discriminador faça a predição de qual das classes pertence a imagem gerada. Após a predição, a imagem retorna para outro gerador que tentará retornar a mesma para sua classe original. Um segundo discriminador é então utilizado para predizer se essa imagem transformada é ou não da classe original, formando um ciclo.

Existe um grande potencial no uso de GANs para o aumento de dados. No entanto, elas ainda apresentam alguns desafios, como: dificuldade na obtenção de imagens com altas resoluções, instabilidade no treinamento e não convergência e a necessidade de grandes conjuntos de dados para treinamento.

Neural Style Transfer

Uma dentre as técnicas que melhor representam a capacidade de aprendizado das CNNs é a transferência de neural de estilo ou *Neural Style Transfer* [Gatys et al. 2015]. Essa técnica consiste na transferência de características intrínsecas de uma imagem para outra utilizando CNNs. Essa capacidade de transferir o estilo possibilita que essa técnica sirva como ferramenta para o aumento de dados. A transferência ocorre através da seleção de camadas convolucionais e da retro-propagação da *loss* através das épocas, permitindo que o conteúdo da imagem de entrada seja replicado para a imagem base.

A transferência de estilo permite que a imagem originada do processamento possua características de cor, textura e iluminação da imagem de estilo sem que o conteúdo original seja completamente descartado. Isso ocorre devido a possibilidade das CNNs conseguirem aprender tanto características mais gerais quanto mais específicas de uma imagem. Na Figura 3.23 demonstramos um exemplo de transferência de estilo.

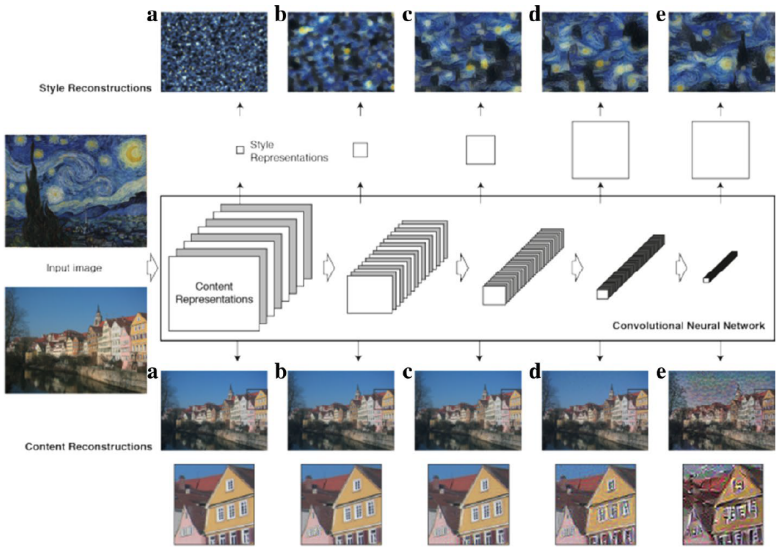


Figura 3.23. Imagens utilizadas para a realização da transferência de estilo e seus resultados. [Shorten and Khoshgoftaar 2019]

Uma das desvantagens desse aumento é o esforço necessário para selecionar os diferentes estilos para serem transferidos. Além disso, se a base utilizada para definir os estilos for pequena, pode ocorrer a inserção de um *bias* nos resultados. Outra desvantagem é o recurso computacional exigido para processar e gerar todas as imagens necessárias para o aumento, mesmo sendo uma alternativa que pode proporcionar uma melhor generalização para CNNs, a utilização de outras técnicas de aumento podem ser mais eficientes. Trabalhos recentes apresentaram formas mais rápidas de transferir o estilo para imagens de conteúdo. No entanto, são limitados pois os estilos disponíveis são predefinidos.

3.3. Aplicações de *Data Augmentation*

O aumento de dados é uma técnica popular amplamente usada para aprimorar o treinamento de redes neurais convolucionais. Embora muitos de seus benefícios sejam bem conhecidos por pesquisadores e profissionais de aprendizagem profunda, seus efeitos implícitos de regularização, em comparação com as técnicas populares de regularização explícita, como *weight decay* e *dropout*, permanecem praticamente sem estudo. De fato, as redes neurais convolucionais para classificação de objetos de imagem geralmente são treinadas com aumento de dados e regularização explícita, assumindo que os benefícios de todas as técnicas sejam complementares [Hernández-García and König 2018].

3.3.1. Estado da arte que aplicam *Data Augmentation*

Diversas técnicas de DA foram propostas na literatura, pois obter grandes quantidades de dados é uma tarefa difícil e de extrema importância para se obter bons resultados com os algoritmos de aprendizagem de máquina. Portanto, alguns trabalhos relacionados serão analisados e detalhados nesta seção.

Existem algumas técnicas de transformações de imagens tradicionais que são mais conhecidas e utilizadas, como vimos na Seção 3.2. No entanto, tem-se várias propostas de variações desses métodos e as novas abordagens utilizadas são os métodos gerativos, como as *Generative Adversarial Networks* (GANs).

Várias aplicações podem ser beneficiadas com as técnicas de DA. A visão computacional, por exemplo, se baseia em detectar objetos em imagens e é melhor utilizada com uma grande quantidade de dados. Dessa forma, diversos trabalhos utilizaram técnicas de DA em seus conjuntos de dados, como os trabalhos apresentados a seguir.

[Claro et al. 2020] aplicaram técnicas de DA para detecção de Leucemia Aguda em imagens de lâminas de sangue. Neste trabalho, os autores desenvolveram CNNs e aplicaram em um conjunto de imagens heterogêneas, utilizando rotação, translação, zoom, *Flipping* e *shear* como técnicas para aumento de dados. O resultado obtido foi de 97,18% de acurácia.

No trabalho de [Zhang et al. 2019], os autores utilizaram DA com três tipos de métodos: rotação de imagem, correção *gamma* e *noise injection* ou injeção de ruído. Nesse trabalho foi utilizado uma CNN como classificador e o *dataset* utilizado era composto por imagens para classificação de frutas. O melhor resultado encontrado foi 94,94%, sendo aproximadamente 5% a mais que os resultados encontrados na literatura.

Os autores [Taylor and Nitschke 2017] propuseram um estudo comparativo entre os esquemas populares de DA para indicar qual é a técnica mais indicada para tal base de dados. Foram testadas técnicas geométricas (ex. corte, rotações) e fotométricas (ex. *color jittering, fancy* PCA) sendo avaliadas por uma CNN. Os resultados utilizando 4-*fold cross validation* foram reportados com as acurácia top-1 e top-5, indicando que a técnica de corte aumentou os resultados obtidos pela CNN.

A proposta dos autores [Shijie et al. 2017] era aplicar diversas técnicas de DA e após a geração da base aumentada, classificar os dados utilizando a CNN Alexnet e nesse caso, considerou *subsets* das bases de dados CIFAR-10 e da *ImageNet* com 10 classes. As técnicas de DA utilizadas foram: GAN/WGAN, corte, rotação *flipping, shifting*, PCA

Tabela 3.1. Resumo dos Trabalhos Relacionados.

Título	Autores e Ano	Base de Dados	Técnica de DA
Convolution Neural Network Models for Acute Leukemia Diagnosis	[Claro et al. 2020]	Imagens de Lâminas de Sangue	Rotação, Translação, Zoom, *Flipping* e *Shear*
Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation	[Zhang et al. 2019]	Classificação de Frutas	Rotação, Correção de *Gamma* e *Noise injection*
Improving Deep Learning using Generic Data Augmentation	[Taylor and Nitschke 2017]	Caltech 101; Pascal VOC	Comparação manual de técnicas fotométricas e geométricas
Research on Data Augmentation for Image Classification Based on Convolutional Neural Networks	[Shijie et al. 2017]	Subsets da CIFAR-10 e ImageNet	Comparação DA tradicional e GANs
Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification	[Frid-Adar et al. 2018]	Tomografia do Fígado	Comparação DA tradicional e GANs para balancear *datasets*

jittering, color jittering, noise, e combinações desses métodos. Os autores concluíram que as técnicas de corte, *flipping*, WGAN e rotação obtiveram melhores resultados em relação às outras técnicas e que combinações de métodos de DA podem ajudar a se obter melhores resultados que os individuais.

Já em [Frid-Adar et al. 2018], os autores propuseram a utilização de métodos de DA tradicionais e a aplicação das GANs para gerar dados sintéticos. Os testes foram realizados com uma base de dados limitadas de tomografia computacional composta por 182 imagens de lesões no fígado. Os resultados mostraram uma melhora significativa. Com os métodos clássicos de DA obtiveram 78,6% de sensibilidade e 88,4% de especificidade. Com a adição dos dados gerados pelas GANs, foram obtidos 85,7% de sensibilidade e 92,4% de especificidade.

3.4. Estudo de Caso

A utilização de GAN's para o aumento de dados vem aumentando com o passar dos anos. Esse fato é justificado pelo aumento do poder computacional que proporciona o treinamento dessas arquiteturas. No entanto, a concepção das GAN's ainda é de difícil entendimento por parte dos estudantes. Tendo em vista essa problemática, nesta seção iremos demonstrar o funcionamento de GAN's por meio da implementação de uma DCGAN com o gerador, discriminador e o processo de treinamento e geração das imagens artificiais.

3.4.1. Base de dados

No problema abordado neste capítulo, apresentamos a base MNIST [Deng 2012], uma das bases mais populares encontrada em diversos experimentos na literatura.. Essa base de dados possui imagens contendo dígitos escritos a mão e conta com cerca de 70.000 imagens para treino e teste. A resolução das imagens de entrada é de 28×28. Na Figura 3.24, apresentamos alguns exemplos da base utilizada no experimentos.

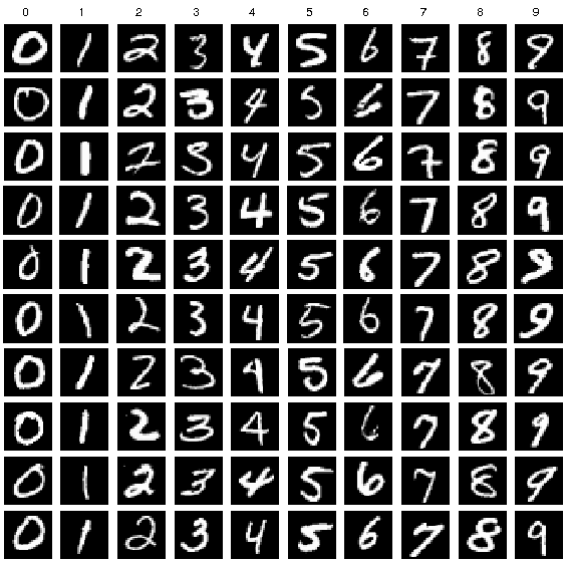


Figura 3.24. Exemplos de imagens provenientes da base de dados MNIST. [Deng 2012]

```
import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from keras import optimizers

batch_size = 64
lr = 0.0003
epochs = 50

(x_treino, _), (x_teste, _) = keras.datasets.mnist.load_data()

all_digits = np.concatenate([x_treino, x_teste])

all_digits = all_digits.astype('float32') / 255

all_digits = np.reshape(all_digits, (-1, 28, 28, 1))

dataset = tf.data.Dataset.from_tensor_slices(all_digits)
dataset = dataset.shuffle(buffer_size=1024).batch(batch_size).prefetch
    (32)
```

3.4.2. Construíndo a GAN

Para construirmos a GAN proposta neste estudo, precisamos definir a arquitetura do gerador e do discriminador. Sendo assim, como entrada, o gerador deve receber um ruído de tamanho 128 que servirá como base para a geração da imagem artificial. A estrutura da arquitetura com os tamanhos dos filtros convolucionais é apresentada no código abaixo.

```

def make_generator_model():

    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(128,))
    )
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
    padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
    padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding
    ='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model

generator = make_generator_model()
generator.summary()
```

```

def make_discriminator_model():

    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                    input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'
    ))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))
```

```
    return model

discriminator = make_discriminator_model()

discriminator.summary()
```

3.4.3. Treinamento

```
class GAN(keras.Model):

    def __init__(self, discriminator, generator, latent_dim):
        super(GAN, self).__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    def train_step(self, real_images):
        if isinstance(real_images, tuple):
            real_images = real_images[0]
        batch_size = tf.shape(real_images)[0]
        random_latent_vectors = tf.random.normal(shape=(batch_size,
    self.latent_dim))

        generated_images = self.generator(random_latent_vectors)

        combined_images = tf.concat([generated_images, real_images],
    axis=0)

        labels = tf.concat(
            [tf.ones((batch_size, 1)), tf.zeros((batch_size, 1))], axis
    =0
        )

        labels += 0.05 * tf.random.uniform(tf.shape(labels))

        # Treinando o discriminador
        with tf.GradientTape() as tape:
            predictions = self.discriminator(combined_images)
            d_loss = self.loss_fn(labels, predictions)
        grads = tape.gradient(d_loss, self.discriminator.
    trainable_weights)
        self.d_optimizer.apply_gradients(
            zip(grads, self.discriminator.trainable_weights)
        )

        random_latent_vectors = tf.random.normal(shape=(batch_size,
    self.latent_dim))
```

```
        misleading_labels = tf.zeros((batch_size, 1))

        with tf.GradientTape() as tape:
            predictions = self.discriminator(self.generator(random_latent_vectors))
            g_loss = self.loss_fn(misleading_labels, predictions)
        grads = tape.gradient(g_loss, self.generator.trainable_weights)
        self.g_optimizer.apply_gradients(zip(grads, self.generator.trainable_weights))
        return {"d_loss": d_loss, "g_loss": g_loss}
```

```
class GANMonitor(keras.callbacks.Callback):
    def __init__(self, num_img=3, latent_dim=128):
        self.num_img = num_img
        self.latent_dim = latent_dim

    def on_epoch_end(self, epoch, logs=None):
        random_latent_vectors = tf.random.normal(shape=(self.num_img, self.latent_dim))
        generated_images = self.model.generator(random_latent_vectors)
        generated_images *= 255
        generated_images.numpy()
        for i in range(self.num_img):
            img = keras.preprocessing.image.array_to_img(generated_images[i])
            img.save("generated_img_{i}_{epoch}.png".format(i=i, epoch=epoch))

gan = GAN(discriminator=discriminator, generator=generator, latent_dim=latent_dim)

gan.compile(
    d_optimizer = optimizers.Adam(learning_rate=lr),
    g_optimizer = optimizers.Adam(learning_rate=lr),
    loss_fn = keras.losses.BinaryCrossentropy(from_logits=True),
)

gan.fit(
    dataset, epochs=epochs, callbacks=[GANMonitor(num_img=3, latent_dim=latent_dim)]
)
```

Com o treinamento concluído, na Figura 3.25 apresentamos alguns resultados obtidos pela arquitetura proposta. Observamos que em alguns exemplos é notável que o discriminador não conseguiu distinguir todos os exemplos entre reais e falsos. No entanto, em outros conseguimos definir bem qual número está presente na imagem. A quantidade de épocas utilizadas no treino é de fundamental importância na obtenção dos resultados, sendo necessárias inúmeras épocas para treinar apropriadamente uma GAN.

3.5. Considerações Finais

Dos aprimoramentos discutidos, transformações geométricas, transformações de espaço de cores, filtros de kernel, imagens misturadas e *random erasing*, quase todas essas trans-

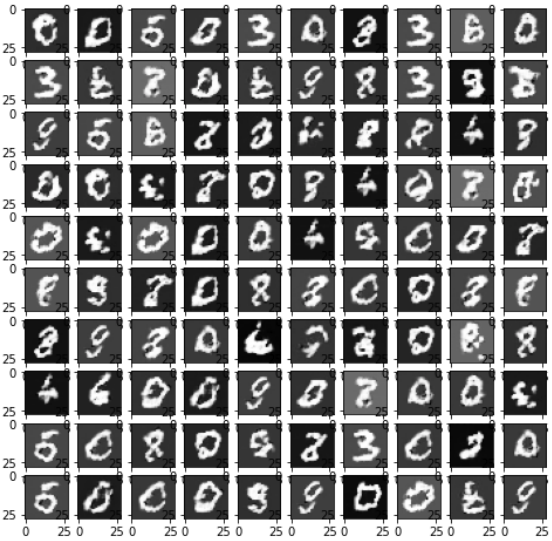


Figura 3.25. Resultado da geração a partir da DCGAN proposta.

formações também vêm com um parâmetro de magnitude de distorção associado. Com uma grande lista de potenciais aumentos e um espaço de magnitudes na maior parte contínuo, é fácil conceituar o enorme tamanho do espaço de pesquisa de aumento. A combinação de aprimoramentos como corte, inversão, mudança de cor e *random erasing* pode resultar em tamanhos de conjuntos de dados massivamente inflados. No entanto, isso não garante que seja vantajoso. Em domínios com dados muito limitados, isso pode resultar em *overfitting*.

Portanto, podemos concluir que DA é importante e pode ser considerado quando é preciso obter mais dados e tentar melhorar os resultados. Para isso, é necessário antes uma análise minuciosa sobre quais técnicas utilizar e ponderar, segundo as vantagens e desvantagens de cada método. Por exemplo, as técnicas de transformações de imagens são muito rápidas, mas é necessário analisar se tais efeitos não podem confundir o classificador. Já com as GANs, os resultados sintéticos podem ser excepcionais, mas necessita de muito processamento e tempo.

Referências

[Acharya and Ray 2005] Acharya, T. and Ray, A. K. (2005). *Image processing: principles and applications*. John Wiley & Sons.

[Aha and Kibler 1991] Aha, D. and Kibler, D. (1991). Instance-based learning algorithms. *Machine Learning*, 6:37–66.

[Batchelor and Waltz 2012] Batchelor, B. and Waltz, F. (2012). *Intelligent machine vision: techniques, implementations and applications*. Springer Science & Business Media.

[Chatfield et al. 2014] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

[Claro et al. 2020] Claro, M., Vogado, L., Veras, R., Santana, A., Tavares, J., Santos, J., and Machado, V. (2020). Convolution neural network models for acute leukemia diagnosis. In *The 27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*.

[Deng 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

[DeVries and Taylor 2017] DeVries, T. and Taylor, G. W. (2017). Dataset augmentation in feature space.

[Frid-Adar et al. 2018] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE.

[Gatys et al. 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style.

[Gomes and Velho 1997] Gomes, J. and Velho, L. (1997). *Image processing for computer graphics*. Springer Science & Business Media.

[Gonzalez and Woods 2000] Gonzalez, R. C. and Woods, R. E. (2000). *Processamento de imagens digitais*. Editora Blucher.

[Heljakka et al. 2018] Heljakka, A., Solin, A., and Kannala, J. (2018). Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision*, pages 22–38. Springer.

[Hernández-García and König 2018] Hernández-García, A. and König, P. (2018). Further advantages of data augmentation on convolutional neural networks. In *International Conference on Artificial Neural Networks*, pages 95–103. Springer.

[Inoue 2018] Inoue, H. (2018). Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

[Kang et al. 2017] Kang, G., Dong, X., Zheng, L., and Yang, Y. (2017). Patchshuffle regularization. *arXiv preprint arXiv:1707.07103*.

[Marques Filho and Neto 1999] Marques Filho, O. and Neto, H. V. (1999). *Processamento digital de imagens*. Brasport.

[Moosavi-Dezfooli et al. 2015] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2015). Deepfool: a simple and accurate method to fool deep neural networks.

[Moreno-Barea et al. 2018] Moreno-Barea, F. J., Strazzera, F., Jerez, J. M., Urda, D., and Franco, L. (2018). Forward noise adjustment scheme for data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 728–734. IEEE.

[Nitesh VC 2002] Nitesh VC, Kevin WB, L. O. K. W. (2002). Smote: synthetic minority over-sampling technique. In *Journal of Artificial Intelligence Research*, page 321–357.

[Pascal and Hugo 2010] Pascal, V. and Hugo, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408.

[Penatti et al. 2015] Penatti, O. A., Nogueira, K., and Dos Santos, J. A. (2015). Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–51.

[Radford et al. 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

[Shijie et al. 2017] Shijie, J., Ping, W., Peiyi, J., and Siping, H. (2017). Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese automation congress (CAC)*, pages 4165–4170. IEEE.

[Shorten and Khoshgoftaar 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60.

[Summers and Dinneen 2019] Summers, C. and Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270. IEEE.

[Takahashi et al. 2019] Takahashi, R., Matsubara, T., and Uehara, K. (2019). Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*.

[Taylor and Nitschke 2017] Taylor, L. and Nitschke, G. (2017). Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*.

[Taylor and Nitschke 2018] Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547. IEEE.

[Vargas et al. 2016] Vargas, A. C. G., Paes, A., and Vasconcelos, C. N. (2016). Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In *Proceedings of the xxix conference on graphics, patterns and images*, volume 1.

[Zhang et al. 2019] Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., and Wang, S.-H. (2019). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, 78(3):3613–3632.

[Zhong et al. 2020] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *AAAI*, pages 13001–13008.

[Zhu et al. 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

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.

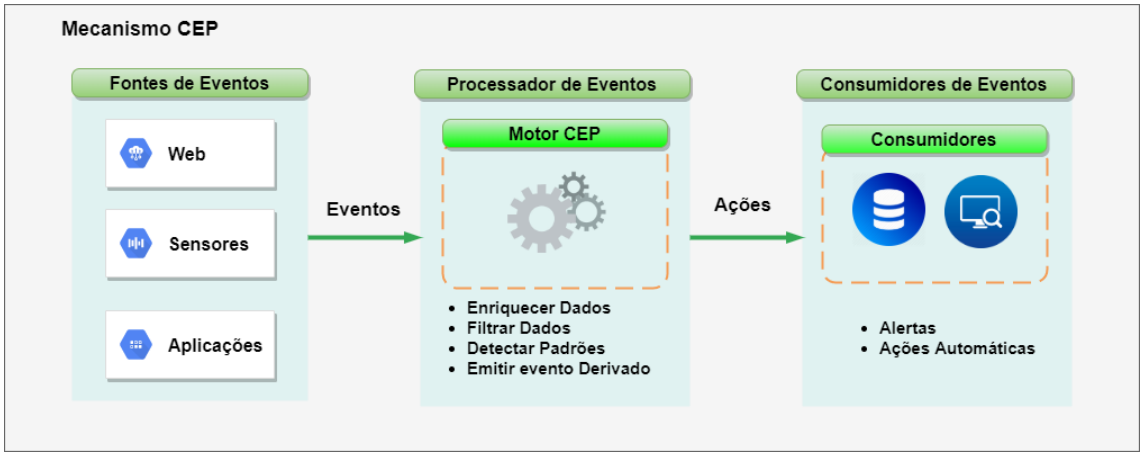


Figura 4.1. Visão geral de um mecanismo CEP.

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.

```
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,

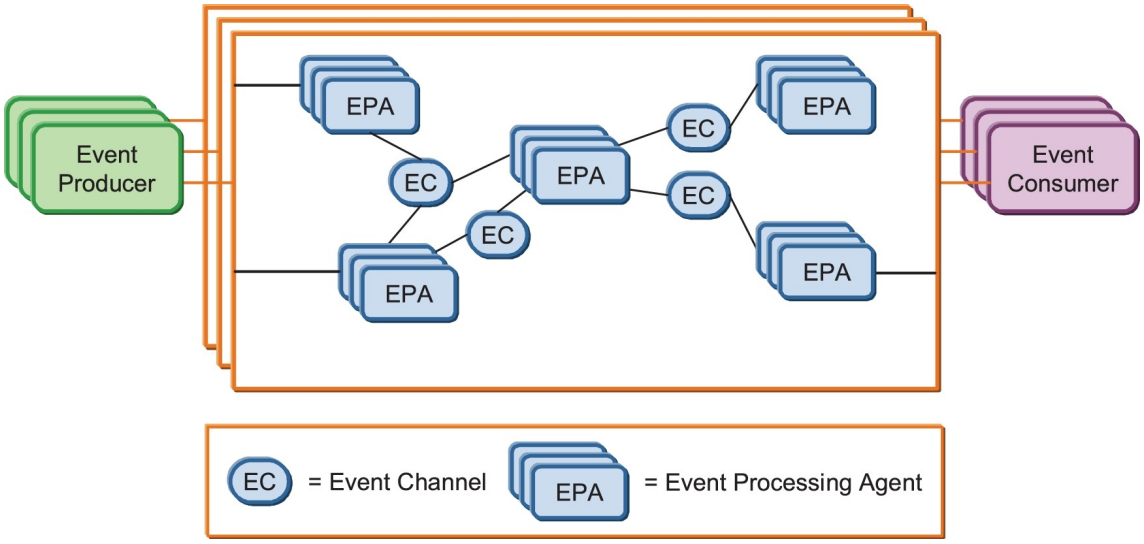


Figura 4.2. Rede de Processamento de Eventos [Etzion et al. 2011].

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.

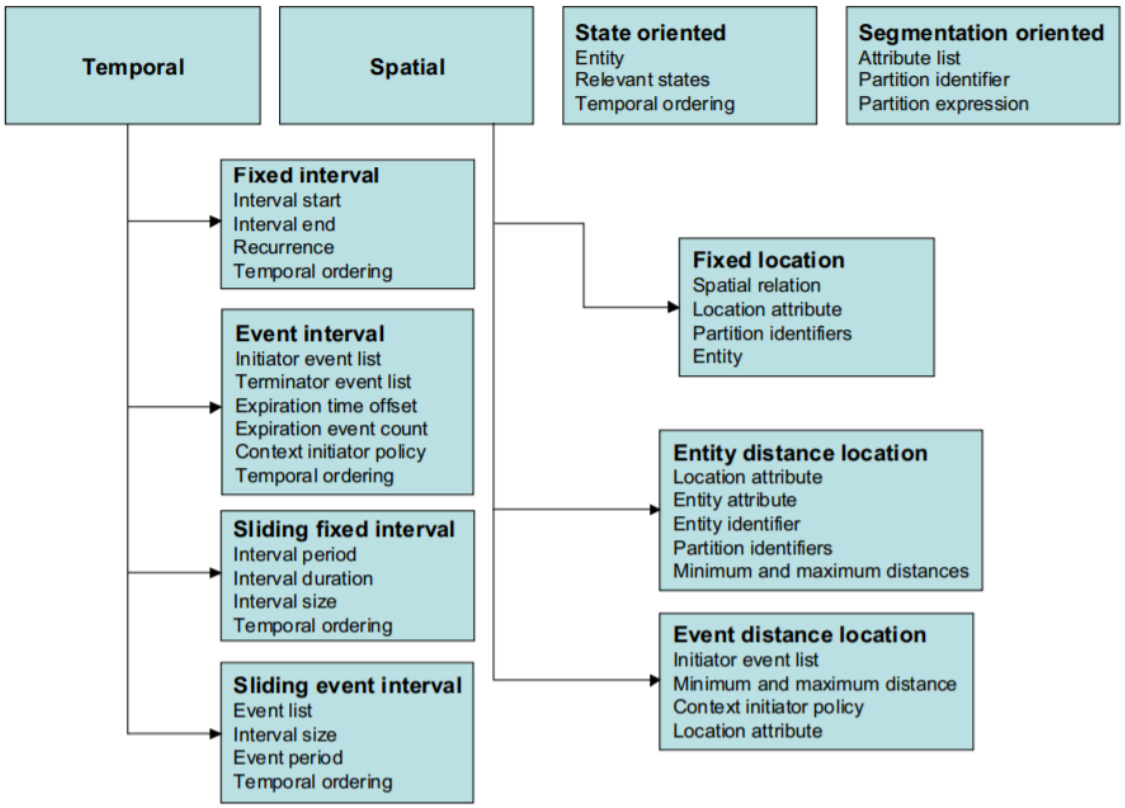


Figura 4.3. Dimensões de contexto [Etzion et al. 2011].

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.

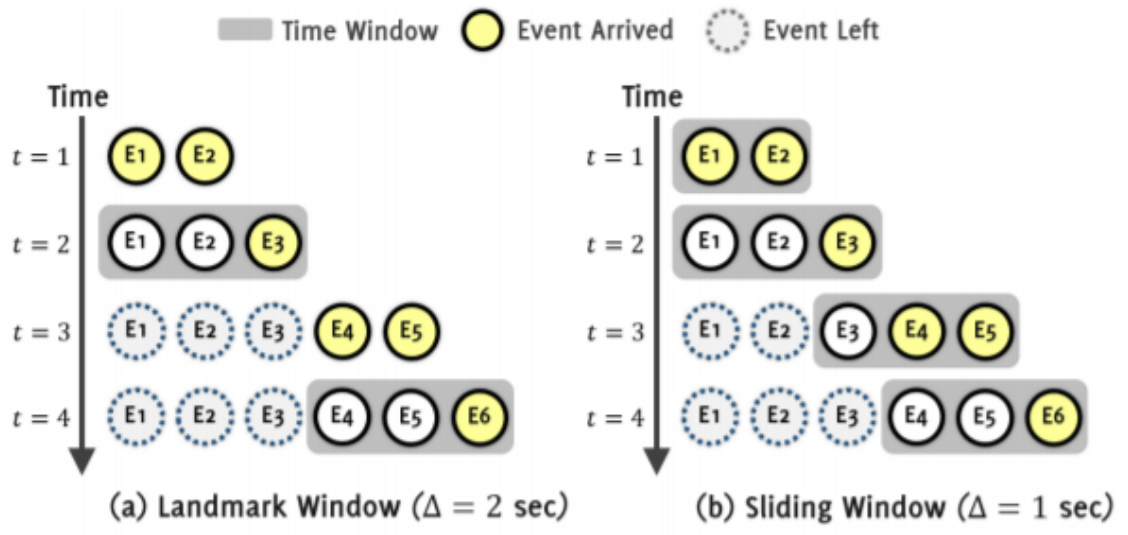


Figura 4.4. Exemplo de janelas de dados *Landmark* e *Sliding* [Roriz 2017].

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.

```
EVERY (
   dur = SocialEvent(duration > 60)
OR
   type = SocialEvent(typeEvent = 'Coversation')
  )
```

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.

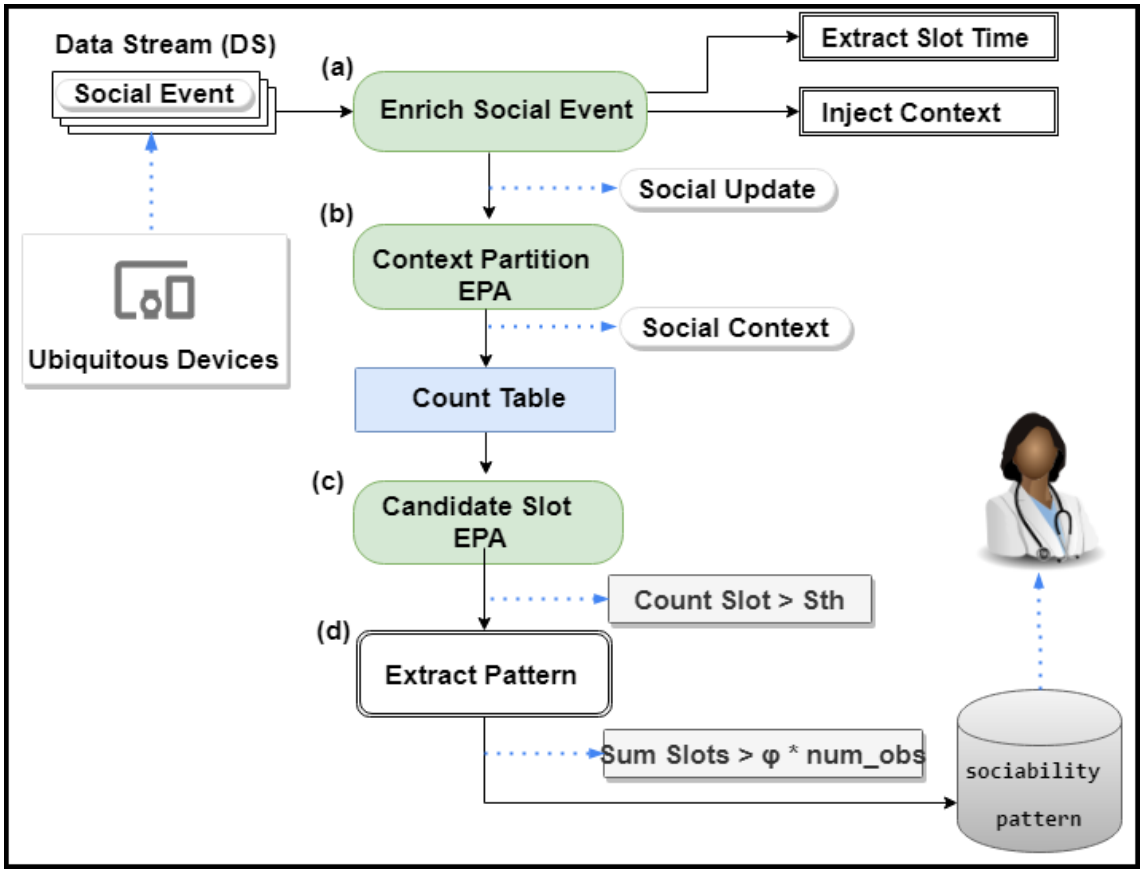


Figura 4.5. Fluxo de processamento da ferramenta desenvolvida.

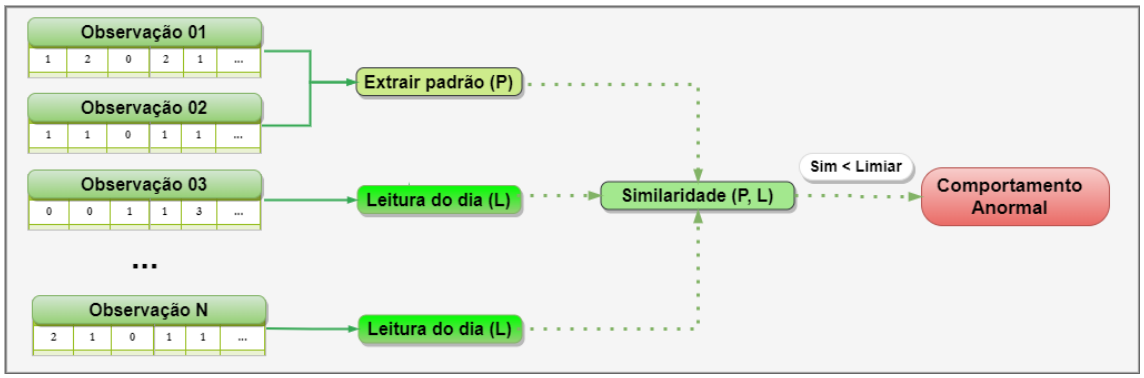


Figura 4.6. Estratégia de detecção de comportamentos sociais anormais.

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.

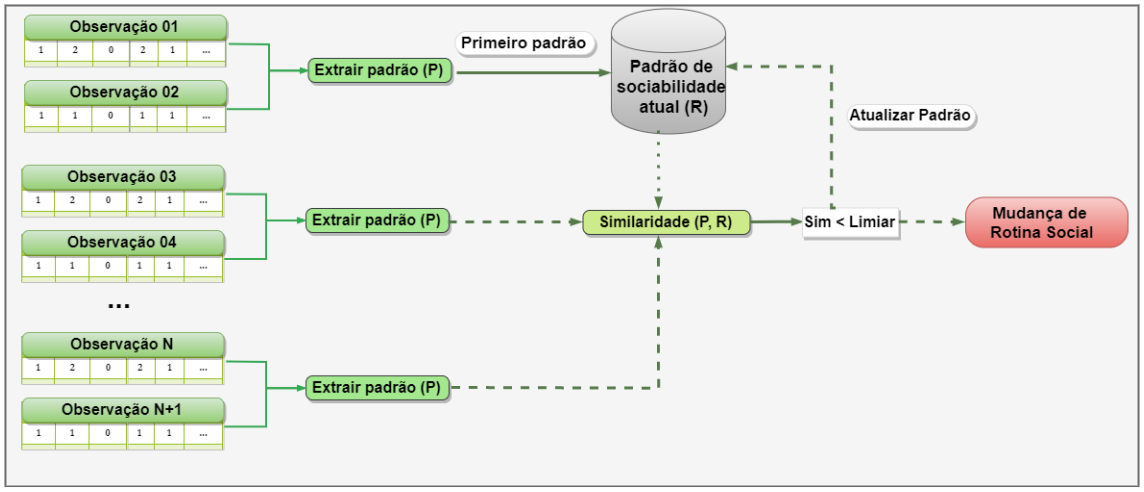


Figura 4.7. Estratégia de detecção de mudança de rotina social.

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.

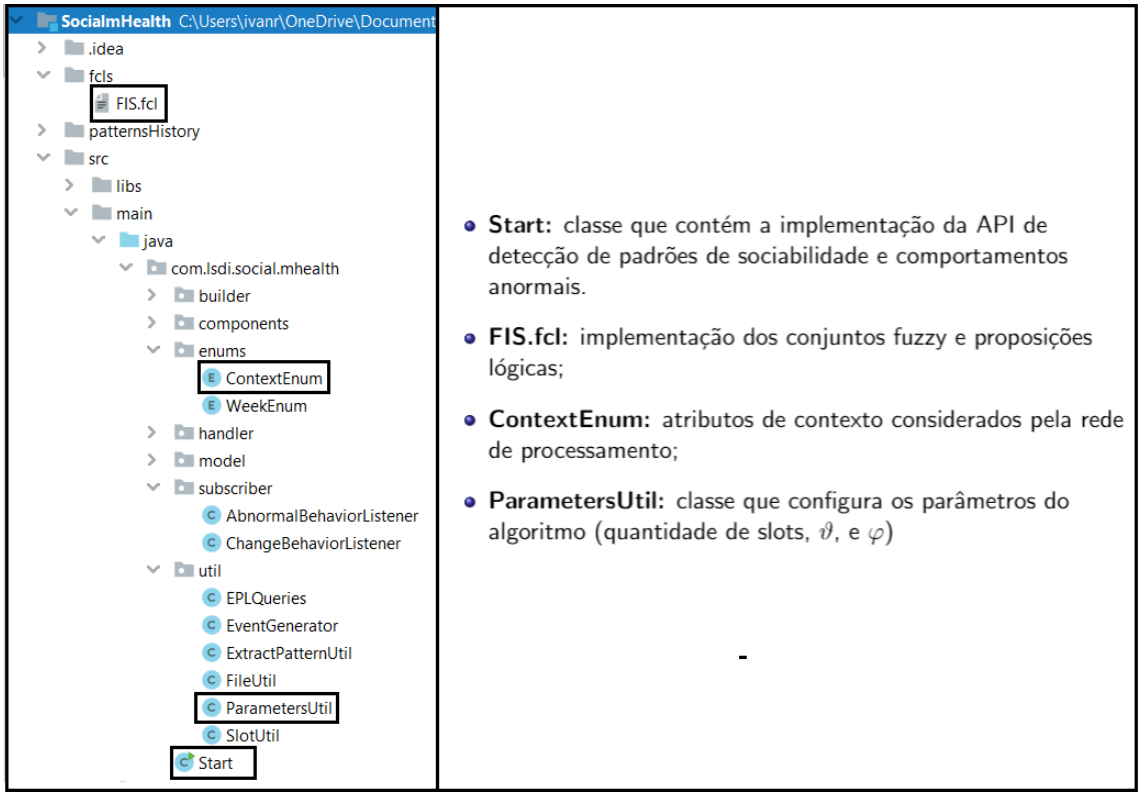


Figura 4.8. Estrutura do código fonte da ferramenta.

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.

Capítulo

5

Desenvolvimento de sistemas Web orientado a reuso com Python, Django e Bootstrap

Cynthia Pinheiro Santiago, Nécio de Lima Veras, Anderson Passos de Aragão, Daniel Albuquerque Carvalho, Luciana Alves Amaral

Abstract

The constant growth of the Internet has caused a significant demand for WEB applications. In this scenario, developers seek to create solutions that deliver the required functionality but that also offer usability, security, scalability, among others. Web development frameworks, providing extensive reuse of architectures and libraries, emerged in this context to reduce the development time and complexity inherent in such systems. In this chapter, we present a complete Web solution with Python and the Django and Bootstrap frameworks through a step-by-step, from the installation of the required software, through the integration of the components to the publication in production.

Resumo

O constante crescimento da Internet tem ocasionado uma demanda significativa por aplicações Web. Neste cenário, desenvolvedores buscam criar soluções que entreguem as funcionalidades requeridas mas que também ofereçam usabilidade, segurança, escalabilidade, entre outros. Os frameworks de desenvolvimento Web, proporcionando extenso reuso de arquiteturas e bibliotecas, surgiram nesse contexto com a intenção de reduzir o tempo de desenvolvimento e a complexidade inerentes a tais sistemas. Neste capítulo, apresentamos uma solução Web completa com Python e os frameworks Django e Bootstrap através de um passo-a-passo, desde a instalação dos softwares necessários, passando pela integração dos componentes até a publicação em produção.

5.1. Introdução

O crescimento global das empresas e suas constantes demandas por novas tecnologias fazem com que o desenvolvimento de software esteja em contínua mudança, especialmente quando o funcionamento do software é direcionado para o ambiente da Internet. Somado

a isso, o desenvolvimento de sistemas para a Internet geralmente envolve partes interessadas (*stakeholders*) mais heterogêneas em comparação com a construção de software tradicional. Ainda assim, os modelos de processo usados para o desenvolvimento desses produtos são emprestados das abordagens tradicionais [Manhas 2017]. Isso não significa que as empresas usem um processo de desenvolvimento ideal, em vez disso, existem várias desvantagens apresentadas por metodologias inadequadas e não versáteis, o que torna o processo de criação um desafio.

A Engenharia de Software (ES) é uma área da Ciência da Computação que abrange todos os aspectos do desenvolvimento profissional de software. Estes aspectos englobam desde a sua concepção até o encerramento [Sommerville 2016], passando pelos processos de planejamento, projeto, codificação e testes. Nos últimos anos, a ideia da oferta de "software como serviço" foi proposta, modificando a forma com que os usuários utilizam a internet e acessam seus dados (armazenados em nuvem). A partir disso, ocorreram mudanças significativas na maneira como o software é distribuído entre os usuários, impactando na forma como os sistemas são desenvolvidos, especialmente, os sistemas para a Web (*World Wide Web*), fazendo emergir tecnologias, infraestruturas e novas metodologias para o desenvolvimento Web.

O desenvolvimento de aplicativos Web está no auge devido ao avanço das tendências tecnológicas e à constante dependência da Internet. Em tempos de isolamento social causado pela pandemia do novo Coronavírus, essa dependência é muito mais perceptível para a maioria das pessoas. Como resultado das necessidades dos desenvolvedores, novas metodologias de desenvolvimento são propostas [Molina-Ríos and Pedreira-Souto 2020]. As equipes de produção do software não constroem um novo produto a partir de um projeto "em branco", elas o programam fazendo reúso de componentes e de outros programas (*frameworks*[1]). Neste sentido, a ES incorpora conceitos de reusabilidade de software durante o desenvolvimento de novos produtos, relacionados com a montagem de partes do sistema a partir de softwares preexistentes [Sommerville 2016]. No ambiente Web, essa abordagem tornou-se dominante para a construção de softwares tento em vista a existência de restrições para tempo e custo, além de exigências maiores sobre qualidade e segurança para os produtos de software.

Diante disso, várias tecnologias para o desenvolvimento de produtos Web vem surgindo ao longo dos anos, destacando elementos como produtividade e velocidade na construção dos produtos, segurança, performance do produto e, obviamente, reuso de componentes em diferentes projetos. Os produtos Web comumente são divididos em dois subprodutos tecnológicos: *front-end* e *back-end* [Ghimire 2020]. Tecnologias *back-end* estão ligadas à linguagens de programação, todavia, preocupam-se com a programação lógica dos sistemas Web e influenciam na forma como os dados são armazenados, acessados e servidos a partir dos "servidores". Por outro lado, tecnologias *front-end* concentram a parte estética e de exibição de conteúdos, também conhecida como desenvolvimento do lado do cliente. Neste capítulo abordaremos a linguagem Python apoiada pelo *framework* Django como principais tecnologias *back-end* e o *framework* Bootstrap como principal tecnologia *front-end*.

[1]Um framework é um conjunto de bibliotecas e ferramentas padronizadas para uma linguagem de programação específica [Ghimire 2020]

Nas seções posteriores serão detalhados alguns pontos acima citados, como Engenharia de software Orientada à Reúso, Desenvolvimento de Software para o ambiente da Web por meio das tecnologias Python, Django e Bootstrap. Além disso, será apresentado um projeto ilustrando o uso das tecnologias detalhadas.

5.2. Engenharia de Software Orientada a Reúso

O desenvolvimento de softwares, de uma forma geral, é tipicamente conhecido por ser um processo caro e lento. A pressão que acompanha todas as etapas de construção, bem como a busca por redução de custos e maior retorno do investimento, associados à redução do tempo disponível para o desenvolvimento podem fazer com que o resultado final e a qualidade do produto fiquem aquém do desejado. Neste sentido, a reutilização de software é uma solução viável, que pode ajudar a lidar com muitos destes desafios. Apesar da ideia de reúso de software ter sido proposta como estratégia de desenvolvimento no final dos anos 60, só a partir dos anos 2000 se tornou realidade. Este fato se deu como resposta às exigências cada vez mais frequentes dos *stakeholders* por menores custos de produção e de manutenção, entregas mais rápidas e softwares de maior qualidade.

Engenharia de software baseada em reúso é uma estratégia onde todo o processo de desenvolvimento de um sistema é orientado a maximizar o reúso de ativos de software já existentes, desde sistemas completos, assim como componentes, *frameworks*, classes, funções e até mesmo conceitos [Sommerville 2016]. A reutilização de software também pode ser definida como "o uso dos conhecimentos e artefatos de engenharia existentes para construir novos sistemas de software" [Frakes and Fox 1996]. Segundo [Sommerville 2016], esta abordagem apresenta algumas reconhecidas vantagens como: (i) os ativos de software reusados (anteriormente experimentados e testados em sistemas em funcionamento) tendem a ser mais confiáveis do que os que são desenvolvidos a partir de um "projeto em branco"; (ii) como o custo de um ativo de software existente já é conhecido e consolidado, isso reduz a margem de erro na estimativa dos custos do projeto; (iii) em vez de repetir o mesmo trabalho em vários sistemas, especialistas desenvolvem ativos de software passíveis de reúso que encapsulem seu conhecimento e que sejam passíveis de utilização inclusive por outros desenvolvedores; (iv) alguns padrões, como os de interface de usuário, podem ser implementados como um conjunto de componentes reusáveis, resultando em menos erros por parte dos usuários uma vez que a interface já lhes é familiar; (v) o reúso de ativos de software em geral acelera a produção do sistema, já que reduz seu tempo de desenvolvimento e validação.

Por outro lado, também são conhecidos obstáculos com relação à adoção do reúso de software como: (i) resistência à modificação de mentalidade dos próprios desenvolvedores quanto ao reuso, (ii) alta curva de aprendizagem de alguns ativos de software a serem reusados, (iii) a dificuldade em manter um repositório de componentes reutilizáveis no que se refere à classificação, catalogação e pesquisa de componentes (às vezes, pode ser mais custoso encontrar um componente que reutilizá-lo) além de outras barreiras detectadas, analisadas e apresentadas em [Almeida 2019].

Ao longo do tempo, muitas técnicas foram desenvolvidas para oferecer suporte ao reúso de software em suas mais distintas granularidades, desde sistemas completos até funções. Neste aspecto, dentre as abordagens de reúso de software disponíveis está a que

faz uso de *frameworks* de aplicação que, segundo [Schmidt et al. 2004], podem ser definidos como um conjunto integrado de artefatos de software (tais como classes, objetos e componentes) modulares e/ou extensíveis que colaboram para fornecer uma arquitetura reusável. Estendendo esse conceito, um *framework* de aplicação Web pode ser entendido como uma coleção de pacotes e módulos compostos por código padronizado e pré-escrito que suporta o desenvolvimento de aplicações Web, tornando o desenvolvimento mais rápido e fácil, e fazendo com que os programas que o utilizem tornem-se mais confiáveis e escaláveis. Tais *frameworks* já possuem componentes internos que "configuram" o projeto, para que o desenvolvedor tenha menos trabalho repetitivo.

Para este tipo de desenvolvimento, os *frameworks* tornaram-se uma parte essencial, já que os padrões Web estão sempre se sofisticando e, consequentemente, a complexidade da tecnologia necessária para atendê-los está cada vez maior, tornando praticamente proibitivos o custo e o tempo necessário para o desenvolvimento de uma aplicação a partir de um "projeto em branco". Os *frameworks* de aplicações Web geralmente têm sua arquitetura baseada no padrão MVC (*Model-View-Controller* ou Modelo-Visão-Controlador, em português) [Gamma et al. 1995] ou em suas variações. Isto permite separar as representações internas dos dados do modo como estes dados são apresentadas para os usuários. Além disso, os *frameworks* Web oferecem suporte à apresentação dos dados de maneiras diferentes e interagem com cada uma dessas apresentações. Quando os dados são modificados por meio de uma das apresentações, o modelo de sistema é alterado e os controladores associados a cada visão atualizam sua apresentação.

A interação do usuário com uma aplicação Web que usa o padrão MVC (Figura 5.1) segue um ciclo natural: "o usuário executa uma requisição (*request*) e, em resposta, a aplicação consulta e/ou altera seu modelo de dados, fornecendo uma visualização atualizada com uma resposta (*response*) ao usuário. Isto é um ajuste muito conveniente para aplicações Web, onde há uma série de requisições e respostas HTTP" [Freeman 2017].

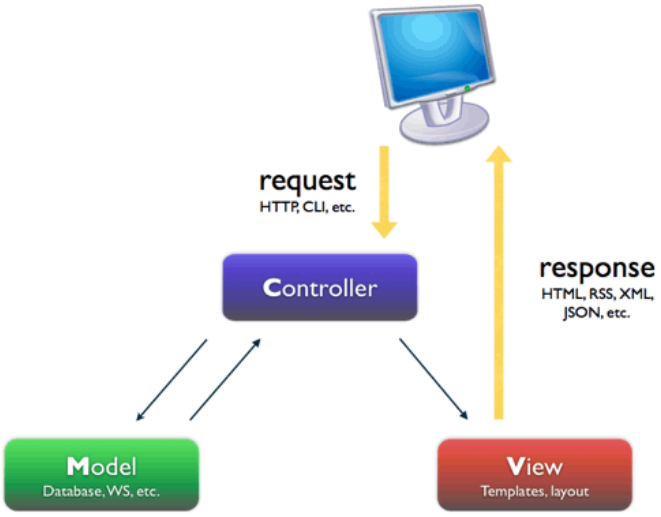


Figura 5.1. Padrão MVC, adaptado de [Dragos-Paul and Altar 2014]

Além disso, a maioria dos *frameworks* Web disponíveis oferece minimamente os

seguintes recursos: (i) suporte para a autenticação de usuário e controle de acesso; (ii) definição de *templates*[2] de páginas Web preenchidos dinamicamente; (iii) interface abstrata para manipulação de banco de dados; (iv) criação e gerenciamento de sessões do usuário; (v) suporte a tecnologias que permitem interação com o usuário (p.e. AJAX[3]) . Ou seja, eles dão apoio à resolução de problemas recorrentes utilizando uma abordagem genérica, permitindo ao desenvolvedor dedicar mais tempo à resolução dos problemas em si, e não em reescrever software que já existe. É interessante frisar que um mesmo sistema Web pode incorporar vários *frameworks*, de forma que cada um deles é projetado para apoiar o desenvolvimento de parte da aplicação.

5.3. Desenvolvimento Web

O termo "Desenvolvimento Web" pode ser definido como a construção, criação e manutenção de sites na Internet ou em uma intranet. Normalmente, o desenvolvimento na Web envolve um *front-end* (*client-side*), código que interage com o cliente através de um navegador, e um *back-end* (*server-side*), código executado no servidor Web, que encapsula a lógica de negócios da aplicação e a interação com o banco de dados. O padrão MVC, citado na Seção 5.2, favorece essa divisão: os desenvolvedores *front-end* concentram-se na camada de Visão (*View*) e os desenvolvedores *back-end* concentram-se na camada de Modelo (*Model*).

Frameworks podem ser utilizados tanto no *front-end* como no *back-end*. *Frameworks* e bibliotecas fazem parte da pilha de tecnologias Web: uma combinação de software, aplicativos, linguagens de programação e ferramentas que são construídas umas sobre as outras para criar um site. Uma pilha de tecnologia aplicável ao desenvolvimento Web pode ser visto na Figura 5.2. Uma biblioteca é uma coleção de ferramentas e recursos específicos que pode ser adicionada à aplicação Web a fim de obter funcionalidades. Diferentemente de um *framework*, uma biblioteca não oferece uma estrutura do ponto de vista arquitetural, mas implementa diferentes comportamentos e ações.

Nas subseções seguintes, apresentamos mais detalhes sobre cada uma das frentes do desenvolvimento de um sistema Web e mencionamos de que forma *frameworks* e bibliotecas podem ser utilizados neste sistema.

5.3.1. *Front-end*

Tudo o que é visto e "navegado" pelo usuário (identidade visual do site, menus, textos, imagens etc.) é criado e mantido pelo desenvolvedor *front-end*. Para isso, são escritos programas para vincular e estruturar os elementos visuais, adicionando interatividade com os usuários. Esses programas são executados através de um navegador (*browser*). No *front-end*, o desenvolvedor transforma a ideia de design da aplicação em realidade, concentrando-se no layout, design e interatividade da aplicação Web. Para tanto, linguagens como HTML (linguagem de marcação), CSS (linguagem de estilo) e JavaScript

[2]Um template é um documento apenas com a apresentação visual e instruções sobre onde e qual tipo de conteúdo deve entrar em cada parcela da apresentação.

[3]O AJAX (Asynchronous JavaScript and XML) é um conjunto de técnicas de desenvolvimento voltado para a Web que permite que aplicações trabalhem de modo assíncrono, processando qualquer requisição ao servidor em segundo plano.[Holdener 2008]

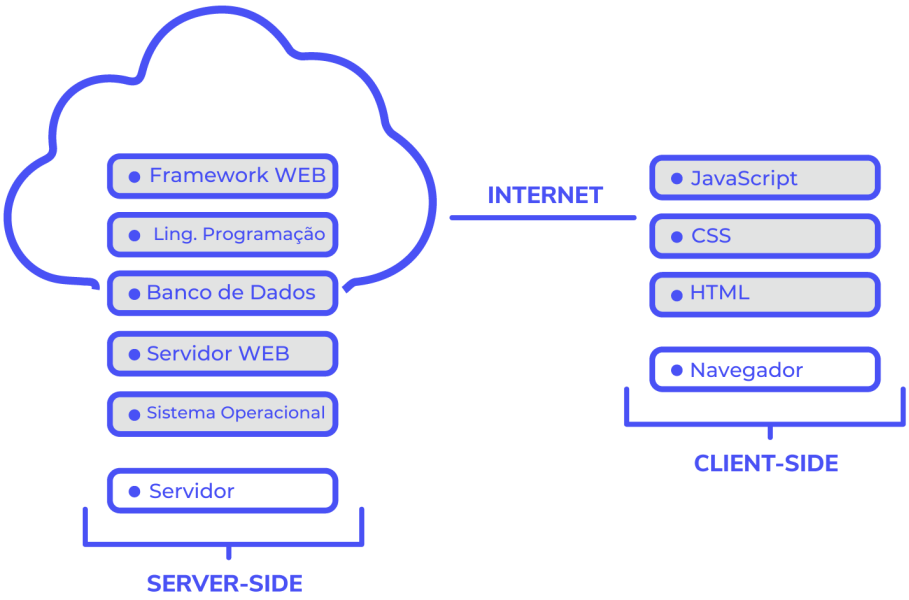


Figura 5.2. Pilha de tecnologias, adaptada de [Fawcett 2019]

(linguagem de script/programação) são utilizadas. A seguir, uma breve descrição de cada uma destas linguagens:

- HTML (*HyperText Markup Language*): é a linguagem de programação básica para desenvolvimento Web *front-end*. Fornece a estrutura para o conteúdo de um site e define palavras, títulos, parágrafos, imagens etc. O HTML consiste em um conjunto de *tags* pré-definidas, representando funções diferentes que "convertem" o conteúdo em um formato legível na tela.

- CSS (*Cascading Style Sheets*): é uma "folha de estilos" que descreve como os elementos HTML aparecerão em uma página da Web. Usa-se o CSS para controlar a apresentação, o estilo e a formatação dos elementos HTML em um site, configurando cores, bordas, imagens de fundo etc. Os arquivos CSS declaram um conjunto de regras, que definem propriedades e seus respectivos valores para os elementos HTML.

- JavaScript: refere-se ao controle do comportamento de uma página Web. É uma das linguagens de programação mais populares e difundidas no mundo. O uso do JavaScript torna os sites interativos, manipulando as especificidades dos elementos HTML e CSS. Com JavaScript, um usuário pode clicar em um botão e disparar uma ação, fazer o site rolar para a parte inferior ou exibir fotos aleatórias, por exemplo.

Na maioria dos casos, os *frameworks* para *front-end* são caixas de ferramentas ("*tool boxes*") que incluem diversas bibliotecas CSS e JavaScript. Entre os *frameworks* e

bibliotecas mais utilizados no *front-end* estão o Ember[4], React[5], Backbone[6], Vue[7], Angular[8] e Bootstrap[9].

O Bootstrap é o *framework front-end* utilizado no projeto ilustrativo deste capítulo. Desde 2013, ele se tornou um dos projetos mais populares na plataforma de compartilhamento de código GitHub[10]. Possui suporte mantido por uma comunidade ativa e um vasto ecossistema, incluindo modelos e extensões criados em torno dele. Lançado inicialmente pelo Twitter[11] para manter a consistência em seus projetos internos de Web design e desenvolvimento, o Bootstrap evoluiu e, desde o lançamento da versão 3, foi licenciado sob a licença MIT de código aberto [Bootstrap 2020a]. Mais detalhes sobre este *framework* na seção 5.5.

5.3.2. *Back-end*

O *back-end* consiste no servidor Web que hospeda o site, uma aplicação para executá-lo e um banco de dados para armazenar as informações. Como o nome sugere, o desenvolvedor *back-end* trabalha na parte de "trás" da aplicação, escrevendo programas que garantam que o servidor, a aplicação Web e o banco de dados funcionem bem juntos. Esse desenvolvedor precisa analisar quais são as necessidades de negócio dos *stakeholders* e fornecer soluções de programação eficientes e seguras. Para isso, são utilizadas uma variedade de linguagens do tipo "*server-side*", como PHP, Ruby, Java ou Python.

Além disso, dada a complexidade das tecnologias envolvidas no *back-end* e a quantidade de requisitos de qualidade que devem ser atendidos (p. e. desempenho, confiabilidade, segurança e disponibilidade do serviço), geralmente são utilizados *frameworks back-end* que auxiliam o desenvolvedor em muitos destes aspectos.

Uma das características comuns que a maioria dos *frameworks back-end* oferecem é uma abstração do banco de dados com a qual ele possa interagir, em lugar do próprio banco de dados (Figura 5.3). Isso é feito através de um Mapeamento Objeto-Relacional (do inglês, ORM: *Object-Relational Mapping*), definido como uma ferramenta que fornece uma metodologia e um mecanismo para sistemas orientados a objetos para armazenar dados de forma segura e por um longo período de tempo em um banco de dados, tendo controle transacional sobre eles, mas sendo expresso, se necessário, como objetos dentro da aplicação [O'Neil 2008].

O ORM libera o desenvolvedor da preocupação de conhecer a estrutura ou esquema do banco de dados. O acesso a dados é todo feito usando um modelo conceitual que reflete os objetos de negócios dos desenvolvedores, correlacionando dados em tabelas de um banco de dados relacional com classes em uma linguagem de programação. Assim, o desenvolvedor trabalha com conceitos da programação orientada a objetos para gerenciar os dados armazenados no banco [Dragos-Paul and Altar 2014].

[4]https://emberjs.com/
[5]https://reactjs.org/
[6]https://backbonejs.org/
[7]https://vuejs.org/
[8]https://angular.io/
[9]https://getbootstrap.com/
[10]https://github.com/
[11]https://twitter.com/

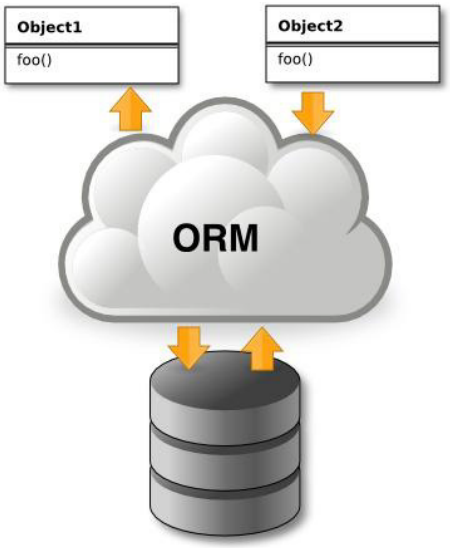


Figura 5.3. Mapeamento Objeto-Relacional (ORM) [Dragos-Paul and Altar 2014]

Um outro ponto em que os *frameworks back-end* auxiliam os desenvolvedores é em relação à segurança da aplicação. A vulnerabilidade de aplicações Web é a causa dos mais diversos tipos de ataques aos usuários. São muitos os pontos em que os desenvolvedores devem ficar atentos como: validação dos dados de entrada dos usuário, autenticação no sistema, gerenciamento de privilégios de acesso, acesso não autorizado à interface de gerenciamento, acesso a dados sensíveis no banco de dados, roubos de sessão, tipo de criptografia dos dados, manipulação indevida de parâmetros do sistema, ataques de negação de serviço, entre outros [Dragos-Paul and Altar 2014]. Por isso, muitos *frameworks back-end* disponibilizam aos desenvolvedores solução robustas em segurança já prontas que evitam em grande parte os inconvenientes mencionados.

Além disso, outra característica fornecida pela maioria dos *frameworks back-end* é o mapeamento de URLs (do inglês, *URL Mapping* ou *URL Routing*), que consiste em criar padrões de URL para a aplicação Web. O *framework* compara requisições HTTP com padrões pré-estabelecidos no sistema (*URL patterns*). Se forem compatíveis, estas requisições são encaminhadas para tratamento em sua função correspondente, que foi previamente mapeada à URL no *framework*. Por outro lado, após o tratamento das requisições, a resposta (*response*) correspondente é encaminhada para uma URL de saída, gerada depois que uma função específica é executada.

Atualmente, existem diversos *frameworks* para *back-end* disponíveis que implementam estas características. Alguns dos mais populares entre os desenvolvedores são: Spring[12], Express[13], Laravel[14], ASP.NET[15], Rails[16] e o Django[17].

[12]https://spring.io/
[13]https://expressjs.com/
[14]https://laravel.com/
[15]https://dotnet.microsoft.com/apps/aspnet
[16]https://rubyonrails.org/
[17]https://www.djangoproject.com/

O Django, *framework back-end* utilizado no projeto ilustrativo deste capítulo, é escrito em Python e vem com o conceito de "baterias incluídas", ou seja, ele dá suporte para a maior parte das necessidades de um desenvolvimento *server-side*, sendo altamente personalizável, escalável, possuindo uma extensa documentação e uma grande comunidade de desenvolvedores. Tornou-se um projeto de código aberto e foi publicado sob a licença BSD em 2005 [Foundation 2020a]. Mais detalhes sobre este *framework* na seção 5.4.

5.4. Python e Django

Python é uma linguagem de alto nível, interpretada, de tipagem forte e dinâmica, desenvolvida na década de 80 por Guido van Rossum no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI), na Holanda. O propósito geral do seu desenvolvimento foi facilitar a programação sem que o programador se preocupasse com o hardware. Tornou-se uma linguagem simples e muito poderosa, utilizada principalmente em áreas como *Data Science*, *Data Analytics*, Inteligência Artificial, *Deep Learning*, Desenvolvimento Web e aplicações de forma geral. Além disso, possui diversas bibliotecas que permitem o reúso de funcionalidades no desenvolvimento de software. Até o presente momento, são 246.267 bibliotecas disponíveis [Foundation 2020b].

A linguagem vêm ganhando popularidade equiparando-se com linguagens tradicionais como C++ e Java. Segundo o RedMonk, uma das maiores empresas que coletam estatísticas para programadores, Python está na segunda colocação no ranking de linguagens com maior popularidade [Stephens 2020]. É uma linguagem multiplataforma, o que garante portabilidade e compatibilidade em diversos sistemas operacionais. Além disso, Python tem código aberto, o que significa que qualquer pessoa pode ver e alterar sua codificação para fazer suas próprias bibliotecas. Suporta também interação com outras linguagens, permitindo que desenvolvedores possam utilizar ferramentas e/ou funcionalidades que não se encontram nativamente em Python, algumas escritas em C/C++, Java, C#, .NET, PHP, entre outras.

Python se propõe a trazer mais comodidade na hora de programar, por isso, diferentemente das demais linguagens, utiliza poucas palavras, o que implica em menos código. Quando começamos a programar em Python, aprendemos que a indentação é primordial, temos que ter um código organizado e limpo para funcionar, assim temos menos erros. Esta linguagem faz uso de palavras reservadas curtas da língua inglesa, facilitando assim, o entendimento e o aprendizado, além de deter de um grande poder de processamento (flexibilidade e simplicidade), fator que pode ser utilizado tanto por aqueles que estão entrando no mundo da programação, como também para os mais experientes.

Por estes diferenciais, diversas empresas optam por codificar uma parte ou mesmo a totalidade de seus produtos em Python. Alguns delas são: a Netflix, maior stream de vídeos, que utiliza Python para aumentar a segurança de seus dados, analisar alertas e gerar relatórios de dados; o Google, que o utiliza em projetos que demandam entregas rápidas e facilidade de implantação como o Google App Engine, uma plataforma de computação em nuvem para desenvolver e hospedar aplicativos em data centers gerenciados pelo Google; o Instagram, maior rede social de compartilhamento de fotos, que migrou suas aplicações para o Python; o Spotify, mais famosa stream de músicas, que usa o Python

para analisar dados e no back-end do aplicativo [Demchenko 2019].

Por outro lado e seguindo a popularidade desta linguagem surgiu o Django, um *framework* para criação de aplicações Web escrito em Python, criado em 2005 por um grupo de programadores do Lawrence Journal-World com a intenção de tornar mais rápido o desenvolvimento de aplicações Web. Este *framework* tornou-se conhecido por fornecer soluções para grande parte dos problemas tradicionais em desenvolvimentos Web, possuindo dezenas de tarefas comuns já prontas para serem reutilizadas, como por exemplo autenticação de usuário, administração de conteúdo, mapas de site, entre outras.

Django também ajuda a evitar erros de segurança comuns e proporciona escalabilidade aos sistemas, ou seja, consegue oferecer capacidade de expansão de um sistema sem perda do seu desempenho, como é o que acontece por exemplo com Mozilla Firefox, Pinterest e Instagram. Muitas empresas escolhem o Django por sua extrema versatilidade, sendo ele utilizado para criação de sistemas que vão desde gerenciamento de conteúdo até plataformas científicas.

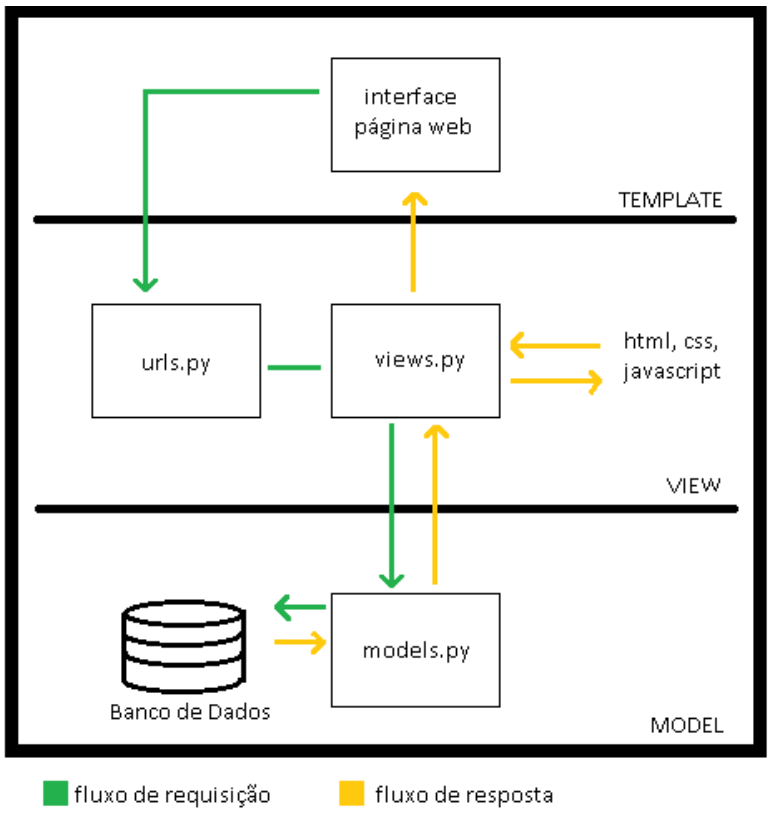


Figura 5.4. Fluxo MTV do Django

Diferente de outros *frameworks* Web que utilizam o MVC, descrito na seção 5.2, o Django utiliza a estrutura MTV (Model-Template-View), representado na Figura 5.4, onde o *framework* gerencia a maior parte da comunicação entre requisições HTTP. Quando é criado um projeto em Django temos um diretório inicial de pastas e arquivos, onde estão os arquivos models.py, views.py e urls.py, responsáveis pelo fluxo MTV. Maiores detalhes sobre a instalação do Django podem ser vistos em [Foundation 2020a]. A

seguir, explicaremos as camadas MTV e o funcionamento do fluxo entre elas:

- *Model*: Django já vem com uma solução para mapeamento objeto-relacional (ORM, do inglês *Object-Relational Mapping*) no qual o esquema do banco de dados é descrito em código Python [Foundation 2020a]. Neste caso, abstrações em Python podem ser usadas para criar consultas complexas sem que seja necessário realizar ações diretas no banco.

- *Template*: Esta é a camada de apresentação, onde as informações são visualizadas pelos usuários. Um *template* consiste de partes estáticas do arquivo HTML de saída e de partes com uma sintaxe especial que descrevem como o conteúdo dinâmico será apresentado. O Django tem um caminho de pesquisa para *templates*, o qual permite minimizar a redundância entre eles. Geralmente, uma *view* recupera dados de acordo com os parâmetros de pesquisa, carrega um *template* e o renderiza com os dados recuperados [Foundation 2020a].

- *View*: As *views* recebem a informação e o tipo da requisição ("POST" ou "GET") do lado do cliente e, em seguida, formatam os dados para que sejam armazenados no banco através dos *models* da camada *Model*. As *views* também se comunicam via *models* com o banco para recuperar dados que são transferidos posteriormente aos *templates*, para a visualização do usuário. Cada *view* é responsável por fazer uma entre duas coisas: devolver um objeto *HTTPResponse* contendo o conteúdo para a página requisitada ou levantar uma exceção como *Http404* [Foundation 2020a].

Em um site convencional, uma aplicação Web aguarda requisições HTTP do navegador. Quando uma requisição é recebida, a aplicação calcula o que é necessário fazer com base na URL e nas informações dos dados enviados via POST ou GET. Dependendo do que for necessário, a aplicação poderá ler ou gravar dados no banco ou executar outras tarefas necessárias para satisfazer a requisição. A aplicação retornará uma resposta, gerando dinamicamente uma página HTML para o navegador exibir, inserindo os dados recuperados em espaços reservados em um *template* HTML. Aplicações Web feitas com Django geralmente agrupam o código que manipula cada uma dessas etapas em arquivos separados: urls.py, views.py e models.py.

O arquivo urls.py é encarregado de armazenar o mapeamento de URLs (*urlpatterns*), redirecionando requisições HTTP para a função (também chamada *view*) apropriada, com base na URL da solicitação. As *views* são o coração da aplicação Web, recebendo requisições HTTP de clientes Web e retornando respostas HTTP.

No arquivo urls.py existe uma lista de mapeamentos do tipo URL/*view* correspondente, como pode ser visto no exemplo apresentado na Figura 5.5. O objeto *urlpatterns* é uma lista de métodos *path()*. O primeiro parâmetro do método *path()* é a URL que corresponderá a uma *view*. Opcionalmente, usa-se sinais de menor e maior (<, >) para definir partes de uma URL que serão capturadas e passadas para a *view* correspondente como argumentos nomeados. O segundo parâmetro do método *path()* é a view correspondente à URL informada e, por fim, o terceiro parâmetro é o nome para chamada no template.

Uma vez que o mapeamento em urls.py é encontrado pelo Django, a *view* correspondente à URL é chamada. As *views* geralmente são armazenadas em um arquivo

```
urlpatterns = [
    path('caminho/', views.equipment_list,name='equipment_list'),
    path('caminho/<int:id>/', views.equipment_detail, name='equipment_detail'),
]
```

Figura 5.5. Exemplo de arquivo urls.py

chamado views.py, como ilustra a Figura 5.6. Por meio deste arquivo são disparadas as respostas às requisições recebidas: podem ser acessados o banco de dados através da camada *Model* ou bibliotecas, caso seja necessário, e é enviado o retorno para renderização na camada *Template*. Todas as funções *view* recebem um objeto HttpRequest como parâmetro e retornam um objeto HttpResponse. Juntamente com o HttpRequest pode haver um ou mais argumentos para a *view*, que são nomeados na URL correspondente e recebidos como parâmetros pela função. Ao final, como mostra a Figura 5.6, a *view* retorna a função de renderização *render()* que envia ao template os dados necessários, juntamente com o *context*, que é a variável que contém dados para visualização, renderizando assim a página para visualização do usuário.

```
from django.shortcuts import render

def EntryList(request):
    context = {'objects':Entry.objects.all()}
    return render(request,'entry_list.html',context)
```

Figura 5.6. Exemplo de arquivo views.py

As aplicações feitas com Django armazenam, gerenciam e consultam dados por meio de objetos do tipo *models*, como mostrado no exemplo da Figura 5.7. Os *models* definem a estrutura dos dados armazenados, incluindo o tipo do dado, seu tamanho máximo, valores padrão, opções de lista de seleção, texto de ajuda para documentação, texto de etiqueta (*label*) para formulários etc. A definição do *model* feita no arquivo models.py é independente do banco de dados escolhido para a aplicação Web. O *framework* integra-se com diversos bancos de dados do mercado mas, por padrão, vem configurado com o SQLite. No entanto, é muito simples configurar no Django a troca de um banco para outro. Uma vez definido o banco, não é preciso escrever código para comunicar-se diretamente ele. É preciso apenas escrever a estrutura dos *models* e, em seguida, o Django, através de seu ORM, cuida de todo o trabalho de armazenamento e recuperação de dados. Se for necessário alterar a estrutura dos *models*, o desenvolvedor precisa apenas fazer as migrações, que são comandos do Django para que ele propague as alterações feitas para o esquema do banco de dados [Foundation 2020a].

```
class Entry(models.Model):
    name = models.CharField(max_length=40)
    date = models.DataTimeField()
    amount = models.IntegerField(default=0)

    def __str__(self):
        return self.name
```

Figura 5.7. Exemplo de arquivo models.py com a definição de *models*

Além de definir a estrutura dos dados, os objetos *models* oferecem um mecanismo simples para recuperação de dados no banco de dados. Por meio de uma *query* (consulta), retornam objetos nos quais serão realizadas as manipulações necessárias nas *views*. Existem vários tipos de consultas disponíveis, que podem ou não conter parâmetros para especificar os objetos a serem retornados, como podemos observar na Figura 5.8. No exemplo em questão, está sendo realizada uma consulta na tabela *Entry*, realizando um filtro por ano de publicação igual a 2005 e ordenando as consultas por ano de publicação (*pub_date*) e por título (*headline*).

```
Entry.objects.all()
Entry.objects.filter(pub_date__year=2005).order_by('-pub_date', 'headline')
Entry.objects.order_by('blog__name', 'headline')
```

Figura 5.8. Mecanismo do Django em *views.py* para recuperação de dados no banco

Uma vez que a requisição foi tratada na *view*, é retornado o resultado à camada *Template*, onde será feita a renderização de um arquivo de saída, tipicamente um arquivo HTML, contendo trechos de código em Python que manipulam os objetos retornados pelas *views* (Figura 5.9). No arquivo HTML, é possível utilizar também bibliotecas JavaScript, folhas de estilo (CSS), ou mesmo fazer uso de *frameworks front-end* como o Bootstrap, apresentado na seção 5.5.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Entry List</title>
</head>
<body>
  <ul>
      {% for object in objects %}
        <li>{{ object.name }} {{ object.date }} {{ object.amount }}</li>
      {% endfor %}
    </ul>
</body>
</html>
```

Figura 5.9. Arquivo HTML com código Python para manipulação de objetos

Além disso, o Django possui diversos outros recursos que são essenciais e aceleram ainda mais o desenvolvimento de projetos Web. Alguns deles são: (i) formulários (ou *forms*), que coletam os dados do usuário e os convertem automaticamente para objetos Python; (ii) mecanismos seguros e robustos de autenticação e permissão, permitindo que os usuários do site criem contas e efetuem login ou logout com segurança; (iii) armazenamento em cache (*caching*), fazendo com que a renderização das páginas aconteça apenas quando necessário, evitando assim a lentidão na aplicação; (iv) disponibilização de uma interface padrão para administração do site, onde é possível criar, exibir e editar quaisquer modelos de dados, entre outros [Foundation 2020a].

5.5. Bootstrap

O Bootstrap é um *framework front-end*, que disponibiliza código fonte para a criação de interfaces Web. É considerado um “kit de ferramentas”, contendo componentes HTML,

CSS e JavaScript cujo uso proporciona rapidez e praticidade ao desenvolvimento. Este *framework* tem como principal objetivo auxiliar na criação de sites amigáveis e responsivos. Além disso, oferece uma grande variedade de *plug-ins*, possui temas compatíveis com vários outros *frameworks* e possui suporte para todos os navegadores. Também oferece extensibilidade usando JavaScript, com suporte interno para *plug-ins* jQuery e uma API JavaScript. O Bootstrap pode ser usado com qualquer IDE ou editor e com qualquer tecnologia ou linguagem back-end como ASP.NET, PHP, Ruby on Rails ou Python [Bacinger 2020].

O jQuery é uma biblioteca JavaScript rápida, compacta e rica em recursos. Utilizá-la simplifica em muito o processo de manipulação de eventos e o uso de animações, com uma API fácil de usar e compatível com vários navegadores [jQuery 2020]. Essa biblioteca é de código aberto e sua principal função é facilitar a utilização do JavaScript no desenvolvimento de sites.

O *framework* Bootstrap surgiu, em um primeiro momento, como uma tentativa de resolver o problema da falta de padronização visual nos sistemas Web desenvolvidos na equipe do Twitter, composta pelos engenheiros Jacob Thorton e Mark Otto em 2010. Na época de criação, a ideia era padronizar as interfaces gráficas do site, para evitar inconsistências [de Leone 2017]. Para otimizar o trabalho, resolveram desenvolver uma estrutura única para ser usada por todos os profissionais da equipe. A ferramenta foi lançada no segundo semestre de 2011 e já em 2012 tornou-se um dos projetos mais populares no GitHub.

Este *framework* facilita o trabalho de desenvolvimento pois dispensa a criação de diversos *scripts*, como acontece normalmente em muitos projetos. Com recursos como *tooltip* (dicas), *menu-dropdown*, *modal*, *carousel*, *slideshow* entre outros, garante ótimas experiências em seu uso e manipulação por parte dos desenvolvedores, com configurações bastante intuitivas [ISBRASIL 2017].

A responsividade proporcionada pelo Bootstrap permite que os usuários possam acessar os sites em todos os tipos de dispositivos de forma otimizada e sem que informações sejam perdidas durante a navegação. Por conta disso, os desenvolvedores não precisam se preocupar em fazer várias versões de um site para todos os tipos e tamanhos de telas disponíveis. O sistema de *grid* (ou grade, em português) do Bootstrap, faz uso de vários tipos de *containers*, linhas e colunas para organizar e alinhar visualmente o conteúdo [Bootstrap 2020b]. *Containers* são estruturas que contêm a informação e que ajudam a dispor o conteúdo no site. A informação contida nos *containers* é organizada em colunas, sendo 12 no total pelo padrão Bootstrap. O *layout*, o alinhamento e o tamanho das colunas da *grid* são customizáveis através de um conjunto de utilitários chamado *flexbox*. Para implementações mais sofisticadas, também é possível utilizar CSSs personalizados, além dos CSSs já disponibilizados pelo Bootstrap.

Este *framework* emprega vários estilos e configurações globais importantes, todos voltados para a normalização de estilos entre navegadores [Bootstrap 2020a]. Os componentes do Bootstrap pode ser incorporados ao projeto de duas formas. A primeira forma é através da inclusão do Bootstrap CDN (do inglês, *Content Delivery Network*). Nesse caso, todo o *framework* será carregado a partir de um repositório externo na internet, ficando a aplicação dependente da disponibilidade deste. O *link* deve ser inserido no cabeçalho do

documento HTML, na forma como é exibido na Figura 5.10.

```
<head>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
  integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
</head>
```

Figura 5.10. Inclusão do Bootstrap CDN

As bibliotecas JavaScript do Bootstrap devem ser incluídas no projeto antes do fechamento do corpo do documento HTML, juntamente com o jQuery.js e o popper.js (Figura 5.11).

```
<body>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="
  sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="
  sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="
  sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
</body>
```

Figura 5.11. Inclusão de bibliotecas JavaScript

A segunda forma de se utilizar o Bootstrap é fazendo o download do código compilado já pronto para usar, com seus pacotes CSS e bibliotecas JavaScript compilados e minificados (Figura 5.12), sendo a partir de então acessados a partir de uma pasta local. Dessa forma, pode-se utilizar o Bootstrap sem necessitar conexão com a Internet, pois os arquivos estão sendo carregados localmente.

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

Figura 5.12. Diretório Bootstrap

Navegadores diferentes definem regras distintas para renderização de CSS. Para solucionar essa divergência entre navegadores, surgiram as regras de "*reset*" do CSS que definem padrões de estilo com o objetivo de reduzir as inconsistências entre os navegadores em itens como altura padrão de linha, margens, tamanhos de fonte dos títulos e assim por diante. Para uma tipografia mais inclusiva e acessível, o Bootstrap usa uma pilha de

fontes nativas que seleciona a melhor *font-family* para cada sistema operacional e dispositivo, além de adotar o tamanho de fonte padrão (*font-size*) dos navegadores (tipicamente 16px), entre outras características [Bootstrap 2020c].

O Bootstrap oferece a possibilidade de se usar diversos componentes já prontos como botões personalizados, formulários, imagens e ícones, *navbars*, tabelas e modais. Com esse framework, a apresentação dos botões torna-se invariável de um navegador para outro, sendo também possível personaliza-los com diversas cores, formatos de bordas e tamanhos. Pode-se usar grupos de botões alinhados com funcionalidades das mais diversas.

Os formulários são elementos usados desde muito tempo no desenvolvimento Web. O Bootstrap estiliza elementos de entradas como <*input*>, <*textarea*> e <*select*> melhorando a visualização dos formulários. A exibição das imagens pode ser responsiva com o Bootstrap, além de ser possível deixa-las com as bordas arredondadas ou nos mais variados formatos, tamanhos e posições. É possível inclusive exibir imagens (ícones) dentro de botões, por exemplo. Com CSS é possível alterar cores, tamanhos e sombras dos mesmos.

Um outro elemento disponibilizado pelo Bootstrap são *navbars*, que são estruturas usadas pra criar menus de navegação de forma padronizada. Em telas maiores, a barra de navegação que contém o menu é exibida na horizontal, já em telas menores geralmente é transformada, de forma responsiva, em um menu suspenso.

Outro componente útil disponibilizado pelo Bootstrap para desenvolvimento de sites é o *modal*, uma janela *pop-up* que se abre sobre o conteúdo da página de acordo com uma ação prévia do usuário. O *framework* possui uma estrutura própria para os modais, com uma documentação específica.

Na seção seguinte, será exemplificado o uso de alguns destes componentes dentro do contexto de um projeto ilustrativo, que usa o Bootstrap como tecnologia *front-end*.

5.6. Projeto Ilustrativo

Para exemplificar o uso dos *frameworks* Django e Bootstrap descritos anteriormente apresentamos, a título de projeto ilustrativo, um sistema Web para controle de empréstimos de equipamentos (CONEQUI), desenvolvido pelo NUPREDS[18] (Núcleo de Práticas Profissionais em Engenharia e Desenvolvimento de Software), do Instituto Federal de Ceará, campus Tianguá. O principal objetivo do sistema é realizar o empréstimo e a devolução de equipamentos disponíveis na secretaria do campus para alunos, professores e funcionários previamente cadastrados, onde a autenticação no sistema e a confirmação das ações são feitas mediante a leitura da impressão digital do usuário.

O diagrama de classes da Figura 5.13 descreve a estrutura do sistema, com suas classes, atributos, operações e as relações entre os objetos. Para exemplificar os conceitos mostrados anteriormente, utilizaremos apenas as classes EquipmentType (Tipo de Equipamento) e Equipment (Equipamento) do diagrama. Ambas são utilizadas nos casos de uso Manter Tipo de Equipamento e Manter Equipamento, modelados cada um na

[18] https://nupreds.ifce.edu.br/

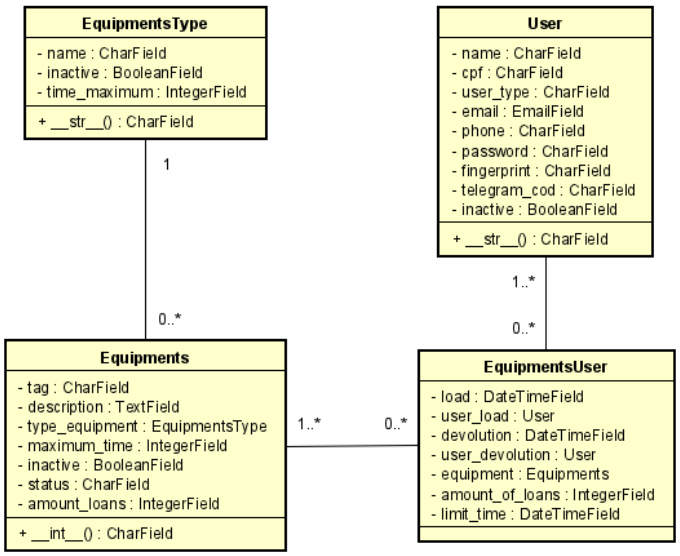


Figura 5.13. Diagrama de Classes do sistema CONEQUI

forma de um CRUD (*Create*, *Read*, *Update* e *Delete*), paradigma que contempla as quatro operações principais executadas em um banco de dados em uma única funcionalidade de sistema.

```
1   from django.db import models
2
3   class Equipment_type(models.Model):
4       name = models.CharField(max_length=20, unique=True)
5       inactive = models.BooleanField(default=False)
6       time_maximum = models.IntegerField(default=0)
7
8       def __str__(self):
9           return self.name
10
11  class Equipment(models.Model):
12      tag = models.CharField(max_length=10)
13      description = models.TextField()
14      type_equipment = models.ForeignKey('Equipment_type',on_delete=models.SET_NULL)
15      maximum_time = models.IntegerField(default=5)
16      inactive = models.BooleanField(default=False)
17      status =  models.CharField(max_length=9,null=False,default='Livre')
18      amount_of_loans = models.IntegerField(default=0)
19
20      def __int__(self):
21          return self.id
22
```

Figura 5.14. Arquivo models.py

A partir do diagrama de classes foram construídos os *models* do Django no arquivo models.py, exibidos na Figura 5.14. A classe Equipment_type (linha 3) e a classe Equipment (linha 11) possuem todos os atributos de suas respectivas representações no diagrama de classes. As classes associam-se através do atributo type_equipment da classe Equipment (linha 14), atributo este que é um objeto do tipo Equipment_type. Uma vez que são especificadas as classes do sistema e seus relacionamentos no arquivo models.py não é necessário preocupar-se com as tabelas no banco de dados, já que o próprio Django ocupa-se disso, de forma transparente para o desenvolvedor, através de seu ORM.

O mapeamento URL/*view* é descrito no arquivo urls.py, como pode ser observado na Figura 5.15. Nele, estão presentes todos os *paths* referentes à classe Equipment, onde

```python
from django.urls import path
from equipments import views

urlpatterns = [
    path('', views.equipment_list, name='equipment_list'),
    path('Ver/<int:pk>', views.equipment_view, name='equipment_view'),
    path('Novo', views.equipment_create, name='equipment_new'),
    path('Editar/<int:pk>', views.equipment_update, name='equipment_edit'),
    path('Inativar/<int:pk>', views.equipment_delete, name='equipment_delete'),
]

```

Figura 5.15. Arquivo urls.py

os quatro últimos correspondem às funções do CRUD Manter Equipamentos: Ver (*Read*), Novo (*Create*), Editar (*Update*) e Inativar (que corresponde ao *Delete*, uma vez que a remoção de equipamentos neste sistema é apenas lógica).

```python
from django.shortcuts import render, redirect, get_object_or_404
from equipments.models import *

def equipment_list(request, templete_name='equipments/equipment_list.html'):
    data = {}
    data['list_equipment'] = Equipment.objects.all().order_by('status','tag')
    return render(request, templete_name, data)

def equipment_view(request, pk, template_name='equipments/equipment_detail.html'):
    equipment = get_object_or_404(Equipment,pk=pk)
    return render(request, template_name, {'object':equipment})

def call_delete_template(request,pk,template_name='equipments/equipment_delete.html'):
    equipment = get_object_or_404(Equipment,pk=pk)
    return render(request, template_name, {'object':equipment})

def equipment_delete(request,pk):
    equipment = Equipment.objects.filter(id = pk).delete()
    return redirect('/')

```

Figura 5.16. Arquivo views.py

As *views*, fazem a recuperação de registros do banco através dos objetos definidos em models.py. Na Figura 5.16 estão algumas *views* do CRUD Manter Equipamentos. Por exemplo, a função equipment_list (linha 4) faz a busca de todos os objetos do tipo Equipamento e exibe esta listagem no template HTML (equipment_list.html) por meio da função *render*. A função equipment_view (linha 9), que corresponde à operação *Read* do CRUD Manter Equipamentos, recebe como parâmetro o identificador do objeto (pk) para recuperá-lo do banco e, sem seguida, o encaminha para ser renderizado no template HTML correspondente (equipment_detail.html). A função equipment_delete (Corresponde ao Delete do CRUD) (linha 17), recebe o identificador de um objeto do tipo Equipamento (pk) para realizar a busca e a inativação deste objeto no banco.

Os templates HTML são renderizados através da função *render* da view. Na Figura 5.17, a página equipment_list.html é renderizada pela função equipment_list da views.py. Nos templates HTML, é possível manipular os objetos por meio de funções Python. Por exemplo, na linha 11, temos um laço para percorrer todos os objetos que foram recebidos pelo template, criando assim uma tabela com todos os objetos do tipo Equipamento cadastrados no sistema. Na linha 16, temos a chamada à URL equipment_view, seguida por um parâmetro, que é o número de identificação do equipamento. Isso vai permitir que o usuário visualize os dados de um equipamento específico. Outras ações possíveis nesta tela são a edição e inativação (remoção lógica) de um equipamento específico (linhas 17

```
<h1>Equipments</h1>
<table border="1">
<thead>
    <tr>
    <th>Etiqueta</th>
    <th>Descrição</th>
    <th>Ações</th>
    </tr>
</thead>
<tbody>
    {% for equipment in list_equipment %}
        <tr>
            <td>{{ equipment.tag }}</td>
            <td>{{ equipment.description }}</td>
            <td>
            <a href="{% url "equipment_view" equipment.id %}">Visualizar</a>
            <a href="{% url "equipment_edit" equipment.id %}">Editar</a>
            <a href="{% url "equipment_delete" equipment.id %}">Inativar/Ativar</a>
            </td>
        </tr>
    {% endfor %}
</tbody>
</table>
<a href="{% url "equipment_new" %}">New</a>
<a href="{% url "home" %}">voltar</a>
```

Figura 5.17. Template HTML para listagem de equipamentos

e 18). A Figura 5.18 exibe a tela resultante deste template.

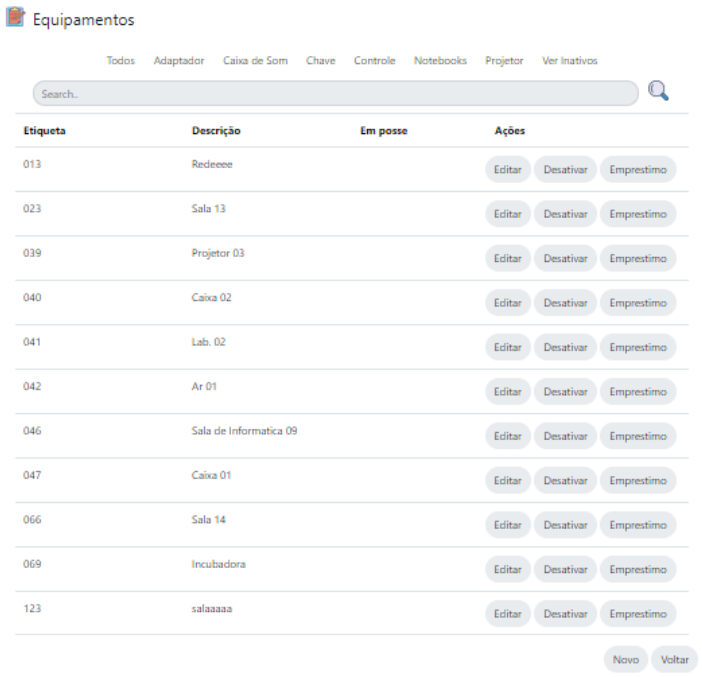


Figura 5.18. Renderização do template de listagem de equipamentos

Na tela da Figura 5.18, vários componentes Bootstrap foram utilizados. Para a barra de pesquisa, por exemplo, foi usado um formulário para que os usuários possam pesquisar equipamentos através de sua etiqueta e/ou descrição. Logo abaixo desta barra responsiva, é exibida uma tabela também responsiva contendo os equipamentos cadastrados que correspondem aos termos da busca.

A responsividade é proporcionada pelo mecanismo Bootstrap que permite que escolhamos quantas colunas da *grid* serão necessárias para exibir os elementos da página,

```
46  <div class="container-fluid">
47    <form class="form-padrao" method="POST" action="{% url 'search' type %}">{% csrf_token %}
48      <div class="input-group row">
49        <div class="col-lg-9 col-md-10 col-sm-12">
50          <input class="form-control mr-2 rounded-pill"  type="text" placeholder="Search.." name="search"
51          required="required" style="background: #e9ecef">
52        </div>
53        <div class="col-lg-1 col-md-3 col-sm-6">
54          <button class="border-0" type="submit" style="background: #FFFFFF;">
55            <span>
56              <img src="{% static "images/lupa.png" %}" style="height: 30px; width: 30px">
57            </span>
58          </button>
59        </div>
60      </div>
61    </form>
62  </div>
```

Figura 5.19. Código HTML da barra de pesquisa

considerando cada um dos tamanhos de dispositivos possíveis (sm para small, md para medium e lg para large), como pode ser observado nas linhas 49 e 53 da Figura 5.19. Ainda nesta Figura, também podemos observar que é possível customizar os botões, inclusive incluindo imagens (linhas 54 a 58).

```
60  <table class="table table-sm" style=" margin:auto; margin-top: 10px;" >
61    <thead>
62      <tr class = "d-flex">
63        <th class="col-lg-3 col-md-4 col-sm-12">
64          <a href="{% url "filter_list_equipment" 'Etiqueta' type %}" style="color: black;">Etiqueta</a>
65        </th>
66        <th class="col-lg-3 col-md-4 col-sm-12">
67          <a href="{% url "filter_list_equipment" 'Descricao' type %}" style="color: black;">Descrição</a>
68        </th>
69        <th class="col-lg-2 col-md-3 col-sm-9">
70          <a href="{% url "filter_list_equipment" 'EmPosse' type %}" style="color: black;">Em posse</a>
71        </th>
72        <th class="col-lg-4 col-md-6 col-sm-12">
73          <a href="" style="color: black;">Ações</a>
74        </th>
75        </tr>
76      </thead>
77    <tbody>
78    {% for equipment in list_equipment %}
79      <tr>
80        <td class="col-lg-3 col-md-4 col-sm-12">{{ equipment.tag }}</td>
81        <td class="col-lg-3 col-md-4 col-sm-12">{{ equipment.description }}</td>
82        {% load my_tag %}
83        <td class="col-lg-2 col-md-3 col-sm-9">{% NameUser equipment.id %}</td>
84        <td class="col-lg-4 col-md-6 col-sm-12">
85            <button class="btn rounded-pill" style="background: #e9ecef">
86                Editar
87            </button>
88            <button type="button" class="btn rounded-pill" data-toggle="modal" data-target="#ModalLabel{{equipment.id}}" style="margin-left: 5px;background: #e9ecef">
89            {% ActiveOrInactive equipment.id %}
90            </button>
91            <button class="btn rounded-pill" data-toggle="modal" data-target="#Modal{{equipment.id}}"style="margin-left: 5px; background: #4de7ff; color: white;">
92            {% else %}
93            <button type="submit" class="btn rounded-pill" data-toggle="modal" data-target="#Modal{{equipment.id}}"style="margin-left: 5px; background: #e9ecef">
94            {% endifequal %}
95            {% LoanOrDevolution equipment.id %}
96            </button>
97        </td>
98    </tbody>
99  </table>
```

Figura 5.20. Código HTML com tabela de listagem de equipamentos

Na Figura 5.20, observa-se o código da tabela HTML onde é exibida a listagem de equipamentos. A disposição dos campos da listagem nesta tabela é determinada pela identificação do tamanho do dispositivo (sm, md ou lg) e pela quantidade de colunas da *grid* necessárias para exibi-los na tela, através do atributo *class* da *tag* <*td*>.

Já na Figura 5.21, pode-se observar um exemplo do uso de modais no nosso sistema. Neste caso, o modal (pequena janela destacada) está sendo utilizado para solicitar ao usuário que informe sua impressão digital para concluir o empréstimo de um equipamento escolhido na tela anterior, que é a que está logo abaixo do modal (tela de listagem de equipamentos). Ao clicar com o botão “Emprestar” na tela de listagem de equipamentos, o modal é acionado e é solicitada a digital do usuário.

Conforme o código na Figura 5.22, para se exibir um modal é necessário um botão ou link que o acionará após ser clicado. *Data-toggle* é usado para ligar o elemento ao tipo

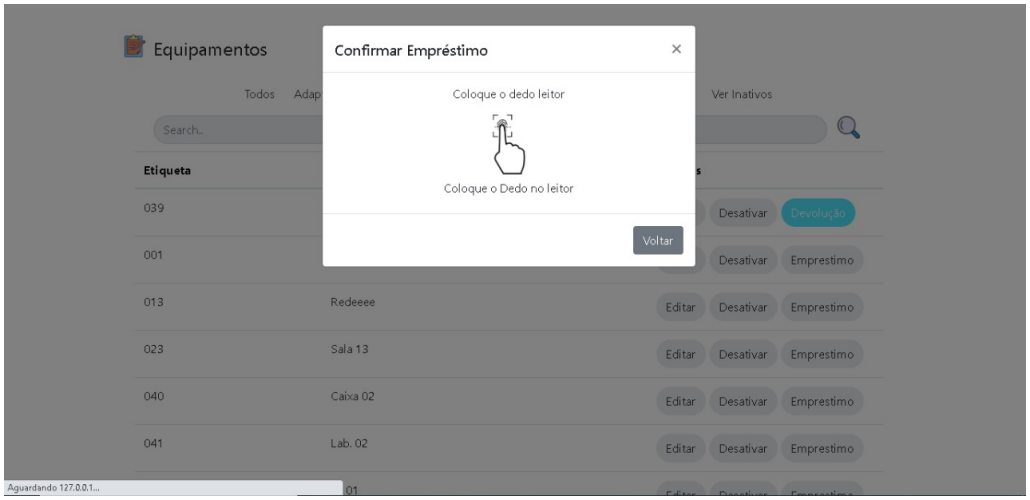


Figura 5.21. Modal para solicitação de impressão digital

```
<button type="submit" class="btn rounded-pill" data-toggle="modal" data-target="#Modal{{equipment.id}}"style=" background: #e9ecef">
    Emprestar
</button>
```

Figura 5.22. Código HTML do botão que aciona o modal

correspondente e é esse atributo que abre a janela modal. Já o *data-target* é usado junto ao modal para referência ao seu alvo, ou seja, funciona como uma forma de ID.

```
90  <div class="modal fade" id="Modal{{equipment.id}}" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
91    <div class="modal-dialog">
92      <div class="modal-content">
93        <div class="modal-header">
94          <h5 class="modal-title" id="exampleModalLabel">Confirmar Empréstimo </h5>
95            <button type="button" class="close" data-dismiss="modal" aria-label="Close">
96              <span aria-hidden="true">&times;</span>
97            </button>
98        </div>
99        <div class="modal-body text-center">
100         {% for message in messages %}
101           <div class="alert alert-danger">
102             <strong>Error!</strong> {{ message }}
103           </div>
104           {% endfor %}
105           <p >Coloque o dedo leitor</p>
106           <span><img src={% static "images/digital.png" %} style="height: 80px; width: 80px;" ></span>
107           {% ifequal equipment.status 'Livre' %}
108           <form name="myForm{{equipment.id}}" id="myForm{{equipment.id}}" method="post" action="{% url "emprestar" equipment.pk %}">{% csrf_token %}
109           {% else %}
110           <form name="myForm{{equipment.id}}" id="myForm{{equipment.id}}" method="post" action="{% url "devolver" equipment.pk %}">{% csrf_token %}
111             {% endifequal %}
112             <button type="submit" class="btn border-0">Coloque o Dedo no leitor</button>
113           </form>
114       </div>
115         <div class="modal-footer">
116           <button type="button" class="btn btn-secondary" data-dismiss="modal">Voltar</button>
117         </div>
118     </div>
119   </div>
120 </div>
```

Figura 5.23. Código HTML do modal

A figura 5.23 corresponde ao código HTML da janela modal. A *div* principal (linha 90) deve ter um ID igual ao valor do atributo de destino de dados usado para acionar o modal (“Modalequipment.id”). A classe *modal* percebe o conteúdo da *div* como modal e coloca o foco nele. A classe *fade* adiciona um efeito de transição. O atributo role igual a *dialog* melhora a acessibilidade para pessoas que usam leitores de tela. A classe *modal-dialog* (linha 91) define a largura e a margem adequadas ao modal. A classe *modal-content* (linha 92) modifica o modal quanto à borda, cor de plano de fundo etc. Dentro

desta *div*, adiciona-se o cabeçalho, o corpo e o rodapé do modal. A classe *modal-header* na linha 93 é usada para definir o estilo do cabeçalho do modal. A classe *modal-title* na linha 94 é usada para dar titulo ao modal. Na linha 95, *data-dismiss* fecha o modal se o usuário clicar nele. A classe *close* estiliza o botão Fechar e a classe *modal-title* estiliza o cabeçalho com uma altura de linha adequada. A classe *modal-body* na linha 99 é usada para definir o estilo do corpo do modal. Nas linhas 100 a 114 está o conteúdo do modal, onde podemos adicionar qualquer marcação HTML como parágrafos, imagens, vídeos etc.

A documentação e a codificação completa deste projeto estão disponíveis para consulta em https://github.com/daniel-root/nupreds.

5.7. Considerações finais

A maneira como o mercado de software desenvolveu-se e a presença de sistemas computacionais na Web fez com que a indústria de software focasse cada vez mais na qualidade e eficiência de seus produtos, utilizando-se da Engenharia de Software como uma das maneiras de garantir qualidade e confiabilidade durante desenvolvimento de produtos de software. Contudo, além da qualidade e da agilidade, os desenvolvedores buscam velocidade e confiança na entrega dos produtos, por meio da reusabilidade de componentes de software durante o processo de construção de um sistema Web.

Diante da necessidade de um desenvolvimento de sistemas Web de forma mais dinâmica e eficiente, este capítulo apresentou uma solução Web completa desenvolvida com a linguagem Python e os *frameworks* Django e Bootstrap, uma vez que ambos viabilizam o uso do conceito de reúso de software. O crescimento e avanço da Internet e os custos cada vez mais baixos de serviços de computação com armazenamento em nuvem viabilizam uma explosão no crescimento do número de aplicações para o ambiente da Web e, apenas o uso de metodologias não garante a qualidade do produto final. Manter a cultura de entrega contínua de software é tão importante quanto o correto uso da Engenharia de Software durante a construção da aplicação para garantir que o dinamismo das exigências dos usuários sejam satisfatoriamente atendidas.

O capítulo abordou algumas das principais tecnologias para desenvolvimento Web vinculadas à linguagem Python e expos, por meio de uma aplicação prática, o uso e a integração da linguagem com os *frameworks* apresentados. O intuíto foi incentivar os novos desenvolvedores a perpetuarem o uso de melhores práticas de desenvolvimento de sistemas web, garantindo sempre eficiência na entrega e qualidade no produto final.

Referências

[Almeida 2019] Almeida, E. S. d. (2019). Software Reuse and Product Line Engineering. In Cha, S., Taylor, R. N., and Kang, K., editors, *Handbook of Software Engineering*, pages 321–348. Springer International Publishing, Cham.

[Bacinger 2020] Bacinger, T. (2020). What is bootstrap? a short bootstrap tutorial on the what, why, and how. [Online; acesso 20-Julho-2020].

[Bootstrap 2020a] Bootstrap (2020a). Documentação boostrap. [Online; acesso 18-Julho-2020].

[Bootstrap 2020b] Bootstrap (2020b). Sistema grid. [Online; acesso 25-Julho-2020].

[Bootstrap 2020c] Bootstrap (2020c). Tipografia. [Online; acesso 26-Julho-2020].

[de Leone 2017] de Leone, L. (2017). Bootstrap: o que é, porque usar e como começar com o framework. [Online; acesso 19-Julho-2020].

[Demchenko 2019] Demchenko, M. (2019). Six huge tech companies that use python: Does it fit your project? [Online; accessed 09-junho-2020].

[Dragos-Paul and Altar 2014] Dragos-Paul, P. and Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*, 69:1172–1179.

[Fawcett 2019] Fawcett, A. (2019). A Beginner's Guide to Web Development.

[Foundation 2020a] Foundation, D. S. (2020a). Documentação do django. [Online; accessed 09-junho-2020].

[Foundation 2020b] Foundation, P. S. (2020b). The python package. [Online; accessed 09-junho-2020].

[Frakes and Fox 1996] Frakes, W. and Fox, C. (1996). Quality improvement using a software reuse failure modes model. *IEEE Transactions on Software Engineering*, 22(4):274–279.

[Freeman 2017] Freeman, A. (2017). *Pro ASP.NET Core MVC 2*. Apress, London, U.K.

[Gamma et al. 1995] Gamma, E., Helm, R., Johnson, R. E., and Vlissides, J. (1995). *Design Patterns*. Prentice Hall.

[Ghimire 2020] Ghimire, D. (2020). Comparative study on Python web frameworks: Flask and Django.

[Holdener 2008] Holdener, A. (2008). *Ajax : the definitive guide*. O'Reilly, Farnham.

[ISBRASIL 2017] ISBRASIL (2017). O que é bootstrap? [Online; acesso 21-Julho-2020].

[jQuery 2020] jQuery, F. (2020). jquery write less, do more. [Online; acesso 25-Julho-2020].

[Manhas 2017] Manhas, J. (2017). Initial framework for website design and development. *International Journal of Information Technology*, 9(4):363–375.

[Molina-Ríos and Pedreira-Souto 2020] Molina-Ríos, J. and Pedreira-Souto, N. (2020). Comparison of development methodologies in web applications. *Information and Software Technology*, 119:106238.

[O'Neil 2008] O'Neil, E. J. (2008). Object/relational mapping 2008: hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1351–1356, Vancouver, Canada. Association for Computing Machinery.

[Schmidt et al. 2004] Schmidt, D. C., Gokhale, A., and Natarajan, B. (2004). Leveraging Application Frameworks. *Queue*, 2(5):66–75.

[Sommerville 2016] Sommerville, I. (2016). *Software engineering*. Pearson, Harlow Singapore.

[Stephens 2020] Stephens, R. (2020). Redmonk slackchat: January 2020 programming language rankings. *Procedia Engineering*.

Capítulo 6

Soluções de Aprendizado de Máquina para Estimar em Tempo Real a Pose Humana em Aplicações de Saúde

Renan Nascimento (UFDPar), José Everton Fontenele (UFDPar), Rodrigo Baluz (UFPI/UESPI), Rayele Moreira (UFPI/UNINTA), Silmar Teixeira (UFDPar/UFPI), Ariel Teles (IFMA/UFDPar/UFMA)

Abstract

Artificial intelligence applications have been proposed as a solution to several health problems. Machine learning algorithms for computer vision have been used to aid in the diagnosis, monitoring and treatment of problems that affect people's health. This short course aims to introduce the TensorFlow library and models applied to real-time recognition of points in the body joints and segments, which represent the position of the human body, contributing to the development of solutions in the area of health informatics. To exemplify, two TensorFlow-based applications are presented: the first one is proposed to assist in postural and range of motion assessment; the second one is a serious game used in the neurofunctional rehabilitation process in upper limbs for post-stroke patients in domestic settings and rehabilitation clinics.

Resumo

Aplicações de inteligência artificial vêm sendo propostas como solução para diversos problemas na área da saúde. Algoritmos de aprendizado de máquina para visão computacional vêm sendo usados para auxílio no diagnóstico, monitoramento e tratamento de problemas que afetam a saúde de pessoas. Este minicurso visa introduzir a biblioteca TensorFlow e modelos aplicados ao reconhecimento em tempo real de pontos nas articulações e segmentos corporais, que representam a posição do corpo humano, contribuindo para o desenvolvimento de soluções na área de informática em saúde. Para exemplificar, são apresentadas duas aplicações que fazem uso do TensorFlow: uma para auxiliar na avaliação postural e da amplitude de movimento; a outra, um jogo sério usada no processo de reabilitação neurofuncional em membros superiores para pacientes pós-AVC em ambientes domésticos e clínicas de reabilitação.

6.1. Introdução

O corpo humano é organizado em sistemas formados por estruturas de variadas complexidades. Dentre eles, os sistemas esquelético e articular, que são interdependentes. O primeiro é construído por ossos de tamanhos variados que se conectam, dando origem ao que conhecemos como articulações do corpo, que por sua vez, formam o sistema articular. Ossos e articulações possuem formatos variados e irregulares, dependendo da região corporal a qual pertencem. Do ponto de vista funcional, os ossos funcionam como alavancas e as articulações como eixos, ajudando na mobilidade do corpo humano [Graaff and Marshall 2003]. Anatomicamente, ossos e articulações são usados como ponto de referência em processos avaliativos, tais como é feito na avaliação antropométrica, método por meio do qual se faz a medição do tamanho do corpo ou de suas partes individualmente. Esse processo é tradicionalmente feito usando uma fita métrica para mensurar o tamanho dos segmentos (e.g., braço, perna, tronco), usando saliências/proeminências ósseas como pontos anatômicos de referência [Sabharwal and Kumar 2008].

Além da antropometria, a identificação de pontos e segmentos anatômicos é importante para outros processos que envolvem estimativa da pose humana, tais como a avaliação postural, que permite identificar alterações no alinhamento corporal e a avaliação goniométrica, técnica de avaliação da amplitude de movimento articular. Apesar do método clínico direto usando fita métrica ser considerado de fácil aplicação, ele depende da identificação manual desses pontos de referência e essa avaliação pode ficar sujeita a vieses, dependendo da experiência do examinador. Além dessas avaliações, a estimativa de pose humana também pode ser útil em processos de avaliação dinâmica, como avaliação da marcha e em processos terapêuticos que visem ganho de mobilidade, como na reabilitação motora. Além dos métodos manuais de avaliação que envolvem inspeção, palpação e mensuração com a fita métrica, os métodos baseados em modelos computacionais têm sido empregados como ferramentas de avaliação, tais como a fotogrametria, a qual pode fazer uso de Visão Computacional (VC).

A interação entre homem e máquinas faz parte do cotidiano das pessoas no mundo industrializado e essa relação tem evoluído de tal forma que hoje já falamos de máquinas que conseguem enxergar e fazer coisas tão bem ou melhor que o ser humano. Hoje temos dispositivos com capacidade de processamento cada vez maior e com funções variadas que vão além daquelas que dispunham na sua concepção original. A interação homem-máquina atingiu um nível no qual é possível ser guiado por elas por lugares desconhecidos sem se perder, ser orientado sobre condições climáticas, fazer reconhecimento facial de animais e humanos, entre outras funções. Essa habilidade das máquinas enxergarem, processarem e interpretarem imagens é conhecida como VC [Forsyth and Ponce 2002].

No caso da fotogrametria, que é um método computadorizado que pode ser usado para estimar postura, amplitude de movimento e medidas antropométricas, a base é o processamento de uma fotografia do paciente, realizadas na vista anterior, posterior e laterais. Métodos como esse alavancaram o uso da VC para além do processamento de imagens médicas (e.g., tomografia, Raio-X, ressonância magnética), passando a ser usada como recurso para estimativa e avaliação dinâmica do ser humano. Além da fotogrametria, o uso de VC pode ser também aplicado na técnica de gameterapia para a reabi-

litação de pacientes com sequelas motoras após Acidente Vascular Cerebral (AVC). A gameterapia se apresenta como um método auxiliar que utiliza cenários interativos dos jogos sérios para potencializar a humanização do tratamento e a experiência do paciente durante a execução dos exercícios necessários para a recuperação das funções motoras [Domínguez-Téllez et al. 2020].

Avanços computacionais têm permitido utilizar máquinas em tarefas variadas que incluem diferentes níveis de complexidade. Boa parte desses avanços estão associados ao uso de algoritmos de aprendizado de máquina (do inglês, *Machine Learning* - ML) que processam e interpretam datasets de forma inteligente. Baseado no processamento de datasets de uma amostra, algoritmos treinados identificam padrões existentes e, a partir deles, fazem inferências à medida que novos dados são incorporados ao sistema [Aneja et al. 2019]. Esses algoritmos ampliaram ainda mais o alcance dos modelos de VC, permitindo interpretar diferentes características de imagens e dados, extraindo deles informações que permitam compreendê-las, interpretá-las e aplicá-las a algum propósito. A possibilidade de processar grande conjuntos de dados e extrair deles informações úteis, para alcançar um dado objetivo de forma mais rápida e precisa, fazem com que esses algoritmos sejam usados em diferentes aplicações de saúde.

A partir do uso de algoritmos de ML, processos de identificação dos pontos e segmentos anatômicos passam a ocorrer de forma automatizada. Essa automatização pode favorecer o desenvolvimento de instrumentos e métodos terapêuticos dinâmicos que podem facilitar a recuperação e adesão dos pacientes ao tratamento. Nesse contexto, o objetivo deste capítulo é introduzir a biblioteca *TensorFlow* (TF) e suas extensões aplicadas ao reconhecimento em tempo real de estruturas e movimento do corpo humano, além de apresentar algumas de suas possíveis contribuições para as pesquisas na área de informática em saúde.

Este capítulo está organizado da seguinte maneira. A Seção 6.2 apresenta a biblioteca TF. Na Seção 6.3, são apresentados duas extensões de VC do TF, o *PoseNet* e o *MediaPipe*, para a estimação da pose humana em tempo real, enquanto a Seção 6.4 ilustra exemplos de aplicações de saúde que fazem uso dessas extensões. Por fim, a Seção 6.5 conclui o capítulo com nossas considerações.

6.2. A Biblioteca *TensorFlow*

O TF foi desenvolvido originalmente por pesquisadores e engenheiros que trabalham na equipe do *Google Brain*, dentro da organização *Machine Intelligence Research* da Google, para realizar pesquisas em ML[1]. É uma ferramenta poderosa para treinamento em larga escala, pois utiliza eficientemente centenas de servidores, os quais possuem unidades de processamento gráfico (do inglês, *Graphics Processing Units* - GPUs) e unidade de processamento de tensor (do inglês, *Tensor Processing Units* - TPUs) para treinamento e execução de modelos treinados [Abadi et al. 2016]. As GPUs são processadores dedicados ao processamento e renderização de gráficos em tempo real. Já as TPUs são unidades de processamento de algoritmos de Inteligência Artificial (IA) e Redes Neurais Convolucionais (do inglês, *Convolutional Neural Networks* - CNNs).

[1] `https://github.com/tensorflow/tensorflow`

Em particular, o TF é especializado em Redes Neurais Artificiais (do inglês, *Artificial Neural Networks* - ANNs). Uma ANN é uma ferramenta de inteligência computacional inspirada no sistema neural humano [Basheer and Hajmeer 2000], que visa replicar o processo de comunicação entre neurônios, gerando um modelo de estruturas interconectadas capazes de realizar cálculos para processamento de dados e representação de conhecimento. Uma CNN é um algoritmo de aprendizado profundo que pode captar uma entrada, atribuir importância (i.e., pesos e vieses que podem ser aprendidos) a vários aspectos/objetos do elemento de entrada, e ser capaz de diferenciar ou classificar um do outro.

6.2.1. *Google Colaboratory*

O *Google Colaboratory*, ou simplesmente *Colab*, é uma plataforma executada em nuvem e funciona em tempo real, contendo servidores com GPUs e TPUs. A plataforma foi desenvolvida para tornar mais fácil o trabalho de estudantes, cientistas de dados e pesquisadores da área de IA. A interface do *Colab* apresenta resultados de processamento dos algoritmos em tempo real, o que torna a experiência do usuário (i.e., do desenvolvedor) mais dinâmica. Além disso, ele permite que códigos possam ser compartilhados e os usuários possam trabalhar colaborativamente ao mesmo tempo [Google 2019]. O *Colab* é baseado no *Jupyter Notebook* [Carneiro et al. 2018].

O *Colab* é uma plataforma de ambiente interativo e vem configurada por padrão com a linguagem *Python*. Além disso, ele já vem com as principais bibliotecas úteis para ML prontas para uso, como o TF, o *Pandas*, e o *Matplotlib* [Google 2019]. O ambiente de desenvolvimento do *Colab* é composto por notebooks (i.e., um documento utilizado para o desenvolvimento de algoritmos na plataforma com a extensão .ipynb). Um notebook é composto por lista de células. As células de código são onde se pode escrever e executar códigos fonte, além de visualizar as saídas, que mostram os resultados após a execução. Os resultados mais comuns são textos, tabelas e gráficos [Carneiro et al. 2018]. As células também podem conter textos explicativos que são formatadas usando uma linguagem de marcação chamada *markdown*.

A partir de uma conta do *Google*, utilizando o *Google Drive* ou acessando diretamente o site do *Colab*, é possível criar um novo notebook, como ilustrado na Figura 6.1. Para criar um novo notebook, primeiro clicamos em novo, depois em Mais->Google Colaboraotory.

Abaixo é mostrado uma célula de código (Figura 6.2) usada para a escrita e execução do código fonte, a qual funciona em tempo de execução e pode ser executado individualmente. A execução do código pode ser feita clicando do botão “Play”. Uma célula de texto (Figura 6.3) é útil para adicionar comentários ao notebook. Para editar um bloco de texto, basta clicar duas vezes na célula.

Abaixo é mostrado dois exemplos de como funcionam as células. No primeiro exemplo (Figura 6.4), são criados dois blocos: o primeiro foi inicializado uma variável **v** recebendo uma soma; no segundo bloco, a função *print()* para exibir o valor da variável, já com o resultado da soma. O outro exemplo (Figura 6.5) é da célula de texto escrita na linguagem de marcação *markdown*. A célula é composta por uma barra de ferramentas para ajudar na edição. Do lado esquerdo a parte de marcação da linguagem e do lado

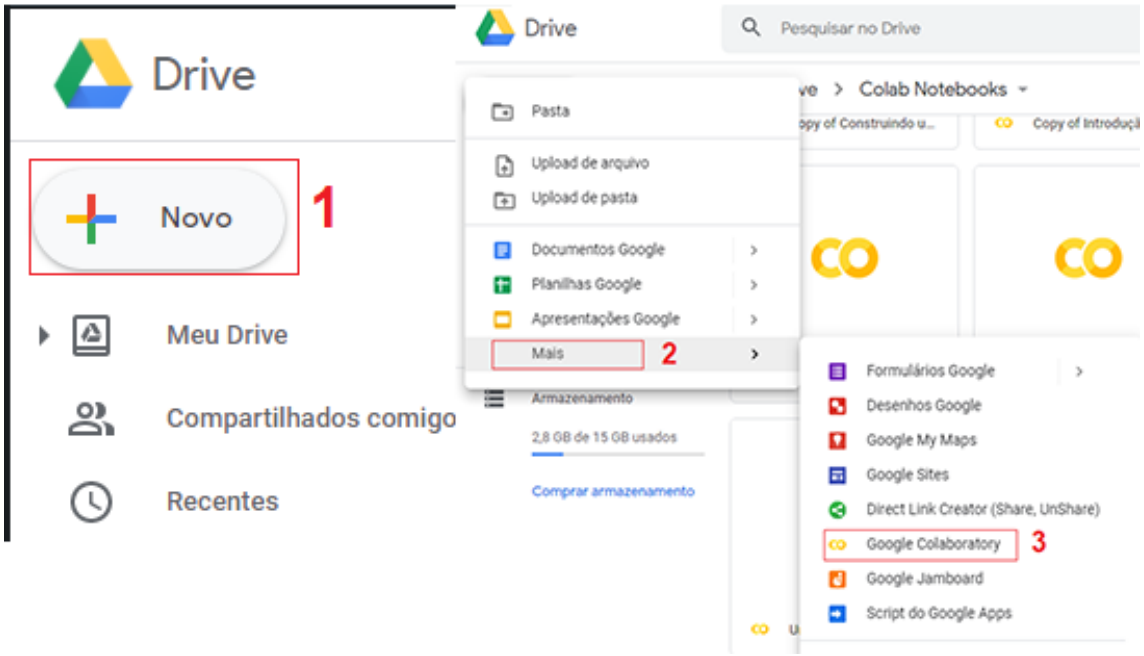


Figura 6.1. Criação de notebook no *Google Colab*.

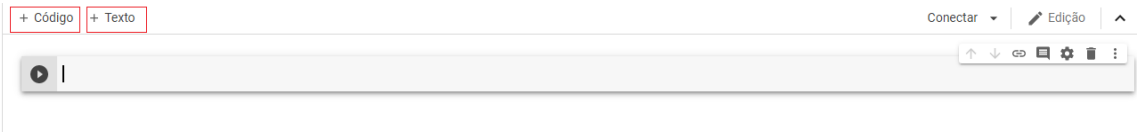


Figura 6.2. Célula para código fonte.

Figura 6.3. Célula para texto usando a linguagem de marcação *markdown*.

direto o resultado.

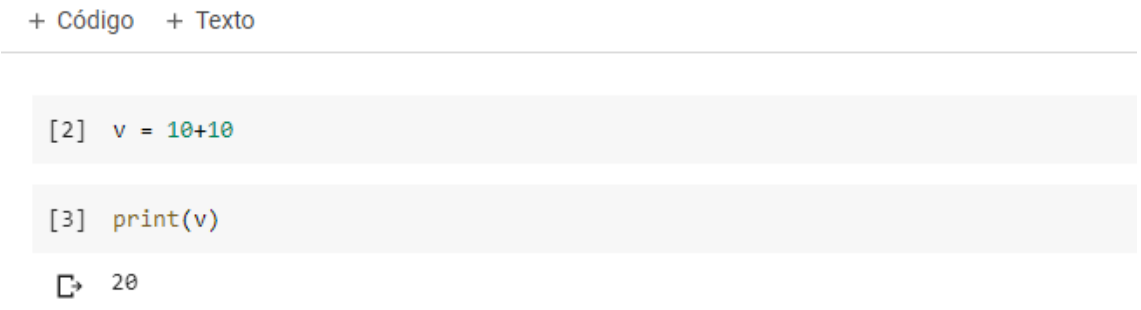


Figura 6.4. Exemplo da célula de código.

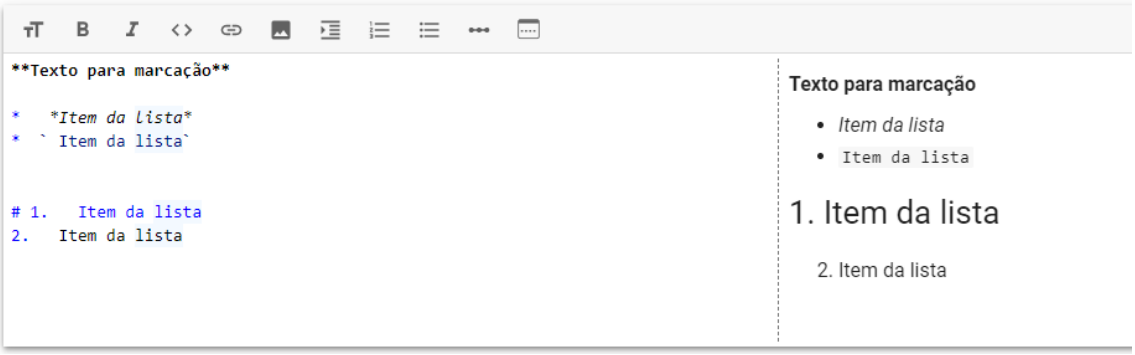


Figura 6.5. Exemplo da célula de texto.

6.2.2. Versões

Até o momento da escrita desse capítulo, o TF está na versão 2.3.0. Nessa seção, serão apresentadas as principais mudanças da versão 1.0 para a 2.0. A primeira versão publicada do TF (v1.0) data de 2018 e já possuía portabilidade para as linguagens *Python*, *JavaScript*, *C++*, *Java*, *Go* e *Swift*, também presentes nas versões mais recentes. Segundo a documentação oficial, a API *Python* é a mais completa, mas as APIs de outras linguagens também podem ser facilmente integradas a projetos e oferecer algumas vantagens de desempenho na execução. Como forma de disseminar a biblioteca, é incentivado à comunidade de desenvolvedores a implementar e manter o suporte para outras linguagens de programação. Dentre elas, tem-se versões em *C#* (i.e., *TensorFlowSharp* e *TensorFlow.NET*), *askell*, *Julia*, *MATLAB*, *R*, *Ruby*, *Rust* e *Scala*.

A versão TF 2.0 veio com uma proposta de simplificar e facilitar seu uso. Uma das principais alterações propostas foi a incorporação do *Keras* (*tf.keras*), a qual é uma API de alto nível do TF. Ela é utilizada para criar e treinar modelos de aprendizado profundo, abordando uma prototipagem rápida e com recursos avançados, tendo como vantagens a facilidade de usar modelos compostos e fácil de entender.

Outras mudanças propostas foram a forma de acesso às variáveis e constantes. Na versão 1.0, a manipulação e atualização de variáveis eram possíveis apenas com a inicialização de uma sessão (*session = tf.Session()*). Então houveram algumas alterações da API, em que muitas classes, funções e tipos de dados foram renomeados ou removidos, e nomes dos argumentos de funções também foram alterados. Por exemplo, na versão 2.0 não existe mais a classe *tf.ConditionalAccumulator*. Houveram também correções no preenchimento automático, recurso agregado à plataforma que completa o código enquanto está digitando. Contudo, todas essas mudanças foram motivadas pela consistência e clareza de sua utilização.

6.2.3. Modelos e Extensões

Um módulo é uma parte autônoma de um grafo do TF, com os respectivos pesos e recursos, que podem ser reutilizados em diferentes tarefas. Dentre os modelos disponíveis, podemos citar os que estão disponíveis nos repositórios *TensorFlow Model Garden*[2],

[2]https://github.com/tensorflow/models

TensroFlow Hub[3] e *TensorFlow.js models*[4]. O *TensorFlow Model Garden* contém os modelos oficiais do TF que usam as APIs de alto nível do TF disponibilizados através do *GitHub*[5]. O *TensorFlow Hub*[6] é um repositório de modelos de ML. Em particular, ele fornece modelos salvos, treinados previamente, que podem ser reutilizados para resolver novas tarefas com menos tempo e menos dados para o treinamento.

Já o *TensorFlow.js models* é um repositório de modelos treinados previamente que foram portados para a linguagem *JavaScript*. Os modelos *TensorFlow.js* são executados no navegador e no ambiente *Node.js*[7]. Alguns dos modelos presentes no repositório são o *PoseNet* para estimativa de pose, o *Coco SSD* para o reconhecimento de objetos, e o *DeepLab V3* para a segmentação semântica, em que a partir de uma imagem é retornado a descrição dos elementos presentes.

6.2.4. Exemplo de Rede Neural Artificial com o *TensorFlow*

Como exemplo, apresentamos o desenvolvimento, treinamento e teste de uma CNN utilizando o TF, disponível em: `https://github.com/RenanFialho-Dev/ercempi_2020`. O *dataset* utilizado para a construção é o *Fashion MNIST*[8], o qual contém um conjunto de treinamento com 60 mil imagens com resolução de 28x28 pixels em escala de cinza, e um conjunto de teste com 10 mil imagens com mesmas características. O dataset possui imagens de roupas e calçados.

Inicialmente, são importadas as bibliotecas TF e *NumPy*. Esta segunda é uma biblioteca para computação numérica, utilizada para operações matemáticas. A partir do TF, é realizada a importação do dataset *MNIST*, o qual é, em seguida, carregado em uma tupla de *arrays* tipados segundo a biblioteca de tipos de dados do *NumPy* (*np.array*). Como ilustrado na Figura 6.6, temos a tupla (*X_treino, Y_treino*) para o conjunto de treinamento com seus rótulos, e *X_teste, Y_teste* para o conjunto de imagens de teste.

```
[14] import numpy as np
     import tensorflow as tf
     from tensorflow.keras.datasets import fashion_mnist
```

Carregando o dataset

```
[11] (X_treino, X_teste), (Y_treino, Y_teste) = fashion_mnist.load_data()
```

Figura 6.6. Bibliotecas importadas e tupla de *arrays* para armazenar o dataset.

Com o dataset carregado, realiza-se a normalização dos dados. O processo de

[3] `https://tfhub.dev/`
[4] `https://www.tensorflow.org/js/models`
[5] `https://github.com/tensorflow/models/tree/master/official`
[6] `https://tfhub.dev/`
[7] Node.js é um interpretador *JavaScript*: `https://nodejs.org/en/`
[8] `https://keras.io/api/datasets/mnist/`

normalização faz com que os dados fiquem dentro de um limite de valores, no nosso caso, entre 0 e 1. Ao realizar esse procedimento, melhoramos a qualidade dos dados, evitando dados ruidosos e garantimos que a nossa CNN irá treinar mais rapidamente. Em seguida, utiliza-se a função *reshape* da biblioteca *NumPy*, a qual realiza uma transposição dos dados para uma nova forma sem que seus dados sejam perdidos. Por exemplo, uma matriz de 4 linhas e 3 colunas após *reshape* ficará com 3 linhas e 4 colunas. Em nosso exemplo, *reshape(-1, 28*28)* retorna uma nova matriz com 60 mil linhas, correspondente a quantidade de imagens que temos, e 784 colunas, que representam os valores de cor em cada pixel, como visto na Figura 6.7.

```
#normalização
X_treino = X_treino/255.0
X_teste = X_teste/255.0

#reshape dos dados
X_treino = X_treino.reshape(-1, 28*28)
X_teste = X_teste.reshape(-1, 28*28)
```

Figura 6.7. Normalização e *reshape* dos dados.

A partir dos dados preparados, construímos a estrutura da nossa CNN. Primeiramente, acessamos a biblioteca *tf.keras* e definimos uma classe do tipo *models* a uma variável, que atribui propriedades de camadas em um objeto com recursos de treinamento e inferência. Seguimos adicionando as configurações para as camadas de entrada, intermediárias (também chamadas de densas, as quais recebem essa definição porque todos os neurônios são conectados aos neurônios da camada seguinte), e camada de saída, utilizando o comando *model.add()*. Cada camada adicionada através da função *tf.keras.layers.Dense()* contará com os hiper-parâmetros (estes são variáveis de configuração contendo os parâmetros que controlam o próprio processo de treinamento): número de unidades/neurônios, função de ativação e tamanho da entrada.

Após a camada de entrada, é adicionada uma camada de *dropout*. Esta é uma técnica de regularização em que definimos aleatoriamente os neurônios de uma camada para zero. Dessa forma, durante o treinamento, esses neurônios não serão atualizados. Como alguma porcentagem de neurônios não será atualizada, todo o processo de treinamento é longo e temos menos chances que a rede aprenda mais sobre o conjunto de treinamento ao invés de aprender a generalizar, chamado também de superajuste (i.e., *overfitting*). Para a camada de saída, o número de unidades/neurônios corresponde a quantidade de classes presentes no dataset (i.e., 10 no *Fashion MNIST*) com uma função de ativação *softmax*, conforme a Figura 6.8.

Com estas configurações, a CNN deve ser compilada, e estará preparada para executar a fase de treinamento e teste. A função *compile* recebe os seguintes parâmetros: um otimizador (*optimizer*), um parâmetro perda (*loss*), e um parâmetro contendo métricas (*metrics)* a serem avaliadas pelo modelo durante o treinamento e teste. O otimizador é um parâmetro que representa uma função de ativação, que será implementando no nosso modelo. Dentro do pacote *keras.optimizers* encontramos as funções que podem ser implementadas dentro do seu conjunto de classes definidas. O parâmetro perda recebe uma

```
[10] #definição do modelo
     model = tf.keras.models.Sequential()
     model.add(tf.keras.layers.Dense(units=32, activation='relu', input_shape=(784, )))
     model.add(tf.keras.layers.Dropout(0.4))
     model.add(tf.keras.layers.Dense(units=64, activation='relu'))
     model.add(tf.keras.layers.Dropout(0.6))
     model.add(tf.keras.layers.Dense(128, activation='relu'))
     model.add(tf.keras.layers.Dense(units=10, activation='softmax'))
```

Figura 6.8. Implementação do modelo de rede do TF.

função do tipo *tf.keras.losses* que visa calcular os erros entre classificações verdadeiras e previstas. Já o parâmetro contento as métricas é utilizado para avaliar o modelo durante o treinamento e o teste, e pode ser dos seguintes tipos: uma sequência de caracteres (nome de uma função interna), uma função ou uma instância *tf.keras.metrics.Metric*.

O treinamento da rede recebe como parâmetros de entrada o conjunto de treinamento (*X_treino*), com suas classes (*Y_treino*) e o número de épocas (*epochs*) para treinar o modelo, como ilustrado na Figura 6.9). Na Figura 6.9 também é mostrado o início do processo de treinamento com a contagem de épocas, tempo, valor de perda e acurácia.

Compilando o Modelo

```
[ ] model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['sparse_categorical_accuracy'])
```

Treinamento

```
model.fit(X_treino, Y_treino, epochs=10)

Epoch 1/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.9707 - sparse_categorical_accuracy: 0.6313
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.7188 - sparse_categorical_accuracy: 0.7365
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.6664 - sparse_categorical_accuracy: 0.7596
```

Figura 6.9. Compilação e treinamento do modelo de CNN.

Finalizado o treinamento da CNN, é possível avaliar a acurácia passando os dados de teste. A função *evaluate* recebe como parâmetro o conjunto de teste (*X_Teste*) e o conjunto de classes de teste (*Y_Teste*), e retorna uma tupla contendo o valor de perda (*error score*) e a acurácia para o conjunto fornecido. Na Figura 6.10 é possível ver que a CNN do exemplo obteve uma acurácia de 81%.

```
Teste_perda, Test_acuracia = model.evaluate(X_teste, Y_teste)

313/313 [==============================] - 0s 1ms/step - loss: 0.5603 - sparse_categorical_accuracy: 0.8125
```

```
[8] print("Acurácia para os dados de teste: {}".format(Test_acuracia))

Acurácia para os dados de teste: 0.8125
```

Figura 6.10. Avaliação da CNN.

6.3. Estimando a Pose Humana em Tempo Real

6.3.1. *PoseNet*

O *PoseNet*[9] é um modelo de VC para a estimativa de pose/postura humana a partir de uma imagem ou vídeo. Ele é um dos modelos disponibilizados pela biblioteca *TensorFlow.js*. Sua tarefa não é apresentar ou identificar quem está presente na imagem, mas sim determinar onde está localizado 17 pontos do corpo humano.

O *PoseNet* compreende duas opções de configuração que podem ser usadas: uma para a identificação de pose para uma única pessoa (i.e., *single pose*) e e outra para múltiplas pessoas (i.e., *multiple poses*). Ambas buscam identificar poses através de imagem ou vídeo. A imagem/vídeo é passada para um modelo de CNN já treinado e, ao final do seu processamento, retorna um objeto JSON do tipo *Pose* para uma pessoa, ou um conjunto de *Poses*, para a configuração de múltiplas pessoas.

Um objeto *Pose* compreende os seguintes dados: um valor de acurácia geral (i.e., um *score* que descreve o quanto o modelo tem confiança dos achados) e os pontos-chave (i.e., os *keypoints*). Um *keypoint* é composto de: coordenadas do tipo *position*, acurácia (i.e, *score*) para o ponto identificado, e a descrição (*part*) do ponto localizado. Existem 17 *keypoints* que o algoritmo é capaz de identificar, são eles: nariz, olho esquerdo, olho direito, orelha esquerda, orelha direita, ombro esquerdo, ombro direito, cotovelo esquerdo, cotovelo direito, pulso esquerdo, pulso direito, quadril esquerdo, quadril direito, joelho esquerdo, joelho direito, tornozelo esquerdo, e tornozelo direito.

Inicialmente, é necessário entender os parâmetros de configuração do modelo do *PoseNet*, e como utilizá-los. Ao iniciar um objeto do tipo *PoseNet*, é necessário definir a arquitetura (*architecture*), resolução/passo de saída (*outputStride*), resolução de entrada (*inputResolution*), multiplicador (*multiplier*), quantidade de bytes (*quantBytes*) e *modelUrl*. A seguir estes atributos são descritos.

- Arquitetura: as arquiteturas presentes são *MobileNetV1* e *ResNet*. Ao instanciarmos um modelo com *MobileNetV1*, estaremos usando um modelo mais leve, rápido e, consequentemente, com acurácia menor, pois foi desenvolvido para ser executado em dispositivos móveis. A *ResNet* possui uma acurácia maior e, portanto, mais lenta. Ambas cumprem o seu papel, a escolha delas depende do poder de processamento do dispositivo onde será executado;

- Resolução/Passo de saída: este atributo pode receber os valores 8, 16 e 32. Para a arquitetura do tipo *ResNet*, os conjuntos disponíveis são 16 e 32. Para esse atributo, quanto menor o valor, maior será a resolução de saída e mais preciso será o modelo. Porém, a velocidade de processamento é comprometida;

- Resolução de entrada: esse atributo especifica o tamanho em que a imagem/vídeo é redimensionada antes de ser inserida no modelo *PoseNet*. A resolução mínima e padrão do modelo é de 257 pixels. Quanto maior o valor, mais precisos serão os resultados, porém se torna mais lento, por consumir mais tempo de processamento.

[9] `https://github.com/tensorflow/tfjs-models/tree/master/posenet`

Ao contrário, quando passado um valor menor como parâmetro, aumenta-se a velocidade e se obtém uma menor precisão;

- Multiplicador: atributo usado apenas pela arquitetura *MobileNetV1*. É um valor numérico que pode ser 1.0, 0.75 ou 0.5. Quanto maior o valor, maior o tamanho das camadas de convolução da CNN, e mais preciso é o modelo, se tornando mais lento;

- *quantBytes*: este argumento controla os bytes usados para quantização dos pesos da CNN. As opções disponíveis são 4, 2 e 1 bytes por quantização;

- *modelUrl*: um parâmetro opcional que pode conter um endereço de URL personalizado do modelo. Ele é útil para o desenvolvimento local ou para países que não têm acesso ao modelo hospedado no *Google Cloud Platform*.

Com o entendimento geral do modelo *PoseNet*, podemos nos aprofundar em utilizá-lo. Desenvolvemos aqui um exemplo para estimar a pose de uma única pessoa (i.e., *single pose*) a partir de uma *webcam* em uma página HTML. Ao passar os dados capturados da *webcam* para o modelo *PoseNet*, veremos como resultado um objeto com os pontos-chave retornados.

Inicialmente criamos uma página HTML chamada *index.html* com uma estrutura padrão HTML5 e importamos os módulos *@tensorflow/tfjs*, *@tensorflow-models/posenet* e *camera.js* por meio da *tag script*, como ilustrado na Figura 6.11. Como pode ser visto, o *PoseNet* é um arquivo *JavaScript* contendo as funções que manipulam o vídeo e a implementação do modelo. Este exemplo completo pode ser encontrado em: `https://github.com/RenanFialho-Dev/ercempi_2020`

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/posenet"></script>
<script src="camera.js"></script>
```

Figura 6.11. Importação dos pacotes do TF, *PoseNet* e *camera.js*.

Dentro do arquivo *camera.js*, criamos a função construtora *Camera*. No construtor, criamos um objeto *videoTag* através do acesso ao elemento vídeo presente no HTML, e outro objeto *videoConfig* para guardar as configurações de vídeo. Como boa prática, as funções e métodos que manipulam nossa classe não estão presentes no construtor. Para evitarmos consumo de memória desnecessário, pois quando um objeto é criado, é alocado memória para ele. Ao invés disso, utilizamos o recurso de *prototype*, que é um objeto associado a uma função construtora. Colocando os métodos neste objeto, todas as instâncias usarão as mesmas cópias de cada método.

No nosso *prototype*, implementamos no objeto câmera duas propriedades e uma *promise*. A propriedade *start* contém uma função anônima que faz acesso ao método *MediaDevices.getUserMedia* solicitando ao usuário permissão para usar uma entrada de mídia. No nosso caso, a *webcam* do dispositivo. Com permissão concedida pelo usuário, recebemos um *mediaStream* contendo o vídeo que é repassado a propriedade *srcObject*

da *tag* video. A Figura 6.12 apresenta a implementação da propriedade *start*, usado para inicializar a câmera no navegador.

```
Camera.prototype = {
    start: function () {
        this.ligado = true

        navigator.mediaDevices.getUserMedia(this.videoConfig).then(function (mediaStream) {
            this.videoTag = document.querySelector('video');
            this.videoTag.srcObject = mediaStream;
        }).catch(function (err) {
            console.log("Não foi possível iniciar o vídeo" + err);
        })
    },
```

Figura 6.12. Método *start* para inicializar a câmera no navegador.

A propriedade *stop* também é executada por uma função anônima. Por ela obtemos o fluxo do elemento de vídeo de sua propriedade *srcObject*, acessando a variável *videoTag*. Em seguida, a lista de faixas do fluxo é obtida chamando seu método *getTracks()*. A partir daí, tudo o que resta a fazer é percorrer a lista de trilhas usando o método *forEach()* e chamando o método *stop()* de cada trilha. Por fim, *srcObject* é definido como nulo para interromper o link para o objeto *mediaStream* para que ele possa ser liberado, como ilustrado na Figura 6.13.

```
    stop: function () {
        this.ligado = false
        const stream = this.videoTag.srcObject
        const tracks = stream.getTracks();

        tracks.forEach(function (track) {
            track.stop();
        });

        this.videoTag.srcObject = null;
    },
```

Figura 6.13. Método *stop* para interromper o fluxo do elemento vídeo.

A função *estimateSinglePoseOnVideo* implementa o modelo do *PoseNet* e retorna um objeto *Pose*. Criamos uma variável *configNet*, que é instanciada com as configurações do modelo, a qual pode ser vista na Figura 6.14. Uma vez que a nossa variável está pronta, acessamos a função *estimateSinglePose* passando os parâmetros de *input*, *flipHorizontal* e *decodingMethod*. A entrada (*input*) é o vídeo que estamos captando, o espelhamento (*flipHorizontal*) recebe um valor verdadeiro ou falso (quando verdadeiro, replica o efeito de movimento espelho) e, por último, o método de identificação que usaremos, o *single-person* (Figura 6.14).

Dentro do corpo (*body*) da nossa página, inserimos as *tags* de vídeo e *canvas*. A *tag* vídeo é responsável por exibir o conteúdo captado pela câmera do dispositivo e o *canvas* estará exibindo o mesmo conteúdo que o vídeo. Através dele acrescentamos marcadores dos pontos-chave encontrados pelo modelo do *PoseNet*. Dentro da *tag script*,

```
async estimateSinglePoseOnVideo () {
    if (this.ligado == true){
        this.configNet = await posenet.load({
            architecture: 'MobileNetV1',
            outputStride: 16,
            inputResolution: { width: 300, height: 300 },
            multiplier: 0.75
        });

        await this.configNet.estimateSinglePose(this.videoTag, { flipHorizontal: false,
        decodingMethod: 'single-person' }).then(result => {
            this.resultPoses = result;
        }).catch((err) => {
            console.log('Falha ao analisar', err);
        });

        this.configNet.dispose();
    }
    return this.resultPoses;
}
```

Figura 6.14. Função assíncrona *estimateSinglePoseOnVideo*.

instanciamos 3 variáveis: *canvas* e *vídeo*, que acessam a interface de programação dos elementos que possuem seus nomes, e a variável *ctx* recebe o contexto de desenho da *tag canvas*. Para realizar o acesso à câmera, criamos um objeto *camera* e realizamos a chamada da função *start*.

Para passarmos os dados capturados pela câmera para o *PoseNet*, acessamos o evento *onprogress* da *tag* vídeo, verificamos se o vídeo não está parado ou finalizado, e então fazemos a chamada da função *estimateSinglePoseOnVideo* do objeto *camera* anteriormente instanciado. Obtemos como retorno o objeto *Pose* contendo as informações dos pontos-chave localizados. Em seguida, passamos o objeto *Pose* para a função *poseDetectionFramePoints()*, que percorrerá o *array* de *keypoints* e passará as coordenadas (x,y) para a função *drawKeypoints()* (Figura 6.15), que realizará o desenho dos pontos na tela.

Para sincronizar a imagem captada pela câmera e a representação do *canvas*, adicionamos um evento *camera.addEventListener* ao vídeo enquanto estiver executando. O contexto do *canvas* executa a função *drawImage()*, que é atualizada frequentemente. E como resultado do nosso exemplo temos o vídeo captado pela câmera e uma réplica do vídeo com marcações de pontos identificados pelo modelo de CNN, como visto na Figura 6.16.

6.3.2. *MediaPipe*

O *MediaPipe*[10] é um *framework open-source* para o desenvolvimento de soluções de ML, desenvolvido e mantido pela *Google*. O *MediaPipe* provê modelos treinados que podem ser reutilizados de forma simples, pois visa ajudar pesquisadores e desenvolvedores a criarem suas próprias soluções. Ele disponibiliza modelos que podem ser implementados em diversas plataformas, tais como *Android*, *iOS*, *Windows*, *Linux* e Web.

A arquitetura do *framework* adota os seguintes conceitos [Lugaresi et al. 2019]: grafos (`Graph`), nós (`Calculators`), pacotes (`Packets`) e fluxos (`Streams`). A

[10] `https://github.com/google/mediapipe`

```
function drawKeypoints(ctx, x, y, r, scale = 1) {
    if (x !== 0 && y !== 0) {
        ctx.beginPath();
        ctx.arc(x, y, r, 0, 2 * Math.PI);
        ctx.fillStyle = '#4DB680';
        ctx.fill();;
    }
}

function poseDetectionFramePoints(pose) {
    this.setValuesOnInput(pose);
    pose.keypoints.forEach(value => {
        if (value.score >= 0.75) {
            drawKeypoints(ctx, value.position.x, value.position.y, 5);
        }
    });
}

function getProgressVideo() {
    video.onprogress = () => {
        if (!video.paused && !video.ended) {
            camera.estimateSinglePoseOnVideo().then(resultPoses => {
                poseDetectionFramePoints(resultPoses)
            });
        }
    }
}
```

Figura 6.15. Funções *getProgressVideo()*, *poseDetectionFramePoints()* e *drawKeypoints()*.

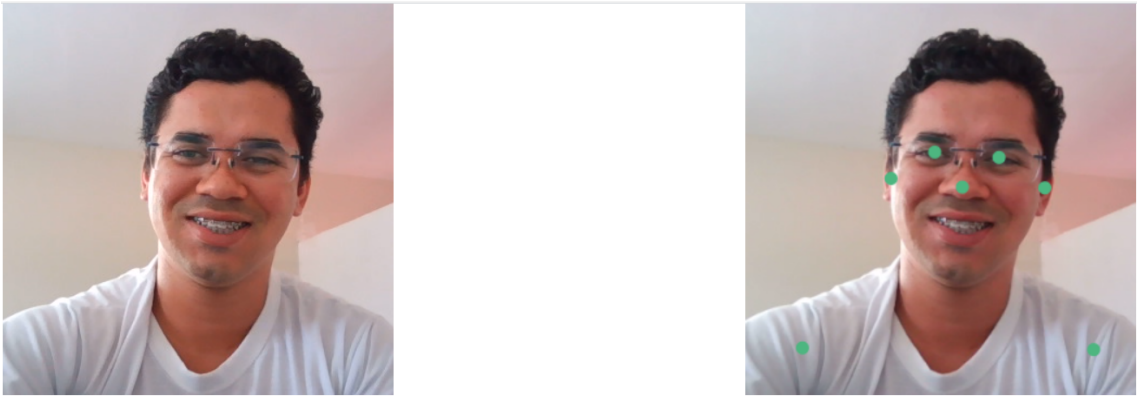


Figura 6.16. Estimativa do *PoseNet* em tempo real.

arquitetura é organizada em uma pipeline de um grafo de nós. No grafo, os nós são conectados por fluxos de dados. Cada fluxo representa uma série temporal de pacotes de dados. Nós e fluxos definem o grafo de fluxo de dados. Os pacotes que fluem pelo grafo são agrupados por seus *timestamps* dentro da série temporal (i.e., eles são as unidades básicas de fluxos de dados). A pipeline é flexível, permitindo inserir ou substituir qualquer nó do grafo, além de customizá-los. Os grafos podem ter qualquer número de entradas e saídas, podendo ramificar ou mesclar os fluxos de dados. Os dados entre os nós fluem de forma bidirecional. Nos nós, é onde boa parte do processamento de um grafo ocorre. Sua função é consumir pacotes e sua interface é responsável por definir o número de portas de entrada e saída para os fluxos, as quais são identificadas por índices.

Dentre os diversos modelos disponíveis no *MediaPipe*[11], existem soluções para a detecção de face (*Face Detection*), malha de rosto (*Face Mesh*), mãos (*Hands Detection*), segmentação de cabelo (*Hair Segmentation*), detecção de objetos (*Object Detection*), dentre outros. Em particular, o *MediaPipe Hands* (MH)[12] é uma solução de rastreamento das mãos e dedos de seres humanos. Nele, foram usadas técnicas de ML para inferir 21 pontos-chave da mão a partir de um único quadro. O método utilizado pelo *MediaPipe* consegue um bom desempenho em tempo real funcionando em *smartphones*. Ele utiliza dois modelos de ML: o primeiro é aplicado à imagem completa e delimita uma caixa ao redor da mão, portanto é utilizado para detectar a mão; já o segundo modelo trabalha somente na região delimitada previamente para identificar pontos-chave.

Nesta seção daremos ênfase ao modelo MH. A seguir é mostrado um exemplo prático de construção de uma página Web para a detecção de uma das mãos. Este exemplo pode ser acessado em: `https://github.com/RenanFialho-Dev/ercempi_2020`. Por meio de uma *webcam* acessada por uma página HTML, os dados captados pela *webcam* são passados para o modelo *HandPose*. Como resultado, tem-se 21 pontos-chave. Para esse exemplo, foi criado uma página HTML5 e importadas as bibliotecas *TensorFlow* e *HandPose* por meio da tag *script* (Figura 6.17).

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/handpose"></script>
<script src="camera.js"></script>
```

Figura 6.17. Importando as bibliotecas tensorflow, handpose e camera

No *prototype*, implementamos no objeto câmera duas propriedades (*start* e *stop*) e uma função assíncrona para estimar a pose da mão chamada *estimateHandsPoseOnVideo*. Para isso, utilize os códigos já explicados anteriormente para as Figuras 6.12 e 6.13.

A função *estimateHandsPoseOnVideo* é responsável por estimar a pose da mão. A variável *modelo* foi criada pare receber o objeto retornado por essa função, o qual contém as configurações do modelo do *MediaPipe*. Após a variável *modelo* obter o objeto, podemos usar a função *estimateHands*, passando os parâmetros *input* e *flipHorizontal*. O *input* é responsável por fazer a captura do video e *flipHorizontal* recebe um valor booleano responsável por espelhar o video no sentindo horizontal (Figura 6.18).

No *body* da página, foram inseridas as *tags video* e *canvas*. A *tag video* exibe o conteúdo capturado pela câmera do dispositivo e o *canvas* exibe os *frames* do vídeo com as marcações dos pontos-chave identificados. Na *tag script*, foram criadas 3 constates: *canvas*, *video* e *ctx*. As duas primeiras são responsáveis por capturar e renderizar. A variável *ctx* é utilizada para desenhar na *tag canvas*. Os índices dos dedos são guardados no objeto *fingerLookupIndices*, contendo 25 posições, onde esses pontos são armazenados em *arrays* representando cada dedo (o índice 0 é responsável pela palma da mão), como visto na Figura 6.19.

[11] `https://google.github.io/mediapipe/solutions/solutions.html`
[12] `https://google.github.io/mediapipe/solutions/hands`

```
async estimateHandsPoseOnVideo() {
    const model = await handpose.load();
    await model.estimateHands(this.videoTag, { flipHorizontal: false }).then(result => {
        this.resultPoses = result;
    }).catch((err) => {
        console.log('Falha ao analisar', err);
    });
    return this.resultPoses;
}
```

Figura 6.18. Função *estimateHandsPoseOnVideo*.

```
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext('2d');
const video = document.getElementById("video");
let fingerLookupIndices = {
  thumb: [0, 1, 2, 3, 4],
  indexFinger: [0, 5, 6, 7, 8],
  middleFinger: [0, 9, 10, 11, 12],
  ringFinger: [0, 13, 14, 15, 16],
  pinky: [0, 17, 18, 19, 20]
};
```

Figura 6.19. Declaração de variáveis para uso do modelo.

O evento *onprogress* é responsável por capturar os dados da câmera e passar para o *HandPose*. A função *estimateHandsPoseOnVideo* é estanciada pelo objeto *camera* para obter os pontos-chave. Em seguida é verificada se a mão aparece na câmera. Caso verdadeiro, é passado o *array* dos pontos para a função *drawKeypoints*, responsável por desenhar os pontos-chave (Figura 6.20).

A função *drawKeypoints(keypoints)* (Figura 6.21) executa um *loop* para percorrer o *array* dos pontos-chave, passando as coordenadas (x, y) para a função *drawPoint* (Figura 6.22), a qual desenha um único ponto por vez. A medida que o *loop* é executado, os pontos são desenhados na tela. Em seguida é executado um outro *loop* para desenhar as linhas entre os pontos. A constante *points* recebe o mesmo *array* de pontos-chave usados para desenhar a linha e é passada para a função *drawPath*, responsável por desenhar uma única linha (Figura 6.23). O resultado do exemplo é um vídeo capturado pela câmera e exibido na *tag video* e os *frames* sendo exibidos no *canvas* com a marcação dos

```
function getProgressVideo() {
  video.onprogress = () => {
    camera.estimateHandsPoseOnVideo().then(resultPoses => {
      if (resultPoses.length > 0) {
        drawKeypoints(resultPoses[0].landmarks)
      }
    });
  }
}
```

Figura 6.20. Função *getProgressVideo()*.

pontos-chave identificados pelo *HandPose*, como visto na Figura 6.24.

```
function drawKeypoints(keypoints) {
  const keypointsArray = keypoints;

  for (let i = 0; i < keypointsArray.length; i++) {
    const y = keypointsArray[i][0];
    const x = keypointsArray[i][1];
    drawPoint(ctx, x - 2, y - 2, 3);
  }
  // mapeamento das linha
  const fingers = Object.keys(fingerLookupIndices);
  for (let i = 0; i < fingers.length; i++) {
    const finger = fingers[i];
    const points = fingerLookupIndices[finger].map(idx => keypoints[idx]);
    drawPath(ctx, points, false);
  }
}
```

Figura 6.21. Função para mapear pontos-chave.

6.4. Aplicações na Saúde

Algoritmos de ML têm sido aplicados na saúde como alternativa para melhorar a acurácia de métodos de triagem, diagnósticos e terapêuticos, usados no monitoramento

```
function drawPoint(ctx, y, x, r) {
  ctx.beginPath();
  ctx.arc(x, y, r, 0, 2 * Math.PI);
  ctx.fillStyle = '#FFFFFF';
  ctx.fill();
}
```

Figura 6.22. Função responsável por desenhar um único ponto.

```
function drawPath(ctx, points, closePath) {
  const region = new Path2D();
  region.moveTo(points[0][0], points[0][1]);
  for (let i = 1; i < points.length; i++) {
    const point = points[i];
    region.lineTo(point[0], point[1]);
  }

  if (closePath) {
    region.closePath();
  }
  ctx.stroke(region);
}
```

Figura 6.23. Função responsável por desenhar uma única linha.

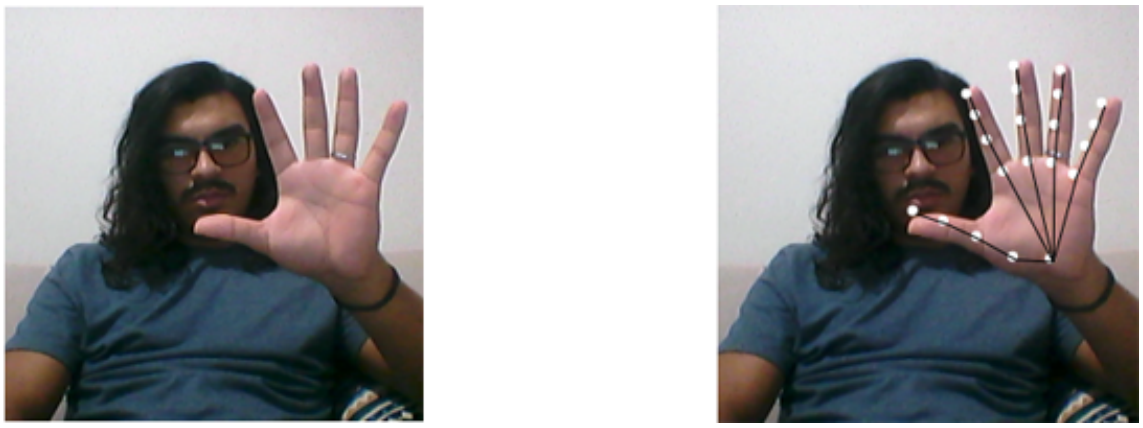


Figura 6.24. Mapeamento da mão em tempo real usando o *HandPose*.

e recuperação da saúde humana. A seguir são apresentados e discutidos exemplos de soluções de ML para estimar em tempo real a pose humana em aplicações de saúde.

6.4.1. Avaliação Postural

No contexto da avaliação dinâmica da posição do corpo humano, algoritmos de ML possibilitam detectar articulações do corpo humano e a partir delas estimar a posição e movimento humano [Voulodimos et al. 2018]. Esse recurso já foi aplicado, por exemplo, para estimar o posicionamento de membros de bebês prematuros em unidades de terapia intensiva e, a partir disso, avaliar o estado de saúde deles [Moccia et al. 2019]. Além disso, ele já foi testado no monitoramento semi-automatizado da posição do tronco superior de pacientes adultos em leitos clínicos [Chen et al. 2018] e em outras situações que podem ser de interesse clínico, por exemplo, o monitoramento das posições do corpo durante o sono [Liu et al. 2019].

Como indicado acima, a estimativa de pose humana baseada em algoritmos de aprendizado de máquina pode auxiliar no monitoramento de situações relacionadas à saúde humana, mas sua aplicação ainda pode ser ampliada para otimizar outros métodos de avaliação em saúde já existentes. Um exemplo que pode ser citado é sua aplicação ao processo de avaliação postural do corpo humano em pé. No contexto destacado aqui, enfatizamos não apenas o rastreamento dos movimentos do corpo e seus variados segmentos, mas a busca por desalinhamentos da coluna vertebral, considerando possíveis inclinações que possam ser sugestivas para a ocorrência de desvios nos planos sagital e coronal, como escoliose, hiperlordose e hipercifose.

A identificação de articulações corporais é um dos passos iniciais envolvidos no processo de avaliação postural realizado por profissionais de saúde. Os métodos tradicionalmente usados para avaliação postural ainda exigem a marcação manual usando marcadores reflexivos posicionados nas articulações. Nem mesmo aplicações para *smartphone* utilizadas com esse propósito se mostraram capazes de realizar essa marcação de forma automatizada. Nesse sentido, uma vez que algoritmos de ML já se mostram capazes de fazer essa identificação de modo automático, então podem ser aplicados ao processo de avaliação da postura, otimizando a marcação de pontos anatômicos de referência [Moreira et al. 2020]. A integração entre algoritmos de ML e recursos disponíveis nos smartphones permite o desenvolvimento de aplicações, como o que vem sendo feito por pesquisadores do Laboratório de Neuroinovação Tecnológica & Mapeamento Cerebral (NitLab) da Universidade Federal do Delta do Parnaíba (UFDPar): um protótipo de aplicação móvel para avaliação postural que permite identificar e conectar pontos anatômicos através de linhas e, assim, determinar o tamanho dos segmentos corporais que influencia na ocorrência de desvios posturais, como ilustrado na Figura 6.25, em que é utilizado o *PoseNet*.

6.4.2. Avaliação Goniométrica

Algoritmos de ML que permitem rastrear o posicionamento do corpo humano baseado na identificação das articulações corporais também podem ser aplicados como modelos inteligentes para avaliação goniométrica, ou seja, para avaliar a amplitude dos movimentos. Amplitude de movimento refere-se à quantidade de movimento realizado

Figura 6.25. Exemplo de aplicação de ML para reconhecimento e demarcação anatômica.

por um determinado segmento do corpo, por exemplo, o cotovelo, quadril e joelho, em torno de seu eixo articular. Essa amplitude pode variar em graus e pode ser afetada por condições patológicas que afetam as próprias articulações ou tecidos adjacentes como músculos, tendões, fáscias e tecido neural.

Nesse contexto, não basta movimentar-se, é necessário que os movimentos do corpo ocorram dentro de um amplitude funcional, ou seja, com quantidade suficiente para que o ser humano consiga realizar suas atividades do dia a dia. Considerando então a importância da amplitude de movimento para o desempenho funcional do ser humano, também faz-se necessário empregar métodos de avaliação que permitam quantificar esses movimentos.

Um método clínico tradicionalmente usado para esse fim é a inspeção visual, por meio da qual um avaliador solicita ao paciente que eleve o segmento avaliado no sentido do movimento de interesse (flexão, extensão, abdução adução ou rotação), e analisa se o segmento se move ou não dentro de uma amplitude funcional. Nesse modelo avaliativo, não é possível quantificar os graus de movimento, mas como alternativa para essa limitação, foi desenvolvido o goniômetro universal. Trata-se de uma régua com um eixo de 360 graus que é posicionando no centro da articulação e, a medida que o segmento se move, uma das hastes dessa régua é levada pelo examinador, no sentido do movimento realizado [Gogia et al. 1987, Carvalho et al. 2012].

Assim como na avaliação postural, para a avaliação goniométrica é necessário palpar a articulação e identificar o segmento móvel e usá-los como ponto de referência para o

posicionamento do eixo e das hastes fixa e móvel do goniômetro. Essa avaliação manual pode ser comprometida em função do deslocamento do goniômetro durante o movimento angular do segmento corporal avaliado. Nesse contexto, algoritmos de ML podem ser usado em modelos de avaliação goniométrica automatizada para otimizar esse processo de identificação articular, reduzindo riscos de viés produzidos durante a avaliação manual [Moreira et al. 2020].

6.4.3. Reabilitação Neurofuncional

Os acidentes vasculares cerebrais (AVC) são a terceira principal causa de incapacidade no mundo [Johnson et al. 2016, Raffin and Hummel 2018], associada a baixa qualidade de vida. Existe um número considerável de indivíduos com problemas neurológicos que vivem com alguma deficiência [Baumann et al. 2011] e destes, 80% apresentam deficit motor, provocado, por exemplo por uma hemiparesia que reduz a capacidade cinético-funcional do paciente [Scherbakov et al. 2013]. Apenas metade dos pacientes que sobrevivem ao AVC alcançam a recuperação funcional de seu membro superior parético [Lee et al. 2012], o que afeta seriamente o autocuidado e a participação na sociedade.

A recuperação da função motora, especialmente a função do membro superior, é um é um fator-chave para determinar o nível de independência funcional após AVC [Veerbeek et al. 2011]. Recuperar ou melhorar a capacidade de movimentar segmentos corporais ativamente facilita o desempenho das atividades de vida diária do paciente. Considerando que a maioria dessas atividades envolve o uso dos membros superiores, é crucial desenvolver mecanismos terapêuticos que facilitem sua recuperação e consequente uso funcional, mesmo após o AVC [Choo et al. 2015]

Nesse propósito, profissionais da saúde especializados na reabilitação motora pós-AVC têm utilizado diversas técnicas a fim de tentar acelerar a recuperação do paciente. Dentre estas, a gameterapia [Domínguez-Téllez et al. 2020], um método que utiliza ambientes de videogames (i.e., jogos sérios) no auxílio à reabilitação funcional dos pacientes. Alguns desses jogos sérios apresentam ambientes de gamificação [Ferreira et al. 2014] voltados especialmente para a reabilitação dos membros superiores de pacientes com sequelas motoras após AVC [Fathima et al. 2018]. Entretanto, as tecnologias computacionais associadas a gameterapia se baseiam no uso de sensores, dispositivos vestíveis, braços robóticos ou plataformas proprietárias de videogames, como *Wii*, *Kinect*, *Xbox*, as quais apresentam um elevado custo de aquisição, tornando o uso não popularizado como tecnologias sociais.

Aplicações baseadas em jogos sérios e realidade virtual tornam o programa de treinamento menos monótono, favorecendo o engajamento do paciente [Ayed et al. 2019, Chen et al. 2019]. Baseado nisso, o uso da gameterapia para a reabilitação motora deve ser incentivado, tornando-se rotina não apenas nas clínicas de reabilitação, mas também no ambiente domiciliar. Para que isso se torne viável é necessário oferecer aplicações mais acessíveis aos profissionais, bem como ao próprio paciente. Neste contexto, o TF se apresenta como uma solução robusta que permite aos desenvolvedores de software, treinar e implementar modelos de ML que podem ser usados na gameterapia. Entre as vantagens de se trabalhar com esta biblioteca, é a possibilidade de integração em várias plataformas e ambientes web. Em particular, o *PoseNet* é aplicável a diversas situações que envolvem

posicionamento e rastreamento da posição do corpo. Ao integrar o *PoseNet* às tecnologias utilizadas por navegadores web, torna-se possível implementar um ambiente favorável a gameterapia.

Um protótipo de uma plataforma web tem sido desenvolvido por pesquisadores do NitLab da UFDPar, o qual usa o TF e o *PoseNet* para o controle dos movimentos do paciente em uma interface de jogo sério, como ilustrado na figura 6.26. O contexto do jogo envolve o livre movimento dos membros superiores, aqui representados pelo elemento visual alvo. A dinâmica no jogo visa o paciente estourar os balões exibidos na tela levando o alvo ao encontro do balão. Ao iniciar o jogo, a câmera captura a imagem do usuário e a partir da movimentação do braço, a posição do alvo no cenário do jogo é alterada.

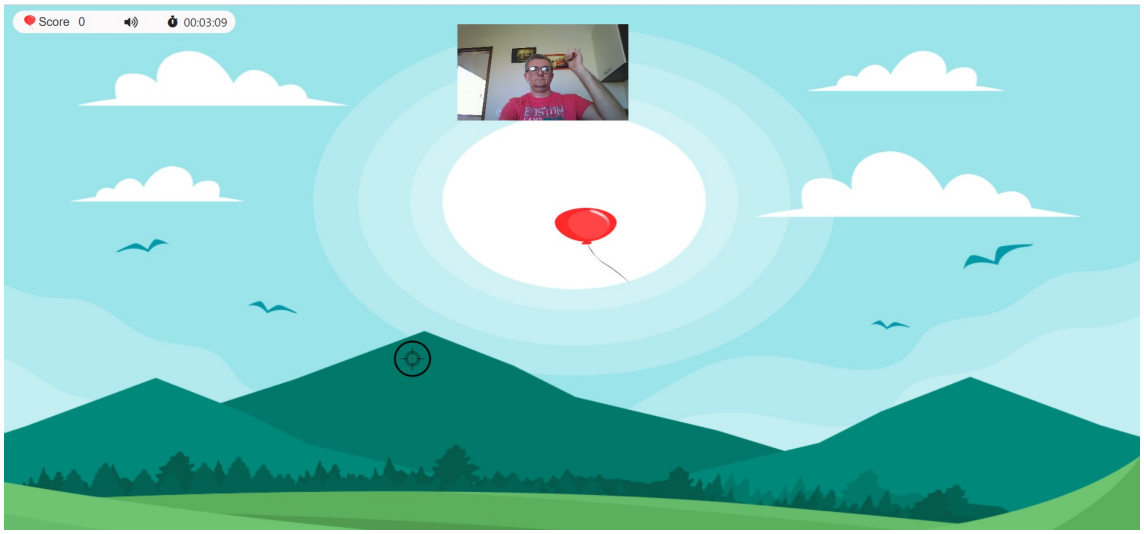


Figura 6.26. Interface de um Jogo Sério para Reabilitação Motora - Gameterapia.

Para o protótipo inicial, estamos usando apenas a posição 9 no *array* de pontos-chave do *PoseNet*, que representa a posição *leftWrist* (pulso esquerdo). Na estrutura do alvo e do balão, no cenário do jogo, existe um elemento colisor, que cria um retângulo os envolvendo. Esse colisor é responsável por retornar uma ação na dinâmica do jogo (um estouro do balão) quando os dois objetos entram em contato.

6.5. Considerações Finais

Este capítulo apresentou a biblioteca *TensorFlow* e suas aplicações no reconhecimento em tempo real de estruturas e movimento do corpo humano, contribuindo para as pesquisas na área de informática em saúde. Mais especificamente, os modelos de ML para visão computacional *PoseNet* e *MediaPipe* foram apresentados, os quais foram desenvolvidos para a estimativa de pose do corpo humano a partir de uma imagem ou vídeo. Para exemplificar soluções de ML para estimar em tempo real a pose humana em aplicações de saúde, as quais fazem uso das tecnologias mostradas, este capítulo também apresentou algumas pesquisas em desenvolvimento no NitLab da UFDPar. Inicialmente, uma solução para auxiliar no processo de avaliação postural e goniométrica. Outra, um jogo sério em desenvolvimento que deverá ser utilizado no processo de reabilitação neurofuncional

de membros superiores para pacientes pós-AVC em ambientes domésticos e clínicas de reabilitação.

A tecnologia está presente em diversos setores da economia, com inovações e soluções usadas no cotidiano da sociedade para aprimorar o modo de vida das pessoas. Como visto neste capítulo, a inteligência artificial aplicada à saúde está cada vez mais presente, com novos procedimentos e técnicas, seja na identificação e prevenção de doenças, no desenvolvimento de tratamentos para casos complexos, como no aumento da eficiência dos serviços de saúde. Estas inovações atingem todos os processos, desde o registro do paciente ao monitoramento de dados, dos testes laboratoriais aos aparelhos que possibilitam que o próprio paciente faça seu monitoramento.

Consentimento Informado

Todas as figuras com fotos de pessoas utilizadas neste capítulo foram autorizadas para fins de pesquisa.

Referências

[Abadi et al. 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA. USENIX Association.

[Aneja et al. 2019] Aneja, S., Chang, E., and Omuro, A. (2019). Applications of artificial intelligence in neuro-oncology. *Current Opinion in Neurology*, 32:1.

[Ayed et al. 2019] Ayed, I., Ghazel, A., Jaume-i Capó, A., Moyà-Alcover, G., Varona, J., and Martínez-Bueso, P. (2019). Vision-based serious games and virtual reality systems for motor rehabilitation: A review geared toward a research methodology. *International journal of medical informatics*, 131:103909.

[Basheer and Hajmeer 2000] Basheer, I. A. and Hajmeer, M. (2000). Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31.

[Baumann et al. 2011] Baumann, M., Lurbe-Puerto, K., Alzahouri, K., and Aïach, P. (2011). Increased residual disability among poststroke survivors and the repercussions for the lives of informal caregivers. *Topics in stroke rehabilitation*, 18(2):162–171.

[Carneiro et al. 2018] Carneiro, T., Medeiros Da Nobrega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., and Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, 6:61677–61685.

[Carvalho et al. 2012] Carvalho, R. M. F. d., Mazzer, N., and Barbieri, C. H. (2012). Analysis of the reliability and reproducibility of goniometry compared to hand photogrammetry. *Acta Ortopédica Brasileira*, 20:139–149.

[Chen et al. 2018] Chen, K., Gabriel, P., Alasfour, A., Gong, C., Doyle, W. K., Devinsky, O., Friedman, D., Dugan, P., Melloni, L., Thesen, T., Gonda, D., Sattar, S., Wang, S., and Gilja, V. (2018). Patient-specific pose estimation in clinical environments. *IEEE Journal of Translational Engineering in Health and Medicine*, 6:1–11.

[Chen et al. 2019] Chen, Y., Abel, K. T., Janecek, J. T., Chen, Y., Zheng, K., and Cramer, S. C. (2019). Home-based technologies for stroke rehabilitation: a systematic review. *International journal of medical informatics*, 123:11–22.

[Choo et al. 2015] Choo, P. L., Gallagher, H. L., Morris, J., Pomeroy, V. M., and Van Wijck, F. (2015). Correlations between arm motor behavior and brain function following bilateral arm training after stroke: a systematic review. *Brain and behavior*, 5(12):e00411.

[Domínguez-Téllez et al. 2020] Domínguez-Téllez, P., Moral-Muñoz, J. A., Salazar, A., Casado-Fernández, E., and Lucena-Antón, D. (2020). Game-based virtual reality interventions to improve upper limb motor function and quality of life after stroke: Systematic review and meta-analysis. *Games for Health Journal*, 9(1):1–10.

[Fathima et al. 2018] Fathima, S., Shankar, S., and Thajudeen, A. A. (2018). Activities of daily living rehab game play system with augmented reality based gamification therapy for automation of post stroke upper limb rehabilitation. *Journal of Computational and Theoretical Nanoscience*, 15(5):1445–1451.

[Ferreira et al. 2014] Ferreira, C., Guimarães, V., Santos, A., and Sousa, I. (2014). Gamification of stroke rehabilitation exercises using a smartphone. In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, pages 282–285.

[Forsyth and Ponce 2002] Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

[Gogia et al. 1987] Gogia, P. P., Braatz, J. H., Rose, S. J., and Norton, B. J. (1987). Reliability and validity of goniometric measurements at the knee. *Physical Therapy*, 67(2):192–195.

[Google 2019] Google (2019). Welcome To Colaboratory.

[Graaff and Marshall 2003] Graaff, V. D. and Marshall, K. (2003). *Anatomia humana*, volume 6.

[Johnson et al. 2016] Johnson, W., Onuma, O., Owolabi, M., and Sachdev, S. (2016). Stroke: a global response is needed. *Bulletin of the World Health Organization*, 94(9):634.

[Lee et al. 2012] Lee, M. M., Cho, H.-y., and Song, C. H. (2012). The mirror therapy program enhances upper-limb motor recovery and motor function in acute stroke patients. *American journal of physical medicine & rehabilitation*, 91(8):689–700.

[Liu et al. 2019] Liu, S., Yin, Y., and Ostadabbas, S. (2019). In-bed pose estimation: Deep learning with shallow dataset. *IEEE Journal of Translational Engineering in Health and Medicine*, 7.

[Lugaresi et al. 2019] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., and Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines.

[Moccia et al. 2019] Moccia, S., Migliorelli, L., Carnielli, V., and Frontoni, E. (2019). Preterm infants' pose estimation with spatio-temporal features. *IEEE Transactions on Biomedical Engineering*, pages 1–1.

[Moreira et al. 2020] Moreira, R., Teles, A., Fialho, R., [dos Santos], T. C. P., Vasconcelos, S. S., [de Sá], I. C., Bastos, V. H., Silva, F., and Teixeira, S. (2020). Can human posture and range of motion be measured automatically by smart mobile applications? *Medical Hypotheses*, 142:109741.

[Raffin and Hummel 2018] Raffin, E. and Hummel, F. C. (2018). Restoring motor functions after stroke: multiple approaches and opportunities. *The Neuroscientist*, 24(4):400–416.

[Sabharwal and Kumar 2008] Sabharwal, S. and Kumar, A. (2008). Methods for assessing leg length discrepancy. *Clinical Orthopaedics and Related Research*, 466(12):2910–2922.

[Scherbakov et al. 2013] Scherbakov, N., Von Haehling, S., Anker, S. D., Dirnagl, U., and Doehner, W. (2013). Stroke induced sarcopenia: muscle wasting and disability after stroke. *International journal of cardiology*, 170(2):89–94.

[Veerbeek et al. 2011] Veerbeek, J. M., Kwakkel, G., van Wegen, E. E., Ket, J. C., and Heymans, M. W. (2011). Early prediction of outcome of activities of daily living after stroke: a systematic review. *Stroke*, 42(5):1482–1488.

[Voulodimos et al. 2018] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018:1–13.

Capítulo 7

Aprendizagem Profunda em Unity com ML-Agents

Vitor Azevedo Silva, Brenno Yves Damasceno Morais,
Suzana Matos França de Oliveira e Danilo Borges da Silva.

Abstract

Deep Learning (DL) approach is applied in different research areas, mainly in games. However, to use algorithms present in these techniques can be relatively complex. These chapter goals on practicality, made possible by the creation of virtual environments on Unity, and the training of agents with ML-Agents. Two learning environments are showed to demonstrate how simple it is to train an agent, with a graphical interface, and use advanced DL approaches such as Proximal Policy Optimization. Lastly, is shown how to produce training charts and interpret it, and tips for advancing studies on the topic.

Resumo

As técnicas de Aprendizagem Profunda (AP) estão sendo utilizadas em diferentes áreas de pesquisa, principalmente em jogos. Porém, o uso dos algoritmos presentes nessas técnicas pode ser relativamente complexo. Este capítulo é focado na praticidade possibilitado pela criação de ambientes virtuais na Unity e pelo treinamento de agentes com o ML-Agents. Dois ambientes de aprendizagem são exibidos para demonstrar como é simples realizar o treinamento de gum agente com interface gráfica e uso de técnicas avançadas de AP como o Proximal Policy Optimization. Por fim, é mostrado como produzir gráficos de treinamento e interpretá-los, e dicas para avançar nos estudos sobre a temática.

7.1. Introdução

A extensão dos campos de Aprendizagem Profunda permite o florescimento de inúmeras ideias e motivações para novas pesquisas. Com o pacote ML-Agents da Unity, as possibilidades são facilitadas. Explorar e desafiar os limites dessa área, realizar testes realísticos e desenvolver agentes que podem se tornar prelúdio para algo maior, são apenas algumas das possibilidades. Este trabalho tem como objetivo explorar esse poderoso pacote à medida que apresenta a Unity, plataforma de modelagem ideal para realizar com fidelidade grandes ideias, incluindo simulações e jogos.

Visto que o pacote ML-Agents possui inerentemente intimidade com áreas da Inteligencia Artificial é recomendada uma noção básica dos conceitos que o compõem, tais como redes neurais e aprendizado de máquina, de modo que o usuário possa compreender com totalidade suas funções e fundamentos.

As redes neurais podem ser entendidas como modelos computacionais inspirados na abstração de um neurônio real, cuja finalidade é simular o processo de aprendizado e resolução de problemas complexos. Com essas redes é possível reconhecer padrões, relacionar, agrupar e classificar dados e informações, com o objetivo de aprimorar continuamente sua busca por melhores resultados. Sua arquitetura é baseada no sistema neurológico humano, onde possui "receptores" dispostos em uma ou várias camadas. Para cada receptor é atribuído um peso que é recalculado a medida que a rede aprende [Poole and Mackworth 2010].

O aprendizado de máquina *(Machine Learning)* é considerado uma ramificação da inteligência artificial, sendo assim, um subcampo da ciência da computação. Tem como principal finalidade o desenvolvimento de técnicas computacionais sobre armazenamento de "experiência" e sistemas com a habilidade de adquirir conhecimento sem intermédio do desenvolvedor. Destacam-se os seguites tipos de treinamento [Lanham 2018]:

- ***Unsupervised Training*** (Treinamento Não Supervisionado): é um método que examina os dados de forma autônoma e desenvolve uma classificação. A classificação é baseada em certas métricas que podem ser descobertas pelo próprio treinamento.

- ***Supervised Training*** (Treinamento Supervisionado): é um método comum em que a maioria das abordagens de *Machine Learning* utilizam para predizer ou classificar. Requer dados de entrada e dados de saída rotulados, necessitando um conjunto de dados de treinamento para construir um modelo.

- ***Imitation Learning*** (Aprendizagem por Imitação): é um método em que o treino é executado observando ações designadas e imitando-as.

- ***Curriculum Learning*** (Aprendizado por Currículo): é um método avançado de aprendizado que funciona subdividindo o problema em níveis de complexidade.

- ***Reinforcement Learning*** (Aprendizagem por Reforço): é um método baseado na Teoria do Controle [Doyle et al. 2013], permite que a rede treine sem dados iniciais ou modelo pré-determinados. O treinamento é modelado com base em recompensar ações que aumentam a probabilidade de sucesso no objetivo determinado.

Esse trabalho destaca a técnica *Reinforcement Learning* (RL) que é frequentemente utilizada para treinamento de agentes, os quais tem a habilidade de perceber o estado do ambiente e emitir uma resposta conforme os estímulos. Os agentes tornam-se inteligentes por meio da modelagem dos dados extraídos dos sensores, recompensando os acertos e punindo os erros, aumentando sua acurácia na tarefa de atingir o objetivo determinado. Este processo pode ser observado na Figura 7.1. Aqui, o ambiente será produzido com o uso da Unity.

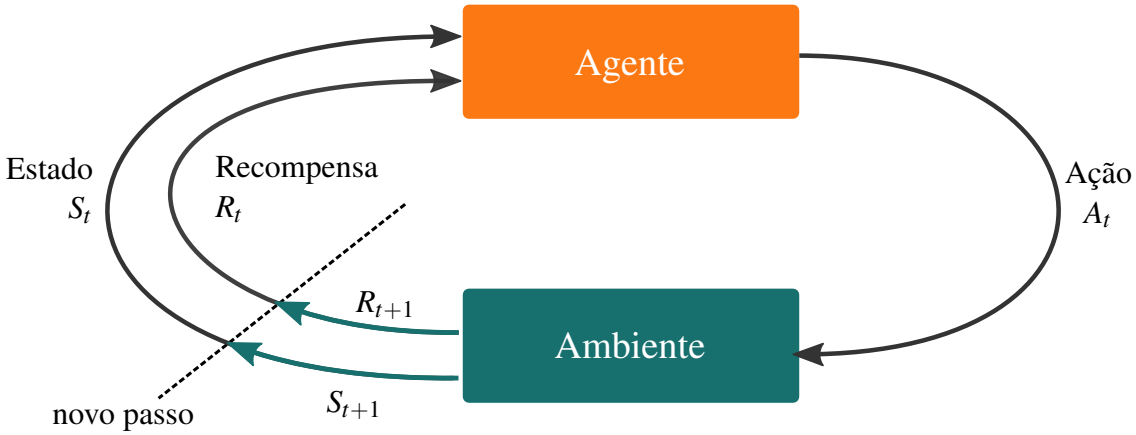


Figura 7.1. Ciclo de Aprendizagem por Reforço. O agente interage com o ambiente e recebe uma recompensa (R_t), positiva ou negativa, norteando seus passos e definindo seus objetivos de acordo com seu estado (S_t) e suas ações (A_t) em cada passo do treinamento [Sutton and Barto 2018].

A **Unity** é um Motor de Jogo (*Game Engine*) em um ambiente de desenvolvimento integrado (IDE – *Integrated Development Environment*) com a finalidade de facilitar a construção dos jogos.

Um motor de jogo é um programa ou um conjunto de bibliotecas que têm a capacidade de agrupar e produzir com totalidade os elementos de um jogo em tempo real, ou seja, reúne recursos de renderização, animação, sons, dinâmicas, física, etc, de modo que o desenvolvimento dos jogos seja simples para pessoas de diversas áreas como programadores ou artistas.

A Unity possui suporte para o desenvolvimento de jogos em duas ou três dimensões, simulações realísticas, podendo incluir funcionalidades como uso de Realidade Aumentada, Realidade Virtual e Inteligência Artificial É uma ferramenta bastante eficiente para criar ambientes e com o uso do ML-Agents os agentes podem ser treinados na Unity [Mills and Stufflebeam 2005].

As demais seções deste capítulo estão divididas em mostrar as componentes necessárias para criação do ambiente e treinamento do agente, Seções 7.2 e 7.3. Na realização da configuração dos ambientes de aprendizagem, Seção 7.4, e na observação de gráficos do treinamento, Seção 7.5. Por fim, a Seção 7.6 destina-se a apresentação de dicas de estudos em aprendizagem profunda com Unity e ML-Agents.

7.2. Unity

Nesta seção é apresentado como instalar o Unity e como identificar alguns elementos observados em sua IDE com sua devida finalidade. Para melhor identificar elementos de interface e os arquivos utilizados optou-se por sublinhar esses elementos.

Para utilizar a Unity acesse o site oficial[1], faça o download e instale o UnityHub. Abra esse programa e vá até a seção Installs (instalações), clique em Add (adicionar) e

[1] https://store.unity.com/download?ref=personal

em seguida escolha a versão compatível mais nova da Unity. Neste trabalho utilizou-se a versão 2019.4.2f1.

Para criar um projeto, abra o UnityHub, acesse a aba Projects, e selecione NEW no canto superior direito. Com isso, uma nova janela se abrirá, ao lado esquerdo localize 3D para criar um projeto com suporte 3D. Escolha um nome de sua preferência e selecione Create. A plataforma de edição é subdividida em janelas (Figure 7.2), cada qual com suas funções práticas, desde modificar o comportamento, física, lógica e aparência do cenário, até organizar os objetos do jogo, a posição de cada um, etc.

Hierarchy(hierarquia): é uma janela que lista todos os objetos da cena atual. Na Unity todo objeto é um GameObject (Figura 7.2a).

Scene (cena): é uma janela que apresenta uma perspectiva interativa do cenário em desenvolvimento. É possível utilizar esse painel para selecionar, reposicionar ou redimensionar objetos como: cenários, personagens, câmera e iluminação (Figura 7.2b).

Game (jogo): é onde aparece o resultado final do jogo, pelo ponto de vista das câmeras definidas na cena. É possível selecionar a escala do jogo, o tamanho, resolução e outros aspectos em relação à imagem (Figura 7.2c). Há também opções de executar e pausar a execução do jogo para observar cada *frame* em diferentes resoluções.

Project (projeto): é a janela que organiza e apresenta todos os arquivos relacionados ao projeto. Do lado esquerdo, Figura 7.2d, é mostrado um painel organizado de forma hierárquica em que é possível navegar pelas pastas que estruturam o projeto. Ao selecionar uma pasta a Unity mostra todos os seus arquivos.

Inspector (inspetor): contém informações detalhadas acerca dos GameObject da cena. É possível alterar a propriedade e inserir componentes ao GameObject, como a característica de ser um objeto rígido ou scripts escritos em linguagem C# (Figura 7.2e).

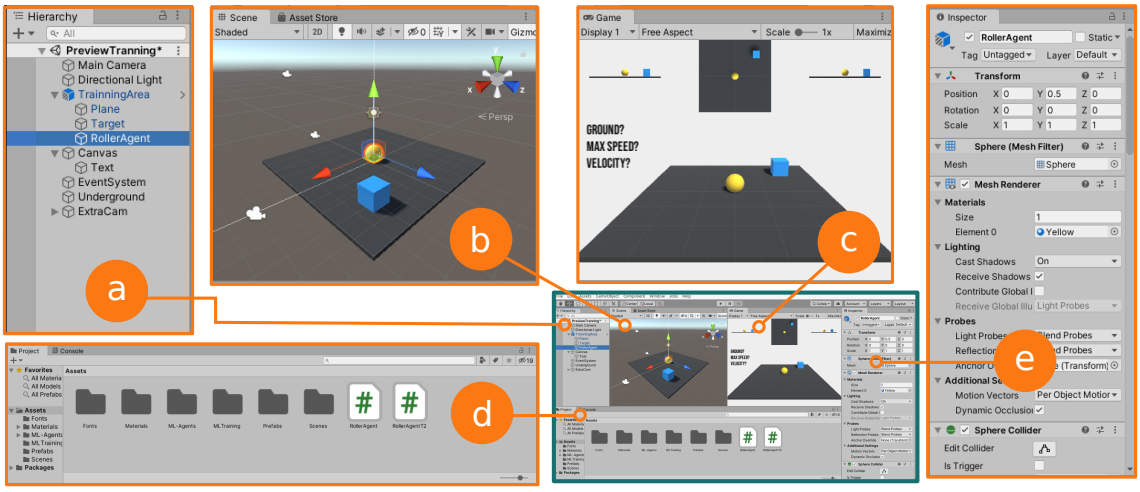


Figura 7.2. Elementos da interface da Unity. (a) *Hierarchy*, onde contém todos os objetos do cenário. (b) *Scene*, onde apresenta uma perspectiva do ambiente. (c) *Game*, onde é apresentado o resultado final do jogo. (d) *Project*, onde apresenta os arquivos relacionados ao projeto. (e) *Inspector*, onde contém informações acerca de um objeto selecionado.

7.3. ML-Agents

O **Unity Machine Learning Agents Toolkit** (ML-Agents) é um projeto de código aberto que permite a utilização de ambientes virtuais e simulações para o treinamento de agentes inteligentes [Juliani et al. 2018]. Os agentes podem ser treinados com o uso de técnicas de *Machine Learning* (ML) como *Reinforment Learning*, *Imitation Learning*, *Curriculum Learning* e outros métodos disponibilizados em bibliotecas matemáticas.

A comunicação entre esses ambientes é feita por meio de uma *Application Programming Interface* (API) – interface de programação de aplicativos – que consta de um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por outros aplicativos [Pereira 2019]. O ML-Agents disponibiliza uma API desenvolvida na linguagem Python de fácil usabilidade, fornecendo assim códigos implementados de do estados-da-arte compostos no TensorFlow[Community 2020a].

O TensorFlow é um *framework open-source* (código aberto) com suporte para Python, Javascript, C/C++, Go e Ruby desenvolvido pelo Google Brain, subgrupo do Google voltado para pesquisas na área de Inteligência Artificial com o objetivo de auxiliar no aprendizado de máquina. Por ser um dos pacotes pré-requisitados pelo ML-Agents, o TensorFlow é instalado automaticamente em versão compatível com a versão instalada do ML-Agents.

A união da Unity, ML-Agents e o Tensorflow permite que desenvolvedores projetem, treinem e visualizem suas aplicações em duas ou três dimensões. Os agentes treinados podem ser utilizados para inúmeros propósitos, incluindo o controle de comportamento de NPCs (personagens não jogáveis), testes automáticos na versão beta em jogos recém construídos, desenvolvimento de protótipos de robôs que atuaram no mundo real, além de avaliar diferentes designs antes do lançamento [Community 2020a]. Os componentes principais do ML-Agents podem ser estruturados seguindo o diagrama na Figura 7.3 e são caracterizados por:

- **Ambiente de Aprendizado**: contém o Scene da Unity e todos os objetos. Este será o ambiente na qual os agentes irão observar, agir e aprender.
- **API Python de baixo nível**: contém uma interface Python simples que permite interagir e manipular um ambiente de aprendizado.
- **Comunicador externo**: conecta o ambiente de aprendizado com a API Python de baixo nível.
- **Treinadores Python**: contém os algoritmos de *Machine Learning* responsáveis por treinar os agentes. Os algoritmos se encontram no pacote Python `mlagents`, a sua execução se deve ao comando `ml-agents-learn` que permite como parâmetros selecionar o método de treinamento e as opções em um documento formatado.

O Agente do ML-Agents consiste de um conjunto de componentes anexado a um GameObject, denominado Agente, que é responsável pela manipulação e gerenciamento de observações que, ao realizar a ação, recebe uma recompensa positiva ou negativa e

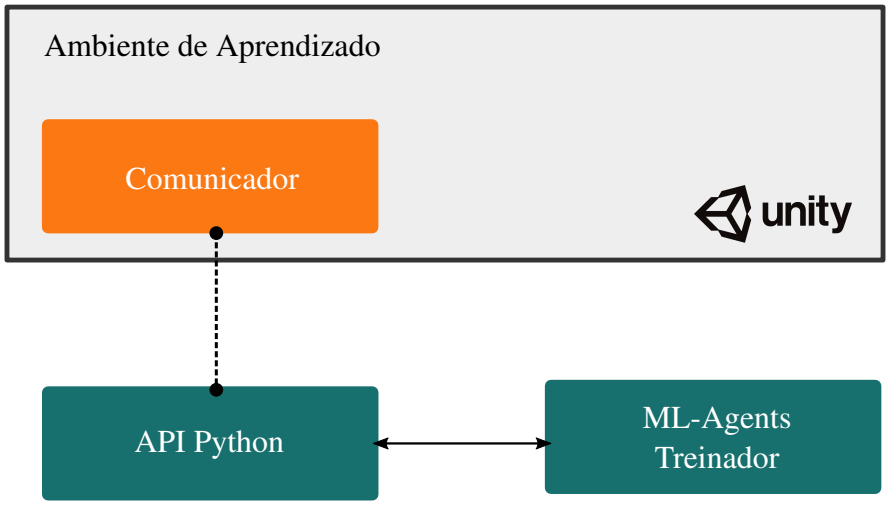


Figura 7.3. Diagrama de blocos simplificado do ML-Agents.

sempre será conectado a um comportamento (*behavior*). O comportamento define atributos específicos do Agente, como por exemplo o número de ações que ele irá executar. O Agente é identificado unicamente por um campo chamado *Behavior name*. Um comportamento também pode ser compreendido como uma função que recebe observações e recompensas do agente e retorna ações, podendo ser classificada em três tipos distintos [Community 2020a]:

- **Comportamento de aprendizagem**: é aquele que ainda não está definido, mas está prestes a ser treinado.
- **Comportamento heurístico**: é aquele que é definido por um conjunto de regras codificadas implementadas no código.
- **Comportamento de inferência**: é aquele que inclui um arquivo de rede neural treinado, o qual criado em decorrência do treinamento do comportamento de aprendizagem, tornando-se um comportamento de inferência.

O treinamento realizado em cada comportamento é associado a um conjunto de vários episódios. É denominado episódio (no contexto de RL) uma sequência de estados do ambiente, ações e recompensas, que terminam em um estado final. Por exemplo, do início de um jogo da velha até o momento em que o jogador perde, empata ou vence. O comportamento é responsável pela a definição de três importantes fases de treino, que ocorrem ciclicamente em cada episódio simulado (Figura 7.1):

- **Observações** (*observations*): é aquilo que o agente percebe sobre o ambiente. As observações podem ser numéricas e/ou visuais. As numéricas medem os atributos do ambiente do ponto de vista do agente. Para os ambientes mais interessantes, um agente exigirá várias observações numéricas contínuas. As observações visuais, por outro lado, são imagens geradas a partir das câmeras conectadas ao agente e representam o seu campo de visão.

- **Ações** (*actions*): são as ações que o agente pode executar. Semelhante às observações, as ações podem ser contínuas ou discretas, dependendo da complexidade do ambiente e do agente. Se for um ambiente de grade simples, em que apenas a localização do agente é importante (norte, sul, leste, oeste), é desejável utilizar observações discretas. Se o ambiente for mais complexo e o agente puder se mover livremente, será mais apropriado usar ações contínuas para descrever o seu posicionamento de acordo com a dimensão do ambiente.

- **Recompensa** (*reward*): é um valor escalar que sintetiza o desempenho do agente. O sinal de recompensa não precisa ser fornecido a todo momento, mas apenas quando o agente realiza uma ação boa ou ruim.

7.3.1. Uso do ML-Agents e suas possibilidades

O uso do ML-Agents promove possibilidades interessantes para diferentes áreas. A utilização do ambiente realístico da Unity é propício inclusive para testes de agentes que irão atuar no mundo real, como carros autônomos e estabilidade de drones. No ambiente de desenvolvimento de jogos, um fator importante é a jogabilidade que, com o uso de *Machine Learning*, pode oferecer uma experiência personalizada para o jogador. Alguns exemplos de ambientes de treino são:

- **Agente único**: apenas um agente, que recebe a própria recompensa.

- **Agente único simultâneo**: múltiplos agentes independentes que recebem recompensas independentes mas com os mesmos *Behavior Parameters* (parâmetros do comportamento).

- **Agente jogando contra si**: em um jogo que permite dois jogadores, dois agentes com as mesmas recompensas e mesmos *Behavior Parameters* jogam contra si.

- **Multiagentes cooperativo**: múltiplos agentes interagindo com ou sem o mesmo *Behavior Parameter* mas possuindo o mesmo sistema de recompensa.

- **Multiagentes competitivos**: múltiplos agentes interagindo com ou sem o mesmo *Behavior Parameter* e possuindo o sistema de recompensa inverso.

Nos ambientes produzidos neste trabalho utilizou-se como ambiente de treino o primeiro caso: **Agente único**.

7.3.2. Instalação do ML-Agents

O kit de ferramentas do ML-Agents armazena vários componentes, entre eles o devkit da Unity C#, que será integrado nas cenas do jogo, assim como três pacotes Python: `mla gents`, que contém algoritmos de ML, permitindo treinar o Agente no ambiente criado facilitando a comunicação entre o cenário da Unity e esses algoritmos; `mlagents_envs`, que contém a API python que irá interagir com o cenário da Unity; e `gym_unity`, que oferece uma função de encapsulamento para a cena suportando a interface OpenIA Gym. A OpenIA Gym pode ser definida como um *toolkit* que testa e compara diferentes métodos

de aprendizagem por reforço em um ambiente de simulação, selecionando o melhor, com o objetivo de maximizar os ganhos da interação com o ambiente.

O kit do ML-Agents contém uma pasta chamada Projects que possui vários exemplos de ambientes com a finalidade de ajudar o usuário a iniciar o seu aprendizado, demonstrando algumas possibilidades, além da quantidade de recursos do kit e das ferramentas da Unity, a versão do kit utilizada neste minicurso será a `release_3`, porém existem novas versões. Para baixar o kit basta clonar o repositório do projeto hospedado no Github.

Clonagem de Repositório. É preciso ter o git instalado, caso o contrário, acesse o site[2] para instalação. E execute a seguinte linha de comando:

```
$ git clone --branch release_3 https://github.com/Unity-Technologies/ml-agents.git
```

Instalar o Pacote ML-Agents na Unity. Para fazer uso do ML-Agents deve-se importar o pacote com.unity.ml-agents e associá-lo ao projeto. Na Unity, navegue até o a aba Window, situado na barra superior, e logo após em Package Manager. Em seguida, localize o botão + e selecione Add package from disk..., navegue da pasta raiz do ML-Agents (instalado pelo git clone) até a pasta com.unity.ml-agents, abra-a e selecione o arquivo package.json (Figura 7.4).

Concluindo esse procedimento de instalação do ML-Agents em um projeto da Unity, pode-se prosseguir para criação dos projetos de aprendizagem.

7.4. Projetos de Aprendizagem Profunda

Nesta seção são apresentados dois Ambientes de Aprendizado: A (Seção 7.4.1) e B (Seção 7.4.2). No Ambiente de Aprendizado A, destina-se a criação de um ambiente inspirado no tutorial apresentado pelo ML-Agents [Community 2020a], com pequenas modificações.

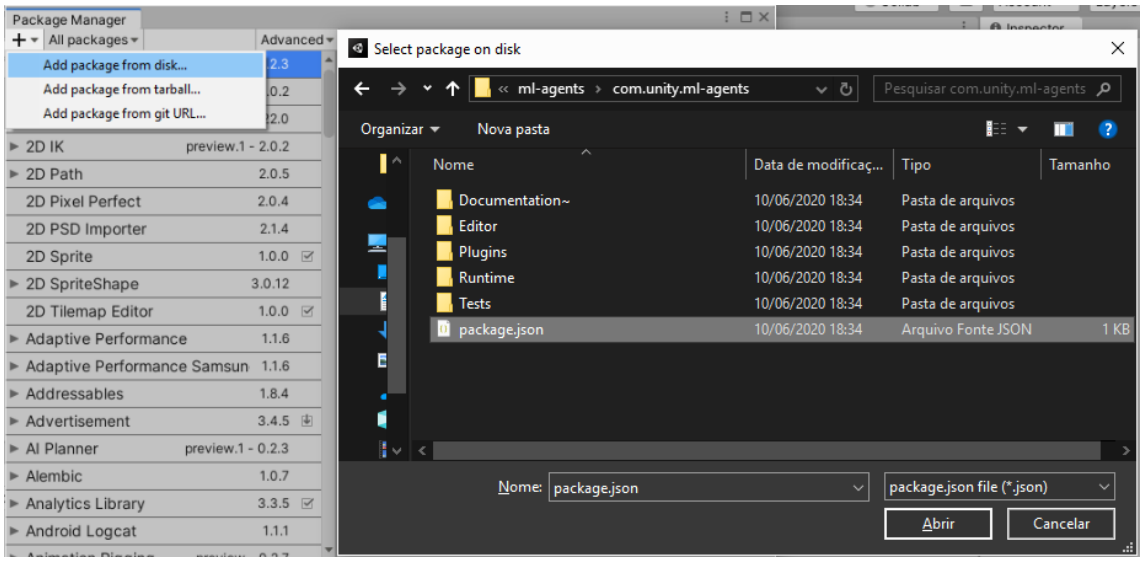


Figura 7.4. Adicionando pacote com.unity.ml-agents no editor Unity

[2] https://git-scm.com/

O Ambiente de Aprendizado B teve como base Ambiente A, porém com modificações na cena e restrições das ações do Agente com possibilidade de intervenção do usuário em tempo real. Em ambos os ambientes o objetivo do agente é alcançar um alvo.

Os códigos citados na nesta seção estão disponíveis no repositório hospedado no Github[3] para consulta. Em cada Ambiente de Aprendizado foram realizadas as seguintes etapas:

1. Criação do ambiente de treinamento na Unity (*scene*) com a configuração dos componentes necessários;
2. Implementação do script C# do Agente; e
3. Realização do treinamento do Agente com a visualização do resultado.

Considera-se que para prosseguir na criação e configuração dos ambientes descritos o projeto na Unity possua o pacote ML-Agents instalado, como foi mostrado na Seção 7.3. Para melhor visualizar as informações e procedimentos tratados nesta seção adotaram-se marcações diferenciadas para relacionar nomes e parâmetros que pertencem a Unity com sublinhado, e texto referente a código com fonte diferenciada.

Para começar, abra o UnityHub e crie um novo projeto 3D chamado LearningProject. A imagem padrão exibido pelo Unity terá como nome da *scene* SampleScene. Com o propósito de organização, foram criados mais dois *scenes*, para isso deve-se selecionar nas opções da aba de *scene* a opção Save Scene As..., ao abrir a janela especifique o nome de cada uma das *scenes* como: AAmbient e BAmbient, que corresponderão aos ambientes citados nas próximas subseções (7.4.1 e 7.4.2). Ambos devem ser colocados na pasta Scene presente na pasta Assets.

Existem GameObjects criados automaticamente que pertencem a ambos ambientes: Main Camera e Direcional Light. Altere os parâmetros do Main Camera pelo Inspector como segue: Position para $(0, 2.5, -8)$, Rotation para $(20, 0, 0)$, Clear Flags para Solid Color, e em Background coloque na paleta o código hexadecimal #FFFFFF que corresponde a cor branca.

Na construção dos Ambientes também foram utilizados materiais – configuração de cor e textura – presentes no repositório do ML-Agents. Para isso, foi copiada a pasta presente no subdiretório do SharedAssets do ML-Agents com o nome Materials e colocado dentro da pasta Assets do ProjetoAprendizagem.

7.4.1. Ambiente de Aprendizado A

O Ambiente de Aprendizado A é simples (Figura 7.6b), sendo composto por três tipos de GameObject: Plane, um solo limitado; Sphere, uma esfera de raio unitário; e Cube, uma caixa também de raio unitário. Esses GameObjects ficarão agrupados em um GameObject vazio e criado um Prefab para que possamos posteriormente replicar a mesma estrutura para o treinamento posterior.

[3] https://github.com/devcavi/mlagentscourse

Criação do Prefab: Crie um GameObject vazio dentro da aba Hierarchy, para isso acesse a aba GameObject a opção Create Empty, e renomeie o objeto criado com o nome TrainingAreaA. Esse será o nome do Prefab. Para prosseguir, deve-se transformar esse objeto em Prefab e posteriormente serão adicionados os elementos do ambiente de treinamento. Crie na pasta Assets uma subpasta com o nome Prefabs e abra essa pasta. Para criar um Prefab basta arrastar o objeto TrainingAreaA para a pasta Prefabs. Feito isso, clique no Prefab TrainingAreaA, dentro da pasta Prefabs, para continuar configurando o ambiente.

Agora, irá aparecer na aba Hierarchy somente o objeto TrainingAreaA. Clique sobre este objeto com o botão direito e selecione dentro da opção 3D Object os objetos Sphere, Cube e Plane (uma de cada vez). O próximo passo será renomear esses objetos: Sphere $\rightarrow$ RollerAgent, Cube $\rightarrow$ Target e Plane $\rightarrow$ Floor. Para renomear, selecione o objeto com o botão direito e siga para a opção Rename. O próximo passo será a configuração de cada um desses três objetos via interface da Unity para terem uma estrutura similar a apresentada nas Figuras 7.5a e 7.5b.

- Floor. A modificação feita será a inserção de uma textura (material), para isso acesse a pasta Materials e localize o arquivo GridMatFloor. Selecione esse arquivo e arraste até o objeto Floor. Ao fazer isso, verá que o material foi aplicado ao objeto.
- Target. Inicialmente, realize a inserção do material com nome AgentBlue, da mesma forma que realizado no Floor. No Inspector do objeto, modifique a Position para $(3.5, 0.5, 2.5)$.
- RollerAgent. Aplique também um material ao objeto, nome Yellow. Altere sua Position para $(0, 0.5, 0)$. Neste objeto será adicionado uma componente chamada Rigidbody. Para isso, via Inspector, clique no botão Add Component e procure por essa componente, selecionando-a. Isso possibilita ao objeto interagir de acordo com as leis físicas dentro da Unity.

Para fazer com que o RollerAgent seja um Agente usado para treinamento, outras três componentes deverão ser inseridas ao mesmo. Da mesma forma que foi adicionado a componente Rigidbody, insira a primeira componente da seguinte forma: selecione New script e coloque como nome AgentA e clique em Create and Add. Abra o script AgentA e insira o Código 7.1.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Unity.MLAgents;
5  using Unity.MLAgents.Sensors;
6
7  public class RollerAgentA : Agent
8  {
9      Rigidbody rBody;
10     public Transform target;
11     public float speed = 10;
12
13     void Start () {
14         rBody = GetComponent<Rigidbody>();
15     }
16
17     public override void OnEpisodeBegin()
18     {
```

```
19          if (this.transform.localPosition.y < 0)
20          {
21              this.rBody.angularVelocity = Vector3.zero;
22              this.rBody.velocity = Vector3.zero;
23              this.transform.localPosition = new Vector3(0, 0.5f, 0);
24          }
25
26          target.localPosition = new Vector3(Random.value * 8 - 4,
27                                             0.5f, Random.value * 8 - 4);
28      }
29
30      public override void CollectObservations(VectorSensor sensor)
31      {
32          sensor.AddObservation(target.localPosition);
33          sensor.AddObservation(this.transform.localPosition);
34          sensor.AddObservation(rBody.velocity.x);
35          sensor.AddObservation(rBody.velocity.z);
36      }
37
38      public override void OnActionReceived(float[] vectorAction)
39      {
40          Vector3 controlSignal = Vector3.zero;
41          controlSignal.x = vectorAction[0];
42          controlSignal.z = vectorAction[1];
43
44          rBody.AddForce(controlSignal * speed);
45
46          float distanceTotarget = Vector3.Distance(this.transform.localPosition,
47                                                    target.localPosition);
48
49          if (distanceTotarget < 1.42f)
50          {
51              SetReward(1.0f);
52              EndEpisode();
53          }
54
55          if (this.transform.localPosition.y < 0)
56          {
57              EndEpisode();
58          }
59      }
60  }
```

Código 7.1. Descrição do script de interação do Agente com o ambiente.

Os elementos do Código 7.1 serão discutidos a seguir, porém objetivo do RollerAgent, será alcançar o Target a partir de qualquer posição dos limites determinados pelo Floor. Para isso a rede terá que aprender como gerar as forças que serão aplicadas no agente, que possui propriedades de corpo rígido, para que o objetivo seja alcançado.

Inicializando o agente: O processo de treinamento baseia-se em episódios em que o agente tenta alcançar seu objetivo. Cada episódio encerra no momento em que o RollerAgent atinge o Target (linha 49) ou fracassa, caindo do Floor (linha 55). Toda vez que o RollerAgent atinge o Target, o episódio encerra e o Target é movido para uma nova posição aleatória (linha 26); se o Target cair da plataforma, ele será movido novamente para cima do Floor.

Atributos da classe: O `RollerAgentA` possui três atributos, um privado (`rBody`) e os demais públicos (`Target` e `speed`). O fato de serem públicos permite com que possam ser atribuídos em tempo de execução pela interface da Unity. A funcionalidade de cada parâmetro pode ser sintetizada:

- `rBody`: é responsável pela interação física do agente no cenário. Por meio desse parâmetro que as forças serão aplicadas (linha 44) e a velocidade do agente será rastreada (linhas 33 e 34). Na inicialização, o `rBody` é capturado por meio do RollerAgent que possui uma componente Rigidbody (linha 14).

- `target`: é o objetivo do agente, ou seja, o Target. Ao executar o jogo, deve-se setar esse elemento na componente Roller Agent A via Inspector.

- `speed`: é um atributo do tipo real (`float`), responsável por armazenar o módulo da velocidade máxima no qual o RollerAgent irá se movimentar durante o treinamento. Será inicializado com o valor 10.

Pode-se observar que a classe `RollerAgentA` herda funções e propriedades pertencentes à classe `Agent` (linha 7), logo a descrição dos métodos que serão comentados à seguir são sobrecarregamentos (*override*) de funções herdadas:

- `Start()`: nesta função o atributo `rBody` é inicializado adquirindo a componente Rigidbody do RollerAgent com o método `GameObject.getComponent<T>()` (linha 14).

- `OnEpisodeBegin()`: esta função é chamada no início de cada episódio, será responsável por mudar os atributos para que o Target seja colocado em uma posição aleatória (linha 26), e que o RollerAgent seja movido para o centro da plataforma, resetando suas velocidades angular e linear no começo de cada episódio (linhas 21 a 23).

- `CollectObservations(VectorSensor sensor)`: esta função é responsável pelas coletas de dados da observação do ambiente. O tamanho deste vetor é determinado pela componente Behavior Paramenters que informa a quantidade de observações. A quantidade de dados que serão adicionados deverão ser do tamanho especificado nessa componente. Esse exemplo contém 8 (oito) observações: três para posição do Target (linha 31), três para posição do RollerAgent e dois para a velocidade do agente nos eixos x e z (linhas 33 e 34). Observa-se que o personagem não se desloca no eixo y, por isso a observação deste eixo não se faz relevante para o treinamento.

- `OnActionReceived(float[] vectorAction)`: esta função é responsável pela as ações do agente. Como o objetivo é o deslocamento no plano então será necessário aplicar uma força para o deslocamento no eixo x e z. As ações também estão associadas às saídas da rede. Neste caso, cada saída é um valor contínuo (real) no intervalo entre -1 e 1, isso é determinado também na componente Behavior Parameters. As forças serão adicionadas por meio da função `AddForce()` (linha 43), na qual a força aplicada corresponde a um vetor, criado à partir da saída da rede com a magnitude da velocidade máxima permitida (`speed`). Nesta função também é criado o sistema de recompensas caso o RollerAgent atinja o alvo (linha 49), sendo baseado na distância entre o RollerAgent e o Target (linha 45).

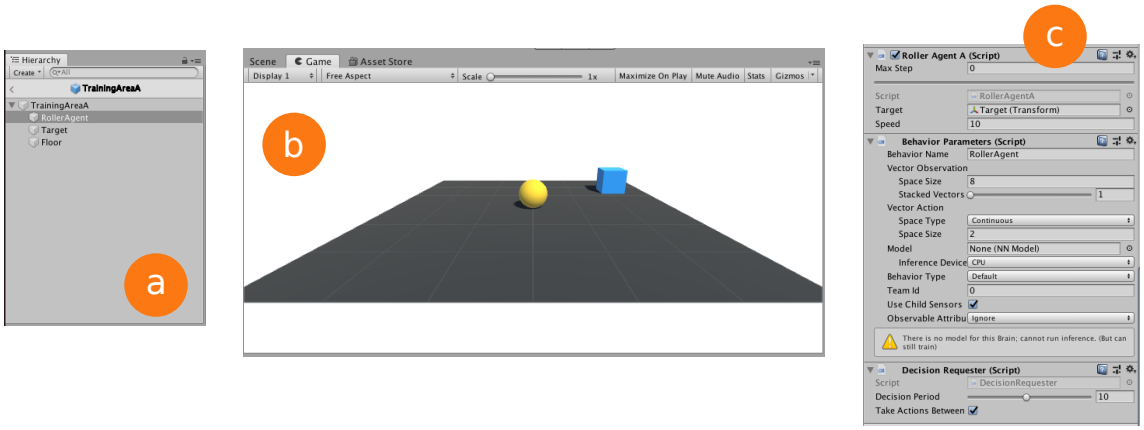


Figura 7.5. Configuração do ambiente de treinamento. a) Estrutura dos objetos. b) Visualização do Jogo pela câmera do Unity. c) Configuração das componentes necessárias do agente para a aprendizagem com ML-Agents.

Os dois componentes que faltam ao RollerAgent são responsáveis pela criação, configuração e comunicação para o treinamento da Rede Neural. Adicione as componentes Behavior Parameters e Decision Requester. Configure-os de acordo com dados da Figura 7.5c. Observe na imagem a configuração da quantidade de observações (Vector Observation) e ações (Vector Action), além do tipo de saída da rede (Continuous). A próxima etapa é definir a configuração do treinamento pretendido, para isso é necessário criar um arquivo de configuração com extensão yaml.

A última peça para concluir a configuração do agente para o treinamento ocorre de forma externa à Unity. Dentro da raiz do projeto LearningProject, crie uma pasta com nome config e dentro dela crie um arquivo chamado roller_agent.yaml. O conteúdo deste arquivo é disposto no Código 7.2.

```
1  behaviors:
2    RollerAgent:
3      trainer_type: ppo
4      hyperparameters:
5        batch_size: 10
6        buffer_size: 100
7        learning_rate: 3.0e-4
8        beta: 5.0e-4
9        epsilon: 0.2
10       lambd: 0.99
11       num_epoch: 3
12       learning_rate_schedule: linear
13     network_settings:
14       normalize: false
15       hidden_units: 128
16       num_layers: 2
17     reward_signals:
18       extrinsic:
19         gamma: 0.99
20         strength: 1.0
21     max_steps: 500000
22     time_horizon: 64
23     summary_freq: 10000
```

Código 7.2. Configuração de treinamento.

Alguns dados do Treinamento: No Código 7.2, o Behavior Name é definido na linha 2, que deve ser idêntico ao definido na componente Bahavior Parameters. É a partir

dessa informação que a configuração da rede e os dados para treinamento, definidos no arquivo yaml, serão cruzados. Na linha 3, foi definido o tipo de algoritmo utilizado no treinamento; o *Proximal Policy Optimization* (PPO) [Schulman et al. 2017] é o método indicado pelos desenvolvedores do ML-Agents, pois demonstrou ser mais geral e estável do que outros algoritmos de RL. Os parâmetros associados ao PPO, como seus limites, podem ser consultados na referência [AurelianTactics 2018a].

A principal contribuição do PPO é garantir que uma nova atualização da política não mude muito da política anterior. Isso leva a uma menor variação no treinamento ao custo de algum viés, mas garante um treinamento mais suave e também certifica que o agente não siga um caminho irrecuperável de ações sem sentido [Trivedi 2019].

Outras informações sobre a rede se encontram nas linhas 13 a 16 (Código 7.2). Além das camadas de entrada e saída, definidas na componente Behavior Parameters, essa rede possui duas camadas ocultas com 128 nós cada (linha 15). Outro parâmetro importante é o `max_steps` (linha 21) que define o limite de passos de treinamento, quando esse número é alcançado o algoritmo de treinamento é finalizado. Agora, o agente está com todas as configurações necessárias para o treinamento, a seguir serão definidos os passos realizados para executar o algoritmo de treinamento.

Antes de começar o treinamento, deve-se criar um ambiente virtual em Python, cuja finalidade é separar projetos, dependências e bibliotecas em um único lugar. Aqui utilizou-se o ambiente Linux, porém os mesmos passos podem ser realizados para outro sistema operacional. Abra o terminal e execute os comandos a seguir, dentro da pasta do projeto (LearningProject):

```
$ sudo apt-get install python3-venv
$ python3 -m venv env
```

Ative o ambiente virtual e instale as atualizações dos pacotes `pip` e `setuptools` e a seguir instale o pacote `ml-agents 0.17` (release_3):

```
$ source env/bin/activate
(env) $ pip install --upgrade pip
(env) $ pip install --upgrade setuptools
(env) $ pip install mlagents==0.17
```

Para realizar o treinamento, com o ambiente virtual ativado (`env`), deve ser inserido o seguinte comando:

```
(env) $ mlagents-learn config/roller_agent.yaml --run-id=AgentA
```

a opção `--run-id` determina um nome dado ao treinamento que será executado. O arquivo de configuração também deve ser informado (roller_agent.yaml). Esses são os parâmetros obrigatórios para execução de qualquer treinamento.

Ao executar o comando aparecerá no terminal algo similar a Figura 7.6a, onde será necessário dar um "play" no Unity para iniciar o treinamento (Figura 7.6b). Vale ressaltar que o comando de treinamento (`mlagents-learn`) possui diversas opções, onde destacam-se:

- `--env`: responsável por informar o executável do ambiente, fazendo com que não seja preciso clicar em "play" na Unity.

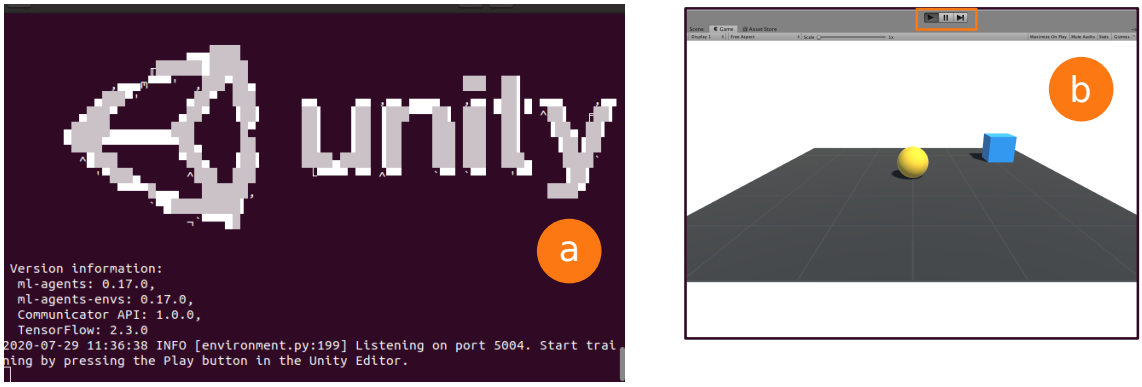


Figura 7.6. Iniciando o treinamento do agente. a) Tela de espera para execução do treinamento vinculado ao Unity. b) Acionamento do "Play" (botão pressionado do retângulo em destaque) para iniciar o treinamento.

- `--resume`: continua o treinamento de um determinado `--run-id`.
- `--force`: faz com que um treinamento seja executado sobrescrevendo um `--run-id` existente.

O treinamento deste Agente demorou cerca de 2h11" em um computador Intel® Core™ i7-6500U CPU 2.50GHz×4. Para minimizar esse tempo pode ser inserido vários agentes na Scene replicando-se o Prefab do TrainingAreaA, inserindo cada um em uma posição diferente. Com 16 áreas de treinamento minimizou-se esse tempo para 17".

O comando de treinamento cria uma pasta no projeto chamado results onde ficarão os arquivos relacionados ao treinamento do agente. Ao final do treinamento acesse a pasta results/AgentA. O arquivo RollerAgent.nn é o resultado do treinamento, que deve ser inserido no Behavior Paramenters no campo Model na Unity. Com isso, é possível visualizar o resultado do treinamento ao acionar botão "play". Essa inserção deve ser via o Prefab do TrainingAreaA para que todos os agentes criados a partir dele possam "enxergar" o treinamento realizado. O resultado pode ser visto na Figura 7.7, onde toda vez que o agente acerta o objetivo a posição do objetivo é alterada e reaparece no ambiente.

7.4.2. Ambiente de Aprendizado B

Este é o Ambiente de Aprendizado B, onde pequenas alterações foram realizadas em relação ao Ambiente de Aprendizado A. Agora o Target pode mudar sua posição aleatoriamente também no eixo y e o usuário tem a possibilidade de mudar a velocidade máxima (`speed`) do RollerAgent em tempo de execução do jogo pela interface da Unity.

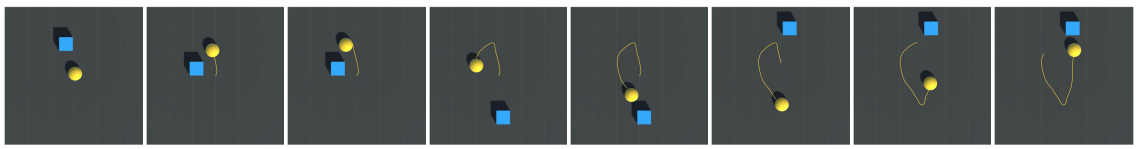


Figura 7.7. Animação do agente treinado. *Frames* da visão superior para melhor observar o movimento do agente em conjunto com um rastro (leitura da esquerda para direita).

Além dessas mudanças, o Agente é restito a ficar o menor tempo possível no ar, ou seja, as forças aplicadas em direção ao eixo y seriam limitadas de acordo com a necessidade, isso é o Target se encontrar fora do alcance na direção y. Para que tudo isso seja possível acontecer, foram introduzidas mudanças na função de recompensa e nas configurações das componentes do Agente e em suas observações e ações.

Para configuração do BAmbient, execute as mesmas etapas descritas na criação do AAmbient, alterando as nomenclaturas do Prefab de treinamento para TrainingAreaB e do script do Agente para RollerAgentB. Concluída a criação do ambiente, serão realizadas as alterações.

Alteração no Prefab RollerAgentB: Selecione o GameObject Plane e localize o atributo Tag no Inspector (abaixo do nome do objeto). Crie uma nova *tag* com o nome Floor. Essa marcação será útil para saber quando o Agente está no solo a partir do script.

Alteração do script do RollerAgentB: Foram inseridos três parâmetros no script: `randomVel`, indicando se haverá (ou não) aleatoriedade na velocidade máxima; e `speedRand`, indicando o valor da velocidade máxima em cada episódio. A variável `speed`, que já existia no Ambiente de Aprendizado A, guardará agora o valor máximo permitido para a velocidade máxima, neste caso foi determinado 250. Todos os atributos do RollerAgentB estão descritos no Código 7.3. Por fim, a variável `touchingGround` será responsável por informar se o RollerAgentB está ou não tocando o Floor. Para que isso aconteça de forma correta deve-se especificar o nome da Tag associado ao Floor que é uma constante (linha 7).

```
1  Rigidbody rBody;
2  public Transform target;
3  public float speed = 250;
4  public bool randomVel = false;
5  float speedRand;
6  bool touchingGround;
7  const string k_Ground = "Floor";
```

Código 7.3. Atributos da script do Agente para interação com o ambiente.

A função `OnEpisodeBegin()` deve ser modificada de acordo com o Código 7.4. Na linha 3, caso o Agente alcance uma altura superior a 10 ele é reinicializado. Na linha 9, o `speedRand` recebe uma nova velocidade máxima que varia no intervalo $[30, \texttt{speed}]$. Na linha 12, foi adicionado um intervalo randômico no eixo y do `target`, variando no intervalo $[0.5, 2.5]$.

```
1  public override void OnEpisodeBegin()
2  {
3   if (this.transform.localPosition.y < 0 || this.transform.localPosition.y > 10)
4   {
5       this.rBody.angularVelocity = Vector3.zero;
6          this.rBody.velocity = Vector3.zero;
7          this.transform.localPosition = new Vector3(0, 0.5f, 0);
8          if (randomVel){
9              speedRand = Random.value*(speed-30.0f) + 30.0f;
10         }
11  }
12  Target.localPosition = new Vector3(Random.value * 8 - 4,
13                                     Random.value*2.0f + 0.5f, Random.value * 8 - 4);
14 }
```

Código 7.4. Função que determina o início de cada novo episódio do Agente.

Será necessário adicionar dois novos métodos no Agente: `OnCollisionEnter()` `OnCollisionExit()`; descritos no Código 7.5. O primeiro verifica se houve o contato do Agente no objeto que possui uma Tag com nome definido na variável `k_Ground`, isso é, o objeto Floor (linha 3). A segunda indica se o Agente perdeu o contato com o objeto com essa Tag (linha 9). Essas ações informam, respectivamente, se o `touchingGround` é verdadeiro ou falso. Para melhorar a resposta dessas duas funções, a componente Rigidbody do Agente deve ter o atributo Collision Detection alterada para Continuous.

```
1  void OnCollisionEnter(Collision col)
2  {
3      if (col.transform.CompareTag(k_Ground))
4       touchingGround = true;
5  }
6
7  void OnCollisionExit(Collision col)
8  {
9      if (col.transform.CompareTag(k_Ground))
10      touchingGround = false;
11 }
```

Código 7.5. Funções responsáveis por tratar o toque do Agente no solo.

Na função `CollectObservations(VectorSensor sensor)` foram adicionadas outras observações, como é possível observar no Código 7.6. O total de novas observações somam 3 (três): informação se o agente toca ou não o solo (linha 3), velocidade no eixo *y* do agente (linha 6) e valor da velocidade máxima do Agente no episódio (linha 8 ou linha 10). As duas opções de velocidade dependem da informação se a velocidade é aleatória ou não. Neste trabalho, a variável `randomVel` é verdadeira no momento do treinamento do Agente, porém será trocada para falso uma vez que estiver treinado, dependendo somente da velocidade atribuída pelo usuário.

```
1  public override void CollectObservations(VectorSensor sensor)
2  {
3      sensor.AddObservation(touchingGround ? 1 : 0);
4      sensor.AddObservation(target.localPosition);
5      sensor.AddObservation(this.transform.localPosition);
6      sensor.AddObservation(rBody.velocity);
7      if(randomVel){
8          sensor.AddObservation(speedRand/speed);
9      } else {
10         sensor.AddObservation(speed/250.0f);
11     }
12 }
```

Código 7.6. Função reponsável por capturar as observações do Agente.

Por fim, a função responsável pelas ações foi bastante modificada, principalmente em termos de recompensa (`OnActionReceived()`). O Código 7.7 contém as alterações realizadas. Observa-se que agora são três ações recebidas, indicando a direção da aplicação da força no Agente. A magnitude da velocidade depende se está em modo randômico (linha 9) ou fixo (linha 7). Outra alteração foi na recompensa, onde a função utilizada agora é a `AddReward()`, que é responsável por acumular as recomensas durante o treinamento até sua finalização (`EndEpisode()`). Na linha 13, foi inserida uma penalização ao Agente, com a finalidade de que ele chegue no objetivo o mais rápido possível. De forma semelhante, se o Agente ficar muito tempo fora do solo (no ar), ele é penalizado (linha 15). Somente quando o Agente alcançar o objeto, que ele somará uma recompensa

positiva (linha 18). Caso o personagem caia do solo ou alcance uma altura superior a 10 é fixada uma recompensa ao episódio (linha 23).

```
1  public override void OnActionReceived(float[] vectorAction)
2  {
3      Vector3 controlSignal = Vector3.zero;
4      controlSignal.x = vectorAction[0];
5      controlSignal.y = vectorAction[1];
6      controlSignal.z = vectorAction[2];
7      Vector3 force_app = (controlSignal * speed);
8      if(randomVel){
9          force_app = (controlSignal * speedRand);
10     }
11     rBody.AddForce(force_app);
12     float distanceTotarget = Vector3.Distance(this.transform.localPosition,
13                                               target.localPosition);
14     AddReward(-0.0001f);
15     if(!touchingGround)
16         AddReward(-0.005f);
17     if (distanceTotarget < 1.42f)
18     {
19         AddReward(1.0f);
20         EndEpisode();
21     }
22     if (this.transform.localPosition.y < 0 || this.transform.localPosition.y > 10)
23     {
24         SetReward(-0.5f);
25         EndEpisode();
26     }
27 }
```

Código 7.7. Função reponsável por realizar as acões do Agente e pelo cálculo da recompensa.

Essas inserções de valores à recompensa têm como objetivo moldar o comportamento do Agente, fazendo com que ele alcance o objetivo de forma rápida e mais realista possível, ou seja, sem ficar flutuando no ar o tempo todo. Para utilizar o `AddReward()` é necessário limitar a quantidade de passos (*steps*) de um episódio. Isso deve ser realizado na componente Roller Agent B na interface da Unity no campo Max Steps, atribua esse valor para 2500. Como foi alterado também a quantidade de observações e ações, altere-as respectivamente para 11 e 3 na componente Behavior Parameters.

Realizadas essas últimas modificações, pode-se partir para o treinamento do Agente. Para isso, o ambiente virtual deve ser ativado e dentro do diretório raiz do projeto deve ser executado o comando:

```
(env) $ mlagents-learn config/roller_agent.yaml --run-id=AgentB
```

Pode-se observar que foi utilizado o mesmo arquivo de do treinamento feito no Agente do Ambiente A.

O tempo de treinamento foi de 19” considerando 16 Agentes ao mesmo tempo. Na Figura 7.8, pode-se observar a sequência de frames descrevendo a ação do personagem no Ambiente B, nesta animação foi escolhida uma velocidade máxima de 45.

Na próxima seção, serão apresentados os gráficos produzidos por cada um dos treinamentos relativos aos identificadores `AgentA` e `AgentB`, descritos no comando `mlagents-learn` (`--run-id`).

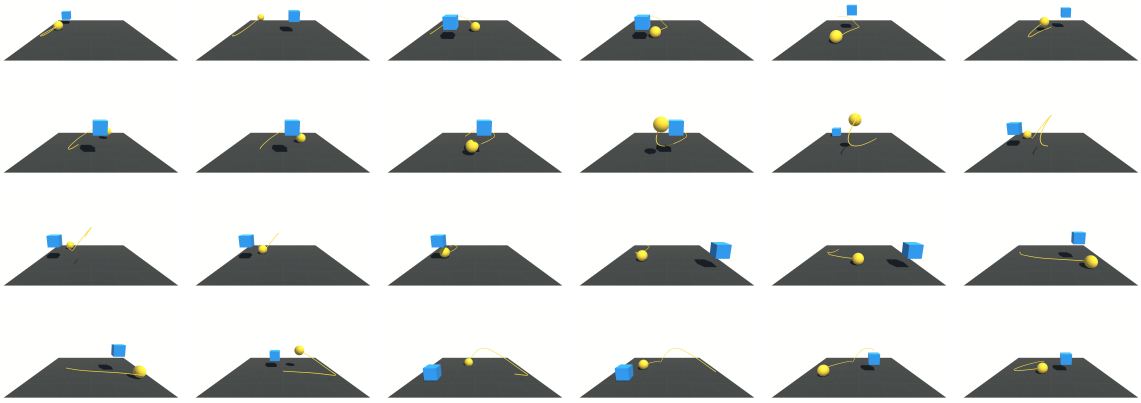


Figura 7.8. Animação do Agente treinado (leitura da esquerda para direita de cima para baixo). A visão escolhida foi em perspectiva para observar a alteração no eixo y da caixa.

7.5. Análise gráfica do Tensorboard

O kit de ferramentas do ML-Agents armazena informações adquiridas durante a sessão de aprendizado, assim os desenvolvedores podem acompanhar o treinamento da rede por meio de gráficos. A ferramenta responsável por mostrar essas informações pertence ao Tensorflow e chama-se Tensorboard [Community 2020b]. Nesta seção é realizada uma análise de alguns gráficos do treinamento, que encontram-se na íntegra no Tensorboard-Dev[4].

O comando `mlagents-learn` salva os arquivos com os dados de treinamento em uma pasta chamada results, organizada pelo nome especificado no `--run-id` que foi designada para na seção de treinamento. Para observar o processo de treinamento antes, durante ou depois, deve-se iniciar o Tensorboard pelo terminal:

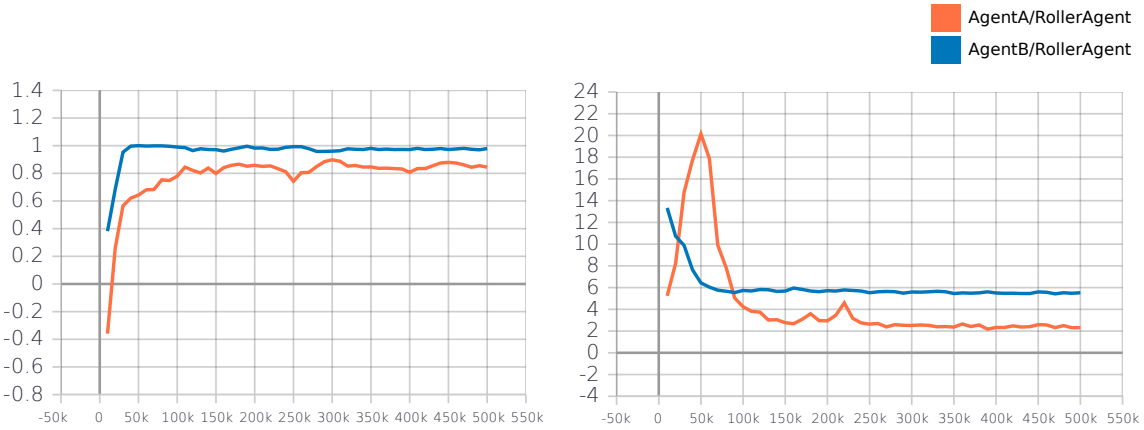
```
(env) $ tensorboard --logdir=results
```

Para executar esse comando, o ambiente virtual deve estar ativado. A localização da pasta deve ser informada na opção `--log-dir`. Por padrão, o Tensorboard opera na porta 6006, podendo ser alterado com a opção `--port` seguido do número da porta desejada. Ademais, os dados serão exibidos no navegador por meio da porta especificada, neste caso `http://localhost:6006/`.

O Tensorboard divide em os gráficos gerados em quatro categorias: *Environment* (Ambiente), dados relacionados aos valores de recompensa adquirida entre os episódios de treinamento; *Is Training* (está treinando), dados que informam se o agente está treinando ainda ou finalizou o treinamento; *Losses* (Perdas), dados que indicam como está se comportando o algoritmo de aprendizagem do agente; e *Policy* (Política), dados que exibem o decaimento dos parâmetros da função de aprendizado durante o treinamento. Destacam-se a seguir alguns gráficos produzidos e sua interpretação.

Na Figura 7.9, tem-se os gráficos que mostram se o personagem conseguiu performar bem o treinamento. Observa-se que o `AgentA` após um curto espaço de tempo atingiu

[4] `https://tensorboard.dev/experiment/Zhm1zVjNTQWrDJ1TPDY0jA/`

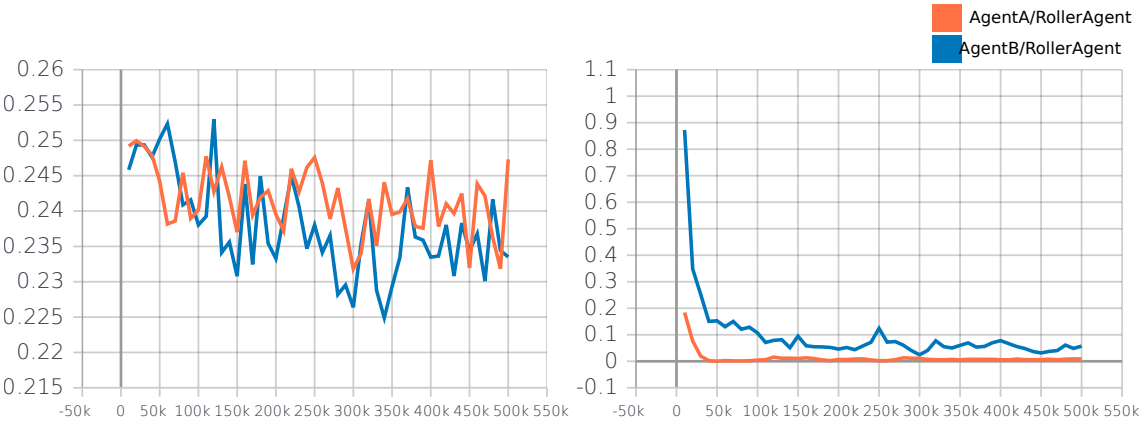


(a) Environment/Cumulative Reward (b) Environment/Episode Length

Figura 7.9. Gráficos relacionados à categoria *Environment*. O eixo x apresenta a quantidade de passos de treinamento, onde é no máximo 500K (mil). (a) exibe a recompensa alcançada pelo Agente durante o aprendizado. (b) indica a duração média de cada episódio.

frequentemente a recompensa máxima (1.0), contudo o `AgentB` não consegue alcançar o valor máximo devido aos descontos aplicados durante cada passo do treinamento (Figura 7.9a). O desconto aplicado ao `AgentB`, influencia também o tempo gasto no treinamento, levando a uma duração de episódio relativamente menor do que em relação ao `AgentA`, pois ele é forçado a chegar o mais rápido possível no objetivo (Figura 7.9b).

Na Figura 7.10, pode-se observar gráficos relacionados ao *Losses*. Pode-se mensurar o quanto função de perda de política está se modificando, em média esses valores tendem a ser menores que 1 (Figura 7.10a). Essas variações estão relacionadas também à aleatoriedade do modelo de treinamento. Percebe-se que o gráfico ao lado, Figura 7.10b, indica a perda média da atualização da função de valor, quado inicia-se o treinamento o valor é grande e com o treinamento bem sucedido tende a diminuir e se estabilizar.



(a) Losses/Value (b) Losses/Policy

Figura 7.10. Gráficos relacionados à categoria *Losses*. (a) Correlaciona o quanto a política (processo para decidir ações) está mudando. (b) indica o quanto a função de aprendizado está na direção correta.

Existem outros gráficos que possuem a tendência em comum de decrescerem ao longo do tempo, esses são: Epison, Beta e Learning Rate. Para uma descrição completa do significado desses e de outros parâmetros consulte [AurelianTactics 2018b].

7.6. Conclusão e Considerações Finais

O pacote ML-Agents, em conjunto com a plataforma Unity, fazem com que a criação de cenários, bem como de desenvolvimento e treinamento de Agentes Inteligentes, seja feita de forma simples para pessoas que são da área de Inteligência Artificial, ou não. Dois exemplos foram apresentados a fim de ilustrar o uso dessas ferramentas, onde foi visto que o algoritmo para treinamento do Agente pode ser utilizado como uma caixa preta.

Com a finalidade de observar a Unity sob novos parâmetros e se aprofundar no assunto, é possível analisar diversos outros exemplos de ambiente. São recomendados três canais relacionados, da plataforma YouTube: *Bot Academy*[5], Sebastian Schuchmann[6] e *Immersive Limit*[7]. A plataforma *Medium*[8] também dispõe diversos artigos acerca do assunto.

Todos os arquivos dispostos no projeto de aprendizagem e mencionados no Tensorboard, assim como os tutoriais em vídeo estão disponíveis no repositório git do Grupo de Estudos CAVI (*Computer Graphics, Animation, Visualization and Inteligence*)[9].

[5] https://www.youtube.com/BotAcademy
[6] https://www.youtube.com/SebastianSchuchmannAI
[7] https://www.youtube.com/ImmersiveLimit
[8] https://medium.com/
[9] https://github.com/devcavi/mlagentscourse

Referências

[AurelianTactics 2018a] AurelianTactics (2018a). Ppo hyperparameters and ranges. https://medium.com/aureliantactics/ppo-hyperparameters-and-ranges-6fc2d29bccbe. Acesso em: 30 de jul. de 2020.

[AurelianTactics 2018b] AurelianTactics (2018b). Understanding ppo plots in tensorboard. https://medium.com/aureliantactics/understanding-ppo-plots-in-tensorboard-cbc3199b9ba2. Acesso em: 30 de jul. de 2020.

[Community 2020a] Community, U. (2020a). Overview ml-agents. https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md. Acesso em: 30 de jul. de 2020.

[Community 2020b] Community, U. (2020b). Using tensorboard to observe training. https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Using-Tensorboard.md. Acesso em: 30 de jul. de 2020.

[Doyle et al. 2013] Doyle, J. C., Francis, B. A., and Tannenbaum, A. R. (2013). Feedback control theory. Courier Corporation.

[Juliani et al. 2018] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., and Lange, D. (2018). Unity: A general platform for intelligent agents.

[Lanham 2018] Lanham, M. (2018). Learn Unity ML-Agents – Fundamentals of Unity Machine Learning. Packt Publishing Ltd., Livery Place 35 Livery Street Birmingham B3 2PB, UK.

[Mills and Stufflebeam 2005] Mills, F. and Stufflebeam, R. (2005). Introduction to intelligent agents. http://www.mind.ilstu.edu/curriculum/ants_nasa/intelligent_agents.php. Acesso em: 30 de jul. de 2020.

[Pereira 2019] Pereira, W. (2019). Api: conceito, exemplos de uso e importância da integração para desenvolvedores. https://take.net/blog/devs/api-conceito-e-exemplos. Acesso em: 30 de jul. de 2020.

[Poole and Mackworth 2010] Poole, D. L. and Mackworth, A. K. (2010). Artificial Intelligence: foundations of computational agents. Cambridge University Press.

[Schulman et al. 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[Sutton and Barto 2018] Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[Trivedi 2019] Trivedi, C. (2019). Proximal policy optimization tutorial (actor-critic method). https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9afffbf6. Acesso em: 30 de jul. de 2020.

Capítulo

8

Como estimar um Software?

Métricas para aferição de Esforço, Prazo e Custo de um Produto de Software

Washington Henrique Carvalho Almeida, Fernando Escobar, Luciano de Aguiar Monteiro, Aislan Rafael Rodrigues Souza e Sérgio Sierro Leal

Abstract

This chapter deals with a recurring question for students, professionals and entrepreneurs in information technology area: how to estimate effort, time and cost related to software development? An overview of usable metrics will be presented for the proper estimation, based on the necessary effort, of the time and cost for systems development. Besides, basic concepts about metrics and a pricing process based on the CEK model will be presented, a proposal for using Canvas, EAP and Kanban.

Resumo

Este capítulo versa sobre uma dúvida recorrente para alunos, profissionais e empreendedores da área de tecnologia da informação: como estimar o esforço, prazo e custo relacionados ao desenvolvimento de um software? Serão apresentados conceitos básicos sobre métricas, listadas as principais características inerentes à formação de preço de software e as métricas para sua correta precificação. Além disso, será detalhado o Modelo CEK, que utiliza Canvas, EAP e Kanban, para orientar o detalhamento do exemplo de precificação de um projeto de software.

8.1. Introdução

O projeto de sistemas é uma atividade da Engenharia de Software estreitamente relacionada com o planejamento e o resultado desta é um produto de software. Todavia, a indústria de desenvolvimento prima por dois aspectos: softwares com qualidade e com um custo mensurável, estas duas perspectivas são contempladas por meio de um conceito usado em diversas engenharias, a medição. Por exemplo, para se compreender como está a qualidade de um código desenvolvido deve-se avaliar a porcentagem de erro por linha

de código e aplicando esse mesmo raciocínio, para se estimar o tamanho de um software ou o seu custo, deve-se definir e empregar alguma métrica.

A área de conhecimento da Engenharia de Software responsável pelos aspectos de medidas de sistemas é a de métrica de software e seu foco principal é usar métricas quantitativas para gerenciar o ambiente de desenvolvimento de software na organização, fornecendo uma visão interna da qualidade do produto de software [O'Regan 2014].

Um questionamento extremamente importante e oportuno em uma fase inicial de planejamento é o que medir em um projeto de sistemas, considerando que o termo "métrica de software" pode abranger muitas atividades, todas envolvendo algum grau de mensuração de software. Neste sentido, [Fenton e Bienan 2014] evidenciaram em seus estudos as principais métricas, elencadas a seguir.

- Modelos e medidas de estimativa de custos e esforços
- Coleção de dados;
- Modelos e medidas de qualidade;
- Modelos de confiabilidade;
- Métricas de segurança;
- Métricas estruturais e de complexidade;
- Avaliação de maturidade de capacidade;
- Gerenciamento por métricas; e,
- Avaliação de métodos e ferramentas.

Dentro dos modelos e medidas de estimativa de custos e esforços, uma das principais dificuldades de desenvolvedores, mesmo os mais experientes, é como estimar o custo de um software. Essa problemática impacta diretamente no planejamento do projeto de sistemas, especificamente em relação ao *lead time* (tempo total entre o início e fim para desenvolvimento de um item do *backlog* do projeto), a quantidade de profissionais envolvidos no projeto e, em decorrência, seu custo final. É importante destacar que um planejamento equivocado nesta fase implicaria no insucesso no projeto e prejuízos financeiros para a empresa.

Como medida para mitigar esta problemática, o presente capítulo visa apresentar uma visão geral sobre as principais métricas utilizadas para estimar software com base no esforço necessário, do prazo e do custo. A abordagem proposta apresentará as principais características da formação de preço de software, incluindo abordagem ágil.

Complementarmente, será contextualizada uma forma de precificação utilizando o Modelo CEK, a partir da integração das ferramentas Canvas de Projeto, Estrutura Analítica do Projeto (EAP) e Kanban, na qual o projeto é norteado desde a sua concepção, planejamento e acompanhamento de sua execução, de forma colaborativa, ágil, controlável e efetiva [Escobar et al. 2019]. O Modelo CEK, proposto pelos autores, foi apresentado pela primeira vez no VII Escola Regional de Computação Ceará, Maranhão, Piauí (ERCEMAPI 2019).

Concluindo, é exemplificada a proposta a partir do exemplo de um projeto de software, o Projeto Moto Easy e a criação de seu mapa de custos.

Esse capítulo está organizado da seguinte forma: 10.1 – Introdução; 10.2 – Definições; 10.3 – Principais Características da Formação de Preço de Software; 10.4 –

Métricas para Precificação; 10.5 – Modelo CEK; 10.6 – Exemplo de aplicação; e 10.7 – Conclusão.

8.2. Definições

Medir software é uma premissa essencial em projetos de sistemas. Desenvolvedores medem as características de seu software para entender se os requisitos são consistentes e completos, se o design é de alta qualidade e se o código está pronto para ser lançado. Gerentes de projeto medem atributos de processos e produtos para fazer projeções de quando o software estará pronto para entrega e se o orçamento será excedido; complementarmente, organizações usam medições de avaliação de processos para selecionar fornecedores de software [Fenton and Bieman 2014].

Gerentes de projetos possuem como proposição básica que o resultado de um projeto é único e exclusivo; esta proposição, por conseguinte, se aplica aos projetos de desenvolvimento de sistemas. Cada projeto de software compartilha desafios distintos relacionados à tecnologia, pessoas e cronogramas. [Bhardwaj and Rana 2016] destacam em seu estudo, que um dos principais desafios enfrentados pelo gerente de projetos de software é identificar as principais métricas de software para controlar e monitorar a execução do projeto.

Os autores explicam, ainda, que cada projeto de desenvolvimento de software, apesar de único e exclusivo, compartilha algumas métricas comuns que podem ser usadas para controlar e monitorar a execução do projeto. Essas métricas são tamanho de software, esforço, duração do projeto e produtividade. Proporcionam, desta forma, uma visibilidade ao gerente de projeto sobre os seguintes aspectos: o que entregar (tamanho), o que foi entregue no passado (produtividade) e quanto tempo levará para entregar mantida a capacidade atual da equipe (tempo e esforço).

A métrica de software busca definir um método padrão para medir certos atributos do processo, produto ou serviço, que se traduzem em indicadores chave de desempenho (KPI). [Jethani 2013] explica que a medição de atributos ajuda a obter informações valiosas sobre problemas e processos, permitindo identificar e gerenciar riscos, ajudando na detecção e resolução precoces de problemas. A medição de atributos fornece informações que melhoram a tomada de decisão objetiva no prazo.

Uma métrica pode possuir sua telemetria baseada em uma única origem ou pode ser uma combinação de dados de várias fontes. Qualquer ponto de dados rastreado acaba se tornando uma métrica que pode ser usada para medir o desempenho de uma equipe. [Davis 2015] destaca alguns exemplos destas métricas, conforme a seguir.

- Velocidade - o desempenho relativo de sua equipe ao longo do tempo
- Linhas de código alteradas (CLOC) - o número de linhas de código que foram alteradas
- Hora-Extra – quantidade de horas trabalhadas a mais, que não estavam previstas no projeto.

Métricas de Software, conforme abordado anteriormente, possuem vários objetivos, como avaliar a qualidade da aplicação e desempenho da equipe. Todavia, o

foco do presente capítulo é estimar software, a partir do qual, algumas métricas específicas devem ser analisadas com mais detalhes.

- Tamanho do software é a medida da funcionalidade do software que está sendo entregue ou espera-se que seja entregue. O tamanho do software é uma medida numérica dos requisitos de sistemas que são definidos qualitativamente pelo usuário, depende apenas do que entregar e não de como entregar. O tamanho do software é uma medida numérica dos requisitos funcionais [Bhardwaj and Rana 2016].
- Esforço é o tempo acumulado gasto por toda a equipe do projeto. O esforço geralmente mede horas pessoais, meses pessoais ou dias pessoais, mas o horário pessoal é a unidade mais adequada e inequívoca, já que outras unidades requerem conversão de hora para dia ou mês. Essa conversão não é padrão porque em alguns países (países especialmente desenvolvidos) há 8 horas de trabalho por dia, enquanto em alguns países (países em desenvolvimento) são mais de 8 horas [Bhardwaj and Rana 2016].
- Tempo representa a duração do calendário do projeto. Ele informa as datas esperadas de início e término do projeto. Não há relação linear entre tempo e esforço. O gerente de projeto deve entender que podemos reduzir a duração geral do projeto adicionando mais membros da equipe, mas além de certo ponto, o aumento no tamanho da equipe não resultará na redução da duração do projeto, mas aumentará a duração do projeto [Bhardwaj and Rana 2016].

Projetos ágeis de desenvolvimento de software possuem algumas características específicas, pois os requisitos podem ser alterados ao longo da evolução do projeto, tornando-se desta forma necessário usar métricas distintas ou adaptadas dos projetos de desenvolvimento tradicionais. Para [Padmini, Dilum and Perera 2015], embora algumas das métricas tradicionais possam ser adaptadas para uso no desenvolvimento ágil, algumas não são adequadas. Isso ocorre devido ao processo ágil de desenvolvimento ser iterativo e incremental, que está disposto a incorporar mudanças ao longo do processo.

Em uma Revisão Sistemática (RS) de [Canedo and Da Costa 2018], a questão principal versou sobre métricas e métodos usados para fazer estimativas de esforço, prazos e custos para o planejamento ágil de projetos de software. Os autores chegaram às métricas apresentadas a seguir.

- Story Point é uma métrica usada no gerenciamento e desenvolvimento ágil de projetos para determinar (ou estimar) o esforço (dificuldade) de implementar uma determinada história. Nesse contexto, uma história é uma necessidade comercial específica atribuída à equipe de desenvolvimento de software;
- Estimativa de esforço utiliza o Planning Poker, uma das técnicas mais populares para equipes ágeis, no planejamento e estimativa de esforço antes de iniciar cada iteração;
- Estimativa de tamanho são agrupadas com as técnicas de estimativa de esforço porque, no contexto do desenvolvimento ágil, a estimativa de esforço geralmente é derivada das estimativas de tamanho e dos cálculos de velocidade.

[Canedo and Da Costa 2018] ainda evidenciaram em sua RS dois trabalhos bem inovadores: (1) na área de **estimativas de projetos ágeis**, que sugere uma estrutura que

depende do uso da lógica fuzzy e visa ajudar na produção de estimativas precisas, propondo-se modificar o método de pontos de caso de uso (UCP) para torná-lo adequado para o desenvolvimento ágil de software, nomeando o novo método como a versão interativa do UCP (iUCP); e, (2) na área de **previsão de esforço** derivado de por mudanças nos requisitos de software. O modelo integra a análise do impacto das mudanças com o modelo de estimativa de esforço COCOMO, para melhorar a precisão das estimativas de esforço decorrentes de mudanças em projetos de desenvolvimento ágil de software.

8.3. Principais Características da Formação de Preço do Software

Em regra, quando estamos em processo de formulação do preço de um software para algum cliente, costumamos calcular apenas o custo do desenvolvimento e o lucro do desenvolvedor, conforme [Sommerville 2019]. Entretanto, nem sempre o relacionamento entre o custo e o preço final costuma ser tão simples assim.

De acordo com [Pereira 2004], precificar software é uma arte, os produtos de software podem ser vendidos por unidade/licença, pelo número de usuários simultâneos, número de servidores, por upgrades (atualização de versão), dentre outras formas. A estratégia do preço a ser utilizada é impactada pelas características de como o software será vendido, além de impactar a competividade entre as empresas fabricantes de software.

O processo de precificação requer das empresas uma constante análise e monitoramento dos fatores que influenciam o ambiente interno e externo. Podemos observar alguns desses fatores na Tabela 1.

Tabela 1 - Fatores que afetam a precificação. Fonte: [Sommerville 2019] adaptado.

Fator	Descrição
Termos contratuais	Um cliente pode permitir que o desenvolvedor mantenha a propriedade do código-fonte, o que possibilitaria o reuso em outro projetos.
Incerteza da estimativa de custos	Quanto mais incerto o cenário, uma organização pode aumentar o percentual de contingência além do seu lucro.
Saúde financeira	Empresas com problemas financeiros podem baixar seu preço para ganhar um contrato. É melhor baixar o lucro ou trabalhar em lucro zero do que a falência.
Oportunidade de mercado	Pode haver cotação baixa do preço para entar em um novo segmento de mercado, por exemplo na área pública, conseguir um atestado de capacidade poderá dar chance de novos contratos futuros com melhores percentuais de lucro.
Volatilidade de requisitos	Se houver mudança de requisitos previsíveis, a organização pode baixar seu preço e cobrar altos preços para incorporar mudanças posteriores.

Durante o processo de precificação, devemos levar em consideração aspectos organizacionais, econômicos, políticos e comerciais, além daqueles mostrados na Tabela 1. Possíveis riscos podem alterar o preço do software, seja para cima ou para baixo. Além disso, devemos levar em consideração que, a decisão sobre o preço do projeto deve ser

realizada em grupo, envolvendo as equipes de marketing e vendas, e demais gerências envolvidas na concepção do software.

O preço de um software costuma ser definido a partir de uma proposta preliminar, que será formulada com base nos fatores apresentados na Tabela 1. A partir disso, as negociações acontecem entre cliente e fornecedor, com o objetivo de estabelecer a especificação detalhada do software.

Em muitos projetos de desenvolvimento de software, os fatores fixos não costumam ser os requisitos do projeto, mas sim, requisitos para a precificação. Para que o preço não exceda o custo esperado pelo cliente, os requisitos podem ser alterados [Sommerville 2019].

Por exemplo, digamos que uma empresa esteja oferecendo um contrato para o desenvolvimento de um sistema de controle de frotas de veículos, seja para uma companhia de frete ou para uma companhia que possua uma frota de veículos própria, no qual o objetivo seja acompanhar e monitorar as atividades rotineiras.

Neste ponto, ainda não foi possível levantar, de forma detalhada, todos os requisitos necessários para a concepção do sistema, mas mesmo assim, a empresa apresenta uma proposta com um valor estimado em R$ 800 mil, imaginando que seja um valor competitivo perante as demais fabricantes de software concorrentes, e que esteja dentro do orçamento do cliente.

Uma vez o contrato celebrado entre o cliente e o fornecedor, os requisitos detalhados do sistema são negociados, para que a funcionalidade básica seja entregue. Durante essa etapa, custos adicionais são estimados devido a outros requisitos adicionais, caso extrapole o preço inicial, mas não prejudique a funcionalidade básica negociada durante a celebração do contrato, esses requisitos adicionais podem ser financiados em um futuro orçamento. Dessa forma, o cliente evita que seu orçamento seja consumido mais do que o inicialmente alocado, não aumentando os custos para a compra do software necessário.

Como vimos no exemplo acima, o custo de um projeto costuma ser fracamente relacionamento ao preço indicado para o cliente, devido a isso, uma estratégia que é comumente usada é “preço para ganhar” [Sommerville 2019]. Definir “preço para ganhar” significa que uma empresa tem uma ideia do preço que o cliente espera pagar e, com base nisso, uma oferta de contrato é feita com base no preço esperado pelo cliente. Depois disso, os custos adicionais para outros requisitos são estimados, para serem adicionados, ou não, na composição do contrato.

8.3.1. Modularidade e Custo do Software

De acordo com [Pressman 2016], modularidade é a manifestação mais comum da separação por interesses. Assim, um software baseado em modularidade é dividido em componentes separadamente especificados e localizáveis, algumas vezes denominados módulos, que são integrados para satisfazer os requisitos de um problema.

Um software monolítico, cuja definição remete a um grande software composto de um único módulo, não costuma ser entendido de forma fácil por um engenheiro de

software. O número de variáveis e complexidade geral torna o entendimento de um software monolítico quase impossível.

Como consequência, caso o engenheiro de software decida seguir um modelo monolítico no desenvolvimento do software, a entrega para o cliente dependerá que todas as funcionalidades e requisitos estejam implementados. Seguir esse tipo de modelo demanda um maior tempo de desenvolvimento, algo intimamente relacionado aos custos que serão necessários para finalizar o produto acordado durante a concepção do contrato. Quanto maior o tempo, maior serão os custos, maior será o preço repassado ao cliente, o que pode tornar o contrato inviável, levando a uma possível rescisão.

[Pressman 2016] nos mostra que, em quase todos os casos, devemos dividir o projeto em vários módulos para facilitar a compreensão e, consequentemente, reduzir os custos necessários para construir o software esperado pelo cliente. Entretanto, é preciso ter cuidado com o número de módulos que serão utilizados para dividir o software que está em desenvolvimento. A Figura 1 demonstra a relação entre modularidade e custo do software.

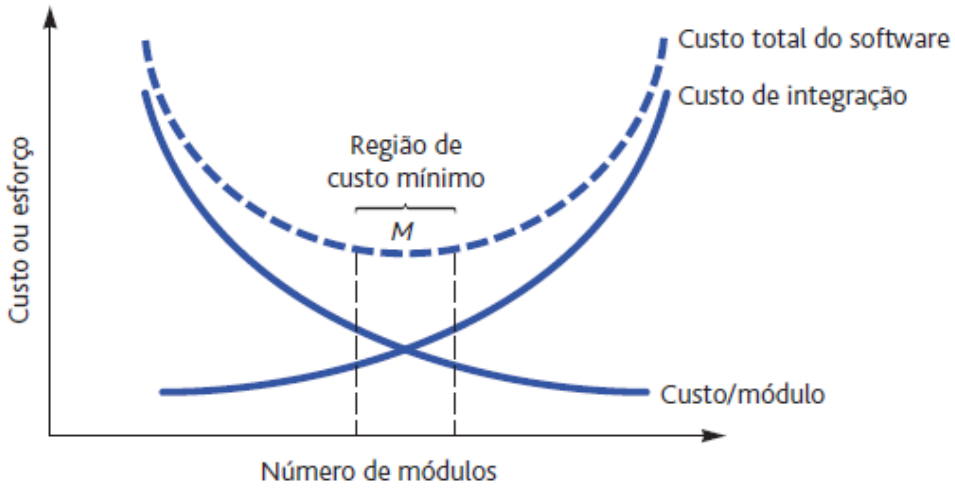


Figura 1 - Modularidade e custo do software. Fonte: [Pressman 2016].

Conforme a Figura 1, o esforço que será demandado para o desenvolvimento de um respectivo módulo individual do software tende a reduzir conforme o número total de módulos aumenta, dividindo o volume de trabalho na codificação. Além disso, é possível entregar ao cliente, de forma gradativa, os requisitos estabelecidos em contrato, conforme os módulos vão sendo finalizados, validados e testados. Dessa forma, segundo [Pressman 2016], dado um mesmo conjunto de requisitos, mais módulos significa um tamanho individual menor.

Por outro lado, é possível analisar um outro cenário a partir da Figura 1. Um ponto importante ao se utilizar a modularidade ao desenvolver um software é a capacidade de integração entre os módulos. Conforme a quantidade de módulos aumenta, maior o esforço associado à integração dos módulos.

Diante disso, ao modularizar um projeto, deve haver um equilíbrio, de modo que o desenvolvimento possa ser planejado mais facilmente, incrementos de software possam ser definidos e entregues, as mudanças possam ser mais facilmente acomodadas, os testes

e a depuração possam ser conduzidos de forma mais eficaz e a manutenção no longo prazo possa ser realizada sem efeitos colaterais graves [Pressman 2016].

8.3.2. Defeitos de Software nos Custos

Um ponto importante que pode impactar nos custos do desenvolvimento de um software se relaciona a possíveis falhas e defeitos. [Pressman 2016] pontua que ambos indicam um problema de qualidade que é descoberto após o software ser liberado para os usuários (ou para outra atividade estrutural dentro da gestão de qualidade).

Para mitigar isso, deve-se manter revisões técnicas, com o objetivo de encontrar erros durante o processo de desenvolvimento, para que não se tornem defeitos depois da liberação do software. Caso seja necessário realizar ajustes em alguma parte do software, esse esforço será refletido em aumento nos custos de desenvolvimento. Como será pontuado adiante, mudanças, de qualquer natureza, geram novos custos, mesmo que seja uma mudança com o objetivo de realizar alguma correção no software.

De acordo com [Pressman 2016], as revisões de software são como um "filtro" para a gestão de qualidade, isso significa que elas são aplicadas em várias etapas durante o processo de engenharia de software e servem para revelar erros e defeitos que podem ser eliminados.

A descoberta de falhas de projeto durante as revisões técnicas ajuda a eliminar grande percentual dos possíveis erros, evitando novos custos com correções, ou alterações, em partes do software que já foram liberadas para uso do cliente e dos usuários.

Conforme ainda pontuado por [Pressman 2016], detectar e eliminar um grande percentual de possíveis erros do software antes da sua liberação, reduz substancialmente o custo das atividades subsequentes dentro do projeto.

8.3.3. Desenvolvimento Ágil na Formação de Preço do Software

Ainda segundo [Pressman 2016], a engenharia de software ágil combina filosofia com um conjunto de princípios de desenvolvimento, os quais têm-se um foco na satisfação do cliente e a entrega incremental antecipada; equipes de projeto pequenas e altamente motivadas; métodos informais; artefatos de engenharia de software mínimos; e, acima de tudo, simplicidade no desenvolvimento geral.

A implementação do desenvolvimento ágil prioriza a entrega mais do que a análise e o projeto, além de uma comunicação ativa e contínua entre desenvolvedores e clientes. Entretanto, desenvolvimento ágil não significa que não será criado nenhum tipo de documento ou documentação, significa que apenas os documentos que vão ser consultados mais adiante no processo de desenvolvimento são criados [Pressman 2016].

Durante o desenvolvimento de um projeto de software, pode acontecer de ser difícil, ou impossível, prever como esse software irá evoluir com o tempo. Por exemplo, no caso de um aplicativo móvel, pode ser difícil prever futuros requisitos do cliente, conforme o uso do aplicativo é disseminado entre os usuários. Pode acontecer também, das condições do mercado mudarem rapidamente, a presença de novas empresas concorrentes, novos softwares com a mesma funcionalidade básica, assim como, as

necessidades do cliente podem mudar, todos esses fatores afetam na forma como o software será desenvolvido e nos custos necessários para esse projeto.

Como [Pressman 2016] sugere, não é possível definir os requisitos completamente antes do início do projeto. É preciso ser ágil o suficiente para garantir uma resposta, mantendo o ambiente de negócios fluido, algo que implica em mudanças constantes. Qualquer tipo de mudança pode sair cara – particularmente se for mal controlada e mal gerenciada. Nesse ponto, a metodologia ágil procura reduzir os custos da mudança no processo de software.

Segundo [Jacobson 2002], a agilidade se tornou um sinônimo para descrever um processo de desenvolvimento de software moderno. Uma equipe ágil precisa ser rápida e capaz de responder de modo adequado às mudanças. Mudanças no software que está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto. Ainda de acordo com [Jacobson 2002], os engenheiros de software devem ser rápidos, caso queiram assimilar as rápidas mudanças citadas anteriormente.

O processo de desenvolvimento de software tradicional apresenta os custos relacionados às mudanças de forma não linear, os quais costumam aumentar conforme o avanço do projeto. Observe na Figura 2, uma curva preta contínua. Conforme pontuado por [Pressman 2016], os custos crescem rapidamente, e o tempo e os custos necessários para assegurar que a mudança seja feita, sem efeitos colaterais, não serão insignificantes.

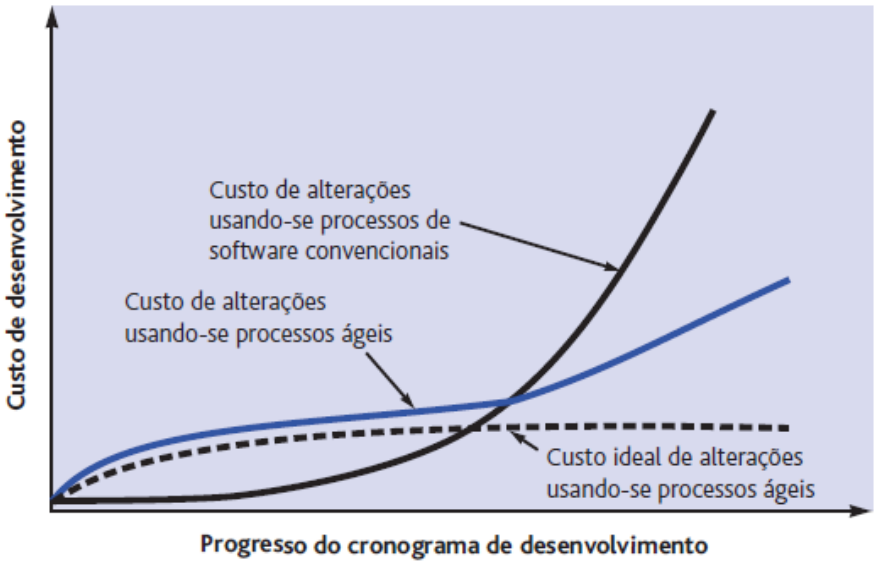


Figura 2 - Custos de mudanças como uma função do tempo em desenvolvimento. Fonte: [Pressman 2016].

O processo de desenvolvimento ágil procura, por meio da aplicação de sua metodologia, dar um direcionamento na forma como lidar com as constantes mudanças durante o desenvolvimento do software. A utilização de uma metodologia ágil ajuda a "achatar" a curva dos custos de mudanças, conforme é possível observar na Figura 1, por meio da curva azul contínua. Assim, como apontado por [Pressman 2016], a equipe de

software consegue assimilar as alterações, realizadas posteriormente em um projeto de software, sem um impacto significativo nos custos ou no tempo.

Percebe-se que é importante ter em mente que possíveis mudanças podem ocorrer durante o desenvolvimento de um software, e avaliar como isso pode impactar no preço já aprovado pelo o cliente, ou seja, no orçamento já alocado para o projeto.

8.4. Métricas para precificação

No dia-a-dia das empresas inúmeras métricas são utilizadas para precificar o custo de desenvolvimento, na pesquisa de [Almeida and Furtado 2019] são levantadas, por meio de pesquisa a empresas desenvolvedoras de software, quais métricas são utilizadas com esses objetivos, ou seja, precificar o custo envolvido no desenvolvimento. As métricas reportadas são: Homem-Hora (HH), Pontos de Função (PF), Unidade de Serviço Técnico (UST), Hora de Serviço Técnico (HST) e outras, que no final são variações ou combinações dessas métricas.

HH é a métrica mais simples e amplamente utilizada em toda a indústria de software, baseada apenas no cálculo de quanto tempo é gasto, com uso da unidade hora, para executar uma determinada tarefa. Alguns estudos mais recentes mostram que, para o cálculo do homem-hora, são usadas a opinião de especialistas, dentro daquele escopo que está sendo desenvolvido e, para dirimir variações grotescas, podem ser utilizadas técnicas como a Delphi.

PF não é uma métrica direta de esforço, mas a partir da medição funcional atrelada à produtividade e tecnologia utilizada com base histórica, pode se chegar ao esforço envolvido e ao custo. Há inúmeros estudos que relatam uma má precisão da utilização da métrica para cenários de manutenção ou sustentação de software.

UST é uma métrica utilizada em contratos de governo na qual há a terceirização do desenvolvimento de software. Para compor a UST, é criada uma unidade de medida as vezes vinculada à hora e são criados pacotes de trabalho que podem ser verificados. A partir dos pacotes de trabalho, é montado um catálogo de serviço previamente definido e são solicitadas as atividades baseadas neste catálogo.

HST é aplicada de maneira similar à UST, mas já define previamente a hora como unidade de medida.

8.5. O Modelo CEK

De acordo com [Escobar et al. 2019], o Modelo CEK, fundamentado na abordagem por projetos aplicada à engenharia de software, por meio da integração das ferramentas de gestão do Canvas de Projeto, da EAP e Kanban, foi proposto como uma das possíveis respostas aos problemas do desenvolvimento de software.

Considerado pelos próprios autores como um método híbrido de gerenciamento de projetos, que combina elementos das abordagens tradicional e ágil, de forma a mitigar os problemas relacionados aos projetos de desenvolvimento de software, em especial às deficiências no levantamento de requisitos e estabelecimento de objetivos, e às falhas de comunicação entre equipe e clientes, tendo como foco a ideação, planejamento e monitoramento da execução, reduzindo desperdícios, apoiando a transformar ideias em projetos colaborativos, ágeis, controláveis e efetivos [Escobar et al. 2019]. O Modelo

CEK está esquematizado na Figura 3, com destaque para as interações entre as ferramentas que o compõem.

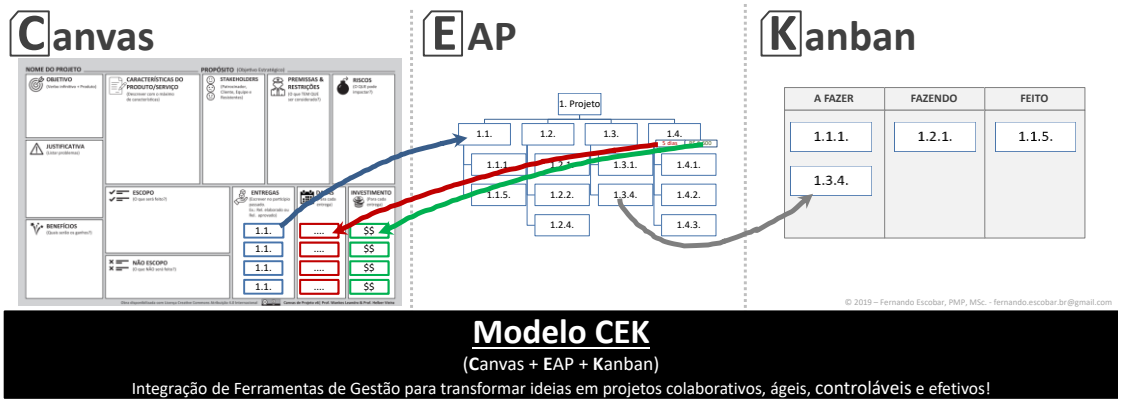


Figura 3 - O Modelo CEK

Proposto por [Leandro and Vieira 2018], o Canvas Project Design é resultado da síntese entre modelos teóricos, boas práticas e diversas sessões de tentativa e erro, incorporando conceitos, abordagens e características do Design Thinking, da Programação Neurolinguística, dos Guias PMBOK e PRINCE2, das Abordagens Ágeis e do Princípio MECE (Mutuamente Exclusivo, Coletivamente Exaustivo). A Figura 4 ilustra o Canvas Project Design.

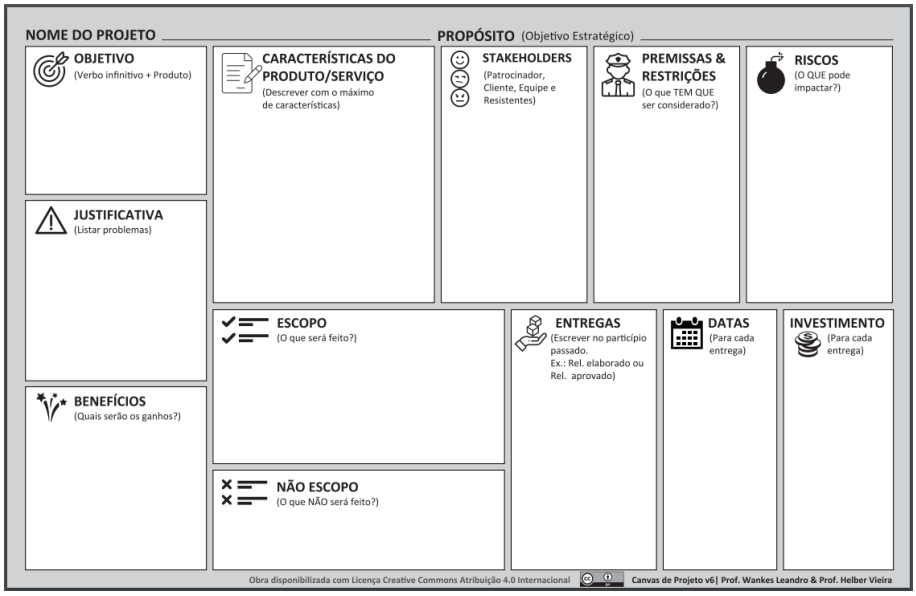


Figura 4 - Canvas Project Design. Fonte: [Leandro and Vieira 2018]

Fundamentado na simplicidade do modelo do Canvas Project Design, com sua abordagem visual, participativa e colaborativa, o Modelo CEK adota o Canvas Project

Design para sua aplicação prática, como ferramenta de gestão para apoiar equipes na fase de concepção de seus projetos [Escobar et al. 2019].

Complementar à etapa de concepção, abordando aspectos relativos ao planejamento, segundo [Escobar et al. 2019], também fundamentado em uma abordagem visual e colaborativa, o Modelo CEK acolhe o processo de criação da EAP.

De acordo com o PMBOK 6 [PMI, 2017], o trabalho relacionado à criação da EAP é o processo de, a partir da técnica da Decomposição, dividir e subdividir o escopo do projeto e suas entregas em partes menores e mais facilmente gerenciáveis, até o nível do pacote de trabalho. A visão estruturada do que deve ser entregue é fundamental para o planejamento e a efetividade do projeto. A Figura 5 apresenta a estrutura de uma EAP.

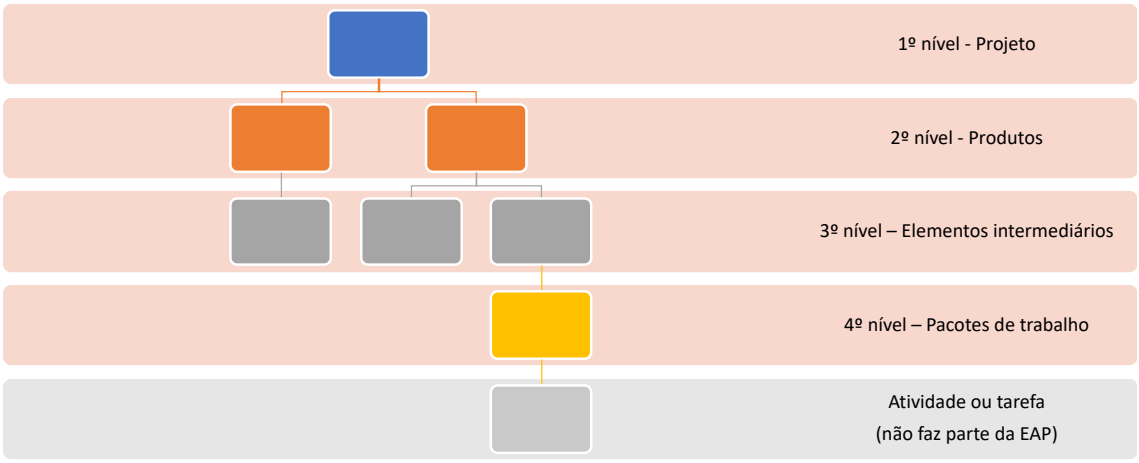


Figura 5 - Estrutura da EAP. Fonte: [Escobar et al. 2019]

Segundo [Escobar et al. 2019], na abordagem proposta pelo Modelo CEK, ao aplicar a técnica de Decomposição, dividindo as Entregas (resultados intermediários) idealizadas no âmbito do Canvas, até o nível do pacote de trabalho, é possível estimar duração e custos. Complementar, ainda segundo os autores, com a técnica da Agregação, as estimativas de duração e custos são somadas nos níveis superiores da EAP, até derivar nos respectivos blocos de Datas e Investimentos do Canvas Project Design. A Figura 6 exemplifica a abordagem para agregação de custos, desde os pacotes de trabalho (nível mais baixo), passando pelos elementos intermediário, o 2º nível da EAP e o projeto como um todo.

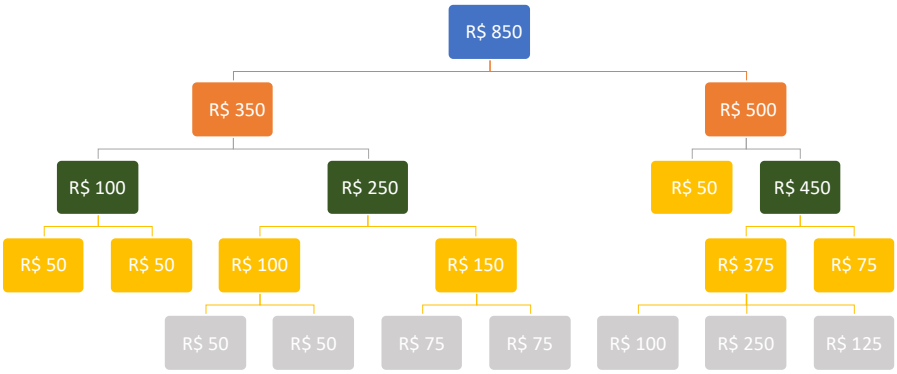


Figura 6 - Exemplo de abordagem para agregação de custos. Fonte: [Escobar et al. 2019]

Para controlar a execução do projeto, por meio do acompanhamento da execução dos pacotes de trabalho, definidos de forma colaborativa com o Canvas e com a EAP,

buscando a efetividade dos projetos de software, de acordo com [Escobar et al. 2019], o Modelo CEK adota o Kanban, buscando reduzir desperdícios, melhorar a comunicação, agilizar o fluxo de entrega de forma contínua em um sistema puxado.

De acordo com [Khaled Yacoub et al. 2016], o Kanban possui cinco princípios fundamentais: (1) visualizar o fluxo de trabalho; (2) limitar o trabalho em andamento; (3) medir e gerenciar o fluxo; (4) tornar explícitas as políticas do processo; e (5) usar modelos para reconhecer melhorias e oportunidades. Esses princípios são aplicados a partir de um Quadro Kanban, que permite visualizar o fluxo de atividades do processo em várias colunas. A Figura 7 ilustra um exemplo de Quadro Kanban.

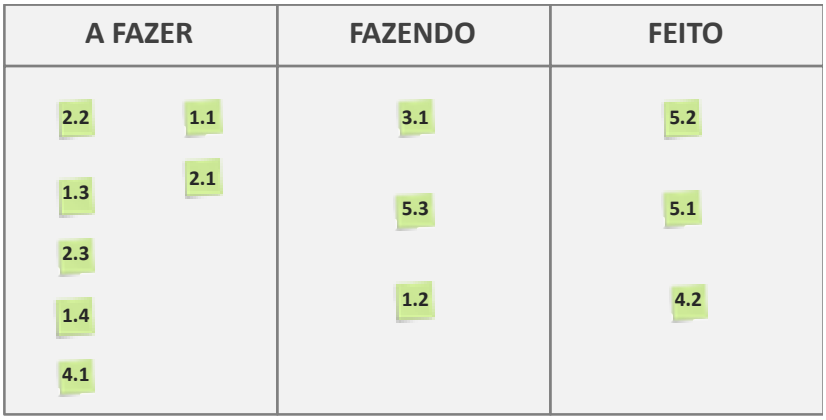


Figura 7 - Quadro Kanban. Fonte: [Escobar et al. 2019]

O Modelo CEK adota a sistemática do Quadro Kanban, com os Pacotes de Trabalho da EAP que ainda não foram priorizados, sendo colocados na coluna A FAZER (backlog do projeto); os Pacotes de Trabalho que já foram priorizados e que tiveram seu trabalho iniciado (limitados a 1 Pacote de Trabalho por equipe ou responsável – em atenção ao limite do Work In Progress, um dos princípios do Kanban) são movidos para a coluna FAZENDO, até que sejam concluídos e movidos para FEITO – à medida que os cartões são movidos para a coluna FEITO, novos espaços são liberados para a coluna FAZENDO, em um sistema puxado e de fluxo contínuo [Escobar et al. 2019].

8.6. Exemplo – o Projeto Moto Easy

Para exemplificar a proposta, fundamentada no Modelo CEK, será apresentado o mapa de custos criado para o Projeto Moto Easy.

Em diversas cidades do Brasil existe um serviço peculiar de transporte de usuários realizados através de motos. Os prestadores destes serviços são denominados mototaxistas. Esses profissionais são prestadores de serviço autônomos que são contratados diretamente pelo cliente. O Moto Easy irá realizar a gestão de corridas desta categoria, integrando os usuários aos profissionais por meio de uma solução desenvolvida na plataforma mobile e aplicação web para gestão dos serviços.

Observe que a formação de custos pode ser aplicada a qualquer tipo de projeto, independente da abordagem de desenvolvimento se tradicional (cascata ou waterfall) ou ágil. Entretanto, é pertinente observar que na abordagem ágil, a questão do retrabalho (e

de seu custo derivado) que tem que ser estimada e monitorada, pois a variação pode ser grande o que impactará no custo final.

8.6.1. O Projeto Moto Easy

A Figura 8 apresenta o Canvas Project Design preenchido para o Projeto Moto Easy, até a definição das Entregas, que irão compor o 2º nível de nossa EAP. Ao final, com o exercício de decomposição, estimativa de prazo e custo, e agregação, será possível derivar as Datas e Investimento para cada uma das Entregas.

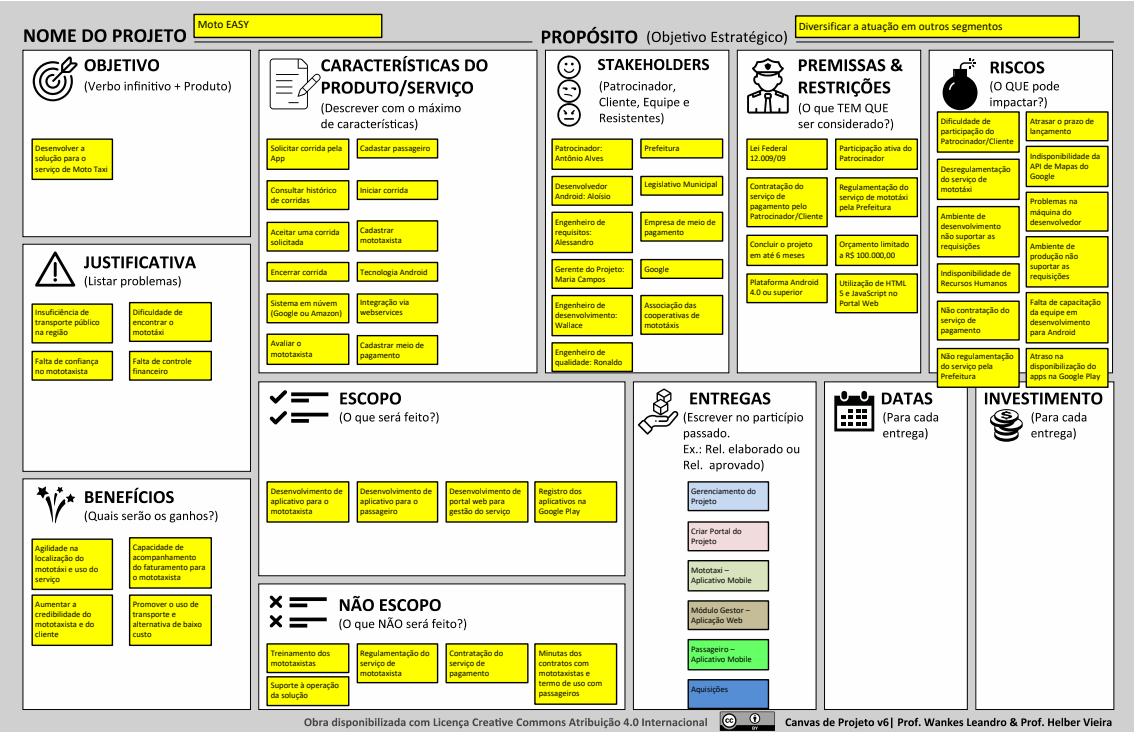


Figura 8 - o Canvas para o Projeto Moto Easy

O projeto foi realizado para entregar os produtos relacionados a seguir, diretamente vinculados às necessidades identificadas durante o levantamento realizado inicialmente com o Cliente:

- desenvolvimento de aplicativo para dispositivos móveis na plataforma Android destinado aos usuários de transporte, que será utilizado para a realização de chamadas aos serviços prestados pelos motos taxistas.
- desenvolvimento de aplicativo para dispositivos móveis na plataforma Android, para uso pelos profissionais prestadores do serviço de transporte de passageiros, os mototaxistas.
- desenvolvimento de sistema na plataforma web, para utilização pelo Cliente na gestão dos serviços oferecidos por meio da solução construída com o projeto.

A Figura 9 exemplifica a EAP para o Projeto MotoEasy.

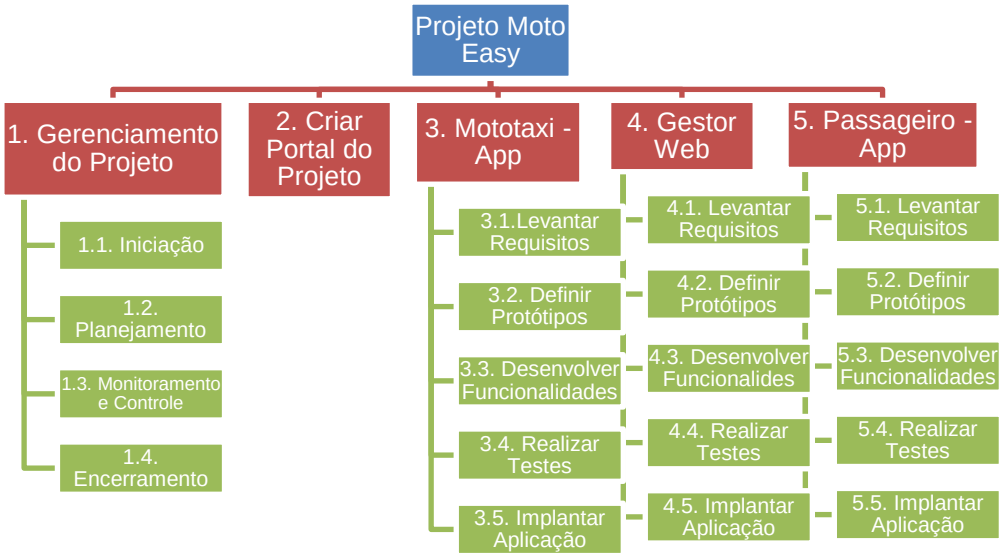


Figura 9 - EAP para o Projeto Moto Easy

8.6.2. O Mapa de Custos

Então definida a EAP, os custos têm que ser levantados numa visão *bottom-up*, partindo dos pacotes de trabalho. Assim foram feitos e estão apresentados nas tabelas a seguir, começando pelo gasto com pessoal.

Existem diversas formas de precificar, mas as mais utilizadas fazem uma composição de todos os custos do projeto e um percentual de lucro, assim foi realizada a estrutura de custos baseadas nos seguintes itens: material permanente, serviços, aquisição de software, pessoal, reserva técnica e lucro.

Na Tabela 2 é apresentado o gasto com material permanente para o projeto. É importante relatar o contexto: o projeto foi desenvolvido *part-time* por alunos de mestrado e foi necessária a aquisição de alguns materiais para facilitar a codificação e testes do software, mas em um cenário de uma *startup* devem ser contabilizados todos os demais custos envolvidos.

Tabela 2. Gasto com material permanente.

Descrição	Quantidade	Valor unitário	Valor total
Celular Samsung Galaxy S7	1	R$ 2.800,00	R$ 2.800,00
Celular Motorola G4	2	R$ 1.100,00	R$ 2.200,00
Notebook Dell Vostro	1	R$ 2.800,00	R$ 2.800,00
TOTAL GERAL			R$ 7.800,00

Na Tabela 3 são demostrados os custos com aquisição de licenças de software. Os demais softwares são todos baseados em licenças gratuitas ou softwares livres, não caracterizando custos adicionais.

Tabela 3. Custos com licenças de software

Descrição	Quantidade	Valor unitário	Valor total
Licença Mensal Adobe Creative Cloud	8	R$ 175,00	R$ 1.400,00
Corel Draw X8	1	R$ 2.399,00	R$ 2.399,00
TOTAL GERAL			R$ 7.800,00

Os serviços descritos na Tabela 4 compõem os gastos com serviços, que abrangem a contratação de serviços em nuvem, conexão de internet e gastos com energia elétrica da equipe envolvida.

Tabela 4. Gastos com serviços

Descrição	Quantidade	Quantidade de meses	Valor Unitário	Valor total
Assinatura Internet Plano 4G	7	8	R$ 110,00	R$ 6.160,00
Hospedagem para estrutura Amazon	1	8	R$ 300,00	R$ 2.400,00
Energia Elétrica *home office*	7	8	R$ 40,00	R$ 2.240,00
TOTAL GERAL				R$ 10.800,00

E, para finalizar a formação de custos, a folha de pagamento pessoal, colocando um valor médio na hora de cada membro da equipe, como se tratava de uma equipe multidisciplinar, inerente a uma equipe ágil, cada um dos membros atuou no papel e o que variou foi o tempo de alocação em cada função, essa estrutura pode variar conforme o projeto e a estrutura de custos de cada empresa.

Tabela 5. Folha de pagamento pessoal Projeto Moto Easy

	Valor Hora	R$ 75,00						
Função	**Horas trabalhadas no projeto**	**Valor Bruto**	**INSS Retenção**	**INSS Patronal**	**Contribuição para terceiros**	**Risco ambiental do trabalho**	**Encargos**	**Prolabore líquido**
			11%	20%	5.8%	1%		
Gerente de projeto	95	R$ 7.125,00	R$ 783,75	R$ 1.425,00	R$ 413,25	R$ 71,25	R$ 2.693,25	R$ 6.341,25
Engenheiro de requisitos	70	R$ 5.250,00	R$ 577,50	R$ 1.050,00	R$ 304,50	R$ 52,50	R$ 1.984,50	R$ 4.672,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de Qualidade	70	R$ 5.250,00	R$ 577,50	R$ 1.050,00	R$ 304,50	R$ 52,50	R$ 1.984,50	R$ 4.672,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
TOTAL GERAL FOLHA DE PAGAMENTO							R$ 16.868,25	R$ 39.716,25

Todos esses custos devem ser agregados de forma a compor a Tabela 6, com uma visão geral, colocando um percentual de 10% de Reserva Técnica, para cobrir riscos inerentes ao projeto, e 35% de lucro. Em contratações com o governo, os lucros são limitados, pois já existem legislações que proíbem lucros exorbitantes, mas na esfera privada, onde estão inseridas a maioria das startups, o lucro pode ser ajustado. Observe que na venda final para o cliente, esses custos e planilhas não são apresentados, o que está estruturado nesse trabalho é formação dos preços.

Tabela 6 - Resumo dos custos do projeto

Descrição	Valor (R$)
Pessoal	67.384,50
Material Permanente	7.800,00
Aquisição de Software	3.799,00
Serviços	10.800,00
Reserva Técnica (10%)	8.978,13
Lucro (35%)	31.424,23
TOTAL	130.186,08

A Tabela 7 apresenta um cronograma de desembolso, uma boa prática de gestão para, por exemplo, a cobrança do cliente por pacotes de trabalho no tempo, o que manterá a saúde financeira da empresa, com o fluxo de capitais.

Tabela 7 - Cronograma de desembolso de recursos financeiros

Mês	Pessoal	Material permanente	Software	Serviços	Outros	Total
Janeiro	3.000,00			1.350,00	5.050,32	9.400,32
Fevereiro	10964,08	7.800,00	3.799,00	1.350,00	5.050,32	28.963,40
Março	10964,08			1.350,00	5.050,32	17.364,40
Abril	10964,08			1.350,00	5.050,32	17.364,40
Maio	10964,08			1.350,00	5.050,32	17.364,40
Junho	8.264,08			1.350,00	5.050,32	14.664,40
Julho	8.264,08			1.350,00	5.050,32	14.664,40
Agosto	4.000,02			1.350,00	5.050,34	10.400,36
TOTAL	67.384,50	7.800,00	3.799,00	10.800,00	40.402,58	130.186,08

Por fim, é apresentado um cronograma de execução do projeto com as horas alocadas proporcionando uma visão no tempo e o total de horas alocadas em cada fase do ciclo de desenvolvimento, compondo o Mapa de custos para o Projeto Moto Easy, conforme Tabela 8.

Tabela 8. Cronograma de execução do projeto

Descrição		**Início previsto**	**Término previsto**	**Status**	**Janeiro**	**Fevereiro**	**Março**	**Abril**	**Maio**	**Junho**	**Julho**	**Agosto**	**Horas trabalhadas**
					Dias úteis trabalhado								**3 Horas/Dia**
Iniciação	Definição do escopo	15/01/2017	04/02/2017	☑	14	3							51
Planejamento	Sprint Planning	05/02/2017	14/03/2017	☑		20	12						96
Execução, Monitoramento e Controle	Primeiro Release	15/03/2017	29/04/2017	☑			15	25					120
	Segundo Release	30/04/2017	14/06/2017	☑					27	12			117
	Terceiro Release	15/06/2017	30/07/2017	☑						14	25		117
	Quarto Release	31/07/2017	15/08/2017	☑							1	13	42
Encerramento	Entrega final e encerramento do projeto	16/08/2017		☑								8	24
Total Geral					14	23	27	25	27	26	26	21	567

8.7. Conclusão

Em uma análise preliminar do assunto a precificação aparenta ser uma tarefa fácil de ser realizada, bastando a composição dos custos e lucro dos itens que fazem parte do escopo do produto. Entretanto, o maior problema da formação de custos são os riscos atrelados ao processo e às atividades de desenvolvimento de software, e isso acaba por impactar nos custos finais.

O desenvolvimento de software parte de um contexto atrelado a riscos que, na maioria das vezes, acabam por ser concretizados, conforme buscou-se demonstrar nesse

trabalho, onde foram apresentados diversos aspectos inerentes a composição de custos e fatores conhecidos, tanto no mercado como na literatura acadêmica, que devem ser considerados na composição de preços de uma organização que constrói soluções de software.

Ainda no escopo desse estudo, foi apresentado um modelo emergente, o Modelo CEK, que agrega ferramentas para o amadurecimento do escopo que, em grande parte, costuma ser o maior problema para as empresas ao receber demandas de seus clientes. A medição e controle em um contexto de mudanças traz também a necessidade de entendimento por parte de quem produz da importância em saber se adaptar a esses cenários. Por isso, pode-se dizer que os maiores problemas enfrentados pela engenharia de software na verdade podem ser descritos como falta de engenharia de software, ou seja, não utilizar os inúmeros processos, métodos e ferramentas já consolidadas nessa área de estudo, que ainda pode ser considerada recente, se comparada a outras engenharias.

Por fim, nesse trabalho fica evidente que, sem foco na qualidade, as empresas e *startups* terão dificuldade para sobreviver. Portanto, não basta mais apenas boas intenções e bons profissionais, pois os custos inerentes às atividades acabam por ser um fator impactante em um mercado altamente competitivo.

8.8. Referências

BHARDWAJ, M.; RANA, A. Key Software Metrics and its Impact on each other for Software Development Projects. **ACM SIGSOFT Software Engineering Notes**, v. 41, n. 1, p. 1–4, 2016.

CANEDO, E. D.; DA COSTA, R. P. Methods and metrics for estimating and planning agile software projects. **Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018**, 2018.

DAVIS, C. W. H. **Agile Metrics in Action: Measuring and Enhancing the Performance of Agile Teams**. 1st. ed. USA: Manning Publications Co., 2015.

FENTON, N.; BIEMAN, J. **Software Metrics: A Rigorous and Practical Approach**. 3rd. ed. USA: CRC Press, Inc., 2014.

JETHANI, K. Software metrics for effective project management. **International Journal of Systems Assurance Engineering and Management**, v. 4, n. 4, p. 335–340, 2013.

O'REGAN, G. Software Metrics. In: Undergraduate Topics in Computer Science. Cham: Springer International Publishing, 2014. p. 151–183.

PADMINI, K. V. J.; DILUM BANDARA, H. M. N.; PERERA, I. Use of software metrics in agile software development process. **MERCon 2015 - Moratuwa Engineering Research Conference,** p. 312–317, 2015.

Sommerville, Ian (2019). Engenharia de Software. 10. ed. **Pearson Education**.

Pressman, Roger; Maxim, Bruce (2016). Engenharia de Software, Uma Abordagem

Profissional. 8. **Ed. McGraw-Hill Global Education**

Jacobson, Ivar (2002). A Resounding 'Yes' to Agile Processes—But Also More. **Cutter IT Journal.**

Almeida, W. H. C. and Furtado, F. (2019). Análise sobre métricas nos contratos de fábricas de software no âmbito da administração pública federal. 1. ed. Rio de Janeiro: **Albatroz**. v. 1

ESCOBAR, Fernando et al. (2019). Modelo CEK (Canvas+ EAP+ Kanban): Integração de Ferramentas de Gestão para transformar ideias em projetos colaborativos, ágeis, controláveis e efetivos.

Leandro, W. and Vieira, H. (2018). Canvas de Projeto - Canvas Project Design. 1a. ed. São Paulo: **Riemma**.

PMI (2017). Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK).

Khaled Yacoub, M., Abdel Athim Mostafa, M. and Bahaa Farid, A. (2016). A New Approach for Distributed Software Engineering Teams Based on Kanban Method for Reducing Dependency. **Journal of Software**, v. 11, n. 12, p. 1231–1241

Capítulo 9

Levantamento bibliográfico utilizando a ferramenta TheEnd

Martony Demes da Silva, Gleison de Andrade e Silva

Abstract

The bibliographic survey is one of the main stages in the construction of scientific research. This is due to the need to search, in databases, works related to the research topic. In this way, it is possible to catalog and reference relevant works that encourage research. However, this activity can be complex if a well-defined method and support tools are not used. This chapter aims to teach academics and researchers a method of conducting a bibliographic survey using Systematic Mapping of Literature using the resources of the tool TheEnd. It is hoped, therefore, that the researcher can build a bibliographic survey quickly and systematically .

Resumo

O levantamento bibliográfico é uma das principais etapas da construção da pesquisa científica. Isso se deve pela necessidade de buscar, em bases de dados, trabalhos relacionados ao tema da pesquisa. Desta forma, é preciso catalogar e referenciar trabalhos relevantes que fomentem a pesquisa. Porém, essa atividade pode ser complexa caso não se utilize um método bem definido e ferramentas de apoio. Este capítulo pretende ensinar aos acadêmicos e pesquisadores um método de realizar um levantamento bibliográfico por meio do Mapeamento Sistemático da Literatura utilizando os recursos da ferramenta TheEnd. Espera-se, com isso, que o pesquisador possa construir um levantamento bibliográfico de maneira rápida e sistemática.

9.1. Introdução

Levantamento bibliográfico ou Pesquisa Bibliográfica (PB) é a primeira etapa no processo de investigação científica [Detroz et al. 2015]. PB é o procedimento de consultar bases de dados de trabalhos científicos com o objetivo de descobrir e reunir informações que servirão de base para a construção da investigação a partir de um determinado tema.

De certo, a PB é um levantamento de toda a bibliografia publicada em forma de livros, revistas, artigos, publicações avulsas e imprensas escritas [Lakatos and Marconi 1991]. Com efeito, o pesquisador terá oportunidade de visualizar diversos aspectos oportunos em determinado assunto.

Consequentemente, com a evolução das pesquisas, ocorre um aumento do número de publicação e resultados [Petersen et al. 2008]. Nesse sentido, é importante realizar um levantamento do estado da arte desse tema. Esse levantamento possibilita que pesquisadores que obtenha uma visão geral do tema de interesse.

Porém, a condução de uma PB de forma parcial ou incompleta pode levar a resultados tendenciosos[Detroz et al. 2015]. Isso inviabiliza a replicação dos métodos utilizados e consequentemente conduzindo à conclusões incorretas. Diante desse quadro, o estudante de graduação que tem pouca maturidade em pesquisas bibliográficas pode ser levado a um caminho não linear de pesquisa científica.

Diante desse cenário, a proposta deste capítulo é apresentar uma prática de elaborar uma pesquisa bibliográfica por meio de um método denominado Mapeamento Sistemático da Literatura (MSL). A estratégia terá suporte de um ambiente digital Web e colaborativo *TheEnd* [Braga et al. 2016]. Ao final, o resultado será um levantamento bibliográfico de um determinado tema.

Nesse âmbito, sabe-se que há alguns tipos de PB os quais serão apresentados nesse trabalho. Entretanto, a escolha do Mapeamento Sistemático da Literatura justifica-se por ser um método científico capaz identificar, interpretar e sumarizar os trabalhos relevantes para determinada linha de pesquisa, área ou fenômeno de interesse de forma não tendenciosa e replicável [Kitchenham 2004a].

Como exposto, a proposta pretende utilizar ferramentas específicas de suporte à PB. São elas:

i) O *software TheEnd*: é um serviço Web capaz de auxiliar o planejamento, execução e sumarização dos resultados de um MSL, tornando a pesquisa mais ágil, colaborativa e replicável- *link de acesso à ferramenta*;

ii) A ferramenta *Google Sheets*: com este software, pode-se gerenciar os trabalhos selecionados e as informações extraídas do *Software TheEnd* de modo *on-line*.

O Diferencial desta proposta é a possibilidade de elaborar referencial teórico (e bibliográfico) de modo sistemático, replicável e colaborativo. Essas características facilitam o entendimento pelo o estudante iniciante em pesquisas científicas. Além disso, a ferramenta de MSL citada permite que vários pesquisadores trabalharem ao mesmo tempo em uma mesma pesquisa, centralizando e compartilhando dúvidas e conhecimento.

9.1.1. Objetivos

O objetivo geral do curso é explanar sobre levantamento bibliográfico com enfoque na PB sistemática. Os objetivos específicos são:

- Explicar sobre pesquisa bibliográfica e mapeamento sistemático;
- Apresentar a Ferramenta *Software TheEnd*;

- Apresentar uma prática de uma PB do tipo Mapeamento Sistemático com uso do *Software TheEnd*.

9.1.2. Plano do Curso

O plano do minicurso é ilustrado na Tabela 9.1. Vale destacar que, devido ao momento vigente (2020), a apresentação e prática será realizada na modalidade a distância.

Tabela 9.1. Plano do Minicurso

Item	**Descrição**
Público Alvo	Público em geral, pesquisadores, acadêmicos de graduação e pós-graduação
Requisitos para participar	Conhecimentos básicos de metodologia científica e de informática básica
Carga Horária	A definir
Método didático	Aulas expositivas onlines e prática com softwares
Período	10 de Setembro
Bibliografia	[Kitchenham 2004a, Keele et al. 2007, Braga et al. 2016, da Silva et al. 2020]

9.2. Pesquisa Bibliográfica

O passo inicial do processo de investigação é a Pesquisa Bibliográfica (PB). PB é o levantamento de toda a bibliografia já publicada, em forma de livros, revistas, publicações avulsas e imprensas escritas [Lakatos and Marconi 1991]. A condução desta etapa produz os referenciais teóricos, que são os trabalhos relacionados ou o estado da arte sobre um determinado tema [da Silva et al. 2020].

A PB é fundamental para encontrar lacunas, as quais são oportunidades de novas investigações. Além disso, é possível avaliar o que já foi estudado exaustivamente e que, talvez, o tema esteja saturado.

Diante disso, há a alternativa de realizar um levantamento de trabalhos relacionados de modo organizado, sistematizado e replicável. Esse mecanismo é denominado de Pesquisa Bibliográfica Sistemática (PBS). Ela contribui para aumento do domínio do conhecimento do pesquisador. Além disso, esse método favorece como ponto inicial de análise para possíveis novas contribuições. Na subseção 9.2.1 a seguir serão apresentados os tipos de pesquisas bibliográficas.

Pesquisas publicadas sobre PB são chamados de **estudos secundários**. Os **estudos primários** apresentam relatos de descobertas cientificas como experimentos ou estudos de casos. Uma pesquisa ou estudo secundário, portanto, realiza a revisão de vários estudos primários relevantes com objetivo de sintetizar as evidências de um determinado tema. Há, ainda, o **estudo terciário**, que realiza uma revisão dos artigos secundário relacionados.

9.2.1. Tipos de Pesquisas Bibliográficas

As pesquisas bibliográficas são classificadas em:

- **Tradicional**: são pesquisas que não seguem um processo sistemático definido. A experiência do pesquisador guia o processo de pesquisa. Diante disso, não há garantia de resultados imparciais: o pesquisador pode enfatizar somente os dados que concordam com as hipóteses apoiadas por ele;

- **Bibliométrica**: são PBs que usam métodos estatísticos e matemáticos para a obtenção de informações quantitativas sobre livros e artigos. Identifica e quantifica aspectos da produção bibliográfica em uma determinada área. Nesse tipo de PB, a relevância de um artigo ou a influência de um pesquisador a partir do número de citações feitas a esse artigo contribuem para a coleta dos dados;

- **Revisão Sistemática da Literatura (RSL)**: em inglês, *Sistematic Literature Review* (SLR)) este tipo de PB sistemática que objetiva identificar e avaliar de forma confiável, imparcial e replicável os trabalhos relevantes sobre determinado assunto. Assim, a revisão tem o propósito de identificar e entregar todas as pesquisas relevantes à linha de pesquisa, e não somente aquelas que corroborem para a hipótese apoiada pelo pesquisador. RSL também pode ser denominada de *survey*, *review*, levantamento ou panorama;

- **Mapeamento Sistemático da Literatura (MSL)**: em inglês, *Sistematic Literature Mapping* (SLM) é uma PB que faz visão geral da área de estudo, quantificando os resultados. Assim, é possível identificar tendências e lacunas de pesquisas a serem estudadas [da Silva et al. 2020]. Esse tipo pode ser identificado por outros nomes, como mapeamento sistemático de estudos, *overview* ou meta-análise;

- **Revisão Terciária**: são revisões que produzem levantamento de revisões secundárias já publicadas sobre determinado assunto. Pode ser uma Revisão Sistemática ou Mapeamento Sistemático.

Vale destacar que a revisão sistemática possui maior nível de evidência cientifica, tanto no contexto de validade como de confiabilidade [Bork 2005]. Isso corrobora sobre a importância da construção de estudos de pesquisa bibliográfica sistematizada, como os RSL e MSL.

9.3. Mapeamento Sistemático da Literatura

A construção do levantamento bibliográfico proposto neste capítulo utilizou-se dos procedimentos de Mapeamento Sistemático da Literatura de [Kitchenham 2004b, Petersen et al. 2008]. Um MSL é definido como um conjunto de procedimentos com o objetivo de identificar, interpretar e sumarizar os trabalhos relevantes para determinada linha de pesquisa, área ou fenômeno de interesse de forma não tendenciosa e replicável [Kitchenham 2004b].

As justificativas para escolha do MSL (ou SLM) são:

1. Organizar e sintetizar todas evidências empírica sobre determinado tratamento ou tecnologia;

2. Identificar lacunas na área;

3. Apresentar uma visão geral de determinando tema de pesquisa [Kitchenham 2004b].

Segundo [Kitchenham 2004b] as etapas de construção de um MSL são: **Planejamento, Execução e Sumarização (resultados)**. No planejamento, os pesquisadores elaboram o protocolo de execução do estudo. Nesse sentido, são definidos os i) objetivos da pesquisa; ii) as questões de pesquisa; iii) as bases de dados que serão utilizadas; iv) as strings de busca a serem aplicadas nas bases de dados; v) os critérios para inclusão e exclusão dos trabalhos obtidos e vi) os formulários para extração de informações relevantes, dentre outros aspectos. A Figura 9.1 apresenta os componentes necessários para o planejamento citados anteriormente.

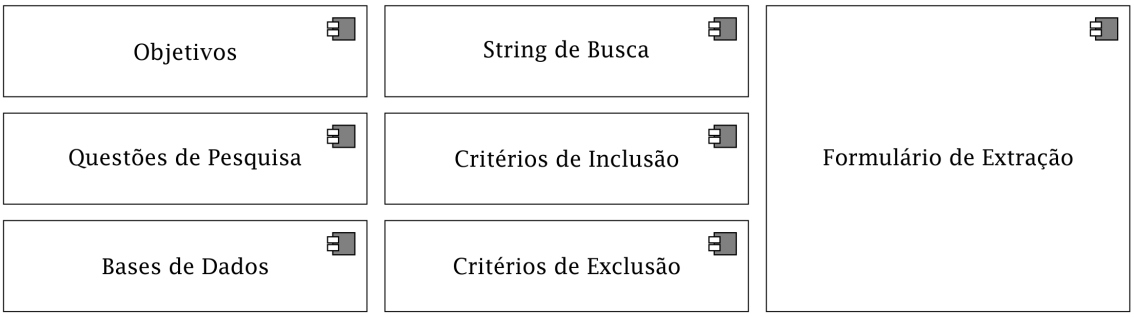


Figura 9.1. Planejamento do MSL

A execução, representada na Figura 9.2, são aplicados os processos ou itens planejados anteriormente. Inicialmente, o pesquisador aplica as *strings* de busca nas bases de dados para obter o conjunto de estudos iniciais. Em seguida, é necessário aplicar os critérios de inclusão/exclusão. Isso é feito realizando a leitura do título e do resumo dos trabalhos obtidos na busca. O resultado são os artigos selecionados. O passo seguinte é leitura completa dos trabalhos selecionados para extração de dados relevantes.

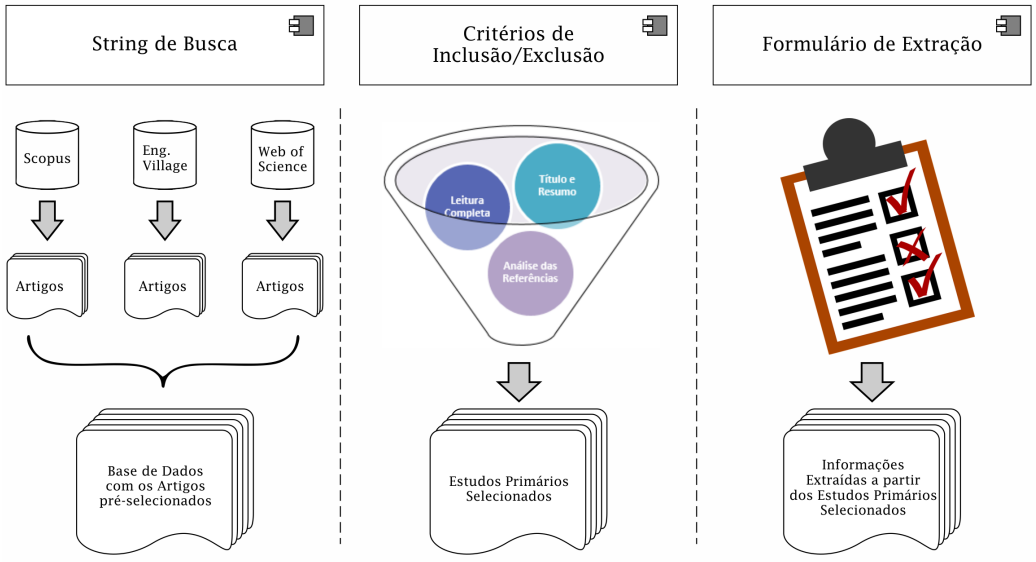


Figura 9.2. Execução do MSL

Por último, na terceira etapa, realiza-se a sumarização dos resultados. As informações são organizadas em tabelas e constrói-se os gráficos para facilitar o entendimento do tema pesquisado. Nesta etapa, ocorre a discussão sobre os resultados obtidos.

Nas próximas subseções deste capítulo serão detalhadas as etapas de um MSL. Para efeito didático, será utilizado um exemplo prático para facilitar o entendimento. A proposta de um MSL deste capítulo é sobre **mineração de dados** e **autismo**. Ou seja, que pesquisas tratam sobre práticas de mineração de dados em prol do autismo.

9.3.1. Planejamento

O Planejamento é a etapa de elaboração do protocolo que guiará o Mapeamento Sistemático. Os elementos que compõe esse plano são fundamentais para desenvolver a pesquisa de modo sistemático. Como citado anteriormente os elementos são: os objetivos, *strings* de busca, bases de dados, questões de pesquisa, critérios de inclusão e exclusão e os formulários de extração.

O primeiro passo é elaborar os **objetivos** da pesquisa. A partir destes, as **questões de pesquisa** e as **strings de buscas** são definidas. Em seguida, as **bases de dados** relevantes ao tema. No planejamento também são construídos os **Critérios de seleção** dos trabalhos, que são objetivos, de inclusão e de exclusão. Por fim, criam-se os **formulários** de extração alinhados com os objetivos da pesquisa.

Vale destacar que esse protocolo deve ser elaborado de forma que a PB fique sistemática, objetiva e replicável. Para isso, é importante realizar testes preliminares à medida em que os elementos são elaborados. Por exemplo, ao pensar nas strings de buscas, é interessante fazer testes nas bases de dados e avaliar se os trabalhos encontrados estão condizentes com o propósito da pesquisa. Além disso, caso seja necessário alterar algum elemento deste protocolo pode tornar necessário realizar novamente toda a PB ou mesmo compromete-la.

9.3.1.1. Questões de pesquisa

A metodologia de [Kitchenham 2004b], ressalta sobre a necessidade da existência de uma Questão Principal(QP), a qual está relacionada com o objetivo do mapeamento. E para alcançar os objetivos da pesquisa, faz-se uso de Questões Secundárias (QSs). Essas QS's também podem categorizar os trabalhos conforme se deseja. Para a prática apresentada neste capítulo as questões estão descritas a seguir:

- **QP** – Como a *data mining* tem sido usada para apoiar causa autista?
- **QS1** – Qual técnica de *data mining* mais usadas?
- **QS2** – Qual finalidade a *data mining*?
- **QS3** – Quais as áreas tem mais *data mining* com autismo?

9.3.1.2. Strings de Busca

As *strings* são as próprias palavras-chaves do tema da pesquisa. Deve-se elaborar essas *Strings* com base nas técnicas ou métodos da pesquisa, no conceito ideia ou área de aplicação. A Figura 9.2 demonstra a composição de strings de busca para o exemplo deste curso.

Tabela 9.2. Strings de busca

Grupo	**Palavra-chave**
Técnica ou método	*data mining*
Conceito ou idéia	*autism, autistic*

9.3.1.3. Base de dados

Deve-se escolher bases de dados ou fontes de buscas de trabalhos científicos condizentes com o tema do MSL. Geralmente, a escolha por essas plataformas é justificada pelo objetivo de coletar o máximo possível de trabalhos com abrangência internacional.

Vale destacar que essas bases de buscas são indexadoras de outras bases de dados. Para este caso exemplo, deve-se escolher bases que possuem trabalhos da área de computação (Exemplo: *Scopus*, IEEE entre outras) e também pode-se utiliza bases da área da saúde (APA, PubMed). Para simplificar a demonstração será utilizado apenas *Scopus*.

9.3.1.4. Critérios de Seleção

Os critérios de seleção visam eliminar artigos que não tem relevância ao tema ou estão fora do escopo desejado. Esses critério são: i) **Objetivos** - para restringir com base em determinados parâmetros, como ano, disponibilidade do trabalho (gratuidade), duplicidade

entre outros; ii) **Exclusão**: visa retirar artigos que não condiz com um tema específico e iii) **Inclusão**: todos os artigos selecionados no mapeamento deve estar dentro desse critério. A Tabela 9.3 apresenta os critérios do exemplo abordado neste capítulo.

Tabela 9.3. Critérios de Seleção

Critérios Objetivos	
CO1	Publicação no período de 2019 a 2020
CO2	Artigos em inglês
CO3	Artigos de acesso gratuito e na íntegra
CO4	Artigos publicados em periódicos ou eventos
CO5	Artigos não duplicados
Critérios de Exclusão	
CE1	Artigos que não propõe uso de *data mining*
CE2	Artigos que não estão relacionados com autismo
CE3	Artigos incompletos (menor que 4 páginas)
CE4	Artigos que não são primários
Critérios de Inclusão	
CI1	Artigos que apresentem indícios do uso de *data mining* com autismo

9.3.1.5. Formulário de Extração

O fomulário é, de fato, as questões da pesquisa, especificamente as questões secundárias, ilustradas anteriormente. Acrescenta-se, neste item, as opções de respostas para as questões. Vale ressaltar que é importante que as questões sejam objetivas, como apresentadas na Tabela 9.4 a seguir.

Tabela 9.4. Formulários de extração

QS1. Qual técnica de *data mining* mais usadas?	
Opções:	Estatística, Visualização, Árvore de decisão, Regras de associação, Redes neurais ou Classificação
QS2. Qual finalidade a *data mining*?	
Opções:	Planejamento, análises, tomada de decisão, outra
QS3. Quais as áreas tem mais *data mining* com autismo?	
Opções:	Saúde, educação, pesquisa, governo ou outra

A etapa de elaboração do formulário para extração dos dados requer uma análise previa de alguns artigos, um experimento preliminar. Isso evita de incluir respostas que não tenha nas opções ou excessivamente a opção outra.

9.3.2. Execução

Nesta etapa é posto em prática o que foi planejado na etapa anterior. Como já citado, aplicaram-se as strings de busca nas quatro bases de dados. O resultado da busca foi um

conjunto inicial de artigos pré-selecionados. Esse número de trabalhos pode ser alto, pois não passou por nenhum critério de seleção.

O próximo passo é a seleção desses artigos por meio dos critérios predefinidos. Em seguida, faz-se a extração dos dados desses artigos selecionados. Essa extração é realizada por meio da resolução das questões do formulário de extração. Para otimizar toda esse processo, utilizou-se um *software* para MSL: A *TheEnd* [Braga et al. 2016]. Esse ambiente permite realizar todas as etapas e processos de MSL. O diferencial desta ferramenta é que o mapeamento pode ser feito de modo colaborativo com vários pesquisadores. A Figura 9.3 a seguir ilustra os botões acesso a todas as etapas do mapeamento da ferramenta.

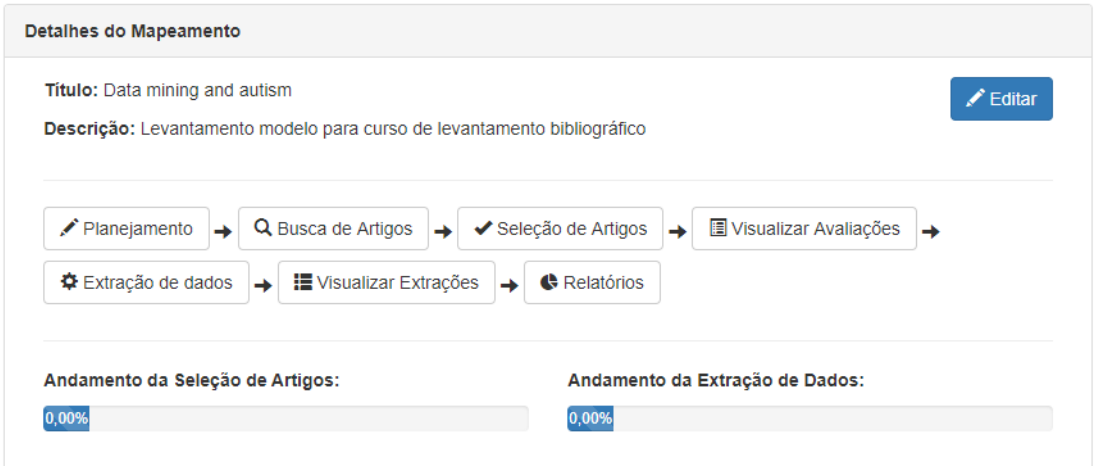


Figura 9.3. Tela inicial de acesso as etapas do MSL na *TheEnd* [Braga et al. 2016]

Por fim, após a extração, os dados produzidos podem ser sumarizados e gerar resultados, que incluem: categorizar as informações, encontrar *gaps*(ou lacunas), tendências e outras. Isso é feito por meio de tabelas e gráficos gerados na etapa de sumarização.

Ressalta-se que os procedimentos do exemplo prático deste capítulo serão apresentados na Seção 9.4.

9.3.3. Sumarização

A sumarização ou o resultado é o momento em que constrói-se os gráficos e as tabelas para ilustrar o resultado do mapeamento. Demonstra-se quantitativamente uma visão geral das informações e permite visualizar as respostas para as questões da pesquisa. Nesse âmbito, os resultados apresentam as tendências, classificação de informações e lacunas dentro do tema da pesquisa.

Dado o embasamento teórico sobre revisão bibliográfica e MSL, a próxima etapa do curso é apresentar o detalhamento da prática, dentro do objetivo do curso. Na Seção 9.4 será apresentado a prática do MSL com uso da ferramenta *TheEnd*.

9.4. Software de MSL: *TheEnd*

A ferramenta *TheEnd* é um software *Web*, colaborativo e disponível gratuitamente no endereço: https://easii.ufpi.br/theend/home/login. Ela foi desenvolvida pelo Laboratório de Engenharia de Software e Informática Industrial - EaSII - do Programa de Pós Graduação em Ciências da Computação. Por meio dessa ferramenta, é possível construir e gerenciar Mapeamentos Sistemáticos.

Diante desse recurso, pretende-se construir o levantamento bibliográfico com as informações coletadas do mapeamento. Nos próximos passos a seguir, objetiva-se apresentar as principais telas e ferramentas.

9.4.1. Acesso e cadastro de um MSL

Como ponto de partida, ao acessar o software *TheEnd*, deve-se cadastrar um usuário com um endereço de e-mail. De posse das credenciais de acesso, o pesquisador pode cadastrar um mapeamento. A Figura 9.4 apresenta a tela de criação de um novo MSL.

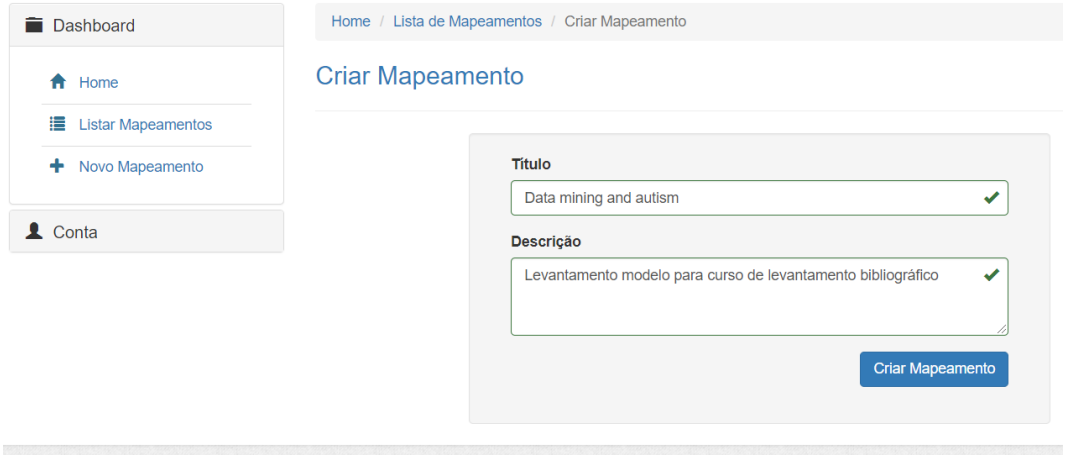


Figura 9.4. Tela que ilustra a criação de um novo do MSL na *TheEnd* [Braga et al. 2016]

Logo após, o pesquisador terá acesso a tela principal do seu MSL, apresentada na Figura 9.5. A imagem apresenta destacado 5 partes numeradas: A área 1 ilustra os botões de acesso as etapas do MSL; na parte 2 e 3 apresentam o andamento da seleção e da extração, respectivamente; na área destacada 4 mostra que é a opção de adicionar mais participantes (pesquisadores) no MSL; e na parte 5 destaca qual é o tipo de participante cadastrado.

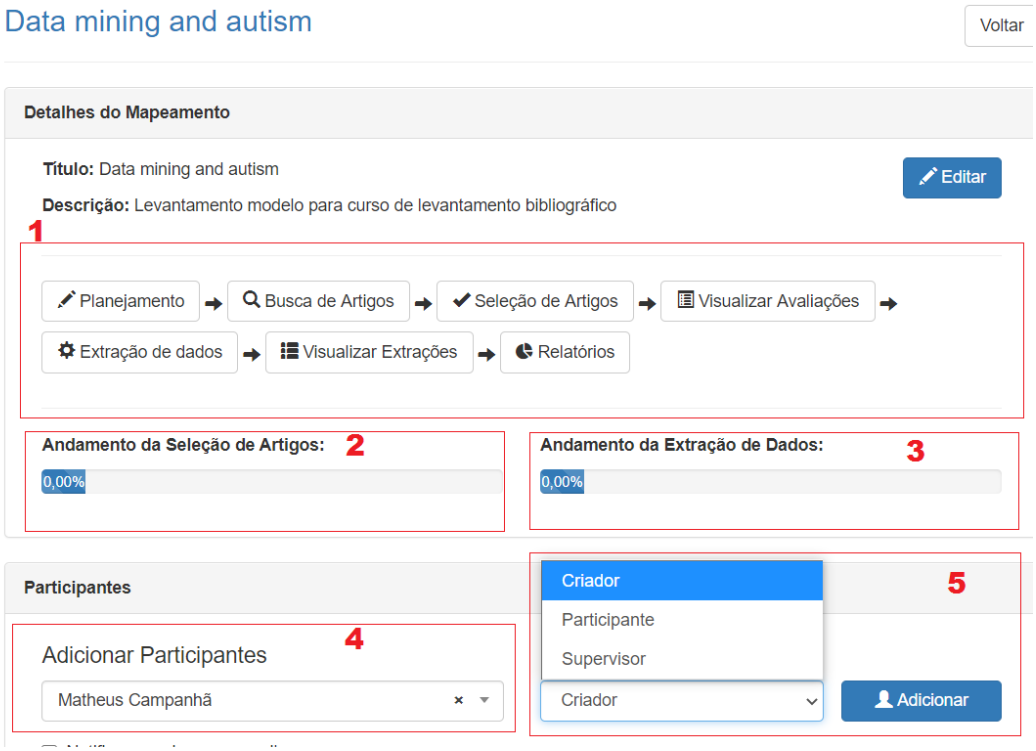


Figura 9.5. Tela principal de uma mapeamento na *TheEnd* [Braga et al. 2016]

Dando um enfoque na Figura 9.5, detalha-se cada etapa do MSL no sistema *TheEnd*. Visando facilitar o entendimento, numerou-se cada botão da etapa do MSL na Figura 9.6 a seguir. Essa figura será referência para entendimento do sistema.

Figura 9.6. Botões para acessar as etapas do MSL na *TheEnd* [Braga et al. 2016]

9.4.2. Planejamento do MSL na prática

Conforme já exposto neste capítulo, o planejamento é a primeira etapa do MSL. Isso é feito, claro, acessando o item 1, identificado na Figura 9.6. Da mesma forma que é proposto por [Kitchenham 2004b], o protocolo é elaborado nessa etapa. A Figura 9.7 apresenta por meio de guias as opções para realizar um cadastro simples dos itens planejados que compõem o planejamento: i) objetivos; ii) questões de pesquisa; ii) strings de buscas; iv) os critérios e o v) formulário. Ressalta-se que as informações desses itens já foram apresentadas na subseção 9.3.1.

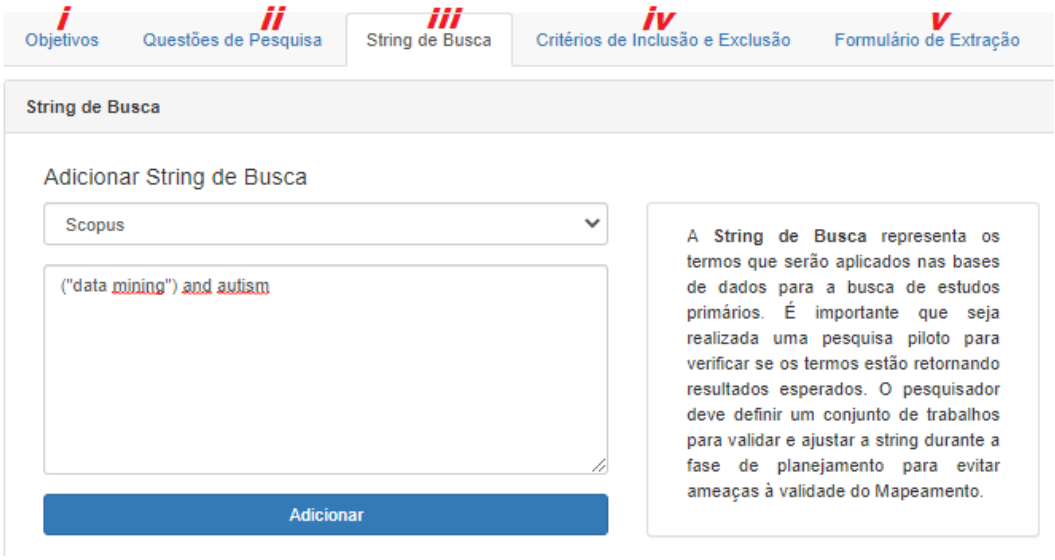


Figura 9.7. Tela de planejamento do MSL na *TheEnd* [Braga et al. 2016]

De todos os itens, vale destacar e ilustrar o cadastro do formulário de extração. A Figura 9.8 apresenta um exemplo de questão de pesquisa cadastrada. Como já explicado na Subseção 9.3.1.5, o formulário é formado a partir das questões de pesquisa com as opções de respostas para tornar o resultado mais objetivo.

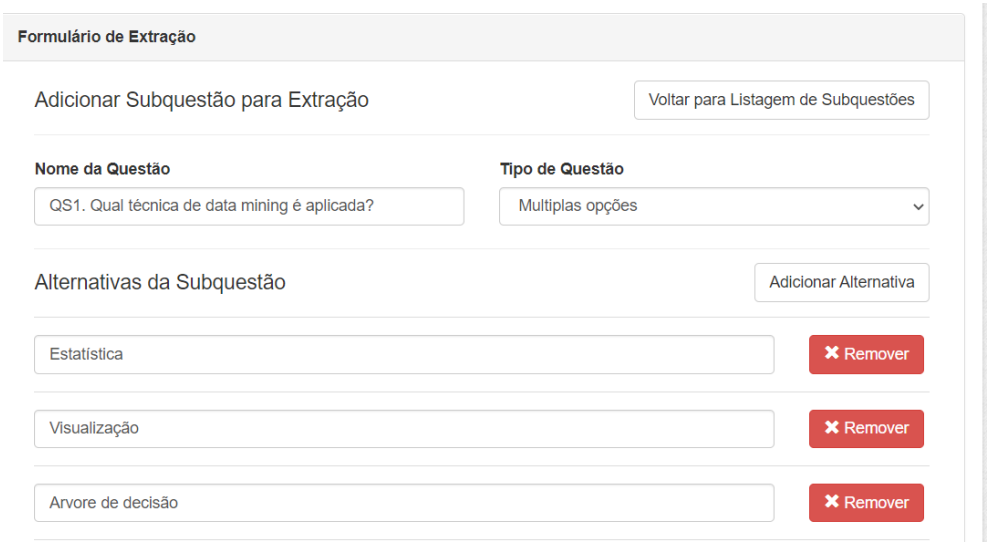


Figura 9.8. Formulário de extração de dados do MSL na *TheEnd* [Braga et al. 2016]

9.4.3. Execução do MSL na Prática

A execução do protocolo planejado inicia-se com a busca dos artigos utilizado as strings de buscas aplicando-as nas bases de dados escolhidas. Como já informado, a base de dados escolhida para essa demonstração é a *Scopus*. A Figura 9.9 apresenta uma busca preliminar na referida base de dados utilizado as strings: "*data mining*"AND "*autism*".

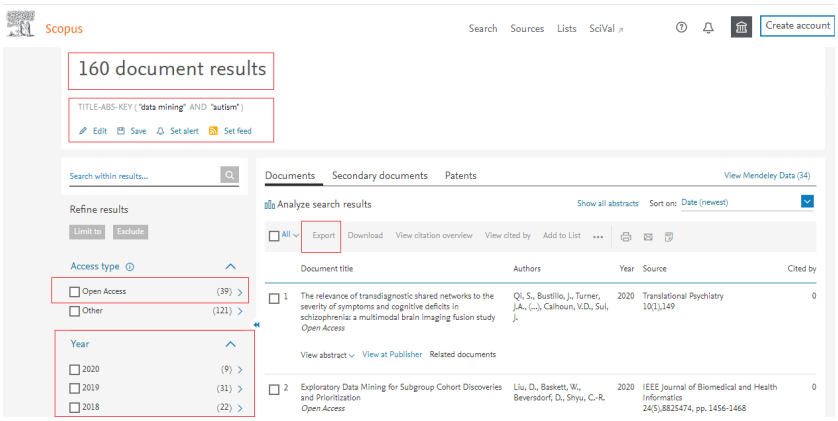


Figura 9.9. Busca de artigos no *Scopus*

A busca inicial encontrou 160 trabalhos. Os próximo passo é aplicar os critérios de seleção, já expostos na Subseção 9.3.1.4. Primeiro aplica-se os critérios objetivos (CO1, CO2, CO3 e CO4). Esses critérios são feitos ainda na plataforma da *Scopus*. Depois disso, ficaram apenas 10 trabalhos.

Os dados desses artigos devem ser exportados no formato *.bib* para inseri-los na ferramenta *TheEnd*. Para tanto, deve-se utilizar a opção de *Export* do *Scopus*, ilustrado na Figura 9.10 a seguir.

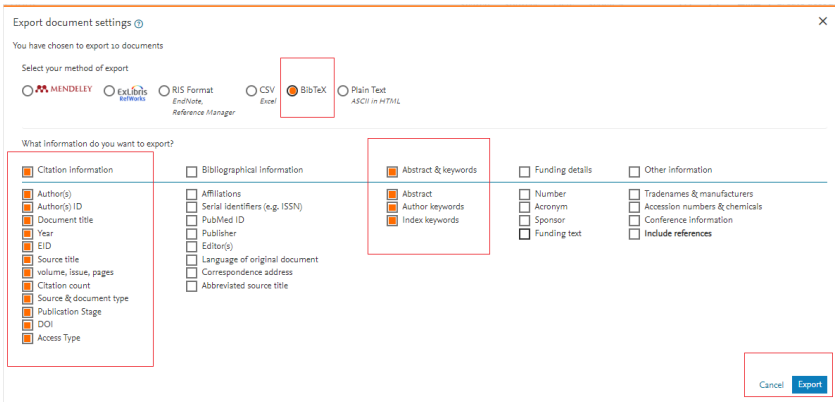


Figura 9.10. Exportação de artigos no formato *BibTex* no *Scopus*

O formato *.bib*, ou *BibTex* para o LaTeXfoi criado para facilitar a separação da bibliografia e uso no texto. Esse formato segue o mesmo conceito da distinção do conteúdo com o estilo do texto utilizada no próprio LaTeX, XHTML, CSS e outros. A Figura 9.11 apresenta um exemplo de conteúdo de um arquivo *.bib*. Esse arquivo contém informações dos artigos selecionados (título, ano, resumo, autores etc).

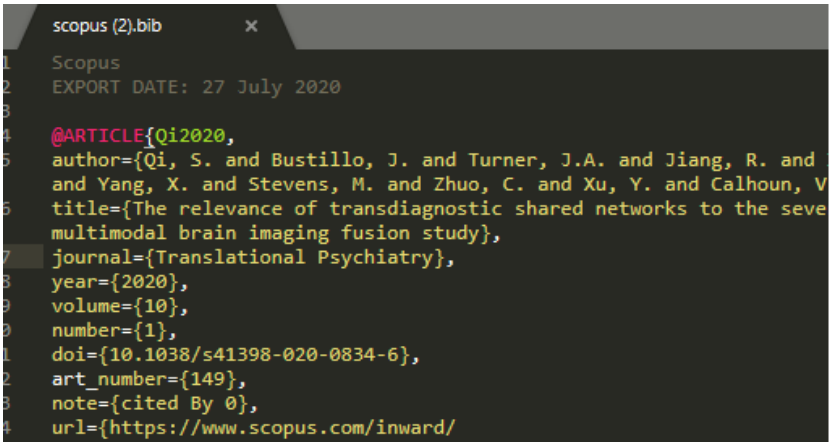


Figura 9.11. Recorte de um trecho do arquivo .bib utilizado neste MSL

Dado o arquivo *BibTex*, o passo seguinte é carregá-lo na ferramenta *TheEnd*. Para isso, deve-se acessar o botão **2**, *Busca de Artigos*, destacado na Figura 9.6. Nesta opção de busca de artigos, é possível inserir informações sobre os artigos de forma manual, ou seja, um de cada vez, ou em conjunto, por meio de arquivo BibTex. A Figura 9.12 destaca essas opções.

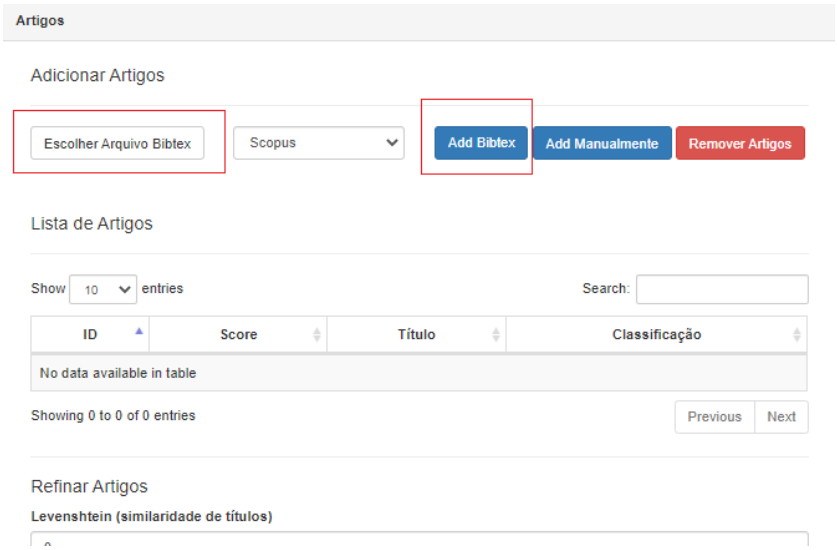


Figura 9.12. Tela de busca de artigos: i) por meio de arquivo *BibTex* ou ii) manualmente

Após carregar os dados dos artigos, na mesma tela é possível *Refinar artigos*. Essa propriedade possibilita remover artigos duplicados ou semelhantes, de mesmo autor, por exemplo.

O próximo passo da execução é selecionar os artigos. Nessa opção é possível por meio do botão *Seleção de artigos*. Ao acessar a tela de seleção, o pesquisador utilizará os critérios de exclusão e de inclusão escolher os artigos almejados, conforme os critérios. Conforme exposto, a escolha se baseia na leitura do título e resumo [Kitchenham 2004b]. A Figura 9.13 apresenta um exemplo com um dado artigo.

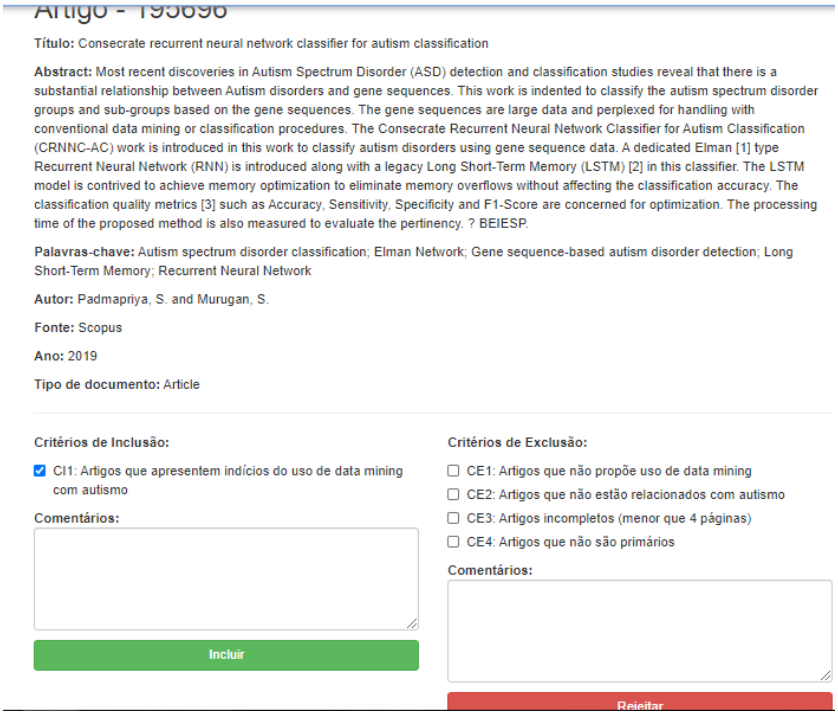


Figura 9.13. Tela ilustrando os critérios de inclusão e exclusão para um dado artigo

Ao finalizar a seleção, o pesquisa passa para o item **4** (da Figura 9.6) - Visualizar Avaliações. Nesta opção, os pesquisadores podem comparar as decisões de seleção (no botão *Comparação das Avaliações*) entre si (caso haja mais de um pesquisador). Assim, se houve discordância, podem decidir sobre qual optar em reunião. Para esse MSL, ao final da seleção, restou apenas **8 artigos** selecionados para etapa de extração.

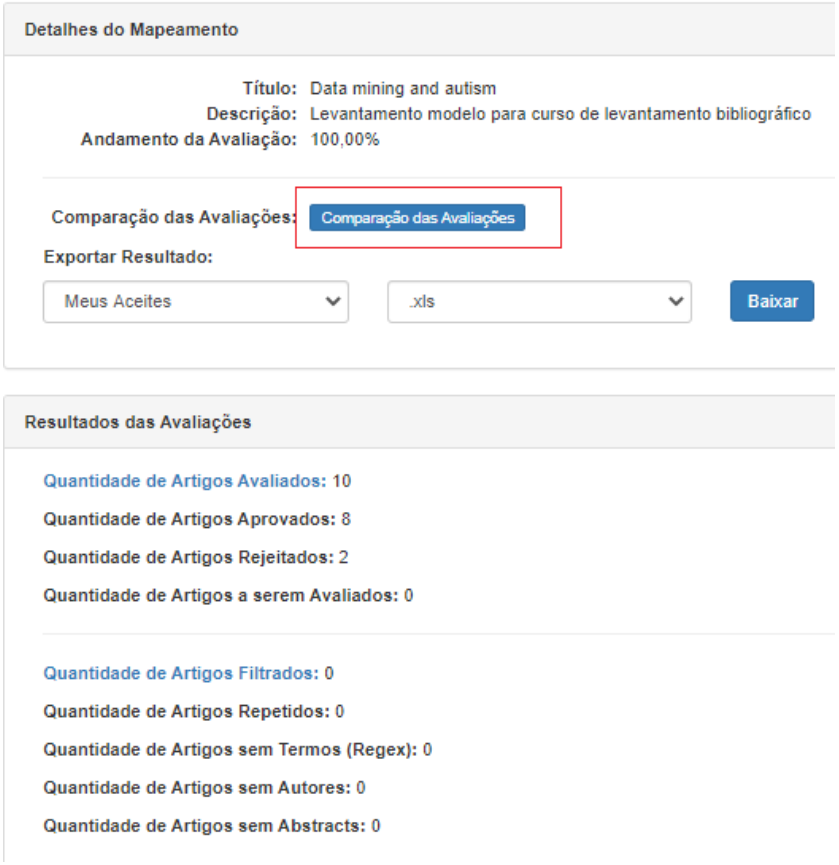


Figura 9.14. Tela com os resultados das avaliações e a opção para comparar as avaliações

O próximo passo é a extração dos dados. Para tal, deve-se responder as questões do formulário de extração. A Figura 9.15 apresenta como estão dispostas as questões. Observa-se que são questões objetivas com múltiplas escolhas.

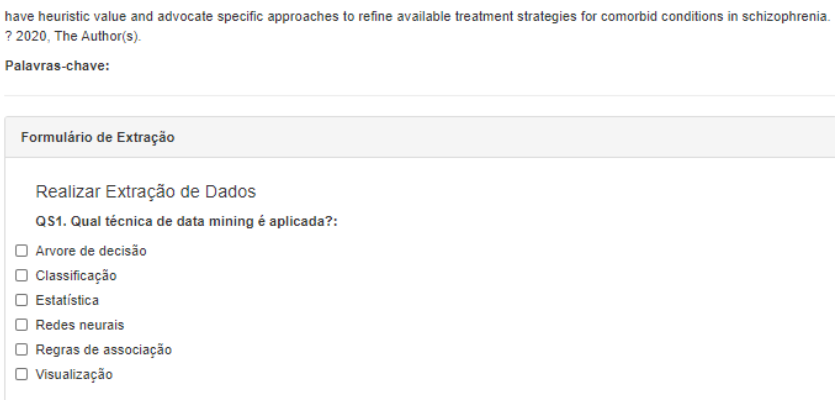


Figura 9.15. Tela ilustrando as questões do formulário de extração

Com mesmo propósito da seleção, na extração também é realizada a comparação de avaliações de extração, acessando o botão destacado com número **6**, na Figura 9.6.

9.4.4. Sumarização do MSL na prática

Por fim, tem-se a etapa de construir gráficos e tabelas para apresentar os resultados. A planilha geral com os resultados consolidados é extraído dentro da opção *Visualizar Extrações*, demonstrado na Figura 9.16, na opção *Exportar Extrações: CSV*. De posse dessa planilha, pretende-se utiliza-la no *software Google Sheets* para gerenciar mudança, trabalhar de modo colaborativo e construir gráficos adicionais que a *TheEnd* não gera.

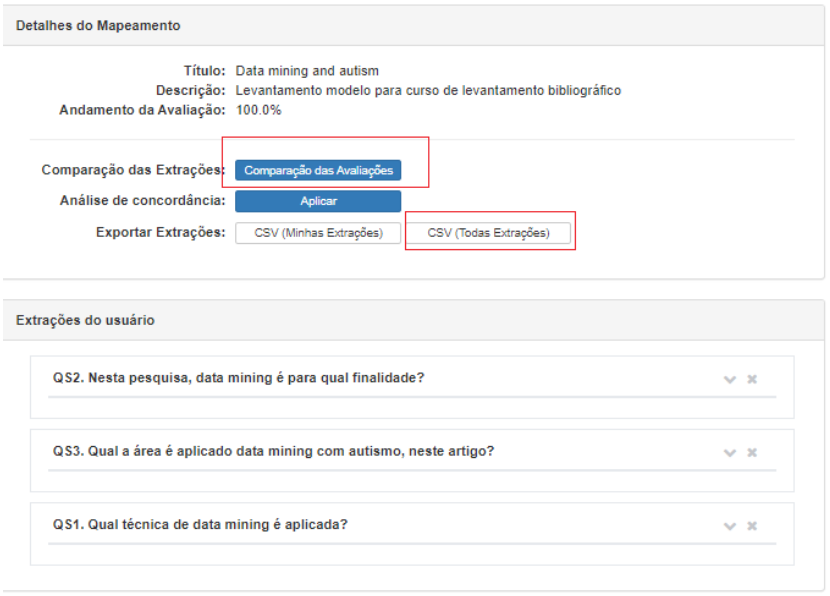


Figura 9.16. Tela apresenta a comparação das extrações, análise de concordância e exportar extrações

Já os gráficos podem são produzidos de maneira automática na própria ferramenta *TheEnd*, no botão *Relatórios*, destacado com número **7**, da Figura 9.6. Observa-se, por

meio da Figura 9.17, que é possível construir diversos gráficos na *TheEnd*, de modo automático.

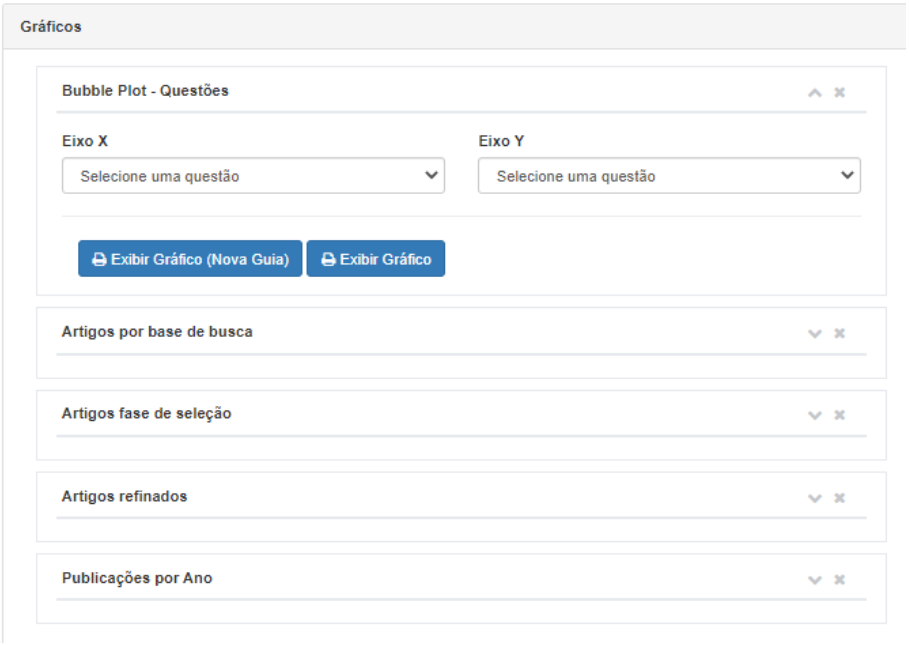


Figura 9.17. Opções de gráficos da *TheEnd*

Dentre as opções de gráfico, o *Bubble Plot - Questões*, ou gráfico em bolha. Por meio desse, é possível avaliar a associação entre respostas das questões de pesquisa. Nesse cenário, é possível mapear os resultados do MSL. A Figura 9.18 apresenta um gráfico em bolha que cruza os dados da área com a técnica de *data mining*.

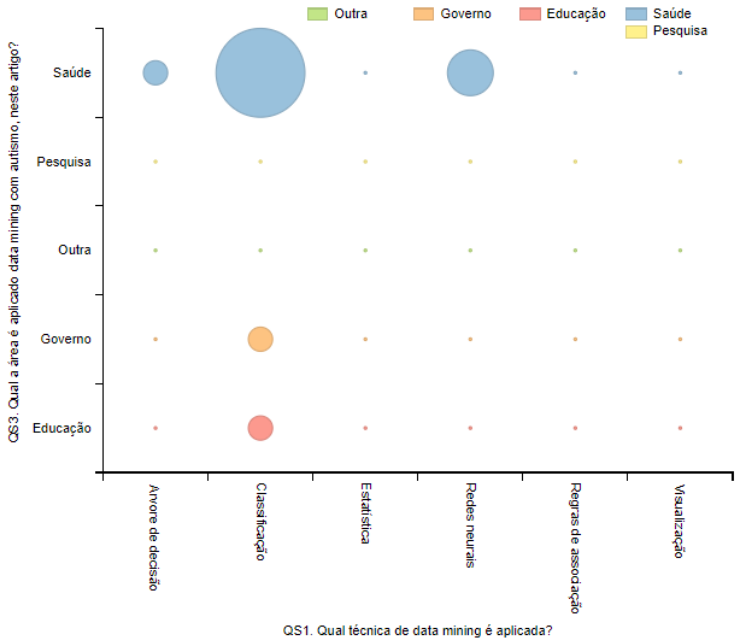


Figura 9.18. Exemplo de gráfico em bolha extraído da *TheEnd*

Com base na Figura 9.18, pode-se visualizar, por exemplo, que não nenhuma pesquisa de educação com uso de redes neurais, regras de associação e visualização. Isso sugere prováveis lacunas e oportunidades de pesquisas. Por outro lado, há considerável quantidade de pesquisas que envolvem a área da saúde e técnicas de classificação. Infere-se que é uma opção que vem sendo bem explorada.

9.5. Considerações Finais

Pretende-se, com este capítulo, que o estudante construa um levantamento bibliográfico por meio do Mapeamento Sistemático da Literatura de maneira objetiva. Para tanto, ele contará com suporte da ferramenta *TheEnd* como alternativa que permite elaborar um trabalho sistemático, colaborativo e replicável.

Os resultados do levantamento produzido por meio do método deste curso, possibilita encontrar lacunas e tendências de pesquisa em um determinado nicho de pesquisa. Com isso há um potencial de publicação desses resultados em meios científicos (congressos e revistas).

Portanto, o participante do curso estará apto a elaborar referencial teórico consistente e organizado.

Referências

[Bork 2005] Bork, A. M. T. (2005). *Enfermagem baseada em evidências*. Guanabara Koogan.

[Braga et al. 2016] Braga, R., Oliveira, P. A., Souza, M., Neto, P. S., Rabêlo, R., and Britto, R. (2016). Estudo prático sobre mapeamento sistemático da literatura utilizando

a ferramenta theend. *Anais Escola Regional de Informática do Piauí*, 1(1):77–91.

[da Silva et al. 2020] da Silva, M. D., Soares, A. C. B., and Moura, I. C. (2020). Aplicação de ferramentas computacionais para o desenvolvimento do ensino de crianças com autismo: um mapeamento sistemático da literatura. *Revista Brasileira de Informática na Educação*, 27(03):351.

[Detroz et al. 2015] Detroz, J. P., Hinz, M., and da Silva Hounsell, M. (2015). Uso de pesquisa bibliográfica em informática na educacão: um mapeamento sistemático. *Revista Brasileira de Informática na Educação*, 23(01):28.

[Keele et al. 2007] Keele, S. et al. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

[Kitchenham 2004a] Kitchenham, B. (2004a). Procedures for performing systematic reviews. Keele university. technical report tr/se-0401, Department of Computer Science, Keele University, UK, 2009-04-28T08:33:13.000+0200.

[Kitchenham 2004b] Kitchenham, B. (2004b). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

[Lakatos and Marconi 1991] Lakatos, E. M. and Marconi, M. d. A. (1991). Metodologia científica. 2ª edição. *São Paulo: Atlas*.

[Petersen et al. 2008] Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, pages 1–10.