ESTIMANDO SIMILARIDADES ENTRE OBSERVAÇÕES ATRAVÉS DE JOGOS COM PROPÓSITO

Renan da Costa Garrot

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2015

ESTIMANDO SIMILARIDADES ENTRE OBSERVAÇÕES ATRAVÉS DE JOGOS COM PROPÓSITO

Renan da Costa Garrot

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Geraldo Bonorino Xexéo, D.Sc.

_____

Prof. Felipe Maia Galvão França, Ph.D.

_____

Prof. Eduardo Soares Ogasawara, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2015

Garrot, Renan da Costa

Estimando similaridades entre observações através de jogos com propósito / Renan da Costa Garrot. – Rio de Janeiro: UFRJ/COPPE, 2015.

XI, 80 p.: il.; 29,7 cm.

Orientadores: Carlos Eduardo Pedreira

Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2015.

Referências Bibliográficas: p. 78-80.

1. Computação Humana. 2. Jogos com Propósito. 3. Medidas de Similaridade. I. Pedreira, Carlos Eduardo II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Gostaria de agradecer em primeiro lugar aos meus pais, Cláudia e Urani, que sempre me deram suporte e incentivo em tudo o que eu fiz na minha vida. Sem eles eu não seria ninguém.

Agradeço especialmente a minha namorada Isabele pelo fundamental apoio emocional dado através de constantes incentivos nos bons e maus momentos, além do auxílio como crítica e testadora oficial dos jogos.

Agradeço aos meus familiares e aos meus verdadeiros amigos, sem citar nomes para não cometer injustiças, pelo suporte direto, nas ideias, textos e burocracias, e indireto, em todos os outros aspectos da vida.

Aos meus orientadores Carlos Eduardo Pedreira e Geraldo Bonorino Xexéo, pelo acompanhamento e incentivo durante toda esta trajetória.

Aos professores Felipe Maia Galvão França e Eduardo Ogasawara por aceitarem fazer parte da banca examinadora deste trabalho, reservando tempo em suas atribuladas agendas.

Agradeço também aos membros do grupo de estudos liderado pelo professor Pedreira, que deram importantes contribuições durante as reuniões pautadas neste trabalho.

Por fim, agradeço a todas as pessoas que jogaram e ajudaram a divulgar os jogos implementados durante este trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESTIMANDO SIMILARIDADES ENTRE OBSERVAÇÕES ATRAVÉS DE JOGOS COM PROPÓSITO

Renan da Costa Garrot

Setembro/2015

Orientadores: Carlos Eduardo Pedreira

Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

A computação humana é um paradigma que utiliza seres humanos para auxiliar na resolução de problemas computacionais. Porém, diferentemente de máquinas, seres humanos necessitam de alguma forma de motivação para realizarem tais tarefas. Jogos com propósito são sistemas de computação humana que tem como fator motivacional o entretenimento. Para isto, os problemas são modelados de forma que sejam resolvíveis através de ações realizadas nestes jogos. Nestes modelos, cada jogador é considerado uma unidade de processamento de um sistema distribuído híbrido homem-máquina que realiza tarefas em troca de diversão. Nos últimos anos foram criados diversos jogos para a resolução de problemas computacionais como a tradução de textos, a geração de rótulos para imagens etc. Entretanto, na maioria destes jogos, a tarefa necessária para a resolução do problema real, como a digitalização de um texto, é delegada diretamente aos jogadores, através de uma competição cujo vencedor é aquele que digita mais rápido, por exemplo. Visto isso, neste trabalho adaptamos e formalizamos um modelo de elaboração de jogos com propósito que isola o problema que está sendo resolvido. O principal objetivo deste modelo é fazer com que os jogadores não tenham conhecimento do problema que é resolvido. Em paralelo, propomos um método para estimar similaridades entre pares de observações baseado nesta formalização, seguido de um algoritmo para encontrar agrupamentos baseado nas similaridades estimadas. Por fim, alguns jogos foram implementados para avaliar o método proposto.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ESTIMATING SIMILARITIES AMONG OBSERVATIONS USING GAMES WITH A PURPOSE

Renan da Costa Garrot

September/2015

Advisors: Carlos Eduardo Pedreira

Geraldo Bonorino Xexéo

Department: Systems and Computer Engineering

Human computation is a paradigm that relies on human beings to support resolution of computational problems. However, differently than machines, human beings need sorts of motivation to execute tasks. Games with a purpose are human computation systems that use entertainment as a motivational factor. To accomplish this, problems are modeled in a way that they can be solved through actions executed in those games. In these models, each player is considered a processing unit that composes a hybrid human-machine distributed system, in which each player executes tasks in exchange for fun. In the last years, many games to resolve computational problems have been created, such as text translation and image labeling. Nevertheless, in most of those games, the necessary task to solve the real problem, e.g., text digitalization, is directly assigned to players in a competition whose the winner is the one who types faster. Given this, in this work we adapt and formalize a model for elaboration of games with a purpose that isolates the problem that is being solved. The main goal of this model is to make players play without knowing the problem being solved. In addition, we also propose a method to estimate pairwise similarities based on this formalization, following an algorithm that finds clusters based on the estimated similarities. Finally, some games are implemented to evaluate the proposed method.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste trabalho foi adaptado, formalizado e implementado um método para a criação de videogames que levam os jogadores a resolverem tarefas computacionais através de suas jogadas, sem que estes tenham conhecimento da atividade que está sendo realizada. A proposta foi especificada para a tarefa de encontrar similaridades entre pares de observações de uma base de dados, entretanto a formalização é adaptável para outras tarefas.

Métodos clássicos para encontrar similaridades ou dissimilaridades entre observações, como os baseados na distância euclidiana, envolvem fórmulas calculadas através de máquinas. Como alternativa, o método proposto utiliza os resultados gerados por seres humanos, através das jogadas realizadas nesses videogames, para calcular estas similaridades.

Observando que as pessoas no mundo todo gastam bilhões de horas por ano realizando tarefas que necessitam de um certo tipo de inteligência, como organizar as cartas em um jogo de paciência, Von Ahn (Von Ahn, 2005) definiu o conceito de computação humana como um paradigma para utilizar processamento humano para resolver problemas computacionais que as máquinas não são capazes ou não são eficientes resolvendo sozinhas. A ideia consiste em considerar cada cérebro humano como uma unidade de processamento de um sistema distribuído híbrido, onde, dependendo do tipo, a tarefa pode ser delegada para uma máquina ou para seres humanos.

Diferentemente das máquinas, seres humanos necessitam de algum tipo de incentivo ou recompensa para realizarem tais tarefas, como pagamento, altruísmo etc. Uma dessas maneiras de incentivo é a diversão. Nela, a tarefa que deve ser realizada é embutida em uma forma de entretenimento. Como estas tarefas, em sua maioria, são delegadas através de computadores, a principal forma de utilizar a diversão como forma de incentivo é através de videogames. Com isso surgiu o conceito de jogos com propósito (Von Ahn, 2006), que consiste, neste contexto, em delegar tarefas que necessitam de processamento humano através de jogos e, assim, fazer com que as pessoas se divirtam ao realiza-las.

Partindo desta ideia, diversos jogos foram desenvolvidos para resolver problemas específicos, como traduzir textos, rotular imagens etc. Entretanto, notou-se que a maioria desses jogos eram desenvolvidos com um propósito específico e acabavam sendo muito diferentes dos jogos tradicionais que a maioria das pessoas estão acostumadas. Como exemplo, no Esp Game (Von Ahn e Dabbish, 2004), jogo criado com o propósito de rotular imagens, eram mostradas imagens a pares de jogadores e estes deveriam adivinhar qual palavra o outro jogador utilizaria para descrever aquela imagem. Ou seja, o objetivo proposto pelo jogo não se diferenciava em quase nada do que seria feito fora do contexto de um jogo, por uma pessoa paga para rotular imagens, por exemplo.

Visando aproximar os jogos com propósito dos jogos tradicionais, foi desenvolvida (Lima, 2013) uma forma de embutir o problema que está sendo resolvido em jogos tradicionais, de forma que os jogadores se sentissem mais familiarizados e, consequentemente, mais estimulados a jogar estes jogos.

As principais contribuições dessa dissertação são:

a) Adaptar e formalizar a ideia de isolar o problema que está sendo resolvido do jogo em si, através do que definimos como ‘Mundo dos Dados’ e ‘Mundo do Jogo’, de forma que os jogadores não tenham conhecimento do trabalho que é implicitamente realizado;
b) Propor um modelo, centrado nesta formalização, que estima similaridades entre pares de observações de uma base de dados através de videogames desenvolvidos com este propósito;
c) Criar um algoritmo para encontrar agrupamentos entre as observações baseado exclusivamente nas similaridades encontradas pelo modelo proposto;
d) Implementar jogos para provar a plausibilidade do modelo proposto, comparando os resultados obtidos com algoritmos clássicos de classificação supervisionada e não supervisionada.

Esta dissertação está organizada da seguinte maneira: O próximo capítulo consiste um resumo dos principais fundamentos desta dissertação. São detalhados, entre outros, os conceitos de computação humana e jogos com propósito anteriormente citados. Também é feita a revisão de dois tópicos de extrema importância: a visualização multidimensional de dados e o conceito de similaridade.

Os itens (a), (b) e (c) da lista de propostas acima são explicado no capítulo 3, que inicia com a ideia geral do que é proposto, seguido da formalização do método, das etapas necessárias para atingir o objeto de estimar similaridades e, por fim, do o algoritmo que utiliza as similaridades estimadas para encontrar agrupamentos entre os dados.

No capítulo 4 são detalhadas as implementações, iniciando com a arquitetura adotada e as tecnologias utilizadas, em seguida são mostradas as bases de dados utilizadas e os jogos implementados para testar o modelo proposto, juntamente com os resultados obtidos por cada jogo.

Por fim, o capítulo 5 mostra as conclusões e observações sobre o que foi feito em adição a algumas ideias para trabalhos futuros que dariam mais versatilidade e robustez ao método proposto.

Capítulo 2

Conceitos Básicos

Neste capítulo abordamos os principais conceitos explorados nesta dissertação. O capítulo é dividido em duas partes, a primeira detalha os conceitos de computação humana e jogos com propósito. Esta primeira parte foi dividida em:

1. Computação Humana – o que é e como surgiu a computação humana;
2. Jogos com Propósito – como realizar computação humana utilizando jogos;
3. Jogos Tradicionais Vs Jogos Com Propósito – principais diferenças e como aproximar os dois conceitos.

A segunda parte consiste em uma revisão de dois tópicos fundamentais para a dissertação, são eles:

1. Visualização de Dados – tópicos sobre representação multidimensional de dados e características pré-atentivas;

2. Medidas de Similaridades – resumo do conceito de similaridade entre observações.

2.1 Computação Humana

Em 1965, Herbert Simon, um dos fundadores da inteligência artificial, fez uma previsão dizendo que em vinte anos as máquinas seriam capazes de realizar qualquer trabalho que os humanos são capazes de realizar. Como sabemos, mesmo trinta anos após a data alvo estimada, esse futuro ainda está muito distante de se tornar realidade. Por esta razão, explorar formas alternativas de processamento é sempre um trabalho interessante e com muitas possíveis aplicações. Além disso, com as pessoas e os objetos cada vez mais interligados, a quantidade e a variedade de informações cresce incomensuravelmente a cada dia. Dessa forma, mais poder de processamento se faz necessário para absorver e organizar todo esse volume de dados.

A Computação Humana consiste na proposta de explorar o poder de processamento dos seres humanos para resolver tarefas que os computadores não conseguem ou não são eficientes resolvendo sozinhos (Von Ahn, 2005).

Na prática a ideia requer a elaboração de uma espécie de algoritmo distribuído para ser executado em um ambiente onde uma parte dos processadores são computadores e outra parte são seres humanos. Dependendo do ponto do algoritmo a próxima tarefa pode ser delegada para um computador ou para um ser humano. Visando maximizar a velocidade e a precisão das respostas, deve-se explorar o que há de melhor em cada um.

Na computação humana é notado uma inversão de papeis, no lugar dos humanos delegarem tarefas para os computadores, são os computadores que delegam tarefas para os humanos resolverem (Gomes *et al*., 2012). Porém, seres humanos não são perfeitos e nem trabalham sem um propósito, por isso, existem alguns pontos, enumerados por Quinn e Bederson (2011), que precisam ser levados em consideração na confecção de uma algoritmo que utilize poder de processamento humano.

Motivação

Diferentemente de máquinas, seres humanos necessitam de algum fator que os incentivem a fazer algum tipo de trabalho. Dessa forma, na hora de construir um algoritmo que utilize Computação Humana é necessário embutir alguma(s) das formas abaixo como motivação para que o trabalho seja realizado.

a) Pagamento – quando uma quantia de dinheiro é oferecida por cada tarefa completada;
b) Altruísmo – quando a pessoa sente que está ajudando a uma causa maior realizando a tarefa proposta;
c) Diversão – quando a realização da tarefa é uma forma de entretenimento e traz bem estar pessoal para a pessoa;
d) Reputação – quando a realização da tarefa traz reconhecimento para a pessoa;
e) Trabalho Implícito – quando a tarefa realizada faz, naturalmente, parte do trabalho realizado pela pessoa.

Controle de Qualidade

Mesmo motivados, os seres humanos podem tentar trapacear o sistema ou até mesmo resolver de forma errada a tarefa, por não ser capaz ou não entender o problema. Logo, é importante que haja um controle de qualidade para garantir que o problema seja resolvido da melhor forma possível. Existem diversas formas de controlar a qualidade de resolução do problema, algumas das mais utilizadas são:

a) Concordância entre os participantes – quando a tarefa só é dada como resolvida quando é dada para duas ou mais pessoas e todas concordam com a mesma resposta;
b) Votação – quando a tarefa é designada para um grupo de pessoas e a resposta aceita é a que foi dada pela maioria das pessoas;
c) Dificultar a trapaça – modelar a forma como a tarefa é passada de forma a ser mais fácil realizar a tarefa de forma correta do que trapacear;
d) Revisão de um especialista – designar um ou mais especialistas para analisar as respostas geradas e decidir se serão aceitas ou não;
e) Revisão externa – quando a resposta gerada por um certo grupo de pessoas é enviada para um outro grupo realizar a validação do resultado.

Agregação de Resultados

Durante o algoritmo, são delegadas pequenas tarefas para que os seres humanos resolvam e retornem uma resposta que poderá ser aceita ou não, dependendo do controle de qualidade adotado.

Ao final, todas estas pequenas tarefas visam chegar a um propósito único, sendo necessário que todos os resultados gerados por estas pequenas tarefas sejam agregados para alcançar tal propósito. Seguem alguns exemplos utilizados de métodos para agregar resultados.

1. Coleção – o objetivo deste método é criar uma base de conhecimento de fatos isolados, utilizando seres humanos para criar, editar, refutar e organizar estes fatos. Um dos objetivos principais desse método é melhorar as pesquisas em inteligência artificial criando grandes bases de dados de fatos de senso comum.
2. Processamento estatístico dos dados – utilizar um meio estatístico para agrupar todas as respostas, por exemplo a média dos palpites. Por exemplo, caso dezenas de pessoas tentem acertar a idade de uma outra pessoa fora do grupo, baseado em conceitos de sabedoria da multidão, a média dos palpites se aproximará bastante da idade real do indivíduo.
3. Aprimoramento Iterativo – trata-se de utilizar a resposta de um indivíduo como pergunta de um outro, dessa forma ir “modelando” a reposta até chegar a um resultado desejado.

Habilidades Específicas

É importante explicitar quais habilidades serão necessárias para a realização da tarefa requisitada. Por exemplo, quando a tarefa consiste na tradução de textos, é importante que as pessoas que estejam realizando tal tarefa tenham um bom conhecimentos das línguas que estão sendo traduzidas.

Entretanto, em muitos casos, a tarefa requisitada não requer nenhuma habilidade específica, podendo ser delegada a qualquer ser humano.

2.1.1 Conceitos Relacionados

Embora exista interseção, é importante salientar que a Computação Humana não é sinônimo de termos comumente encontrados na literatura, como *Crowdsourcing*, Computação Social e Sabedoria das Multidões.

De todos, o termo que gera mais confusão é o *Crowdsourcing*, trata-se de um termo criado por Jeff Howe (2006), surgiu da união da palavra *Crowd* (Multidão) com a palavra *Outsourcing* (Terceirização) e foi definida como:

"Crowdsourcing é o ato de tirar uma tarefa tradicionalmente designada a um agente (normalmente um empregado) e terceirizar para um grupo indefinido e normalmente extenso de pessoas como uma tarefa em aberto."[1] Ou seja, o *Crowdsourcing* visa terceirizar tarefas que antes eram realizadas por uma pessoa específica, como a criação de um novo logotipo para uma empresa, enquanto a Computação Humana utiliza seres humanos para realizar tarefas computacionais. A interseção entre as duas acontece em tarefas que, atualmente, podem ser realizadas tanto por humanos quanto por computadores, por exemplo, a tradução de um texto.

A Computação Social (Schuler, 1994) consiste em tecnologias como blog, wikis e comunidades online. Ela diz respeito a sistemas que facilitam e incentivam as interações sociais, não necessariamente com um propósito específico, como é o caso do *Crowdsourcing* e da Computação Humana. Pode haver interseção no caso de um sistema baseado em Computação Humana precisar da troca de ideias entre alguns participantes para a realização da tarefa demandada.

[1] Tradução do autor para: "Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call." (Howe, 2006)

A Sabedoria das Multidões consiste na defesa de uma ideia que diz que o mesmo trabalho realizado por um grupo grande de pessoas pode será mais bem realizado do que a melhor pessoa do grupo, como é explicitado a seguir:

"Sob as circunstâncias corretas, grupos são impressionantemente inteligentes, e frequentemente são mais inteligentes que a pessoa mais inteligente em seu interior" (Surowiecki, 2005)

Ou seja, a Sabedoria das Multidões requer a organização de um grupo grande de pessoas para realizar qualquer tipo de tarefa, não necessariamente computacional, enquanto a Computação Humana, além de focar em tarefas computacionais, em casos específicos, pode ser realizada por um único indivíduo.

2.1.2 Exemplos

reCAPTCHA

Todos os usuários da internet preenchem muitos CAPTCHAs (*Completely Automated Public Turing test to tell Computers and Humans Apart*) todos os dias. Os CAPTCHAs são formados por letras destorcidas que devem ser digitadas para completar um cadastro ou para enviar uma mensagem. Estas letras servem para garantir que quem está realizando a tarefa é um ser humano e não um robô, evitando que, por exemplo, sejam criadas diversas contas de e-mail para envio de *spam*. Para um ser humano é uma tarefa fácil analisar e reconhecer as letras distorcidas, porém ainda é um trabalho extremamente difícil para um máquina.

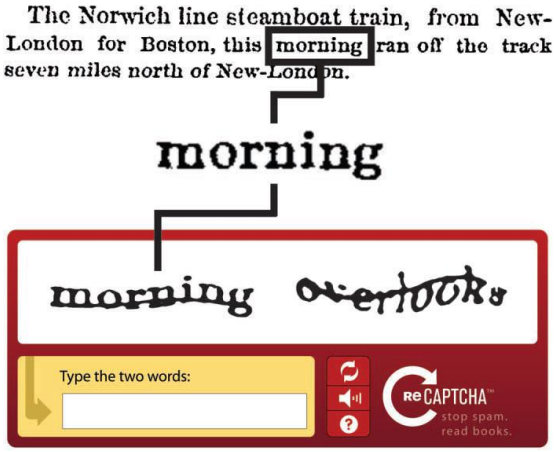


Figura 1 – reCAPTCHA – Sistema de computação humana para reconhecimento de caracteres (Von Ahn et al., 2008)

No ano de 2008 foi estimado que eram resolvidos em média 100 milhões de CAPTCHAs por dia. Percebendo que a humanidade estava desperdiçando milhares de horas por dia resolvendo CAPTCHAs, Von Ahn criou o reCAPTCHA (Von Ahn et al., 2008) (Figura 1) que, além de ter a mesma função de proteção contra *bots* do CAPTCHA, serve como digitalizador de livros.

A ideia consiste em mostrar duas palavras para o usuário, sendo que uma delas o sistema já conhece o texto contido e a outra, desconhecida, foi retirada de um texto que se deseja digitalizar. O usuário deve digitar as duas palavras, sem ter o conhecimento de qual delas é a conhecida pelo sistema.

A palavra conhecida é usada como validador (controle de qualidade), isto é, se o usuário digitar incorretamente esta palavra a resposta não será aceita e serão sorteadas outras duas palavras para completar a operação.

Caso o usuário digite corretamente a palavra conhecida pode-se considerar que se trata de um ser humano, ou seja, o papel original do CAPTCHA já está feito e, além disso, é gerada uma possível digitalização para a segunda palavra. Seguindo um outro modelo de controle de qualidade, quando a mesma transcrição for dada por um certo número de pessoas, a palavra é dada como conhecida. Ao fim, quando todas as palavras do livro forem conhecidas, a tarefa proposta foi concluída, isto é, o livro inteiramente digitalizado.

Duolingo

Um exemplo mais recente de computação humana é o Duolingo[2] (Figura 2), trata-se de uma plataforma de aprendizado de línguas estrangeiras totalmente gratuita. Nela, o usuário escolhe uma língua que deseja aprender e é submetido a diversas tarefas durante o percurso de aprendizado como tradução de pequenas frases, associar imagem ao texto, ouvir pequenos trechos de frases e transcrever, etc.

Esta plataforma é totalmente gratuita para todos os usuários e sem nenhum anúncios, mas ainda assim é rentável para os criadores. O modelo de negócio (HRTALENT, 2014) consiste em mesclar frases conhecidas para testar o conhecimento dos usuários com frases desconhecidas, tiradas de livros, artigos ou páginas da internet, para que seja realizado o serviço de tradução, que pode ser cobrado.

[2] https://www.duolingo.com/

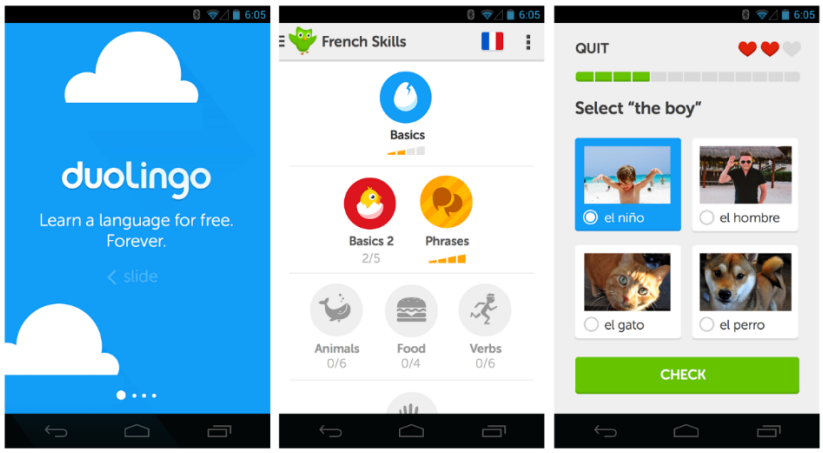


Figura 2 – Duolingo – Sistema de computação humana para tradução de textos

Novamente, como podem haver erros, os resultados das traduções só são aceitos quando efetuados por um número mínimo de usuários (concordância entre os participantes) e enviados para outros usuários (revisão externa) para que digam se a tradução está correta ou não. Dessa forma o serviço de tradução é realizado de forma mais rápida e até mais eficiente, considerando o conceito de sabedoria da multidão, do que por tradutores profissionais.

2.2 Jogos com Propósito

"Todo ano, pessoas gastam bilhões de horas jogando jogos virtuais ao redor do mundo. E se todo esse tempo e energia pudesse ser canalizada em algo útil? E se as pessoas ao jogar estes jogos pudessem ao mesmo tempo, sem nem ao menos perceberem, resolver problemas de larga escala?"[3]

Todos os dias pessoas passam milhões de horas jogando jogos eletrônicos, seja no computador, no celular ou no vídeo game. Ao jogar, cada jogador está realizando algum tipo de trabalho. Seja tentando acertar o adversário, seja tentando decifrar um enigma, etc. A partir do questionamento traduzido acima, Von Ahn decidiu elaborar alguns jogos que explorem este trabalho realizado. Isto é, enquanto a pessoa joga e se diverte também está gerando valor para a humanidade, resolvendo algum problema computacional.

[3] Tradução do auto para: "Each year, people around the world spend billions of hours playing computer games. What if all this time and energy could be channeled into useful work? What if people playing computer games could, without consciously doing so, simultaneously solve large-scale problems?" (Von Ahn, 2006)

A ideia de jogos com propósito está diretamente ligada a ideia de computação humana, na verdade, é uma forma de computação humana que utiliza como motivação, para a realização da tarefa, a diversão do jogador. Portanto, tais jogos servem para resolver problemas computacionais utilizando a capacidade de processamento humano e todos os conceitos explicados no capítulo anterior também são aplicados aqui.

O termo Jogos com Propósito também é utilizado para denominar jogos que tenham um fim específico, como jogos com propósitos educativos para criança, porém não é este o caso estudado neste trabalho.

2.2.1 Exemplos

Esp Game

A rotulação de imagens trata-se da tarefa de encontrar palavras, denominadas rótulos, para descrever uma imagem. Normalmente estes rótulos consistem em objetos contidos nas imagens ou lembranças que a imagem traz para que a vê. Para seres humanos esta é uma tarefa trivial, porém, para os computadores, a geração automática de rótulos é uma tarefa extremamente difícil.

Apesar de fácil para seres humanos, esta é uma tarefa tediosa quando realizada repetidamente sem nenhum incentivo. Para este propósito o ESP Game (Von Ahn e Dabbish, 2004) foi então elaborado com o intuito de criar o incentivo de divertir quem está realizando esta tarefa.

Trata-se de um jogo online que escolhe um par de jogadores de forma aleatória. Eles não se conhecem e nem podem se comunicar. É então escolhida uma imagem com o intuito de ser rotulada. Cada jogador deve tentar adivinhar os rótulos desta imagem. O número de palpites é ilimitado e no momento que ambos os jogadores dão o mesmo palpite, recebem uma certa pontuação e uma nova imagem é sorteada. O jogo segue e a tarefa é acertar a maior quantidade de rótulos em dois minutos e meio.

Uma vez que uma certa quantidade de duplas gerem o mesmo rótulo para uma certa imagem, este rótulo é dado como conhecido e na próxima vez que a imagem é mostrada tal rótulo é mostrado para ambos os jogadores como sendo proibido, forçando aos jogadores a gerarem um novo rótulo para a mesma imagem.

O jogo se mostrou eficiente e em poucos meses no ar conseguiu coletar mais de 10 milhões de rótulos de imagens. Em 2006 o ESP Game foi licenciado para o Google e

passou a se chamar Google Image Labeler, o Google utilizou o próprio banco de imagens, retornado nas buscas, para o jogo classificar. Em 2011 este projeto foi descontinuado pelo Google.

Figura 3 – Esp game – Jogo com propósito de rotular imagens

A Figura 3 mostra um exemplo do jogo ESP Game, inicialmente o jogador analisa a imagem (a). Em (b) são mostrados os rótulos que ele está escrevendo e tentando adivinhar, em (c) são as palavras proibidas, estas palavras já foram transformadas em rótulos conhecidos para esta imagem e, portanto, não é mais interessante que novas duplas gerem este mesmo rótulo. Em (d) é exibido o momento onde o outro jogador acertou o rótulo e, portanto, ambos fizeram pontos. Ainda existe a opção (e) de passar para uma próxima imagem caso esteja muito difícil coincidir o rótulo da imagem atual.

Ao final do jogo é mostrada, para estimular a competição, a pontuação atingida pelos jogadores, a pontuação dos melhores jogadores e os pontos para se atingir o próximo nível.

Type Attack

O Type Attack (Jovian e Amprimo, 2011) é um jogo desenvolvido com o propósito de digitalização de documentos. O processo consiste em converter documentos antigos que só existem em forma impressa em documentos digitais. Tal processo consiste em escanear cada página do documento e realizar o reconhecimento ótico de caracteres, do inglês optical character recognition (OCR).

Os textos foram extraídos do jornal The Straits Times do ano de 1983, incialmente os trechos foram divididos em *snippets* – parágrafos de no máximo cinco linhas para se adequar bem ao jogo. Em seguida os *snippets* passaram por um pré-processamento, parte automatizado e parte manual, para escolher quais trechos iriam para o jogo.

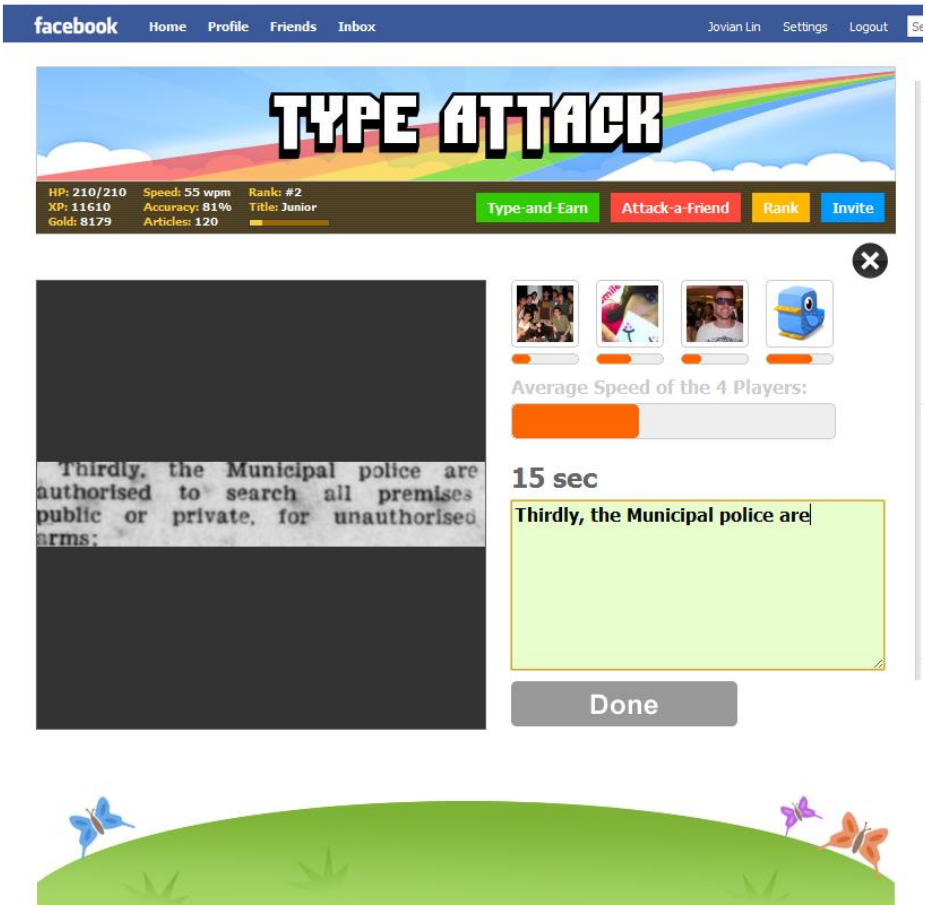


Figura 4 – Type Attack – Jogo com o propósito de transcrição de textos

O jogo, visto na Figura 4, trata-se de uma competição para descobrir quem digita melhor e mais rápido. A cada rodada o jogador tenta digitar um *snippets* em um tempo menor do que outros quatro jogadores que já tenham digitado este mesmo trecho. O controle de qualidade é baseado na concordância entre o que foi digitado pelos demais jogadores e também com o resultado obtido pelo OCR automático do trecho.

Foram realizados testes controlados com 289 jogadores e 505 trechos de textos. Os resultados mostraram que os trechos enviados para o jogo tinham, com o OCR automático, em média, 83,1% de precisão. Com a nova digitalização conseguida através do jogo estes trechos obtiveram 99% de precisão em média, resultados similares aos alcançados por um profissional.

Note que a tarefa desse jogo com propósito é a de digitalização de documentos, a mesma tarefa do reCAPTCHA, explicado na seção de computação humana. A diferença é que no lugar de palavras são digitalizados textos e, no Type Attack a motivação é, como sempre acontece em jogos com propósito, a diversão do jogador, enquanto que no

reCAPTCHA a motivação é o trabalho implícito, já que a digitação é obrigatória para concluir uma outra tarefa que a pessoa esteja fazendo, como a criação de um novo e-mail.

2.3 Jogos Tradicionais Vs Jogos Com Propósito

Esta seção sintetiza as principais diferenças e, como foi proposta por Lima (2013), a ideia de aproximação entre jogos tradicionais e jogos com propósito.

2.3.1 Principais Diferenças

Diferenças de Gênero

Assim como no cinema, os jogos também são categorizados através de gêneros, o gênero auxilia ao jogador a identificar o *gameplay* presente no jogo. Quando dizemos que um jogo é de tiro, mesmo sem nunca termos visto o jogo, imaginamos que teremos que controlar algum personagem que deverá atirar contra outros personagens ou objetos. Os principais gêneros encontrados em jogos tradicionais são:

a) Aventura – possui narrativas fortes, foco na solução de desafios e coleção de itens;
b) Ação – exige uma coordenação elevada entre olhos e mãos. O jogador deve responder em tempo real ao que acontece no jogo;
c) Estratégia – exige raciocínio e planejamento nas jogadas. O jogador deve gerenciar recursos limitados para atingir os objetivos;
d) Simulação – tenta emular condições reais de máquinas e experiências nos jogos;
e) Tiro – baseado em destruir os inimigos através de tiros;
f) Esportes – baseada em esportes e competições atléticas. Reproduzem as regras e estratégias das competições esportivas;
g) Quebra-cabeças – utilizam lógica e reconhecimento de padrões para criar desafios intelectuais. Existem por si só, diferente dos desafios que aparecem nos jogos de aventura.

Os gêneros possuem subgêneros para tornar mais específicos ou evidenciar uma determinada característica do jogo. O gênero tiro, por exemplo, possui subgênero que enfatizam a perspectiva da câmera presente no jogo:

a) Primeira pessoa – a câmera é posicionada de maneira a simular a visão do personagem do jogador. A visão, como um todo, é mais limitada, porém permite uma imersão maior;
b) Terceira pessoa – a câmera é posicionada atrás do jogador. Permite visualizar o personagem do jogador e o seu entorno; e
c) *Shoot 'em up* – a câmera permite uma visão ou de cima ou de lado do jogo. O jogador atira em uma grande quantidade de inimigos enquanto desvia dos perigos.

Foi notado que a maioria dos jogos com propósito até então não se enquadravam em nenhum desses gêneros, como pode ser visto na Tabela 1. A partir desta constatação, foi proposta uma forma de aproximar os jogos com propósito dos jogos tradicionais para que os jogadores em geral, acostumados com os gêneros tradicionais, se sentissem mais familiarizados e, dessa forma, conseguir mais jogadores e, consequentemente, mais processamento humano para a resolução da tarefa desejada.

Tabela 1 – Gêneros de jogos com propósito (Lima, 2013)

Jogo com Propósito	**Gênero de Jogo**
Esp Game	-
Peekaboom (Von Ahn et al., 2006)	-
Foldit (Cooper et al., 2010)	Quebra-cabeça
Phylo (KAWRYKOW et al.)	Quebra-cabeça
Type Attack	-

Diferenças Mecânicas

Na linguagem de jogos o termo mecânica refere-se a uma ação que pode ser tomada pelo jogador. Em um jogo de plataforma, por exemplo, o jogador pode pular, correr, atirar etc., todas essas são mecânicas possíveis deste jogo. Alguns exemplos de mecânicas clássicas de jogos podem ser vistas na Tabela 2.

Tabela 2 – Exemplos de mecânicas de jogos tradicionais (Dillon, 2010)

Mover	Parar	Virar	Pular	Correr
Mirar	Atirar	Recarregar	Esconder	Matar

Comprar	Vender	Dirigir	Acelerar	Voar
Bater	Socar	Chutar	Defender	Quebrar

Foi notado também que, como acontece com os gêneros, as mecânicas existentes em jogos com propósito também diferem muito das mecânicas existentes na maioria dos jogos com propósito, como pode ser visto na Tabela 3.

Tabela 3 – Exemplos de mecânicas de jogos com propósito (Lima, 2013)

Jogo com Propósito	**Mecânica**
Esp Game	Digitar
Foldit	Mover, Ligar, Combinar
Phylo	Mover, Combinar
Peekaboom	Digitar, Revelar
Type Attack	Digitar

Com exceção da mecânica mover, presente em ambas as tabelas, os jogos com propósito tendem a atacar diretamente o problema, por exemplo, o ESP Game que visa a rotulação de imagens consiste em um jogo onde uma imagem é mostrada e o jogador deve acertar qual o rótulo certo para esta imagem, ou seja, a tarefa realizada é a mesma que seria realizada fora do contexto de um jogo. O *gameplay* dos jogos com propósito estudados eram diretamente baseados no método utilizado para resolver o problema. Logo, as ações que esses jogos permitem são parecidas com as realizadas no problema real.

2.3.2 Mecânicas Tradicionais em Jogos com Propósito

Notadas as principais diferenças entre os jogos tradicionais e os jogos com propósito, foi definido o objetivo de aproximar os jogos tradicionais dos jogos com propósito através da proposta de inserção de mecânicas de jogos tradicionais nos jogos com propósito. As etapas para realizar a transformação proposta são:

1. Especificar o problema;
2. Entender os dados do problema;
3. Definir o mapeamento dos dados;

4. Elaborar o *gameplay* do jogo.

Especificar o problema

Um jogo com propósito consiste, como em qualquer algoritmo, em transformar um conjunto de entradas em uma saída. Dessa forma, é preciso que antes de projetar o jogo esteja bem definido qual o problema que o jogo irá resolver. Sendo assim, ao final desta etapa deve-se ter definidos os seguintes itens:

1. Definir o problema – definir qual o problema computacional o jogo com propósito se propõe resolver;
2. Especificar solução – definir qual a saída esperada para o problema, ou seja, quais resultados o jogo com propósito deve gerar;
3. Informações que caracterizam o problema – coletar e organizar a maior quantidade de dados e considerações acerca do problema, com o intuito de facilitar a modelagem do mesmo;
4. Passo-a-passo para obter a solução desejada – descrever os passos necessários para o problema ser resolvido através de computação humana, isto é, uma espécie de algoritmo descrevendo como os dados serão enviados, tratados e respondidos por seres humanos. Caso não seja possível criar este passo-a-passo, talvez o problema não possa ser resolvido através de jogos com propósito.

Entender os dados do problema

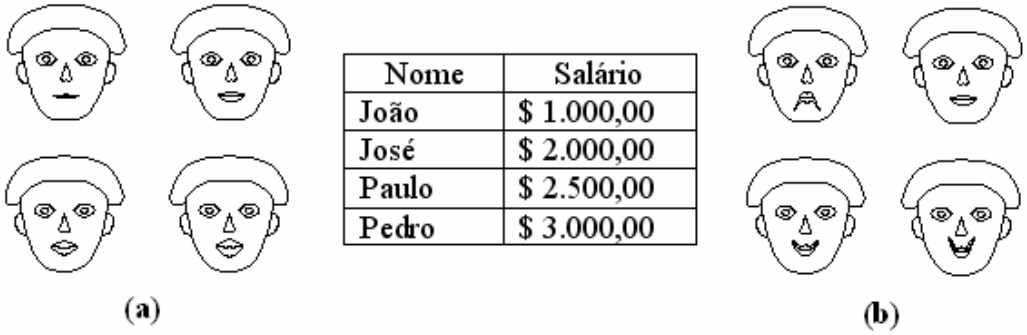
Consiste em definir a melhor maneira de utilizar os dados do problema em um jogo. Saber as informações mais relevantes do problema para que possam ser inseridas no jogo, assim como estudar a possibilidade de descartar as menos relevantes.

O estudo dos dados visa identificar se existe alguma relação entre as variáveis, alguma tendência de comportamento, algum agrupamento especial, alguma estrutura subjacente etc.

Definir o mapeamento dos dados

Definir como os dados do problema serão representados no jogo. Normalmente os dados consistem em um conjunto n-dimensional de atributos. Cada atributo deve ser mapeado em alguma característica do jogo. O passo anterior de entender os dados do problema serve para auxiliar na decisão de mapeamento dos dados.

Existem diversas formas de realizar este mapeamento, um exemplo clássico é a utilização das Faces de Chernoff (Chernoff, 1973), que, em suas palavras: “consiste em representar um ponto de um espaço n-dimensional através do desenho de um rosto no qual as características são representadas pela posição do ponto”.



Nome	Salário
João	$ 1.000,00
José	$ 2.000,00
Paulo	$ 2.500,00
Pedro	$ 3.000,00

Figura 5 – Exemplos de mapeamentos baseado nas Faces de Chernoff (Lima, 2013).

Dois exemplos de mapeamento baseados nas Faces de Chernoff são vistos na Figura 5, ambos mapeando o atributo salário da tabela vista ao centro. Na representação da esquerda (a) o tamanho da boca das faces aumenta quando o salário é maior. Já na representação da direita (b) o tamanho do sorriso dos rostos aumenta em função de seu salário. Esta segunda representação transmite mais informação para o jogador, já que intuitivamente os funcionais com salários maiores estarão mais satisfeitos com seu trabalho.

Elaborar o gameplay do jogo

Segundo Von Ahn (2005), pedir diretamente a solução computacional não é útil, torna o jogo muito algoritmo e pouco entretenimento. É preciso maquiar a pergunta de uma forma que torne o jogo agradável. Esse é o principal desafio nessa etapa de gameplay.

Nesta etapa os passos necessários para a solução do problema, descritos na etapa 1, devem ser convertidos em ações (mecânicas) dentro do jogo (Figura 6). Um passo pode ser realizado por mais de uma ação ou uma ação pode realizar mais de um passo. Podem também haver mecânicas que não necessariamente servem para resolver nenhum passo do problema, presentes apenas para tornar o jogo mais divertido. Ainda podem haver, apesar de não ser o ideal, passos que não são possíveis de converter em nenhuma ação, sendo requisitados diretamente durante o jogo ou resolvidos por uma máquina.

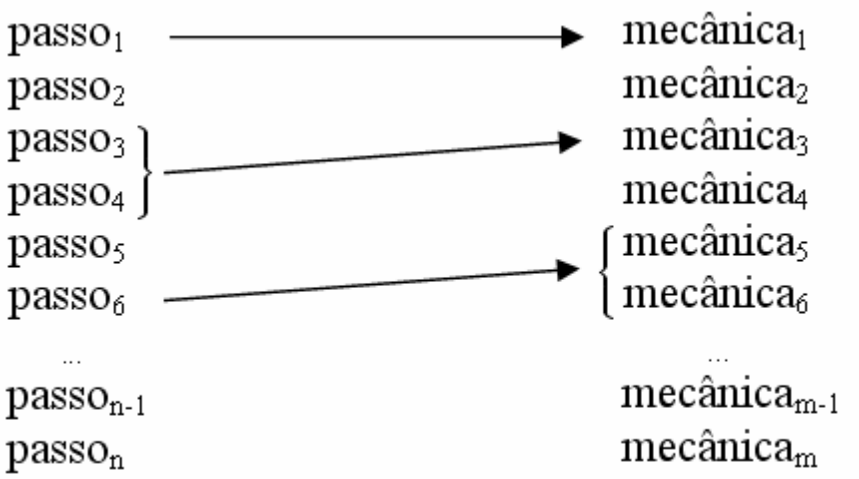


Figura 6 – Passos do problema real relacionados com as mecânicas tradicionais inseridas no jogos. (Lima, 2013)

Ao elaborar o *gameplay* o desenvolvedor do jogo precisa ter em mãos todas as etapas anteriores concluídas. Isto é, todos os dados e passos para a resolução do problema e o mapeamento que irá adotar. Entretanto, o mapeamento pode ter a sua escolha facilitada após a definição das mecânicas e do estilo do jogo. Ao fim dessa atividade é obtida a forma lúdica de solução do problema real para ser utilizada no jogo com propósito definido.

2.3.3 Exemplo

O exemplo abaixo foi criado para ilustrar as ideias de seção anterior. Uma nova lista de salários de funcionários é vista na Tabela 4, esta lista possui oito funcionários e seus respectivos salários. A tarefa computacional que deve ser realizada, através de um jogo com propósito, consiste em ordenar os funcionários em função de seus salários de forma decrescente. Portanto, a saída esperada para o problema é uma lista com os funcionários ordenados de forma que quem recebe mais aparece nas primeiras posições da lista e os funcionários que recebem menos estarão situados no fim da lista.

Tabela 4 – Exemplo de base de dados – Salário dos funcionários de uma empresa

Nome do Funcionário	**Salário**
Ana	R$ 3.000,00
Bia	R$ 1.250,00

Clara	R$ 5.000,00
Denis	R$ 600,00
Eduardo	R$ 2.000,00
Fábio	R$ 2.500,00
Gustavo	R$ 6.000,00
Zélia	R$ 1.000,00

Um possível algoritmo capaz de resolver o problema em questão é:

Passo 1 *– Identificar o funcionário com o menor salário que ainda não foi selecionado;*

Passo 2 *– Colocar o funcionário selecionado no início da fila;*

Passo 3 *– Marcar o funcionário realocado como selecionado;*

Passo 4 *– Voltar ao passo 1 até que não existam mais funcionários não selecionados.*

Figura 7 – Pseudocódigo para a resolução do problema de ordenação decrescente

Ou seja, o último funcionário selecionado no *Passo 1* será aquele que possui maior salário, este será colocado por último no início da fila, por lá permanecendo. O penúltimo selecionado, isto é, o funcionário com o segundo maior salário, ficará em segundo na fila, perdendo lugar apenas para o de maior salário no passo seguinte e assim sucessivamente até que o primeiro selecionado, o funcionário com o menor salário, fique no final da fila realizando a tarefa de ordenação decrescente.

Conseguir descrever os passos do problema significa que o problema pode ser possível de resolver através de computação humana. Assim, as etapas de *Entender o problema* e *Entender os dados do problema* foram definidas.

Na terceira etapa é utilizada a mesma representação mostrada na Figura 7(b). Nela, funcionários com salários maiores estão mais satisfeitos, ou seja, aparecem com um sorriso maior do que funcionários com salários menores.

Para finalizar, é criado o *gameplay* através do “Jogo do Garçom”, neste jogo o jogador é um garçom e o seu objetivo é não deixar nenhum de seus clientes com fome. Um cliente com fome aparenta cada vez mais zangado e pode sair sem pagar a conta. Os clientes que comem ficam felizes e dão boas gorjetas ao sair. Este restaurante possui apenas um prato que é a especialidade da casa. Todos os clientes comem o mesmo prato

e, após comerem, pagam a conta e vão embora satisfeitos. Quando o jogo inicia o restaurante já está cheio e por isso não há ordem de chegada entre os clientes.

As ações (mecânicas) que o jogador deve realizar durante o jogo são:

***Mecânica 1** – Andar até a cozinha e pegar um prato;*

***Mecânica 2** – Carregar, sem deixar cair, o prato até o salão;*

***Mecânica 3** – Escolher qual o próximo cliente que receberá o prato;*

***Mecânica 4** – Entregar o prato para um cliente;*

***Mecânica 5** – Pegar gorjetas de clientes satisfeitos;*

***Mecânica 6** – Repetir o processo enquanto houver clientes com fome.*

Figura 8 – Mecânicas para a resolução do Jogo do Garçom

Uma estratégia vencedora e intuitiva para este jogo é o garçom escolher sempre o cliente que aparentar menos satisfeito para dar o próximo prato, já que os que estão menos zangados tendem a ter mais calma para esperar pela sua vez. Sendo assim, podemos relacionar os passos para a resolução do problema de ordenamento com as mecânicas do jogo da seguinte forma:

Tabela 5 – Relação entre os passos para a resolução do problema de ordenação decrescente com as mecânicas do Jogo do Garçom

Passo do Algoritmo	**Mecânica do Jogo**
	Mecânica 1 – Andar até a cozinha e pegar um prato
	Mecânica 2 – Carregar, sem deixar cair, o prato até o salão
Passo 1 – Identificar o funcionário com o menor salário	*Mecânica 3 – Escolher qual o próximo cliente receberá o prato*
Passo 2 – Colocar o funcionário com menor salário no início da fila *Passo 3 – Marcar o funcionário relocado como selecionado*	*Mecânica 4 – Entregar o prato para um cliente*
	Mecânica 5 – Recolher comida e pegar gorjetas de clientes satisfeitos

Passo 4 – Voltar ao passo 1 até que a fila esteja ordenada	*Mecânica 6 – Repetir o processo enquanto houver clientes com fome*

A tabela acima mostra que as duas primeiras mecânicas do jogo não realizam nenhum passo da resolução do problema, sendo estas criadas apenas para o jogo se tornar mais divertido. O garçom poderia, por exemplo, ser penalizado caso um prato caia de sua bandeja.

A Mecânica 3 está relacionada com o Passo 1 do algoritmo, o jogador, jogando de forma ideal, identifica o cliente menos satisfeito, ou seja, aquele que tem mais fome no momento. Este será o cliente que está mapeando o funcionário de menor salário de acordo com o mapeamento adotado, ou seja, aquele com a face menos sorridente.

Ao entregar o prato para o cliente, Mecânica 4, o jogador realiza os Passos 2 e 3 do algoritmo. O funcionário mapeado por um cliente que recebe um prato é colocado no início da fila e não é possível entregar novos pratos para clientes que já estão comendo (selecionados).

A Mecânica 5, assim como as Mecânicas 1 e 2, não está ligada a nenhum dos passos do algoritmo. Já a mecânica 6 está diretamente ligada ao Passo 4, enquanto houver clientes com fome (não selecionados) o garçom deve continuar entregando os pratos.

Com isso, tem-se uma forma lúdica a para o problema de ordenação decrescente. Ao final do jogo, quando todos os pratos forem servidos, supondo que o jogador jogou de forma ideal, temos a solução do problema.

E este é, em resumo, o modelo proposto para aproximar os jogos com propósito em jogos tradicionais, no caso do Jogo do Garçom existem jogos tradicionais parecidos simuladores de lanchonetes.

2.4 Visualização de Dados

2.4.1 Representação Multidimensional

Existem diversas técnicas amplamente conhecidas para a representação de dados de até três dimensões, mas quando se trata de dados *n*-dimensionais, com *n* acima de três, as técnicas visuais padrões não são suficientes já que o nosso sistema visual não é capaz de

perceber acima de três dimensões. Para tal, foram desenvolvidas técnicas alternativas de representação, de forma a mapear todas as *n*-dimensões de uma base de dados em imagens de duas ou três dimensões.

Keim e Kriegel (1996) organizaram as principais formas de representação de dados *n*-dimensionais em seis classes: geométrica, orientada à pixel, hierárquica, baseada em grafos, baseada em ícones e técnicas híbridas. Chan (2006) rearranjou estes grupos em quatro categorias majoritárias:

a) Projeção geométrica – técnicas que buscam projeções significativas de dados multidimensionais em um ou mais planos cartesianos. Uma das mais conhecidas é a matriz de *scatterplox*.

b) Técnicas orientadas a pixel – são técnicas que se utilizam da coloração de pixels para representação de cada um dos atributos da base. Por exemplo, para representar 10 atributos seriam necessários 10 pixels e a cor de cada pixel representa o atributo.

c) Visualização hierárquica – técnicas que subdividem o espaço de dados em subespaços que são apresentados de forma hierárquica. Um exemplo simples são os eixos hierárquicos, exemplificado na Figura 9.

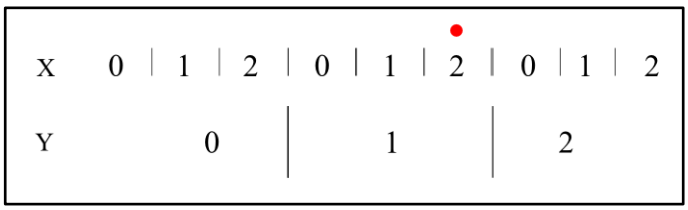


Figura 9 – Exemplo de visualização hierárquica de dados

d) Iconografia – consiste em mapear os atributos de um dado qualquer em um ícone, ou, mais especificamente, um glifo. A iconografia está presente em diversos lugares, por exemplo, um aplicativo que mostra a previsão do tempo geralmente exibe um ícone de um sol, nuvem, nuvem com chuvas etc. para passar a informação ao usuário.

Dos quatro grupos de técnicas apresentados, os três primeiros são eficientes na representação de dados multidimensionais, entretanto eles não possui características lúdicas para serem introduzidas durante um jogo, por sua vez a iconografia atende a esta

demanda, já que, baseado nesta ideia, praticamente qualquer elemento gráfico pode ser utilizado para transmitir informações.

2.4.2 Processamento Pré-atentivo

Não seria entediante se, em um jogo tiro em primeira pessoa, um inimigo aparecesse para confrontar com o jogador a cada dois minutos? Uma boa parte dos jogos é constituído de ações rápidas que acontecem em segundos e até em milissegundos para aumentar a adrenalina do jogador e, assim, tornar o jogo mais divertido. Por esta razão, em um jogo com propósito onde o jogador precisa tomar decisões rápidas sobre determinados objetos do jogo, toda a informação precisa ser transmitida para o jogador no menor intervalo de tempo possível.

O processamento pré-atentivo é a capacidade do sistema visual humano detectar determinadas características de objetos de forma rápida, paralela e com o mínimo de esforço. O cérebro realiza esta tarefa mesmo sem estar com a atenção focada para o objeto (Treisman, 1985).

É considerada pré-atentiva tarefas realizadas paralelamente em diversos elementos em um tempo igual ou menor a 200 milissegundos. Isto porque esse é o tempo que o olho humano leva para se movimentar e qualquer processamento realizado antes disso é feito sobre apenas um "quadro" (Healey, 2015).

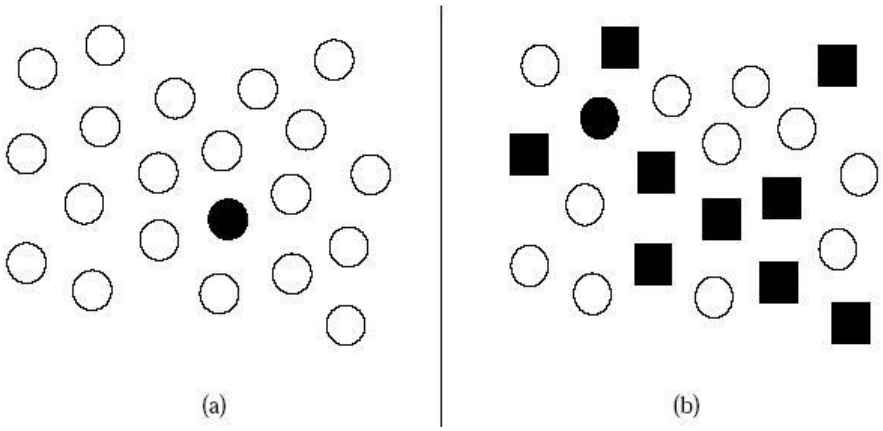


Figura 10 – Exemplos de processamento pré-atentivo. (Healey et al., 1996)

Na Figura 10 (a) o sistema visual humano separa de forma pré-atentiva a bola preta das demais bolas brancas. Já na Figura 10 (b), os objetos pretos são separados dos brancos de forma pré-atentiva, entretanto a bola preta não é destacada dos demais objetos pretos de forma pré-atentiva. Isto sugere uma hierarquia entre as características cor e

forma, apesar de ambas serem pre-atentivas, a cor tem maior destaque que a forma no exemplo mostrado.

Alguns exemplos de características pré-atentivas são (Healey, 2015): cor, intensidade, largura, tamanho, curvatura, comprimento, orientação etc.

2.5 Medidas de Similaridades

Uma medida de similaridade determina o quanto um par de observações é parecido. Normalmente, a similaridade é medida dentro do intervalo [0,1], onde, nos extremos:

Similaridade = 0, se as observações são totalmente distintas; e

Similaridade = 1, se as observações são idênticas.

A forma mais tradicional de calcular a similaridade entre duas observações é baseada em distância. Esta teoria assume que a similaridade é calculada como o oposto da distância, em algum espaço específico, entre as duas observações (Ashby e Ennis, 2007).

As dissimilaridades ou distâncias podem ser subdivididas em duas classes majoritárias: Euclidianas e Não-Euclidianas. As distâncias do tipo euclidiana são baseadas na localização dos pontos em um espaço euclidiano, já as distâncias não-euclidianas são normalmente baseadas em algumas propriedades da observação e não em sua "localização" em um espaço específico.

Uma medida de distância é considerada uma métrica quando satisfaz os seguintes axioma (Kubrusly, 2001):

a) $d(x, y) \geq 0$ (Não-negatividade);
b) $d(x, y) = 0$ se e somente se $x = y$ (Princípio da identidade dos indiscerníveis);
c) $d(x, y) = d(y, x)$ (Simetria);
d) $d(x, z) \leq d(x, y) + d(y, z)$ (Desigualdade do triângulo).

Uma medida de distância não necessariamente é uma métrica. Entretanto, a maioria dos algoritmos requer que pelo menos algumas das condições de métricas sejam respeitadas para funcionar corretamente. Portanto, é interessante, sempre que possível, que a medida de dissimilaridade em questão respeite a maior quantidade possível destes axiomas.

Uma distância muito utilizada pelos algoritmos é a Distância Euclidiana ou Distância L_2 e consiste na ideia de medir o tamanho de uma reta que vai de uma observação p para uma outra observação q ambas contidas em um espaço euclidiano.

Uma outra medida de distância de grande aplicação é a Distância de Manhattan, ou Distância L_1, que consiste da soma das diferenças de cada dimensão de forma isolada. Este nome é dado em alusão a cidade de Manhattan, cujas ruas e quarteirões formam um grid quadriculado quase perfeito. Esta distância é também conhecida como métrica do táxi, pois, supondo-se chamar um táxi de um ponto p para um ponto q deste grid, a menor distância que ele poderá percorrer será a calculada pela distância de Manhattan.

A Distância de Minkowski generaliza as distâncias descritas anteriormente. No extremo, esta distância é conhecida como Distância Chebyshev ou Distância L_∞. A definição formal de cada uma das distância é mostrada na Tabela 6.

Na prática, cada problema requer a utilização de um determinado tipo de distância para se obter bons resultados na tarefa desejada. As distâncias mostradas na Tabela 6 são comumente utilizadas para dados numéricos, contidos em um espaço euclidiano.

Tabela 6 – Exemplos de medidas de dissimilaridades baseadas em distância

Distância de Minkowski	**Distância**	**Fórmula**
$d(p,q) = \left(\sum_{i=1}^{n} (q_i - p_i)^{\lambda}\right)^{\frac{1}{\lambda}}$	Manhattan ou L_1	$d(p,q) = \sum_{i=1}^{n} \lvert q_i - p_i \rvert$
	Euclidiana ou L_2	$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$
	Chebyshev ou L_∞	$d(p,q) = \lim_{\lambda \to \infty} \left(\sum_{i=1}^{n} (q_i - p_i)^{\lambda}\right)^{\frac{1}{\lambda}}$

Para outros tipos de dados existem outros meios de calcular a distância, por exemplo, para se obter a similaridade entre duas *strings* pode-se utilizar a distância de Levenshtein. Esta é baseada no número mínimo de operações necessárias (inserção, deleção ou substituição de um carácter) para se transformar uma *string* s em uma *string* p.

2.5.1 Algoritmos Baseados em Similaridades

Diversos algoritmos de aprendizado de máquina, reconhecimento de padrões, mineração de dados, etc. utilizam medidas de similaridades ou distância como base para encontrar o resultado desejado, que pode ser classificação, agrupamento, regressão, etc.

Este trabalho focou em algoritmos que realizam algum tipo de classificação, por este motivo apenas estes foram brevemente detalhados a seguir. Existem duas classes majoritárias dos algoritmos em questão: supervisionados e não-supervisionados (Han et al., 2011).

Para os algoritmos supervisionados é necessário um conjunto de observações cujas classes são conhecidas, chamado conjunto de treinamento. Este conjunto é usado para treinar o algoritmo e encontrar um modelo. Após treinado, o modelo gerado deve ser capaz de rotular uma nova instância, não contidas neste conjunto de treinamento, em uma das classes existentes.

São exemplos de algoritmos supervisionados baseados em similaridades ou distâncias: k-NN (k vizinhos mais próximos), SVM (Máquina de vetores de suporte), etc.

Os algoritmos de agrupamento ou algoritmos de classificação não-supervisionada têm como objetivo particionar um conjunto de objetos em agrupamentos (clusters). Os agrupamentos encontrados tendem a ser formados por observações mais similares dentro do mesmo cluster e observações menos similares em clusters distintos.

São exemplos de métodos de agrupamento baseados em similaridades ou distâncias: *k*-means, clusterização hierárquica etc.

2.5.2 Similaridades Estimadas pelos Jogos

É considerada observação, neste trabalho, qualquer dado que pode ser mapeado em peças de um jogo. Estes dados podem ou não estar contidos em um espaço euclidiano. As similaridades estimadas serão baseadas na percepção dos jogadores em considerar os objetos parecidos ou não. Na prática os jogadores podem estimar distâncias semelhantes as euclidianas, comparando todos os atributos de forma igual, ou se atentar a alguma propriedade (atributo) que se destaque dos demais.

Dessa forma, não é definido o tipo de similaridade que será estimada, dependerá fortemente do tipo de dado utilizado e da representação adotada. Também não será

possível afirmar se as similaridades estimadas respeitarão todas as condições de métrica, porém, é possível garantir, devido a formulação que será definida no próximo capítulo, que os axiomas de não-negatividade (a) e simetria (c) são sempre respeitados.

Capítulo 3

Estimando Similaridades Através de Jogos

Neste capítulo é explicado o conjunto de passos propostos para estimar similaridades entre observações através de jogos.

Na primeira seção é apresentada a ideia geral da proposta, com o intuito de facilitar o entendimento das demais seções.

Na segunda seção é definida a notação, que serve para formalizar os papéis envolvidos e facilitar futuras alterações no modelo. A formalização, apesar de descrita especificamente para a tarefa de similaridades, foca em isolar o problema que está sendo resolvido do jogo em si, sendo facilmente adaptável para outros tipos de tarefas.

Na terceira seção a tarefa de estimar similaridades através de jogos, baseado na formalização, é detalhada. Esta fase é dividida em doze etapas.

3.1 Ideia Geral

Seja uma base de dados composta por observações que representam bebidas alcoólicas ou não alcoólicas. Esta base possui dois atributos que são o nome e a graduação alcoólica da bebida, como visto na Tabela 7. Desejamos, por exemplo, escolher o conjunto de bebidas que uma criança de dez anos pode ingerir, isto é, o grupo das bebidas não alcoólicas.

Tabela 7 – Exemplo de base de dados – Graduação alcoólica de determinadas bebidas

Nome da Bebida	**Graduação Alcoólica (%)**
Água	0
Café	0
Cerveja	5
Chá	0
Leite	0
Refrigerante	0

Rum	35
Suco	0
Vinho	10
Vodka	40

No jogo *Os Gatinhos da Vovó* (Figura 11) o jogador controla uma vovozinha que cria uma família de gatinhos amarelos. Nos últimos tempos a sua macieira está sendo invadida por gatos azuis selvagens que não deveriam estar ali. Muito chateada, ela decide acabar com a festa dos gatos azuis atirando dentaduras contra eles. O objetivo deste jogo é, portanto, eliminar os gatos azuis sem acertar os amarelos.

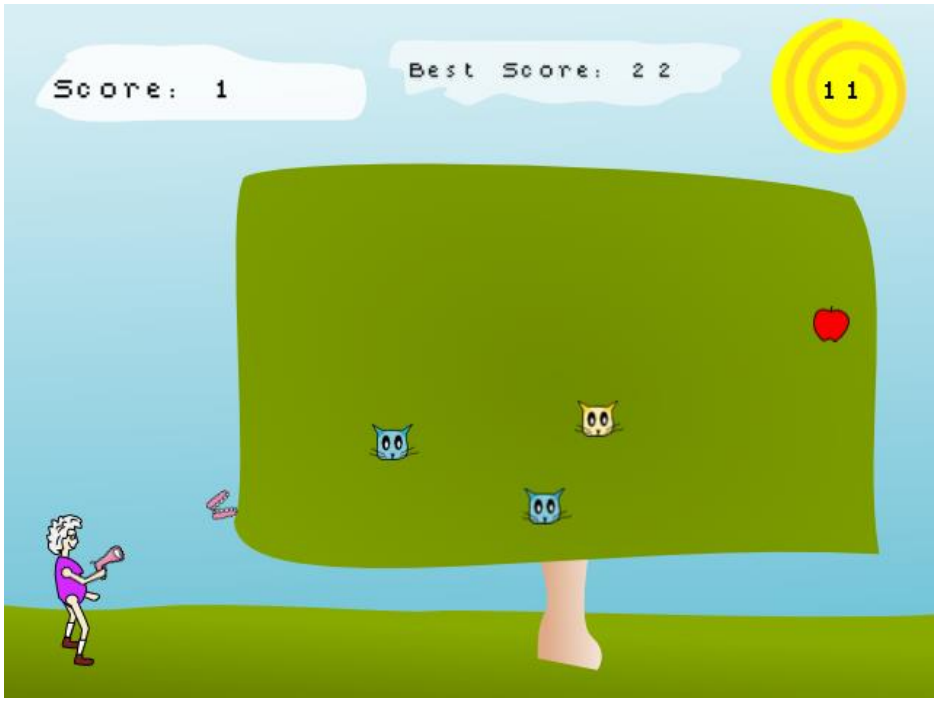


Figura 11 – Os Gatinhos da Vovó – Jogo ilustrativo da ideia geral

Define-se, neste jogo, um mapeamento onde cada gato representa uma das bebidas da Tabela 8. Neste mapeamento, caso a bebida tenha graduação alcoólica igual a zero o gato será amarelo, caso contrário o gato será representado com a cor azul. Ao final de uma partida são gerados dois grupos: o grupo dos gatos mortos e o grupo dos gatos deixados vivos. Caso o jogador tenha jogado de forma ideal, os gatos azuis serão alocados no grupo dos gatos mortos e os gatos amarelos no grupo dos gatos vivos. Assim, considerando que só existem dois tipos de gatos, pode-se afirmar que os gatos azuis são similares entre si, pois o jogador decidiu por matar todos, os gatos amarelos também são

similares entre si, pois o jogador optou por deixá-los vivos, e os gatos azuis não são similares aos gatos amarelos, já que o jogador os tratou de forma diferente. Como dito, cada bebida representa um gato, dessa forma, ao final do jogo, é obtido o agrupamento desejado: o grupo das bebidas alcoólicas (gatos mortos) e o grupo das bebidas não alcoólicas (gatos vivos). É comum, se tratando de um jogo, que um jogador cometa erros e o jogo não seja jogado de forma ideal. Por este motivo, é importante combinar o resultado de vários jogadores na hora de decidir a pertinência de um gato em um grupo. Por exemplo, se oito jogadores decidiram por matar um determinado gato e apenas dois jogadores o deixaram vivo, é inferido[4] que este gato pertence ao grupo dos gatos mortos.

O ciclo visto na Figura 12 resume a ideia geral da principal proposta deste trabalho. O ciclo inicia-se com a escolha de um subconjunto de observações da base de dados que serão enviadas para o jogo. As observações selecionadas são mapeadas em objetos do jogo, estes objetos são agrupados durante o jogo. Os agrupamentos são convertidos em votos que, como será visto, são opiniões acerca da similaridade de duas observações. Os votos aceitos, após um processo de validação, são utilizados para calcular a similaridade entre as observações, chamamos estas similaridades de ligações. Por fim, através do valor das ligações é possível encontrar agrupamentos entre os dados.

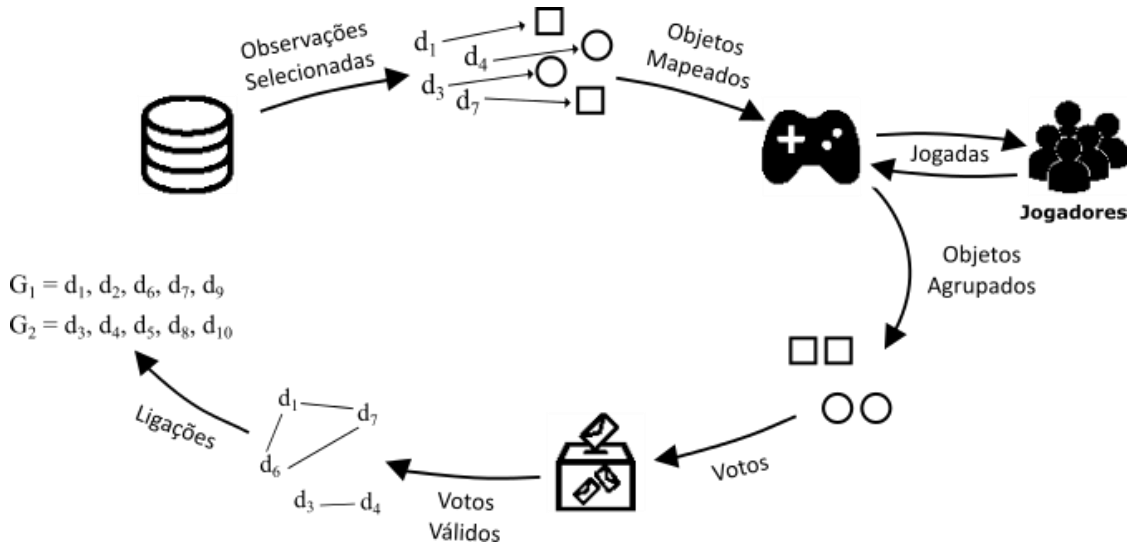


Figura 12 – Ciclo resumido da ideia geral da proposta.

[4] Assumindo um certo erro estatístico.

3.2 Notação e Definições Básicas

Nesta seção é definida uma formalização para a ideia de separação entre o problema que está sendo resolvido do jogo em si. A notação proposta possui pontos específicos para a resolução do problema de similaridade entre dados, mas, como dito, pode ser adaptada para explicitar outros problemas.

A elaboração de jogos com propósito "oculto", isto é, propósito não explicito para o jogador, deve se preocupar com duas visões, ou, como chamamos, com dois mundos. O Mundo dos Dados e o Mundo do Jogo.

No Mundo dos Dados residem as informações referentes a tarefa que está sendo resolvida, enquanto no Mundo do Jogo ficam os jogadores, os objetos, as ações e eventos do jogo que pretende cumprir este objetivo.

3.2.1 Mundo dos Dados

O Mundo dos Dados é composto pelas observações que constituem uma determinada base de dados. Este mundo deve ser invisível ao jogador. É definida, portanto, uma base de dados D com *n* observações d_i, onde $i = 1, 2, \ldots, n$.

$$\mathrm{D} = \{d_1, d_2, \ldots, d_n\}.$$

Além disso, define-se uma *ligação* para cada par de observações $(d_i, d_j)\ \forall\ i, j = 1,2, \ldots, n$.

$\varphi(d_i, d_j) \in [0,1]$ é a ligação entre os dados d_i *e* d_j.

Esta ligação relaciona duas observações da base D quanto ao grau de similaridade. Quanto mais próximo de 1 esta ligação for, mais similar serão as observações. Além disso, cada ligação possui um *tipo de ligação*, este serve para classificar uma ligação em *conhecida positiva*, *conhecida negativa*, *estimada* ou *desconhecida*.

Os valores das ligações são ideias, porém desconhecidos. Dessa forma, o principal objetivo a ser atingido no mundo dos dados é estimar estas ligações através das *estimativas de ligação* $h(d_i, d_j)\ \forall\ i, j = 1,2, \ldots, n$.

$$\varphi(d_i, d_j) \cong h(d_i, d_j)$$

Esta estimativa é calculada em função das jogadas realizadas no Mundo do Jogo. Inicialmente todas – ou quase todas, quando se tem algum conhecimento prévio – as

ligações são do tipo desconhecida. Através das estimativas elas serão classificadas em um dos outros três tipos. Como esta seção é apenas para explicar a notação, o cálculo de $h(d_i, d_j)$ e a classificação das ligações serão vistos durante a descrição do método, na próxima seção.

Por exemplo, seja uma base de dados formada pela altura de 10 indivíduos. Cada observação d_i corresponde a uma altura, em centímetros, de um indivíduo.

$$D = \{171, 162, 154, 189, 177, 188, 195, 165, 150, 190\}.$$

É desejado agrupar os indivíduo em dois grupos: indivíduos altos e indivíduos baixos. Assim, estimativas boas serão próximas a 1 para as alturas similares, $h(d_3 = 154, d_9 = 150) \rightarrow 1$, e próximas a 0 para alturas não similares, como $h(d_9 = 150, d_{10} = 190) \rightarrow 0$.

3.2.2 Mundo do Jogo

O Mundo do Jogo é o ambiente onde acontece o jogo em si e onde são elaborados todos os aspectos visíveis ao jogador. Como exemplo é usado *Jogo das Bolas e Urnas*. Neste jogo são colocadas diversas bolas e duas urnas sobre um tabuleiro. O objetivo do jogador é colocar as bolas nas urnas de acordo com alguma lógica. Uma realização possível deste jogo está representada na Figura 13.

Define-se como *Objeto* qualquer elemento do jogo [5] que pode possuir propriedades e ações. Cada objeto pertence a um conjunto que denominamos *Tipo de Objeto,* objetos do mesmo *Tipo* possuem o mesmo conjunto de propriedades e ações.

Além disso, Objetos podem ser divididos em grupos, entre eles *Peças*, *Tabuleiro*, *Jogadores*, *etc*.

São peças desta realização do jogo: bolas pretas, bolas brancas e urnas.

Embora todas as bolas neste jogo pertençam ao mesmo tipo de objeto, cada objeto possui uma identidade própria, ou seja, cada bola é tratada como um objeto diferente, o mesmo acontece com as duas urnas. Define-se O como o conjunto de todas as peças o_i, $i = 1, 2, \dots, m$ do jogo. Ou seja,

$$O = \{o_1, o_2, \dots, o_m\}.$$

[5] No sentido de Jarviken

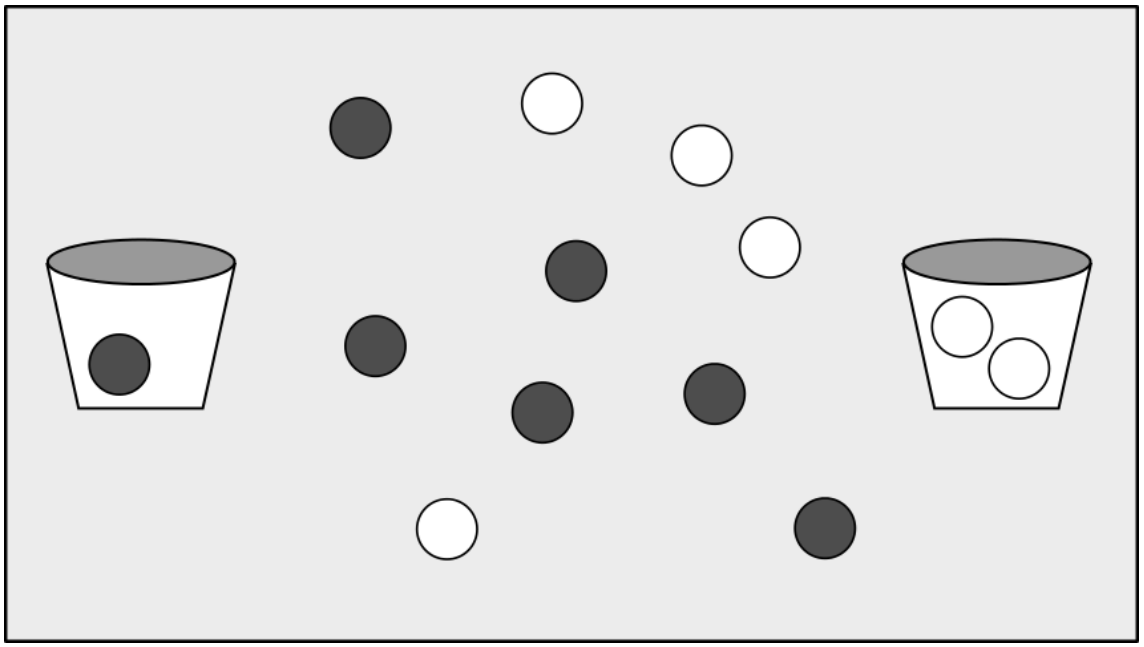


Figura 13 – Jogo das Bolas e Urnas – Exemplo utilizado para descrever os componentes do Mundo do Jogo

Define-se *Propriedade* como uma característica de um objeto. Cada propriedade assume um Valor que possivelmente é alterado durante o jogo. O conjunto de todos os pares $(propriedade, valor)$ de um objeto o_i será chamado de Q_i.

$$Q_i = \{q_{i_1}, q_{i_2}, \dots, \mathrm{q_{i_r}}\}.$$

Um objeto o_i consiste, portanto, em um conjunto Q_i de pares $(propriedade, valor)$ e um conjunto de ações. Todos os objetos possuem a propriedade ID que identifica unicamente o objeto dos demais e nunca se altera. No Jogo das Bolas e Urnas, o objeto o_2 é uma bola branca, com diâmetro unitário e que está na posição x = 50, y = 40 do tabuleiro.

$$o_2 = (Q_2, ações\ de\ o_2) =$$
$$\{ID = 2, cor \rightarrow "branca", diâmetro \rightarrow 1, Posição \rightarrow (50, 40),$$
$$localização \rightarrow tabuleiro\} + ações\ de\ o_2.$$

Cada tipo de objeto pode realizar/sofrer uma série de ações, por exemplo, no Jogo das Bolas e Urnas uma bola pode ser movimentada pelo tabuleiro.

Define-se como *Ação* um comportamento que pode ser realizado pelo objeto, alterando seu estado ou o estado de outro objeto, por isso uma ação precisa receber como parâmetro todos os objetos por ela afetados.

Define-se A_i como o conjunto de todas as n_i ações possíveis de um objeto o_i. Ou seja,

$$A_i = \{a_{i_1}, a_{i_2}, \dots, a_{i_s}\}.$$

No Jogo das Bolas e Urnas os objetos do tipo bola possuem duas ações: *mover*, que altera a posição da bola em questão sobre o tabuleiro e *alterarLocalização*, que altera a propriedade *localização* da bola quando ela é colocada sobre uma das urnas.

$a_1 = mover(x, y) -$ Translada a bola para a posição (x,y).

$a_2 = alterarLocalização(urna) -$ Define que a bola está no tabuleiro ou dentro de uma das urnas.

Reescrevendo o_2:

$$o_2 = (Q_2, A_2) =$$

$$(\{ID = 2, cor \rightarrow \text{branca}, diâmetro \rightarrow 1, Posição \rightarrow (50, 40), localização \rightarrow tabuleiro\},$$

$$\{mover(x, y), alterarLocalização(localizacao)\ \})$$

Nenhum jogo existe sem jogadores. Portanto, *Jogador* será tipo específico de objeto que não pertence ao conjunto de peças. Um jogador não deve ser confundido com a peça que o representa em uma realização do jogo, conhecida como avatar.

Define-se J como o conjunto de todos os j^i, $i = 1, 2, \dots, t$, jogadores que jogaram alguma sessão do jogo. Ou seja,

$$J = \{j^1, j^2, \dots, j^t\}.$$

Um Jogador j^i consiste, como os demais objetos, em um conjunto de propriedades e ações. No jogo das Bolas e Urnas é armazenado o nome a idade do jogador e é permitido a este realizar a ação de *movimentarBola*, esta ação deve receber como parâmetro a bola que está sendo movida e a posição de destino.

$movimentarBola(bola, x, y) -$ Realiza a ação mover(x,y) de *bola*.

No exemplo, o terceiro jogador, j^3, chama-se José e tem 19 anos:

$$j^3 = (\{\ nome \rightarrow \text{"José"}, idade\ =\ 19\ \}, \{\ movimentarBola(bola, x, y)\ \}).$$

Um *Evento* é qualquer mudança de estado em algum objeto do jogo que seja de interesse para o próprio jogo ou para um observador externo. Um evento, ao ocorrer, pode disparar uma série de ações e/ou outros eventos. Para que as ações sejam realizadas um evento precisa receber também como parâmetro todos os objetos envolvidos por ele. Seja E o conjunto de todos os eventos e_i do jogo, com $i\ =\ 1, 2, \dots, w$.

$$E = \{e_1, e_2, \dots, e_w\}.$$

No Jogo das Bolas e Urnas consideramos dois eventos possíveis, são eles: $e_1 =$ "A *bola* encontrar a *urna*", que acontece quando alguma *bola* é posicionada sobre alguma *urna* por algum *jogador*, e $e_2 =$ "Todas as bolas alocadas", que ocorre quando não existe mais bolas sobre o *tabuleiro*. Considerando que o tabuleiro possua a propriedade *quantidadeDeBolas*, que armazena o número de bolas restantes no tabuleiro, e_1 é definido por:

$$e_1(jogador, bola, urna, tabuleiro) -$$

Altera a localização da bola para urna;

Subtrai uma unidade da *quantidadeDeBolas* do tabuleiro.

Quando a *quantidadeDeBolas* do tabuleiro se iguala a zero, o evento e_2 ocorre.

$e_2(tabuleiro)$ – Sistema finaliza o jogo.

A princípio, não há limitações quanto a ações gerarem eventos ou eventos provocarem ações. Ambos os casos são possíveis.

O *Sistema* é também um objeto, não contido no conjunto O de peças, que controla todos os eventos do jogo. O Sistema também é responsável por ações que não envolvem nenhum objeto especificamente, como iniciar o jogo, finalizar o jogo, pausar o jogo etc. Além disso o sistema armazena e trata todas as ações e eventos que ocorreram no jogo.

3.2.3 Entre os Mundos

Foram definidos até agora os dois mundos isoladamente. É preciso, por fim, definir como é feita a transferência de informações entre o Mundo dos Dados e o Mundo do Jogo.

Uma *função de representação* μ_k é uma função que mapeia um dado d_i em um objeto o_j. Pode-se ter mais de uma função representação de dado d_l em r objetos diferentes.

$$\mu_k : \mathrm{D} \rightarrow O, \text{ onde } r = 1, 2, \ldots, r.$$

Considerando apenas os objetos mapeados, a função μ_k é invertível, o que possibilita descobrir qual dado foi mapeado por determinado objeto.

Voltando ao exemplo do Mundo dos Dados, a base D possui observações que representam as alturas de dez indivíduos. Mapeando a altura dos indivíduos no diâmetro das bolas, tem-se o *Jogo das Bolas e Urnas 2* (Figura 14).

A mesma proposta é feita para o jogador, que ele separe as bolas de acordo com alguma lógica. O comportamento esperado é que ele separe as bolas grandes das pequenas, isto é, colocar as bolas grande em uma das urnas a as bolas pequenas em outra.

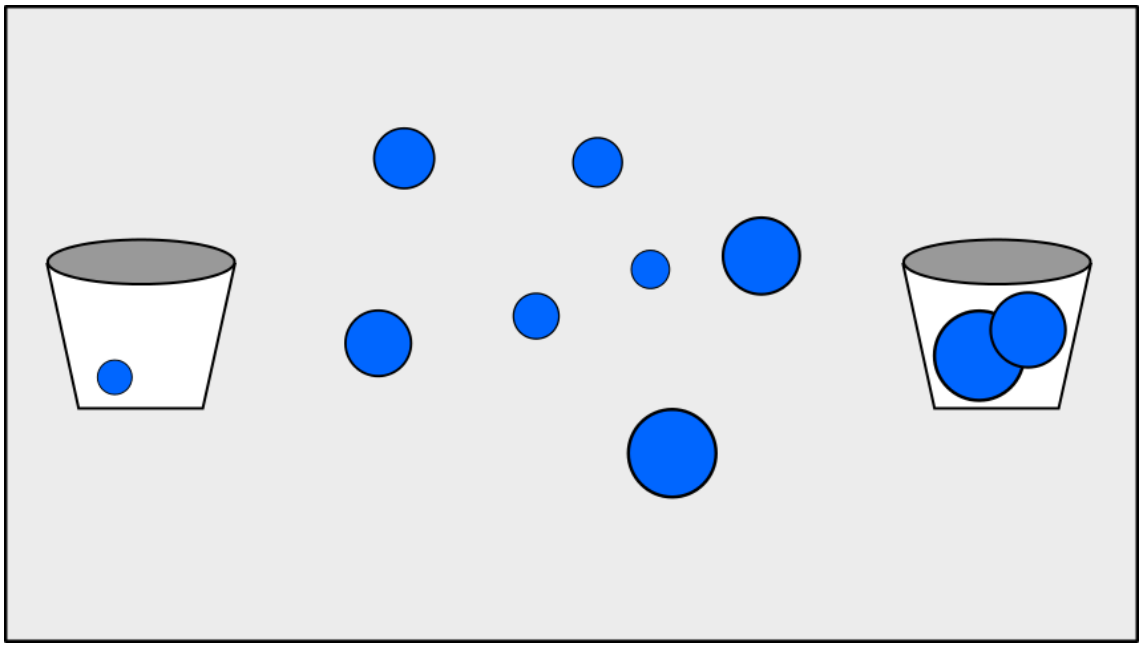


Figura 14 – Jogo das Bolas e Urnas 2 – Exemplo utilizado para descrever a função de representação

Assim, o jogador diferenciou os indivíduos altos e baixos separando as bolas grandes das pequenas, ou seja, sem tomar conhecimento da existência Mundo dos Dados.

A função de mapeamento envia informações do Mundo do Jogo para o Mundo dos Dados, entretanto para estimar as ligações $\varphi(d_i, d_j)$ no Mundo dos Dados, é necessário que alguma informação seja retornada do Mundo do Jogo para o Mundo dos Dados. É definido, para isto, o conjunto de Eventos Úteis U que é um subconjunto do conjunto de eventos E.

$$U \subseteq E.$$

Para um evento ser considerado útil, no contexto de agrupamento, ele deverá enviar para o Mundo dos Dados uma ou mais opiniões do jogador acerca da similaridade de duas ou mais observações. Estas opiniões são chamadas de votos.

Um voto $v_{i,w}^{j}$ é, portanto, uma opinião positiva (=1) ou negativa (=-1) de um jogador acerca da similaridade de um par de observações. Sendo j o jogador que provocou o evento, e i e w as observações d_i e d_w.

Ao final do processamento de um evento útil, o Sistema computa os votos nele contido e os envia ao Mundo dos Dados para atualizar as estimativas das ligações.

Um evento que pode ser considerado útil no jogo das bolas e urnas é o evento e_2 que significa que todas as bolas estão alocadas. Após tratá-lo e finalizar o jogo o Sistema calcula os votos gerado e os envia para o Mundo dos Dados.

Chama-se *sessão* de um jogo, neste modelo, um ciclo completo até a geração de um evento útil. Uma sessão pode iniciar ao iniciar o jogo, ao iniciar uma fase, ao iniciar um determinado objetivo parcial do jogo, etc. De forma análoga, a sessão é finalizada ao final do jogo, da fase, da missão etc.

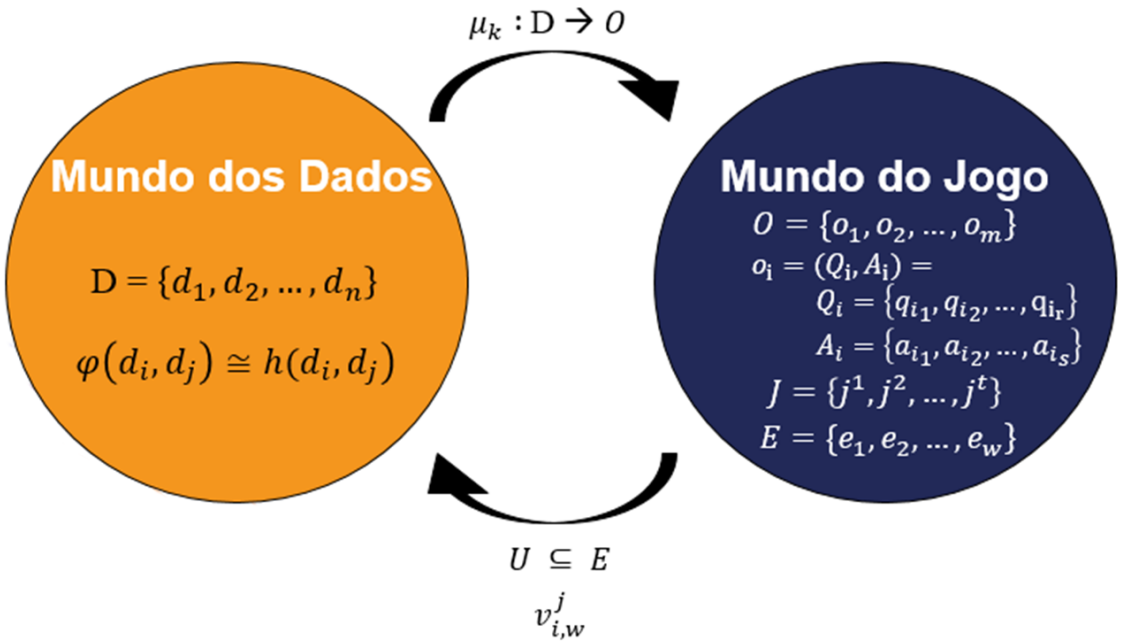


Figura 15 – Divisão entre Mundo dos Dados e Mundo do Jogo

A Figura 15 mostra um gabarito da notação proposta. O Mundo dos Dados é formato pelas observações e as ligações entre elas. O Mundo do Jogo é formado principalmente pelos Objetos, que são subdivididos em Peças, Jogadores etc. e pelo Eventos. A comunicação entre os mundos é feita através do mapeamento das Observações em Objetos e dos Eventos Úteis, que são convertidos em Votos.

3.3 Descrição do Método

Nesta seção é descrito o método proposto para estimar similaridades e agrupar as observações baseado nas estimativas encontradas. A Figura 16 detalha o ciclo visto na Figura 12 e resume as principais etapas de construção do método. A numeração indica a ordem que, preferencialmente, as etapas devem ser elaboradas.

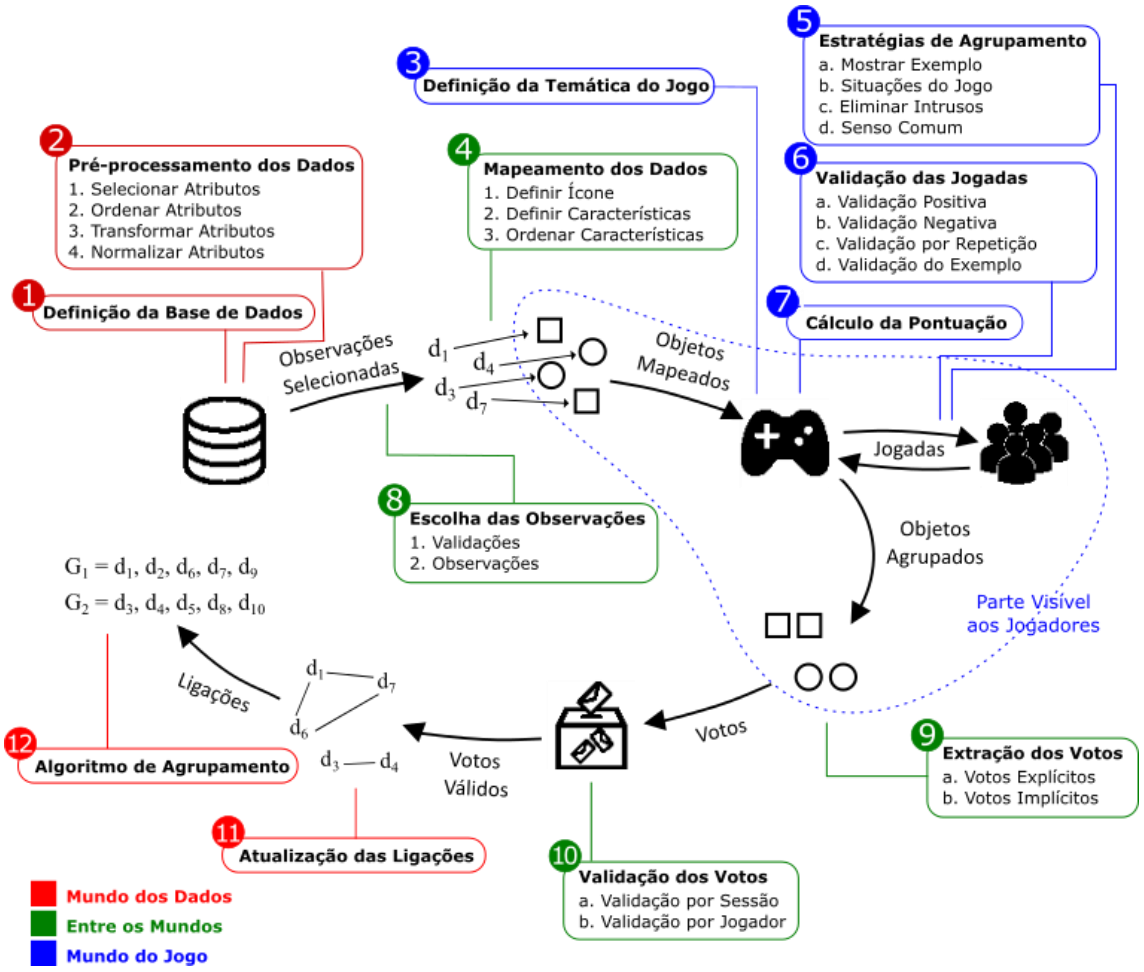


Figura 16 – Principais etapas do método proposto.

3.3.1 Definição da Base de Dados

Inicialmente coletam-se todas as informações sobre os dados que serão agrupados, ou seja, é definida a base D que será submetida ao processo de agrupamento. Além disso, em alguns casos, é necessário um pequeno conhecimento prévio sobre a base, isto é, uma parcela das ligações $\varphi(d_i, d_j)$ de D precisa ser conhecida antecipadamente.

3.3.2 Pré-processamento dos Dados

Em seguida é necessário realizar algumas transformações nos dados da base para possibilitar o mapeamento. As principais transformações que devem ser feitas são:

1. Selecionar os atributos;
2. Ordenar os atributos;
3. Transformar os atributos;

4. Normalizar os atributos.

Selecionar os atributos

Quanto mais atributos na base, mais difícil é a representação multidimensional dos dados, ou seja, mais detalhes podem passar sem que os jogadores percebam durante o jogo. Por isso, um passo importante do pré-processamento consiste em eliminar os atributos irrelevantes.

Esta seleção pode ser feita manualmente, eliminando atributos que intuitivamente não auxiliam na tarefa proposta, como nome, cpf etc., ou pode ser utilizado algum algoritmo, quando se tem informação suficiente, para identificar os atributos mais relevantes.

Ordenar os atributos

Após a seleção é importante que os atributos sejam dispostos em ordem decrescente de relevância, para que o jogo, como será visto, mapeie os atributos mais importantes nas características mais "chamativas" do objeto.

Transformar os atributos

A técnica de representação apresentada na próxima seção se baseia em parâmetros numéricos, como o diâmetro, escala de cor etc. É importante, portanto, a transformação dos atributos simbólicos em numéricos.

Para atributos simbólicos de dois valores pode-se utilizar uma simples conversão binária, exemplo: Feminino = 0, Masculino = 1. Para atributos com mais de dois valores podem ser adotadas técnicas como codificação canônica, no caso de atributos nominais, ou código termômetro quando há relação de ordem entre os atributos.

É importante tomar cuidado ao realizar estas transformações, pois algumas delas aumentam consideravelmente o número de atributos. Além disso, elas são extremas, isto é, um atributo binário que for mapeado, por exemplo, em escalas de cinza, resultará em uma grande diferença de coloração, podendo chamar mais a atenção do jogador do que um atributo contínuo.

Normalizar os atributos

Após todos os atributos serem convertidos para números, eles devem ser normalizados entre 0 e 1 para manter as proporções dos objetos na hora de representa-los.

Apenas com o valor absoluto é impossível para o jogo, que não conhece todas as observações, decidir, por exemplo, se o valor 75 representa uma bola grande ou pequena.

3.3.3 Definição da Temática dos Jogos

Neste passo deve ser feito um rascunho da ideia geral do jogo, isto é, qual o gênero, qual a história contada pelo jogo, qual o tema, etc. É importante que esta etapa seja feita antes da definição da representação dos dados, pois esta deve se basear neste rascunho inicial.

Trata-se de um rascunho, já que as etapas seguintes geralmente alteram significantemente esta ideia inicial para que o jogo cumpra a tarefa desejada.

3.3.4 Mapeamento dos Dados

Jogos eletrônicos, em sua maioria, são representados através de alegorias que combinam imagens e sons. Normalmente, toda a informação necessária para que o jogo possa ser jogado é transmitida através de imagens bi ou tridimensionais, restando para a parte auditiva apenas efeitos de ambientação e imersão do jogador. Por este motivo, neste trabalho, é proposto que toda a informação que o jogador deve processar seja transmitida através de imagens.

É pretendido que o jogador gere, através de suas jogadas, conclusões acerca da similaridade das observações presentes no mundo dos dados. Para que isso seja possível deve-se escolher uma representação para mapear as observações d_i em objetos o_i do jogo. A seguir são descritos os passos propostos para definir e otimizar esta representação.

1. Definir o ícone que será utilizado e quais características parametrizáveis este ícone possui;
2. Definir um número limite de características que será mostrada por este ícone, de modo que o jogador seja capaz de analisar todas elas durante o jogo;
3. Ordenar as características deste ícone para mapear os atributos mais relevantes nas características que mais se destacam;
4. Verificar se é possível explorar a transferência de informação com algum dos atributos;
5. Testar a representação.

Como visto na seção 2.4, a iconografia é a técnica mais apropriada para a representação de dados multidimensionais em jogos. Sendo assim, o primeiro passo é a definição de um ícone que será utilizado como representação gráfica de um objeto do jogo. Estes ícones são conhecidos como glifos.

Um glifo, no contexto de visualização de dados, é uma representação de um dado através de um ícone parametrizável, onde os valores dos parâmetros deste ícone são determinados em função dos atributos de um certo dado.

Em geral é fácil construir um glifo com muitas características parametrizáveis, entretanto o ser humano não é capaz de notar muitas características de uma só vez em uma imagem. Como agravante, no contexto de um jogo, as tomadas de decisões normalmente acontecem em um curto espaço de tempo. Isto gera um limite superior, variante de acordo com o problema, ao número de dimensões que podem ser representadas.

O próximo passo é definir a ordem de prioridade de cada característica. Muitas vezes os glifos possuem características parametrizáveis que se destacam mais do que outras, dessa forma, deve-se definir uma ordem para que o atributo mais relevante, isto é, o atributo que mais contribui para a realização da tarefa, seja mapeado na característica mais perceptível do ícone. É interessante se esta característica for destacada de forma pré-atentiva, como no exemplo da Figura 10 (Página 24).

Um outro ponto que pode ser explorado na definição da representação é a transferência de significado de um atributo para uma característica do objeto. No exemplo da Figura 5 (Página 19), a representação do salário nas Faces de Chernoff fez mais efeito no segundo mapeamento, quando o salário do funcionário foi mapeado no sorriso da face, do que no primeiro mapeamento, proporcional ao tamanho da boca. Isso acontece pois desenvolvemos, durante as nossas vidas, um senso comum que nos diz que estar feliz é "bom" e estar triste é "ruim". Já quanto ao tamanho da boca, não temos essa intuição.

Após definido, o mapeamento adotado deve ser testado para verificar se os jogadores conseguem perceber todos os atributos de forma clara. Um erro comum a ser evitado é, durante o jogo, o jogador comparar um atributo x de um objeto o_i com um outro atributo y, $x \neq y$, de um segundo objeto o_j. Nesses casos o mapeamento deve ser repensado para evitar tal confusão.

Por exemplo, a tarefa de identificar o par de objetos mais semelhantes na Figura 17(a) terá como resposta, dada por muitas pessoas, os objetos 2 e 3, pois nota-se que estes objetos são idênticos desconsiderando a rotação. Porém, caso o mapeamento adotado neste glifo seja como o mostrado na Figura 17(b), isto é, a altura de cada retângulo será proporcional a um dos atributos, os dados representados pelos objetos o_2 e o_3 serão, na realidade, os mais distintos possíveis neste conjunto.

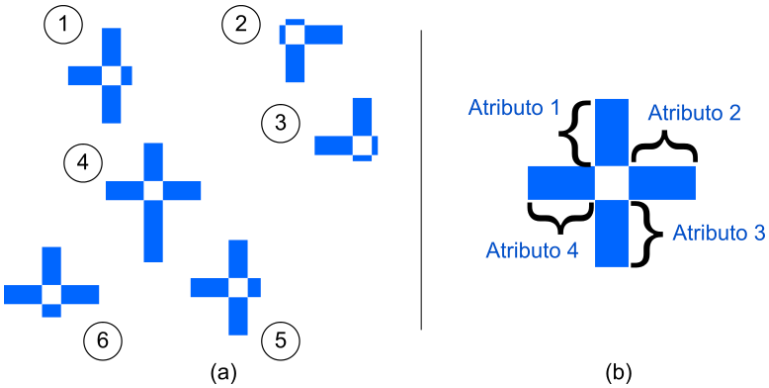


Figura 17 – Exemplo de confusão entre atributos

Uma possível correção é diferenciar cada atributo mapeado com uma cor (Figura 18), neste caso nota-se claramente que as duas observações mais semelhantes do conjunto são as representadas pelos objetos o_1 e o_5.

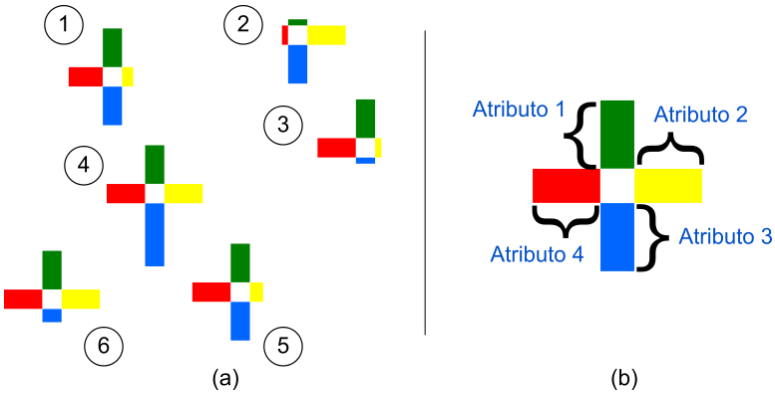


Figura 18 – Possível correção para o exemplo de confusão entre atributos (Figura 17)

Para exemplificar todos os pontos abordados, supõe-se que, em um jogo qualquer, um jogador tem a opção de escolher três moedas, entre as seis de ouro ou prata, presentes na Figura 19.

Figura 19 – Exemplo de mapeamento

A escolha mais frequente são as três maiores moedas de ouro (moedas 1, 3 e 4). É senso comum que ouro vale mais do que prata, logo os jogadores tendem a escolher exclusivamente moedas de ouro, eliminando de forma pré-atentiva as moedas de prata. Em seguida, restam quatro moedas de ouro possíveis e, novamente, é intuitivo que quanto maior a moeda, mais ouro ela possui, assim são escolhidas as moedas 1, 3 e 4.

Na base de dados representada na Tabela 8 é mostrada uma lista de praias e suas respectivas condições climáticas para um certo dia. O objetivo é decidir se uma pessoa deve ir ou não a praia.

Tabela 8 – Exemplo de base de dados - Condições climáticas de determinadas praias

Nome da Praia	**Temperatura(ºC)**	**Previsão**	**Ir a praia?**
Camboinhas	28	Sol	Sim
Peró	29	Chuva	Não
Lopes Mendes	33	Sol	Sim
Copacabana	32	Sol	Sim
Grumari	29	Chuva	Não
Maresias	10	Sol	Não

Após o pré-processamento dos dados, foi eliminado manualmente o nome da praia, o atributo previsão foi *binarizado* e considerado o mais importante e o atributo temperatura foi normalizado, como mostra a tabela abaixo:

Tabela 9 – Exemplo de pré-processamento realizado nos dados da Tabela 8

ID	Previsão	Temperatura(ºC)	Ir à praia?
1	1	0,82	1
2	0	0,85	0
3	1	1	1
4	1	0,96	1
5	0	0,85	0
6	1	0,17	0

Mapeando os dados Tabela 9 nos atributos cor (1 = dourado, 0 = prateado) e raio, respectivamente, de uma moeda, obtém-se como resultado os objetos vistos na Figura 19. Ou seja, ao optar pelas moedas 1, 3 e 4 o jogador, no Mundo dos Dados, escolheu as melhores praias para se frequentar naquele dia.

3.3.5 Estratégias de Agrupamento

Para atingir objetivo do Mundo dos Dados, ou seja, estimar similaridades entre as observações, o desenvolvedor do jogo deve elaborar situações que induzam o jogador a gerar votos.

O primeiro passo é a escolha das mecânicas. Como explicado, este trabalho prioriza mecânicas de jogos tradicionais para que o jogador se sinta familiarizado ao se deparar com um novo jogo.

As mecânicas escolhidas para gerar votos devem poder ser convertidas em uma escolha binária (sim ou não) em relação à similaridade de um determinado par de observações. As mecânicas que têm estas características alteram alguma propriedade de um objeto do jogo, dessa forma, após o evento útil ser gerado, são formados dois grupos: o grupo dos objetos cuja propriedade em questão foi alterada e o grupo dos objeto que não tiveram esta propriedade alterada.

Por exemplo, no jogo mostrado no início deste capítulo, *Os Gatinhos da Vovó* (Figura 11), existem duas mecânicas principais: mover (o braço da vovó) e atirar. A

mecânica mover não altera a propriedade de nenhum objeto mapeado do jogo, portanto, não serve para gerar votos. A mecânica atirar, por sua vez, pode gerar o evento "o gato *x* foi atingido", tal evento altera a propriedade "estado" do gato, de "vivo" para "morto".

O termo "selecionar" será utilizado nas explicações a seguir como uma mecânica genérica que altera a propriedade alvo de um objeto.

Após definidas as mecânicas, é preciso criar situações ou tarefas a serem realizadas no jogo que levem o jogador a utiliza-las. Quatro maneiras, identificadas durante o trabalho, de fazer isso são:

a) Mostrar um exemplo – mostrar para o jogador um objeto como exemplo, para que ele escolha apenas objetos semelhantes;
b) Objetos semelhantes vencem o jogo – criar situações no jogo onde a melhor estratégia possível é a escolha de objetos semelhantes;
c) Eliminar intrusos – mandar o jogador selecionar os objetos mais distintos do grupo;
d) Senso comum – explorar o senso comum dos jogadores, como no exemplo da seção anterior.

Mostrar um exemplo

Uma maneira de induzir o jogador a escolher objetos parecidos é mostrar um exemplo e instruir que o mesmo "selecione" objetos semelhantes.

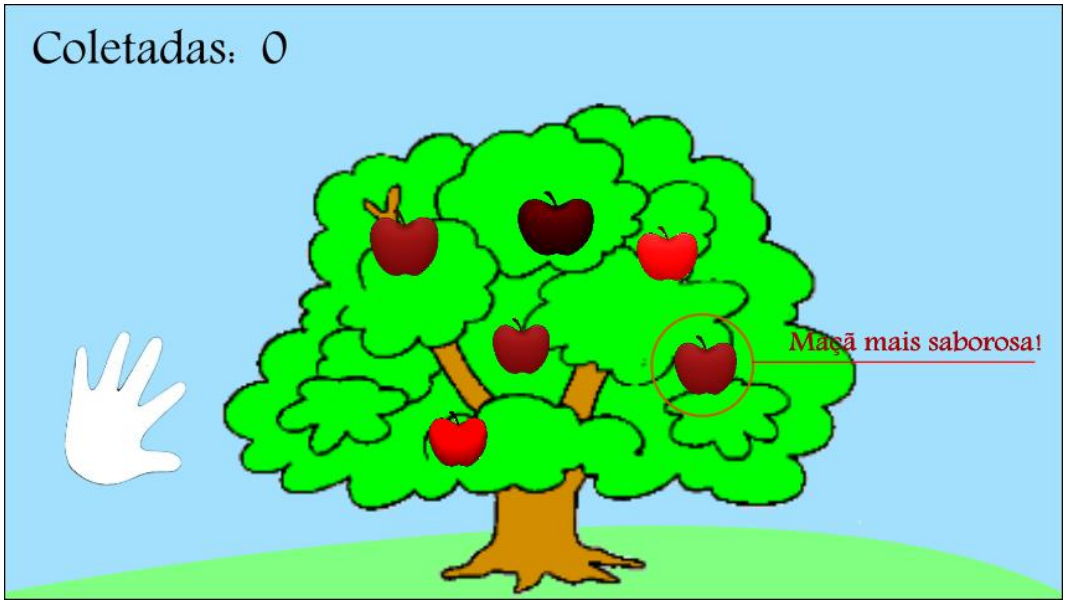


Figura 20 – Jogo das Maçãs – Exemplo de estratégia de agrupamento baseada em exemplo

No *Jogo das Maçãs*, exemplificado na Figura 20, o jogador deve colher maçãs de uma árvore. O jogador é instruído, antes do jogo, a colher apenas as maçãs mais saborosas. É mostrada, ao início, a maçã considerada mais saborosa. Espera-se que a maioria dos jogadores coletem, além da maçã mostrada, outras maçãs semelhantes a ela.

Objetos semelhantes vencem o jogo

Esta ideia consiste em modelar o jogo de forma que a melhor estratégia para vencer o jogo é a escolha de objetos semelhantes.

No *Jogo da Balança*, ilustrado na Figura 21, a cada rodada o jogador deve escolher duas pedras que serão empilhadas simultaneamente, cada uma em um dos lados de uma balança, o objetivo é empilhar o maior número de pedras possíveis sem desequilibrar a balança.

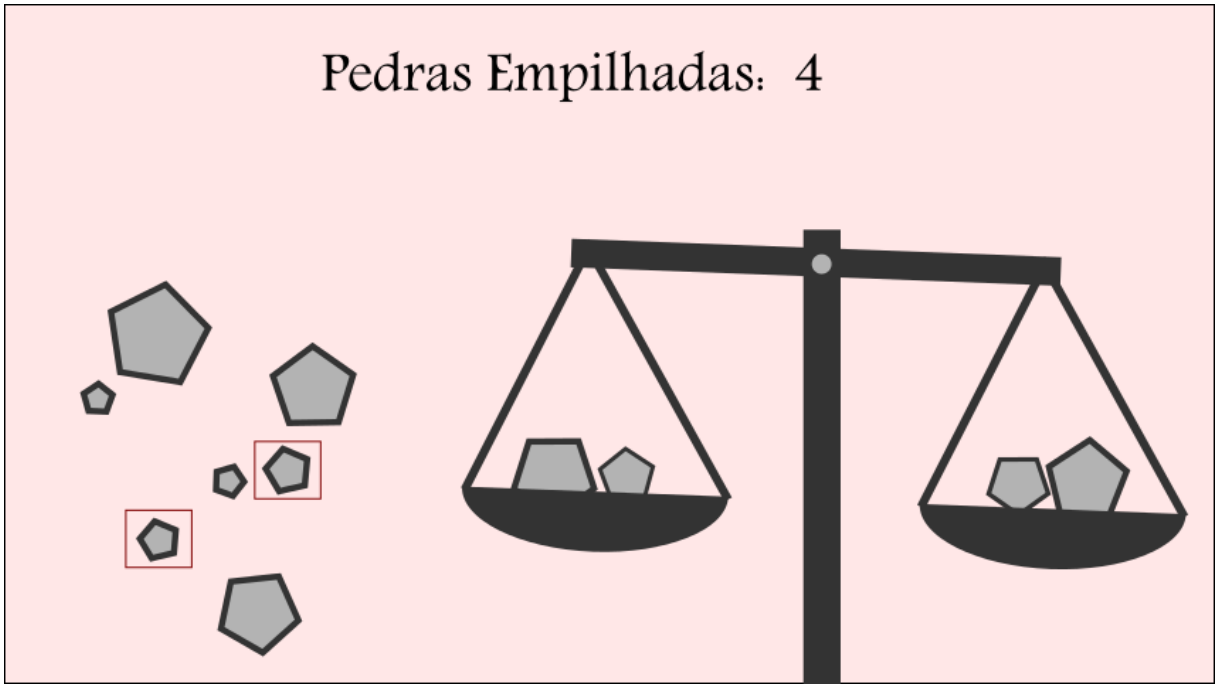


Figura 21 – Jogo da Balança – Exemplo de estratégia de agrupamento baseada em situações do jogo

É intuitivo para os jogadores, mesmo sem instruções prévias, que pedras parecidas tenham o mesmo peso e, portanto, devem manter a balança equilibrada.

Eliminar Intrusos

A terceira forma identificada durante este trabalho de incentivar a geração de votos através das jogadas é o ato de fazer o jogador “selecionar”, novamente utilizando o

termo “selecionar” como uma mecânica genérica, apenas os objetos que ele considerar diferente da maioria, isto é, os intrusos.

No Jogo *Bang Bang*, visto na Figura 22, o jogador tem um rápido intervalo de tempo para atirar no cowboy que ele decidir que é o “intruso”. No exemplo mostrado é esperado que o jogador atire no cowboy do meio, por este ser diferente dos demais.

Figura 22 – Bang Bang - Exemplo de estratégia de agrupamento baseada em eliminar intrusos

Senso comum

A última forma identificada se baseia no conceito de senso comum entre os jogadores, explicado na seção de representação (3.3.4). O exemplo lá mostrado utiliza esta estratégia esperando que as pessoas escolham maiores moedas de ouro, por acreditarem que sejam as mais valiosas.

Uma alternativa possível é explorar, na representação, o gosto pessoal de cada jogador. Por exemplo, mapear as observações em barras de chocolate e instruir o jogador a comer os chocolates mais gostosos.

Espera-se que os jogadores que prefiram chocolates mais doces comam aqueles que possuírem maior quantidade de açúcar, já os jogadores que tenham preferência por chocolates mais amargos tenderão a comer aqueles com maior percentual de cacau. Novamente, votos positivos serão gerados para objetos semelhantes.

3.3.6 Validação das Jogadas

Não basta apenas induzir a geração de votos, é interessante controlar a qualidade dos votos gerados. Para isso, pode-se adotar, durante as partidas, estratégias para a validar as jogadas.

Jogadores que geram bons votos (válidos) devem ser incentivados, passando de fase, ganhando bônus etc. Já aqueles que geram votos indesejados (inválidos) devem ser punidos com perda de pontos, por exemplo. As quatro estratégias de validação identificadas neste trabalho são descritas a seguir.

Validação Positiva

Este tipo de validação consiste em mostrar dois objetos que estejam mapeando duas observações que sejam sabidamente semelhantes, isto é, que tenham ligação conhecida positiva, e aguardar o voto do jogador em relação aquele par de observações. Caso o voto gerado seja negativo, o voto é inválido. Esta validação também considera o voto $v_{i,i}^{j}$ em relação a mesma observação.

Validação Negativa

Trata-se do inverso da validação positiva. Ao mapear duas observações com ligação conhecida negativa, espera-se que o voto gerado pelo jogador em relação a este par de observações seja também negativo.

Validação por Repetição

Consiste em armazenar os votos gerados pelo jogador durante uma certa sessão do jogo. O jogador deve ser penalizado caso, dentro da mesma sessão, gere um voto incoerente, isto é, oposto a algum voto que ele mesmo havia gerado previamente.

Validação do Exemplo

Quando adotada a estratégia de agrupamento baseada em exemplo e a mecânica adotada consiste em "selecionar" objetos parecidos, este, toda vez que aparecer no jogo, deve ser "selecionado". Como no exemplo do *Jogo da Maça* (Figura 20), a maça dita como a mais saborosa deve sempre ser colhida para que a jogada seja válida.

Das quatro estratégias mostradas, a validação negativa e a validação positiva, com exceção da realizada sob a mesma observação, requerem algum conhecimento prévio, em relação às ligações, da base que está sendo agrupada para que sejam adotadas.

Uma alterativa possível, quando não se tem este conhecimento prévio, é mesclar duas bases, uma conhecida outra não, em um mesmo jogo. Dessa forma, votos referentes a observações da base conhecida serão utilizados apenas para realizar validações.

É interessante que o jogador receba um *feedback* rápido em relação a jogada realizada. Por esta razão, durante a sessão, é proposto que sejam utilizadas somente estas validações. Dessa forma, não será necessária, durante a sessão, a comunicação entre os mundos, o que poderia atrasar a resposta ao jogador.

3.3.7 Cálculo da Pontuação

Por fim, deve-se definir como será pontuado o jogador após o fim de uma sessão do jogo. Novamente com o intuito em penalizar/bonificar os jogadores que geraram votos ruins/bons, é interessante utilizar como controle de qualidade a concordância dos votos gerados em relação aos demais jogadores.

Diferentemente das estratégias validação, esta etapa requer comunicação e, consequentemente, processamento no Mundo dos Dados. Por isso deve ser feita ao final de uma sessão (final da fase, fim de jogo etc.), pois o *feedback* do jogador não precisa ser instantâneo.

3.3.8 Escolha das Observações

SGJA – Sistema Gerenciador de Jogos de Agrupamento

É necessária a existência de um sistema que controle a comunicação entre os mundos. Neste trabalho, este sistema é chamado de SGJA - Sistema Gerenciador de Jogos de Agrupamento. Portanto, o SGJA conhece todos os jogos e todas as bases de dados. Cabe a ele decidir, quando requisitado, quais observações serão enviadas para um jogo, validar os votos recebidos etc.

Início de uma Sessão

Ao iniciar uma sessão deve-se selecionar as observações e validações que serão utilizadas durante a sessão. É interessante que cada sessão contenha objetos diferentes, para que o jogo não se torne repetitivo. Como os objetos são, fundamentalmente, uma representação das observações, deve-se enviar, para o Mundo dos Dados, observações diferentes a cada sessão.

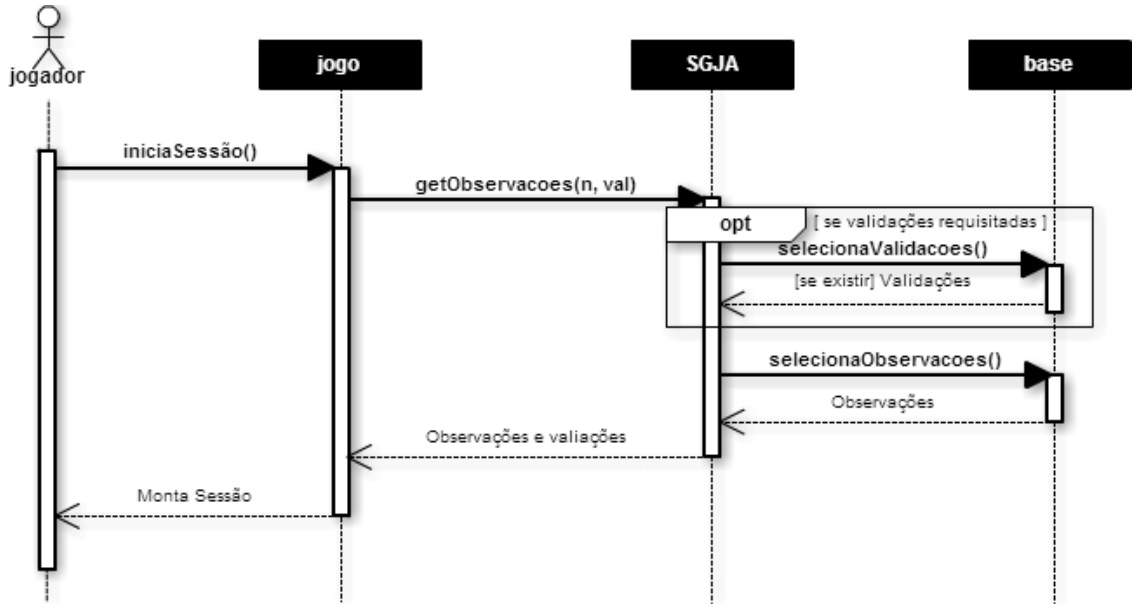


Figura 23 – Ações executadas ao iniciar uma sessão do jogo

A Figura 23 mostra a sequência de ações necessárias para se iniciar uma sessão. Inicialmente o jogador requer o início da sessão (ou a sessão é iniciada automaticamente, em casos de mudança de fase, por exemplo).

Em seguida o jogo requer para o SGJA uma certa quantidade de observações (n) que ele deseja representar durante aquela sessão. O jogo pode também requisitar pares de observações cuja as ligações são conhecidas para utilizá-las como validações (val).

Então o SGJA seleciona, na base que está sendo representada por aquele jogo, os pares de observações requisitados para a validação. Esta requisição pode falhar, retornando vazio, caso ainda não seja conhecida nenhuma validação daquele tipo para esta base.

Logo após o SGJA escolhe mais observações até atingir o número de observações requisitadas. Neste momento, pode ser usada alguma heurística para a escolha das observações que serão enviadas para o jogo, por exemplo, as observações menos enviadas até aquele momento ou as que possuírem menos ligações conhecidas.

Por fim, são retornadas as observações e validações requisitadas e o jogo, então, se encarrega de montar a sessão que será jogada com os dados recebidos.

Final de uma Sessão

Ao fim de uma sessão um evento útil é gerado. Temos, contido nele, o conjunto de todos os objetos do jogo que o jogador teve a opção de "selecionar", utilizando novamente "selecionar" como uma mecânica genérica que altera uma propriedade alvo de um objeto. A partir deste momento o SGJA deve realizar os seguintes passos:

1. Extrair os votos contidos no evento útil;
2. Validar os votos extraídos;
3. Atualizar a estimativa das ligações.

3.3.9 Extração dos Votos

O primeiro passo é extrair os votos contidos em um evento útil. Como exemplo, é suposto um evento útil e_i que contém seis objetos. Destes, três foram "selecionados" durante o jogo. Após inverter a função de representação utilizada, são descobertas as observações representadas por cada objeto. Tem-se, então, dois conjuntos:

$$\text{Observações selecionadas} = \{d_1, d_2, d_3\},$$

$$\text{Observações não selecionadas} = \{d_4, d_5, d_6\}.$$

Chamamos de *voto explícito* um voto gerado através da ação direta do jogador. No exemplo, cada par de observações presente no conjunto das selecionadas gera um voto explícito positivo, pois o jogador disse, explicitamente ao selecionar, que estas observações são parecidas.

Votos gerados a partir do conjunto dos selecionados:

$$v^j_{1,2},\ v^j_{1,3} \text{ e } v^j_{2,3} = 1.$$

O jogador também disse explicitamente, através de suas jogadas, que cada uma das observações do conjunto das selecionadas é diferente das observações do conjunto das não selecionadas. Com isso foram gerados votos explícitos negativos para cada par de observações presentes em conjuntos distintos.

$$v^j_{1,4},\ v^j_{1,5}, v^j_{1,6},\ v^j_{2,4},\ v^j_{2,5}, v^j_{2,6},\ v^j_{3,4},\ v^j_{3,5}, v^j_{3,6} = -1.$$

Denominamos *voto implícito* um voto que pode ser subentendido a partir do conjunto dos objetos não selecionados. Apenas podem ser considerados os votos

implícitos caso a base em questão possua, sabidamente, exatamente dois agrupamentos. Caso contrário, não é possível afirmar que as observações não selecionadas fazem parte de um mesmo agrupamento.

Votos gerados a partir do conjunto das não selecionadas, supondo a existência de dois clusters:

$$v_{4,5}^{j},\ v_{4,6}^{j} \text{ e } v_{5,6}^{j} = 1.$$

Assim são calculados todos os votos contidos no evento útil u_i. Neste caso temos $C_2^6 = 15$ votos. Como mostra o gráfico na Figura 24, a função $f(x) = C_2^x$ tem um crescimento maior do que uma função linear, ou seja, ao dobrar o número de observações analisadas, mais do que dobra o número de votos gerados. Por esta razão é interessante tentar fazer o jogador analisar a maior quantidade de observações possíveis durante uma mesma sessão.

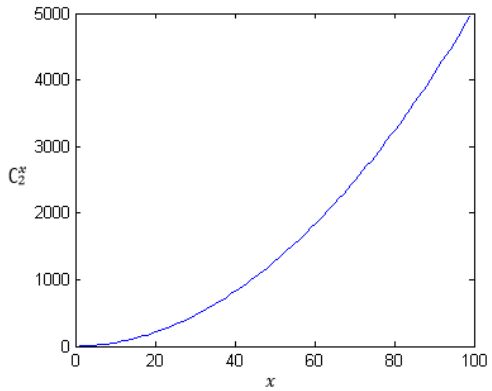


Figura 24 – Gráfico da função $f(x) = C_2^x$

3.3.10 Validação dos Votos

Deve-se calcular o percentual dos votos gerados que concordam com os votos gerados pela maioria dos jogadores acerca do mesmo par de observações. O resultado obtido será o *percentual de concordância* entre os jogadores acerca dos votos desta sessão.

Neste ponto, podem ser feitas duas validações para decidir se os votos serão aceitos ou não: validação por sessão e validação por jogador.

Validação por sessão

Consiste em decidir aceitar ou não os votos contidos no evento útil por ela gerado, baseado exclusivamente na concordância atingida. Para isto, pode-se definir um limiar percentual mínimo da qualidade dos votos daquela sessão.

Validação por jogador

Consiste em armazenar todos os votos de todas as sessões para um mesmo jogador e calcular o percentual de concordância deste com os demais jogadores. Pode ser utilizado como corte um limiar percentual desta concordância ou um percentual do total de jogadores. Dessa forma, apenas os votos dos melhores jogadores são utilizados.

3.3.11 Atualização das Ligações

Para estimar o valor de cada ligação propomos que os votos aceitos sejam armazenados em duas matrizes V^+e V^- com *D* linha *e D* colunas, sendo *D* o número de observações da base de dados, onde cada célula $V_{i,j}^+$ contém o número de votos positivos do par (d_i, d_j) de observações e, por sua vez, $V_{i,j}^-$ contém o número de votos negativos acerca do par (d_i, d_j).

Seja $n_{i,j}$ o total de votos recebidos deste par:

$$n_{i,j} = V_{i,j}^- + V_{i,j}^+$$

Seja $\hat{p}_{i,j}$ a proporção amostral dos votos positivos deste par:

$$\hat{p}_{i,j} = \frac{V_{i,j}^+}{n_{i,j}}$$

Dessa forma, propomos que a estimativa da ligação $\varphi(d_i, d_j)$ seja:

$$\varphi(d_i, d_j) \cong h(d_i, d_j) = \hat{p}_{i,j}.$$

Como trata-se de uma amostra, já que não é possível saber a opinião de todos os jogadores possíveis em relação a um par de observações, é necessário estimar esta proporção assumindo um certo intervalo de confiança (IC).

As jogadas geram, neste modelo, apenas votos positivos ou negativos, por isso, $\hat{p}_{i,j}$ seguirá uma lei Binomial$(n_{i,j}, \hat{p}_{i,j})$. Utilizando o método de Clopper-Pearson, também conhecido como "Intervalo de Confiança Exato" (Clopper e Pearson, 1934), define-se o IC da proporção como (SIGMAZONE, 2015):

$$\hat{p}_{i,j_{ub}} = 1 - Betainv(\frac{\alpha}{2}, n_{i,j} - V_{i,j}^{+}, n_{i,j} + 1)$$

$$\hat{p}_{i,j_{lb}} = 1 - Betainv(1 - \frac{\alpha}{2}, n_{i,j} - V_{i,j}^{+} + 1, V_{i,j}^{+})$$

Onde:

$\hat{p}_{i,j_{ub}}$ é o limite superior do intervalo de confiança;

$\hat{p}_{i,j_{lb}}$ é o limite inferior do intervalo de confiança;

Betainv() é a inversa da c.d.f. da distribuição beta;

α é o percentual de chance de erro do tipo 1, 1 - α é a confiança.

Dessa forma,

$$\hat{p}_{i,j_{lb}} < \hat{p}_{i,j} < \hat{p}_{i,j_{ub}}.$$

Para inferir o tipo da ligação $\varphi(d_i, d_j)$, é necessário estipular um limiar, tal que, caso o IC de $\hat{p}_{i,j}$ esteja inteiramente contido acima deste intervalo assume-se ligação conhecida positiva, por outro lado, caso o IC esteja inteiramente contido abaixo deste limiar a ligação é dada como conhecida negativa. Intuitivamente foi testado o valor médio ($\frac{1}{2}$) como limiar, porém, a quantidade de votos positivos e negativos gerados durante as sessões não é balanceada. Por isso, um palpite melhor para este limiar é a proporção média dos votos:

$$\hat{p}_{média} = \frac{\sum_{i,j} V_{i,j}^{+}}{\sum_{i,j} n_{i,j}}$$

Note que $\hat{p}_{média}$ também possui um intervalo de confiança, porém, como esta é calculada em função de todos os votos de todos os pares possíveis, o seu denominador cresce rapidamente, fazendo o seu IC cada vez menor. Na prática este IC de $\hat{p}_{média}$ é insignificante em relação ao IC de $\hat{p}_{i,j}$. Por este motivo, pode-se utilizar apenas $\hat{p}_{média}$, ignorando o seu intervalo de confiança, como limiar.

Assim, tem-se a classificação do tipo de $\varphi(d_i, d_j)$:

$$\varphi(d_i, d_j) \text{ é } \begin{cases} conhecida\ positiva, & \hat{p}_{i,j_{lb}} > \hat{p}_{média},\ n_{i,j} \geq n_{min} \\ conhecida\ negativa, & \hat{p}_{i,j_{ub}} < \hat{p}_{média},\ n_{i,j} \geq n_{min} \\ estimada, & \hat{p}_{i,j_{lb}} \leq \hat{p}_{média} \leq \hat{p}_{i,j_{ub}},\ n_{i,j} \geq n_{min} \\ desconhecida, & n_{i,j} < n_{min} \end{cases}$$

Deve ser estipulado um número mínimo de votos n_{min} para que uma ligação seja classificada, caso o contrário ela é dita desconhecida.

Ultrapassado n_{min}, caso o limite inferior de $\hat{p}_{i,j}$ esteja acima do limiar estipulado, é inferido que as observações são similares. Caso o limite superior de $\hat{p}_{i,j}$ esteja abaixo de $\hat{p}_{média}$ assume-se que as observações são distintas. Caso contrário, isto é, quando o IC de $\hat{p}_{i,j}$ contém $\hat{p}_{média}$, nada se pode afirmar sobre a similaridade entre d_i e d_j, tem-se apenas o seu valor estimado. Uma vez que um determinado par de observações tem sua ligação conhecida (positiva ou negativa), este pode ser utilizado como validação nas próximas sessões.

3.3.12 Algoritmo de Agrupamento

Ao chegar nesta etapa o principal objetivo do Mundo dos Dados, ou seja, estimar similaridades entre observações, foi atingido. Esta etapa final preocupa-se apenas em utilizar as estimativas encontradas para as ligações e utiliza-las com o intuito de encontrar agrupamentos de observações parecidas.

É importante ter definido o que a pessoa responsável pelos dados realmente necessita. Existem casos em que esta etapa não é necessária, quando se quer, por exemplo, obter apenas observações similares a uma determinada observação, como acontece em sistemas de recomendação. Para isso, é suficiente selecionar as observações conhecidas positivas da observação em questão.

Para encontrar agrupamentos, pode-se considerar as observações e suas respectivas ligações como um grafo onde cada observação representa um vértice e cada ligação forma uma aresta. Esta disposição possibilita a utilização de algoritmos de clusterização baseados em grafos, como *Spectral Clustering* (Schaeffer, 2007), *Markov Clustering* (Van Dongen, 2000) etc.

Entretanto, este trabalho teve como foco principal as etapas anteriores, que envolvem computação humana para conseguir boas estimativas de similaridades entre observações. Notou-se que conhecer todas as ligações, através do modelo proposto, é uma tarefa demorada. Por este motivo, é proposto um algoritmo de agrupamento baseado em clusterização hierárquica aglomerativa (Maimon e Rokach, 2005) que explora, respeitando a ordem de importância, as ligações conhecidas e estimadas.

Como entrada o algoritmo, descrito na Figura 25, recebe a lista de todas as observações *D*, a lista de ligações conhecidas positivas L_+, a lista das ligações conhecidas negativas L_-, a lista de ligações estimadas L_e e uma lista de observações pivô *P*, sendo

cada pivô uma observação inicial de um agrupamento. Como saída são retornados os agrupamentos G_i encontrados. O algoritmo segue os seguintes passos:

Passo 1 *– Criar* |P| *grupos* G_i*, com* $\mathrm{i} = 1, \ldots |\mathrm{P}|$ *, cada um contendo o seu respectivo pivô e remover de* D *as observações de* P*.*

Passo 2 *– Calcular (ou atualizar), para cada observação* d_j *em D, a força de atração de* d_j *para cada grupo, isto é, o somatório das ligações conhecidas positivas em* L_+ *que ligam* d_j *a alguma observação pertencente ao grupo* G_i*.*

Passo 3 *– Inserir, no grupo encontrado, a observação* d_j *que possuir a maior força de atração, não podendo haver empates.*

Passo 4 *– Remover* d_j *de D e voltar ao passo 2 até que nenhuma nova observação seja alocada em algum grupo.*

Passo 5 *– Calcular (ou atualizar), para cada observação* d_j *em* D*, a força de repulsão de* d_j *para cada grupo, isto é, o somatório das ligações conhecidas negativas em* L_- *que ligam* d_j *a alguma observação pertencente ao grupo* G_i*.*

Passo 6 *– Inserir, no grupo encontrado, a observação* d_j *que for menos rejeitada, ou seja, a menor força de repulsão encontrada, não podendo haver empates.*

Passo 7*– Remover* d_j *de D e voltar ao passo 5 até que nenhuma nova observação seja alocada em algum grupo.*

Passo 8 *– Calcular (ou atualizar), para cada observação* d_j *em* D*, a* força de atração estimada *de* d_j *para cada grupo, isto é, o somatório das ligações estimadas em* $\mathrm{L_e}$ *que ligam* d_j *a alguma observação pertencente ao grupo* G_i*.*

Passo 9 *– Inserir, no grupo encontrado, a observação* d_j *com a maior força de atração estimada, não podendo haver empates.*

Passo 10 *– Remover* d_j *de D e voltar ao passo 8 até que nenhuma nova observação seja alocada em algum grupo. Remover* d_j *de D e voltar ao passo 8 até que nenhuma nova observação seja alocada em algum grupo.*

Figura 25 – Pseudocódigo do algoritmo de agrupamento

Caso não se tenha previamente as observações pivôs é necessário encontrar, no grafo formado pelas observações (vértices) e ligações (arestas), um subgrafo completo K_n de ligações conhecidas negativas, onde n é o número de agrupamentos que se deseja encontrar, assim, os pivôs serão as observações deste subgrafo.

Nota-se que é um algoritmo custoso, pois, como proposto, é necessário calcular diversos somatórios a cada iteração. Entretanto é possível otimizá-lo, atualizando, em cada iteração, apenas as forças que envolvem a última observação inserida e as observações que ainda não foram alocadas.

Apesar de custoso, o algoritmo descrito relativamente fácil de implementar e que apresentou bons resultados, que serão detalhados no próximo capítulo.

Capítulo 4

Implementações e Experimentos

Neste capítulo são mostradas a arquitetura e as tecnologias utilizadas para testar os conceitos propostos no capítulo anterior. Em seguida são descritas as bases que foram utilizadas durante os testes realizados e os jogos implementados.

4.1 Arquitetura e Tecnologias

A arquitetura implementada visou manter, como proposto, o conceito de independência entre o Mundo dos dados e o Mundo do jogo. No Mundo dos Dados, as observações, votos e ligações foram armazenados utilizando o servidor de bancos de dados MySQL[6]. O Algoritmo para realização dos agrupamentos, descrito na Figura 25, foi implementado através de um Script em MATLAB[7].

No Mundo do Jogo, os jogos foram desenvolvidos utilizando o *framework* Construct 2[8] que, em sua versão gratuita, exporta os jogos criados para as tecnologias HTML 5/Javascript, suportadas pelos principais navegadores WEB. Além disso, o banco de dados MySQL foi utilizado também no Mundo do Jogo para armazenar o *rank* de pontuação dos jogadores.

O SGJA (Sistema Gerenciador de Jogos de Agrupamento) que realiza a interface entre os mundos modelado como um Webservice escrito em linguagem PHP[9]. A Figura 26 mostra a arquitetura e os principais métodos implementados no SGJA. As setas em preto representam os métodos públicos, isto é, que são visíveis aos componentes externo e chamados quando requisitados. As setas vermelhas representam os métodos privados do SGJA.

Para agilizar os testes e diminuir a parte burocrática, o que poderia desestimular alguns usuários, não foi requerido nenhum cadastro por parte destes na hora de jogar e, por este motivo, não foi feito o controle de quais ou quantos jogadores jogaram cada jogo.

[6] http://www.mysql.com/
[7] http://www.mathworks.com/products/matlab/
[8] https://www.scirra.com/construct2/
[9] http://php.net/

Em contrapartida foi feito o controle dos votos “aceitos” através da validação da sessão. Como limiar, foi definido que caso 75% dos votos recebidos por uma sessão não estivessem de acordo com a maioria dos jogadores, estes votos não seriam computados.

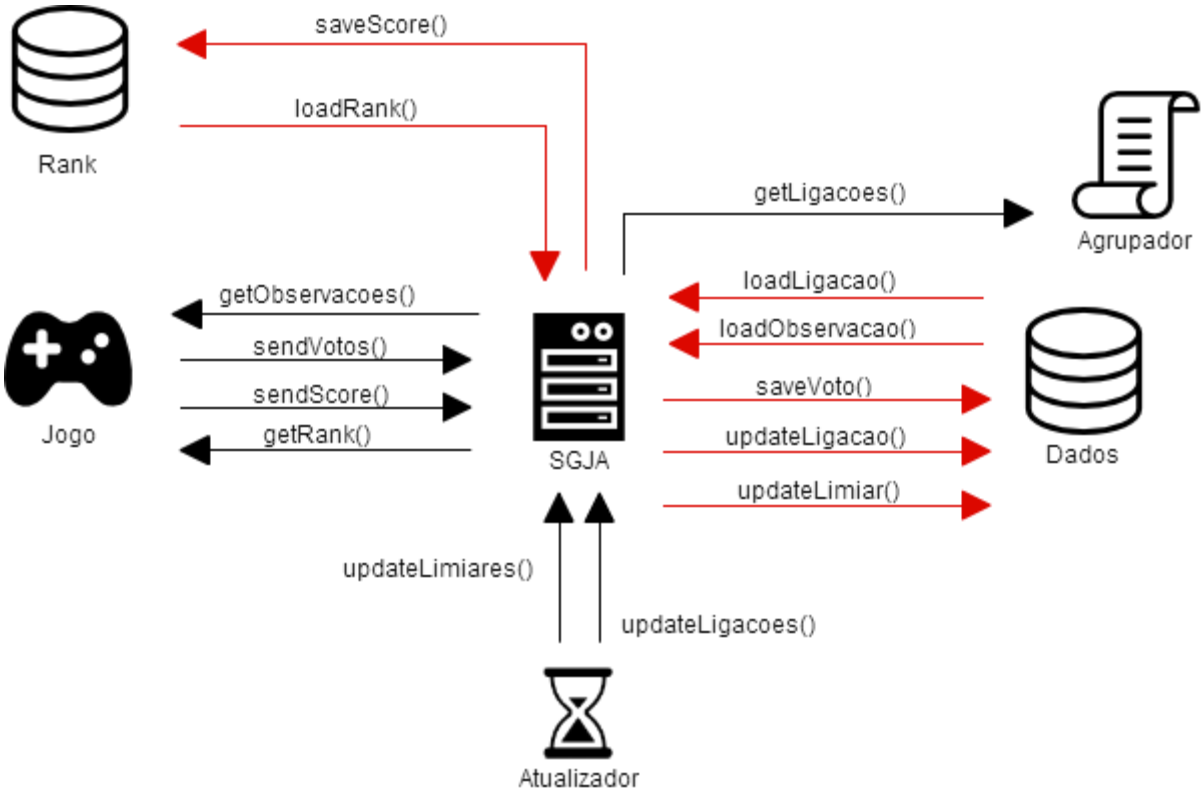


Figura 26 – Arquitetura do SGJA (Sistema Gerenciador de Jogos de Agrupamento)

Como visto na figura acima, um Jogo interage com o SGJA pedindo observações (com validações, quando necessário) e enviando os votos obtidos pelos jogadores. Um Jogo pode também salvar e retornar os scores armazenados.

Na implementação adotada, durante as partidas, os votos são armazenados individualmente. Diariamente, um script *Atualizador* chama o método responsável por atualizar as ligações com os votos aceitos daquele dia. Após isso, são atualizadas as proporções de votos positivos/negativos de cada jogo que serve, como explicado na seção 3.3.11, como limiar para classificar uma ligação em positiva, negativa ou estimada.

O script *Agrupador*, quando executado, recebe as ligações (e seus respectivos tipos) do SGJA e executa o algoritmo para encontrar os agrupamentos das observações. Caso o resultado encontrado ainda não seja satisfatório, pode-se aguardar que novos votos sejam recebidos e executar o script novamente em um momento posterior.

4.2 Jogos e Experimentos

Os jogos implementados foram disponibilizados para o público geral durante um período aproximado de um mês. Não foram realizados, como dito, controles qualitativo e quantitativo dos jogadores. Durante o período disponibilizado alguns jogadores foram observados e questionados enquanto jogavam.

Das bases utilizadas nos experimentos, três delas consistem em dados bidimensionais gerados através de distribuições normais.

A primeira, identificada como *Norm 1*, consiste em dados oriundos de duas distribuições normais com médias centradas em (1,1) e (6,6), respectivamente, e desvio padrão igual a (1,1) para ambas, ou seja, não há sobreposição entre as observações dos dois clusters, tornando o agrupamento mais simples.

A segunda, *Norm 2*, trata-se de observações geradas por duas normais levemente sobrepostas, com médias em (1,1) e (4,4) e desvios padrões também iguais a (1,1). Neste caso, os *clusters* se interceptam levemente. Isto possibilitou testar como o algoritmo tratou as observações próximas a fronteira entre os *clusters*. Tanto a *Norm 1* quanto a *Norm 2* possuem 100 observações, sendo 50 em cada *cluster*.

A última, *Norm 3*, é formada por três clusters isolados, com 30 observações cada, gerados por distribuições normais centradas em (1,1), (1,5) e (10,1). Neste caso eles também estão totalmente isolados, sendo o principal objeto desta base avaliar como o algoritmo e os jogos se comportam com mais de dois clusters. Todas as bases foram normalizadas entre 0 e 1, como proposto na seção pré-processamento. A Figura 27 mostra uma visualização destas bases após normalizadas.

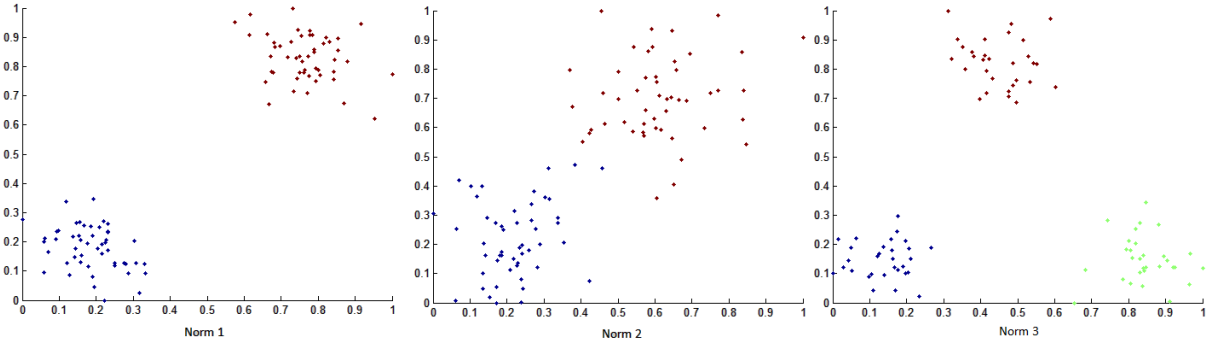


Figura 27 – Bases de dados geradas a partir de distribuições normais

As duas outras bases utilizadas nos testes tratam-se de bases clássicas para testar algoritmos de classificação, retiradas do repositório *UCI Machine Learning Repositor* (Lichman, 2013).

A primeira é a base denominada *Iris* (Fisher, 1936), que consiste em observações que representam flores características de regiões com o clima temperado. Trata-se de uma base de dados com três classes distintas e 50 observações em cada classe, as classes rotulam as íris como: *iris setosa*, *iris versicolor* e *iris virginica*. Esta base possui uma característica interessante onde uma das classes (*íris setosa*) é linearmente separável das duas outras (*íris versicolor e íris virginica*). Assim, para um algoritmo de clusterização, é uma tarefa relativamente simples a separação desses dois clusters, porém não é tão trivial a separação da *íris versicolor e íris virginica*.

Por fim, a base *Wine* (Cortez et al., 2009), foi utilizada no final dos experimentos. Esta base é formada por resultados de análises químicas feitas sobre vinhos oriundos de uma mesma região da Itália, derivados de três cultivares diferentes. As amostras determinaram as quantidades de treze constituintes diferentes em cada observação. No total são 178 observações que formam três classes não balanceadas, com 59, 71 e 48 observações respectivamente. A Tabela 10 resume todas as bases utilizadas durante os testes.

Tabela 10 – Bases de dados utilizadas nos experimentos

Nome	**Nº Observações**	**Nº *Clusters***	**Divisão dos *Clusters***	**Nº Atributos**
Norm 1	100	2	50/50	2
Norm 2	100	2	50/50	2
Norm 3	90	3	30/30/30	2
Iris	150	3	50/50/50	4
Wine	178	3	59/71/48	13

Para explicar e comparar os resultados obtidos, serão utilizados os conceitos de *acurácia* e *cobertura* definidos, neste contexto, como:

$$acurácia = \frac{total\ de\ observações\ corretamente\ agrupadas}{total\ de\ observações\ agrupadas}$$

$$cobertura = \frac{total\ de\ observações\ agrupadas}{total\ de\ observações\ da\ base}$$

A seguir são mostrados os jogos implementados, os prós e contras de cada jogo e os agrupamentos encontrados. Como referencial, alguns dos resultados obtidos foram comparados com resultados encontrados por algoritmos clássicos, como o *k*-means e o *k*-NN. O software Weka 3 [10] foi utilizado para rodar os algoritmos utilizados nas comparações e para realizar seleção de atributos.

4.2.1 Bang Bang

O primeiro jogo desenvolvido para testar o método proposto foi intitulado *Bang Bang*. Sua temática é baseada no velho-oeste americano. A cada rodada é simulada uma batalha entre cowboys. O jogador representa, em primeira pessoa, um cowboy cujo objetivo é atirar apenas nos cowboys suspeitos.

Figura 28 – Telas do jogo Bang Bang

[10] http://www.cs.waikato.ac.nz/ml/weka/

Estratégia de Agrupamento

Na tela inicial do jogo (Figura 28a) é passada uma instrução simples para os jogadores: “Detecte os cowboys intrusos e mate-os.”. Logo, a estratégia de agrupamento adotada consiste em eliminar intrusos. Este jogo foi elaborado para trabalhar apenas com dois *clusters*. Dessa forma, a escolha por mostrar sempre um número ímpar de cowboys em cada batalha faz com que exista sempre um conjunto de cowboys de um cluster maior do que o do outro cluster, estes últimos são os intrusos.

Estratégia de validação e Pontuação

A cada batalha, dos *n* cowboys mostrados, dois fazem parte de uma validação negativa, ou seja, são sabidamente de clusters distintos. A jogada é considerada válida caso um e somente um dos dois for morto. A decisão por matar ou não os demais cowboys, cujos agrupamentos são desconhecidos, é sempre válida.

Durante uma batalha, o jogador possui cinco segundos para matar os cowboys escolhidos (Figura 28b). Caso faça a escolha certa, os cowboys remanescentes não atiram e o jogador avança para a próxima batalha (Figura 28c). Caso a jogada não seja válida, o jogador é morto e o jogo chega ao fim (Figura 28d). A pontuação é igual ao número de batalhas vencidas (Figura 28f).

Mapeamento dos dados

O jogo está preparado para mapear qualquer base bidimensional com dados normalizados. Os atributos são mapeados em duas características do cowboy: o tamanho do bigode e a inclinação do chapéu (Figura 28e). Assim, uma observação no ponto x = 0,9 e y = 0,1 irá gerar um cowboy com bigode grande e chapéu inclinado para a direita, por outro lado, uma observação no ponto x = 0,1 e y = 0,9 irá gerar um cowboy com bigode pequeno e chapéu tombado para a esquerda.

Experimentos e Considerações

Os testes iniciais foram feitos sob a base *norm 1*. Como inicialização, uma observação de cada grupo foi escolhida aleatoriamente e foi criada uma única ligação negativa para que o jogo pudesse realizar as primeiras validações.

Após serem calculadas 1081 ligações, sendo 70 conhecidas e 1011 estimadas, foi executado o algoritmo de agrupamento que conseguiu 100% de acurácia com 100% de cobertura, ou seja, todas as observações foram agrupadas corretamente em seu cluster, provando que o método é plausível.

Apesar do bom resultado obtido com a base *norm 1*, quando se tentou agrupar a base *norm 2*, com sobreposição entre os clusters, o jogo apresentou dificuldades. Posteriormente, foram notados detalhes cruciais no mapeamento e no número de votos gerados em cada rodada.

Quanto ao mapeamento, notou-se, questionando alguns jogadores, que o tamanho do bigode era muito mais perceptível, na hora de decidir os grupos, do que a inclinação do chapéu. Imagina-se que, como senso comum, características inerentes ao corpo humano sejam mais representativas, para diferenciar "tribos", do que características facilmente mutáveis, como a inclinação do chapéu. Como em ambas as bases testadas não há ordem entre os atributos, isto é, nenhum é mais relevante do que o outro, uma representação ideal deveria fazer com que os jogadores dessem igual importância para ambas as características.

Quanto ao número de votos, como, na maioria das rodadas, são mostrados apenas três cowboys, são gerados $C_2^3 = 3$ votos por rodada. Como um dos votos é em relação ao par de observações já conhecidas (validação negativa), sobram apenas dois votos realmente úteis para calcular novas ligações.

4.2.2 Sapoletando

O jogo *Sapoletando* (Figura 29) foi desenvolvido pensando nos problemas identificados no jogo *Bang Bang*. Nele, o jogador controla uma vovozinha que é perturbada por sapos que moram em uma lagoa próxima a sua casa. Ela percebe que quando existem sapos de famílias distintas na lagoa eles brigam entre si. Para acabar com isso, ela decide deixar somente uma família de sapos viva.

Estratégia de Agrupamento

Em cada nível do jogo são mostrados diversos sapos, o objetivo do jogador, como explicado na Figura 29(b), é identificar uma família de sapos e eliminar os sapos que não pertencem àquela família (Figura 29d), ou seja, a estratégia de agrupamento novamente é baseada em eliminar intrusos.

Cálculo dos Votos

O número de sapos mostrados em um nível *X+1* é sempre maior ou igual ao número de sapos mostrados em um nível *X*, aumentando, constantemente, o número de votos gerados em um único evento útil. Considerando que o jogo esteja trabalhando com um número indeterminado de agrupamentos, não é possível afirmar que só serão mostradas duas famílias (*clusters*, no Mundo dos Dados) de sapos em uma fase. Por isso, não é possível inferir os votos em relação aos sapos mortos, já que estes podem ou não fazer parte da mesma família. De qualquer forma, supondo que sejam mostrados 10 sapos e o jogador tenha decido por matar 5, serão gerados: $C_2^{10} - C_2^5 = 35$ votos em um único evento útil.

Figura 29 – Telas do jogo Sapoletando

Estratégias de Validação e Pontuação

Ao gerar um determinado nível o jogo decide, aleatoriamente, utilizar validação positiva, validação negativa ou nenhuma validação. Com o intuito de obter mais votos

positivos corretos, que são os mais interessantes para o Mundo dos Dados, a pontuação é feita em função da quantidade de sapos deixados vivos, induzindo os jogadores a deixar viva a família que possuir mais sapos e eliminar as outras.

Mapeamento dos dados

O mapeamento foi pensado para que, diferentemente do jogo *Bang Bang*, o jogadores dessem igual importância aos dois atributos da base. Para isso, cada atributo foi mapeado no diâmetro da circunferência que representa cada olho do sapo, ou seja, eles serão exatamente iguais quando mapeando os mesmos valores. Tal representação é mostrada na Figura 29(e).

Experimentos e Considerações

Além da realização de alguns testes onde o jogo se mostrou capaz de classificar rapidamente a base *norm 1* (em um teste, com 90% dos votos certos, chegando ao nível 20 duas vezes foi suficiente para agrupar com 100% de acurácia e 100% de cobertura), o jogo também se mostrou capaz de agrupar a base *norm 2*. Com 1313 ligações, sendo 67 conhecidas e 1237 estimadas, foi possível obter 98% de acurácia e 100% de cobertura. Mesmo resultado obtido pelos algoritmos *k-means* e *k-NN*. Visualmente, na Figura 27, percebe-se que o resultado é razoável, pois duas observações do *cluster* azul estão muito próximas ao agrupamento formado pelas observações vermelhas, o que explica os 2% de erro.

Quando foram realizados testes com a base *norm 3* a maioria dos jogadores observados sentiram dificuldades quando apareciam três famílias distintas em um mesmo nível e não havia nenhuma predominante, por exemplo, três sapos de cada família. Isso gerou um alto percentual de votos errados nessas situações.

4.2.3 Dangerous Mine

O terceiro jogo, chamado *Dangerous Mine*, tem a sua história baseada na exploração de uma mina de ouro. O jogador deve se aventurar pela mina coletando pedras de ouro e as depositando nas estações para que o trem as leve para a superfície. São utilizadas as mecânicas mover e coletar, o desafio do jogador é encontrar um caminho para descer pela mina e coletar pedras para a extração de ouro.

Este jogo explorou aspectos ainda não explorados nos dois jogos anteriores, com o objetivo de agrupar de forma mais eficiente bases com mais de dois clusters. Como instrução, o jogador é informado para coletar as "melhores pedras de ouro" e advertido que pedras ruins podem ser explosivas (Figura 30).

Figura 30 – Dangerous Mine – Instruções ao jogador

Como é ser visto na Figura 31, diferentemente dos jogos anteriores, este jogo não foi dividido em níveis. Toda a ação ocorre em apenas uma fase "infinita" que é dificultada com o passar do tempo. Enquanto o jogador tenta encontrar o caminho pela mina, a "tela" do jogo vai subindo lentamente tentando esmagar o mineiro contra a parede. A velocidade de subida da "tela" aumenta com o tempo. Essa estratégia aumenta consideravelmente o número de votos por evento útil, já que o jogador analisa diversas pedras, e decide por coleta-las ou não, durante uma mesma sessão. Por outro lado, como uma sessão dura mais tempo, os jogadores tendem a jogar menos sessões.

Estratégia de Agrupamento

A estratégia de agrupamento utilizada foi baseada em mostrar um exemplo. Antes de iniciar o jogo é mostrado ao jogador a melhor pedra de ouro possível e espera-se que ele colete as pedras mais parecidas com esta. Tal estratégia se mostrou mais eficiente quando existem mais de duas classes, pois como as pedras são mostradas uma de cada vez, ao jogador basta decidir se é parecida ou não com o exemplo, gerando votos positivos

para observações “próximas” ao exemplo, e negativos para as demais, não dependendo do agrupamento em que ela se encontra.

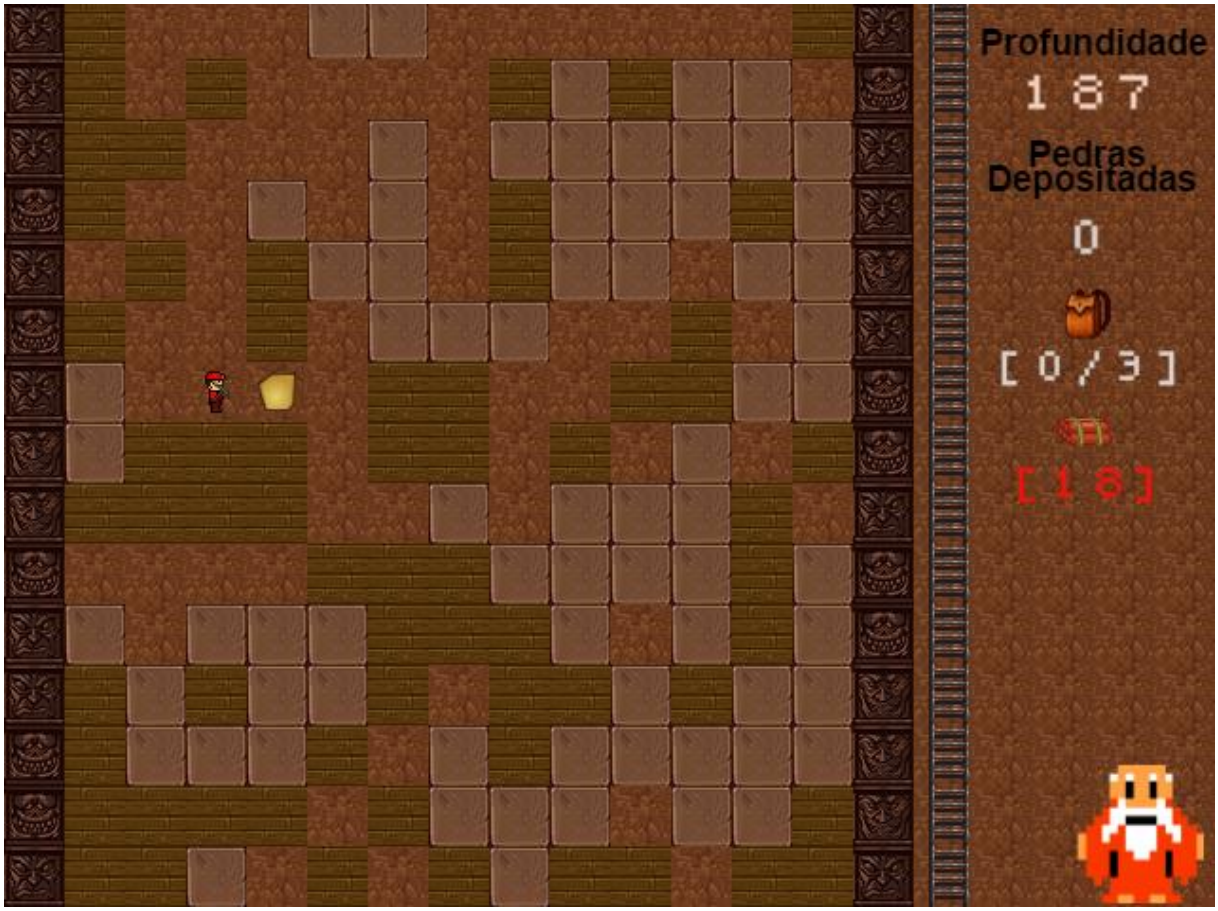


Figura 31 – Dangerous Mine – Durante o jogo

Mapeamento dos dados

O mapeamento se baseou em duas características: cor e tamanho. A coloração é feita em um degrade que varia do dourado até o prateado, quanto mais dourada a pedra for, “melhor” ela será. Quanto ao tamanho, tratando-se de ouro, é intuitivo que quanto maior a pedra, mais ouro será extraído. Ambas as características tentam explorar o senso comum entre os jogadores.

Foi realizado um mapeamento local onde a observação utilizada como exemplo, seja ela qual for, é mapeada sempre na maior e mais dourada pedra (Figura 30, primeira instrução). As demais observações são mapeadas em função da diferença, em módulo, de cada atributo em relação ao exemplo.

Estratégias de Validação

Além de uma validação negativa, ou seja, uma das pedras que aparece pela mina é sabidamente diferente do exemplo e explode quando coletada, foi implementada

também a validação por repetição. Uma mesma observação pode ser representada mais de uma vez durante uma mesma sessão, quando isso ocorre é testada a coerência do jogador, por exemplo, se ele deixou de coletar uma pedra que esteja mapeando a observação d_3, sempre que d_3 for representada naquela sessão ela não poderá ser coletada, caso contrário ocorrerá uma explosão (fim de jogo).

Cálculo da Pontuação

O jogo acaba quando o mineiro é prensado contra a parede ou coleta uma pedra explosiva. Neste momento a pontuação é calculada em função da profundidade atingida, sem nenhuma relação com o Mundo dos Dados, e da quantidade de pedras coletadas. Tal pontuação é ponderada pela "qualidade" das pedras, que nada mais é do que o percentual de votos que estão de acordo com a maioria dos jogadores. Assim, se o jogador coletar pedras ruins e, por falha das validações, não explodir durante o jogo, este será penalizado com uma baixa pontuação, mesmo coletando um número grande de pedras.

Experimentos e Considerações

Durante os experimentos, o jogo se mostrou eficiente para agrupar as bases *Norm 1* e *Norm 3*, esta última foi agrupada com 100% de cobertura e 100% de acurácia com aproximadamente 1660 ligações, sendo 123 conhecidas.

Por outro lado, o mapeamento mostrou o mesmo defeito apresentado no jogo *Bang Bang*, os jogadores deram muito mais importância a coloração da pedra ao tamanho. Acredita-se que o jogo obteve êxito no agrupamento das bases *Norm 1* e *Norm 3* pois estas são, visivelmente, separáveis com apenas um dos atributos. Quando se tentou agrupar a base *Norm 2*, que necessita que ambos os atributos sejam analisados igualmente, os resultados não foram bons, isto é, os clusters não foram devidamente separados.

Sendo assim, a principal contribuição deste jogo não foram os resultados em si, e sim as considerações que geraram novas propostas para a seção de representação dos dados (Seção 3.3.4), o próprio exemplo final da seção citada é baseado na representação adotada neste jogo.

4.2.4 Missão E

O jogo *Missão E* foi criado com o objetivo explorar as melhores características dos três jogos anteriores. O jogo tem como tema uma exploração espacial. O jogador controla um

astronauta que deve coletar amostras de um material desconhecido descoberto no espaço, conhecido como *Esquil*, como explicado na Figura 32 (a).

A mecânica do jogo é inspirada no clássico jogo *snake*. Porém, no jogo original, o jogador deve apenas coletar novas peças para fazer o personagem crescer. Já na variação criada, o jogador deve optar por coletar ou destruir as peças.

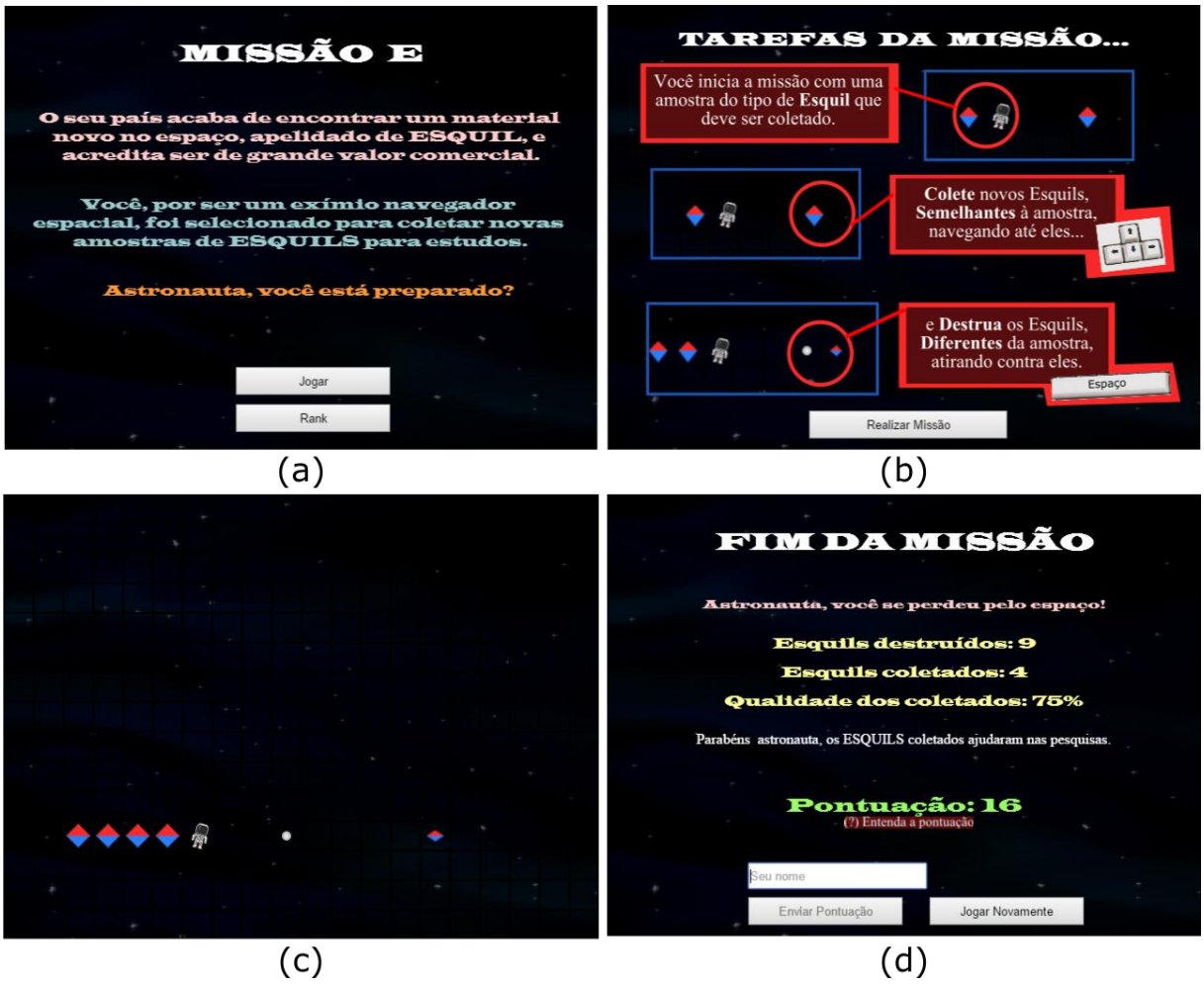


Figura 32 – Telas do jogo Missão E

Estratégia de Agrupamento e Pontuação

As “tarefas da missão” – ou seja, as instruções para que o jogador gere bons votos para o Mundo dos Dados – são passadas na tela mostrada na Figura 32 (b). O jogador inicia com uma amostra de *Esquil* e é requisitado a coletar as semelhantes e destruir as distintas. Ou seja, novamente a estratégia de agrupamento adotada é baseada em um exemplo.

O jogador pontua tanto coletando quanto destruindo as peças. Uma pontuação maior é dada ao coletar para balancear o ônus de carregar a peça, ou seja, controlar uma “cobra” maior. Ao fim do jogo a pontuação é ponderada pela qualidade dos votos.

Estratégias de Validação

A velocidade de movimentação do astronauta aumenta com o passar do tempo e o jogo termina quando ele bate ou atira contra uma peça já coletada, quando coleta uma peça que não deveria coletar (validação negativa ou por repetição), quando explode uma peça que deveria coletar (validação positiva ou por repetição) ou quando sai da tela do jogo.

Mapeamento dos dados

O mapeamento dos objetos foi feito baseado em *star plots* (Chambers, 1983), família de glifos baseados em ícones que lembram estrelas. Cada ponta da estrela mapeia um atributo, o valor do atributo determina o comprimento da ponta por ele representada. O número de atributos que se deseja mapear determina o número de pontas que a estrela irá possuir.

Foi adotado um mapeamento que suporta até quatro atributos, como é mostrado na Figura 33(a). A Figura 33(b) ilustra um mapeamento que não pode ser adotado, pois pode gerar confusão entre os atributos caso o objeto seja rotacionado durante o jogo, como também explicado na seção sobre representação. Por este motivo, foi escolhido o mapeamento da Figura 33(c), como pode ser visto na tela do jogo (Figura 32c).

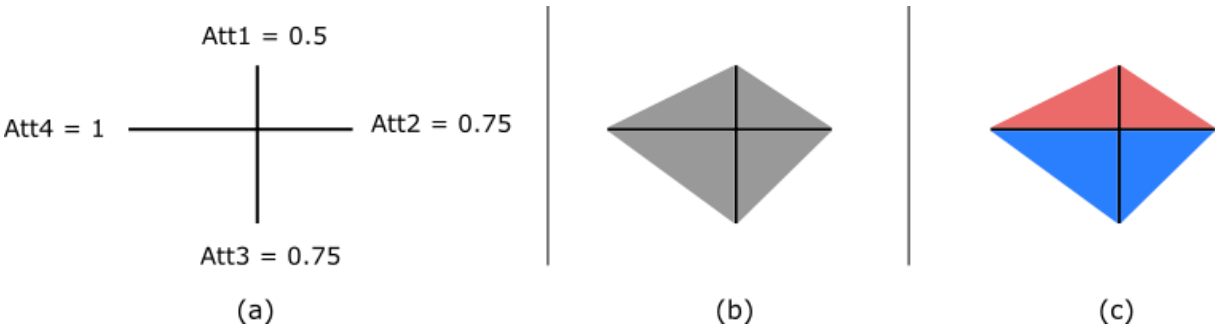


Figura 33 – Mapeamento adotado no jogo Missão E baseado em glifos estrelas

Experimentos e Considerações

O jogo mostrou bons resultados nos testes feitos com as bases de dados gerados *norm 1*, *norm 2* e *norm 3*. Como tratam-se de dados bi-dimensionais, cada atributo da observação foi mapeado em duas pontas das "estrela" que forma o objeto do jogo. O jogo conseguiu agrupar as três bases com resultados iguais aos melhores obtidos nos jogos anteriores, após atingir aproximadamente a mesma quantidade de ligações. Isto é 100% de acurácia e 100% de cobertura para as bases 1 e 3 e 98% de acurácia e 100% de cobertura para a base 2.

Entretanto, quando se tentou agrupar a base *Iris*, que possui exatos quatro atributos, notou-se um efeito inverso ao que aconteceu nos mapeamentos adotados pelos jogos *Bang Bang* e *Dangerous Mine*. Na base *Iris* existe ordem de relevância entre os atributos, ou seja, existem atributos que influenciam mais do que outros na hora de separar os dados. Quando os jogadores comparavam os quatro atributos de forma igual, devido ao mapeamento adotado, uma pequena diferença no atributo mais relevante era ignorada por uma diferença maior em um atributo de menor importância, gerando, com isso, um grande número de votos incorretos.

4.2.5 Os Esquils

Este jogo é simplesmente um novo tema para o jogo *Missão E*.

Para otimizar a qualidade dos votos gerados pelo jogo *Missão E* quando existia ordem de relevância entre os atributos, decidiu-se por alterar o mapeamento de forma que os atributos mais relevantes fossem mais notados. Executando o algoritmo de razão de ganho no *Weka*, notou-se que os atributos 3 e 4 da base *Iris* eram consideravelmente mais relevantes, para a tarefa de agrupamento, do que os atributos 1 e 2. De fato, agrupando com o algoritmo *K-Means* com todos os atributos obteve-se 7,3% menos acurácia do que agrupando com o mesmo algoritmo utilizando somente os atributos 3 e 4. Por este motivo foi decidido por mapear apenas estes dois últimos atributos.

Novo Mapeamento

O novo mapeamento foi feito baseado em um degradê de cores, para o atributo 4, mais relevante de todos, e forma, para o atributo 3. Este novo mapeamento gerou um novo tema para o jogo, que passou a se chamar *Os Esquils*, mostrado na Figura 34.

A nova história explica que *Esquils* são criaturas coloridas que possuem amigos e inimigos. O jogador inicia controlando um determinado *Esquil* (observação exemplo) e deve abraçar novos *Esquils* que forem seus amigos (parecidos) e explodir *Esquils* inimigos (distintos). O *gameplay*, as validações, a pontuação etc., continuaram exatamente iguais ao jogo anterior, sendo modificado apenas o tema para que o novo mapeamento fosse melhor incorporado.

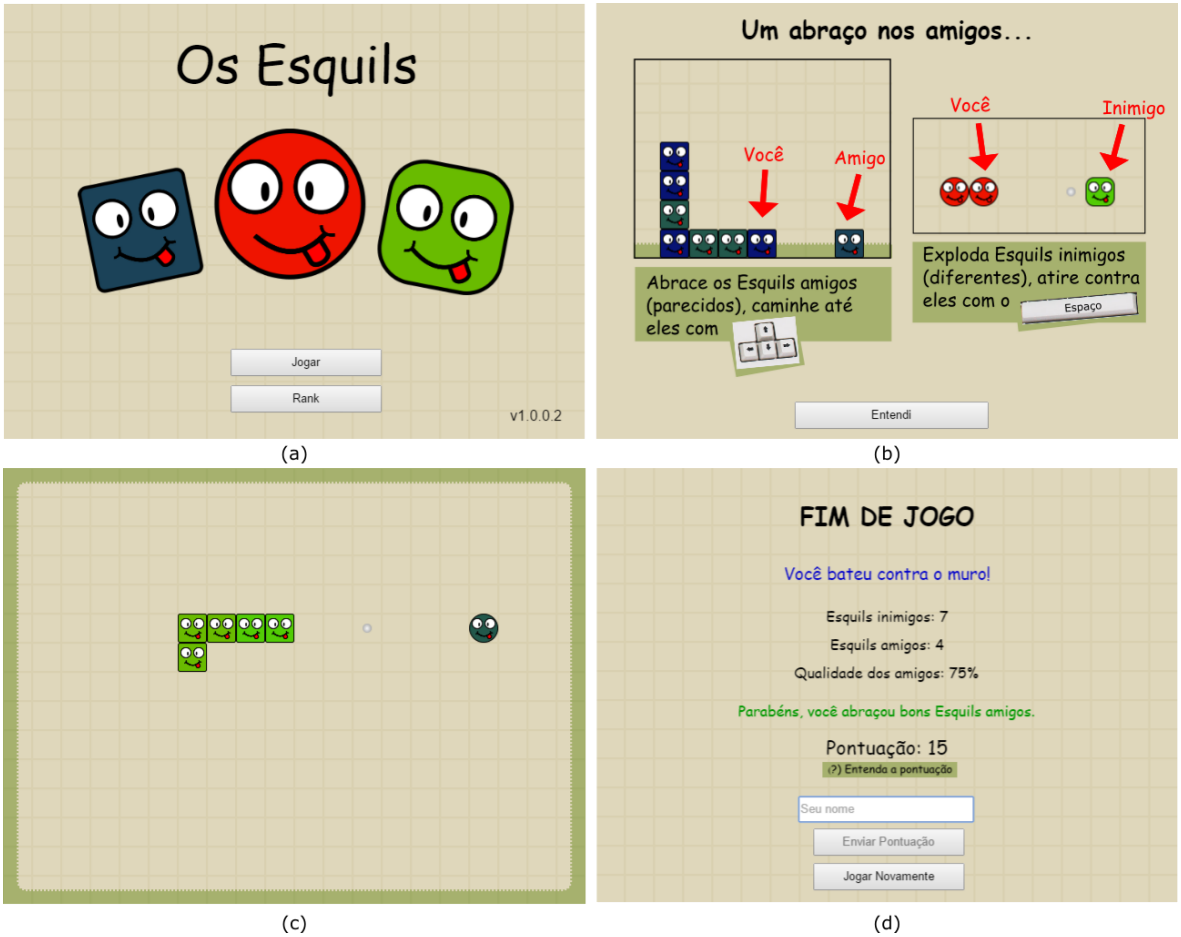


Figura 34 – Telas do jogo Os Esquils

Experimentos e Considerações

Com o novo mapeamento foi atingido o objetivo de agrupar a base *Iris* com 95,7% de acurácia e 100% de cobertura após 3886 ligações, sendo 559 conhecidas e 3327 estimadas. Resultado próximos aos 96% de acurácia atingido pelo *k*-means e aos 94,66% atingidos pelo *k*-NN, com k = 3.

Após atingir os 100% de cobertura na base *Iris*, o mesmo jogo e mapeamento foi utilizado para agrupar a base *Wine*. Como o mapeamento suporta apenas dois atributos, foi utilizado o procedimento PCA (*principal componente analysis*) para encontrar as duas primeiras componentes. O jogo foi executado poucas vezes durante o tempo em que esta base ficou disponível, dessa forma, foram calculadas apenas 1239 ligações e destas somente 34 foram determinadas como conhecidas. Utilizando estes dados, o algoritmo atingiu 87,1% de acurácia e 92,1% de cobertura, agrupando 164 das 178 observações. Resultado que pode ser considerado bom devido ao número de ligações, porém muito aquém quando comparado com o *k*-means, que obteve, nos dados transformados, 97,2% de acerto.

Capítulo 5

Conclusões e Trabalhos Futuros

A contribuição central deste trabalho foi propor um método, baseado na ideia de computação humana mascarada por jogos, para estimar similaridades entre quaisquer conjuntos de observações. O método foi concebido de maneira genérica e flexível para, quando aplicado, ser modelado de acordo com o problema. O método não consiste em um passo-a-passo rigoroso e sim em uma combinação de ideias que podem ser utilizadas e modificadas para se alcançar o objetivo desejado.

Outra contribuição relevante foi a definição da notação, em que o método se baseia, para formalizar a ideia de isolar os dados do problema do jogo que está resolvendo. De forma análoga, a notação é facilmente modificada para atender a outros problemas computacionais.

Ficou claro que um ponto crítico do método proposto é a representação dos dados, isto é, a transferência de informação da base de dados para o jogo. Notou-se que boas representações estão diretamente ligadas as boas estimativas de similaridades e vice-versa.

Foram feitas validações, baseadas nos conceitos de controle de qualidade, que devem ser feitas durante e após o jogo terminar, para que apenas as melhores jogadas sejam utilizadas para a resolução da tarefa em questão.

O algoritmo descrito para a realização de agrupamentos não está otimizado do ponto de vista computacional. Como ponto positivo, ele foi elaborado visando explorar ao máximo as diferenças, isto é, a ordem de importância, entre as similaridades conhecidas e estimadas. Sua concepção também se preocupou em utilizar única e exclusivamente as similaridades encontradas, não dependendo de nenhum outro conhecimento sobre as observações e nem que elas estejam alocadas em um espaço específico, como o espaço euclidiano.

A implementação adotada para testar os conceitos do método exigiu, além de um *back-end* para receber/enviar informações, o desenvolvimento de diversos jogos. Tal tarefa está sujeita ao surgimento de *bugs* que podem demorar a serem identificados, isto é, que só ocorrem em determinadas situações do jogo. Por este motivo, uma grande parte dos resultados obtidos durante os experimentos foram descartados após a percepção

destes erros. Assim, devido à falta de tempo e a reduzida quantidade de jogadores, os testes foram simplificados através da utilização de bases menores e mais facilmente agrupáveis.

Por esta razão, os experimentos serviram, fundamentalmente, para testar a plausibilidade do método. Os resultados obtidos não foram tão expressivos, sendo sempre parecidos com os algoritmos mais clássicos de agrupamento/classificação que, para as bases utilizadas nos testes, são muito mais rápidos e práticos de serem executados do que o método proposto. Por outro lado, como pôde ser visto na seção de experimentos, estes, foram cruciais para o aprimoramento das ideias propostas. Uma boa parte delas surgiram após encontradas falhas conceituais durante os experimentos.

Uma das principais propostas de trabalhos futuros é a realização de testes mais profundos e específicos. É interessante testar a escalabilidade através de bases maiores, por exemplo, detectando ruídos (intrusos) em tais bases. Pode-se também testar a qualidade dos resultados, realizando o mesmo teste várias vezes e analisando através de conceitos estatísticos de confiança. Dessa forma será possível provar, além da plausibilidade, também a robustez do método.

Como exposto, a representação dos dados é um ponto delicado do método, assim, uma outra contribuição importante seria explorar os conceitos psicológicos envolvidos na representação. Quais características são mais perceptíveis ao seres humanos durante o jogo e por quê? Qual o número máximo de características que conseguimos perceber e diferenciar submetidos a adrenalina do jogo? Como generalizar a representação e, consequentemente, o método, de forma a aceitar o maior número de bases possível?

Um outro caminho de trabalho futuro interessante é investigar novas formas de transferência de informações dos dados para os jogos, como no exemplo da Figura 5, onde os funcionários que recebem mais estão mais felizes, sem que os jogadores percebam o que os dados representam em si.

Junto a isto pode-se também pesquisar por novas formas de explorar o senso comum humano, como no exemplo da seção 3.3.4, onde quanto maior a moeda de ouro, intuitivamente mais vantajosa ela será. Esses são conceitos chaves na hora de diferenciar o processamento humano do processamento das máquinas. O método se mostrará realmente útil quando, ao contrário dos testes realizados neste trabalho, for utilizado em bases cujos agrupamentos encontrados por máquinas não sejam bons.

Um outro trabalho futuro possível consiste em encontrar ou calcular alguns limiares, como critérios de parada/aceitação. Algumas questões em aberto são: Qual a qualidade mínima dos votos que devem ser aceitos para garantir que o agrupamento será encontrado de forma correta? Quantas ligações conhecidas e estimadas são necessárias para atingir o melhor agrupamento possível?

Referências Bibliográficas

Von Ahn, L., (2005), *Human Computation*, Carnegie Mellon University, AAI3205378.

Von Ahn, L., (2006), "Games with a Purpose", *Computer*, v. 39, n. 6 (jun.), p. 92–94.

Von Ahn, L., Dabbish, L., (2004), "Labeling Images with a Computer Game". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, p. 319–326, New York, NY, USA.

Von Ahn, L., Liu, R., Blum, M., (2006), "Peekaboom: A Game for Locating Objects in Images". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, p. 55–64, New York, NY, USA.

Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M., (2008), "reCAPTCHA: Human-Based Character Recognition via Web Security Measures", *Science*, v. 321, n. 5895 (dez.), p. 1465–1468.

Ashby, F., Ennis, D., (2007), "Similarity measures", *Scholarpedia*, v. 2, n. 12, p. 4116.

Chambers, J. M., (1983), *Graphical methods for data analysis*. Wadsworth International Group.

Chan, W. W. Y., (2006), *A Survey on Multivariate Data Visualization*

Chernoff, H., (1973), "The Use of Faces to Represent Points in K-Dimensional Space Graphically", *Journal of the American Statistical Association*, v. 68, n. 342 (jun.), p. 361–368.

Clopper, C. J., Pearson, E. S., (1934), "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial", *Biometrika*, v. 26, n. 4, p. 404–413.

Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popović, Z., et al., (2010), "Predicting protein structures with a multiplayer online game", *Nature*, v. 466, n. 7307 (ago.), p. 756–760.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J., (2009), "Modeling wine preferences by data mining from physicochemical properties", *Decision Support Systems*, v. 47, n. 4 (nov.), p. 547–553.

Dillon, R., (2010), *On the Way to Fun: an emotion based approach to successful game design*. Natick, MA, USA, A K Peters.

Van Dongen, S., (2000), "A Cluster algorithm for graphs", *Report - Information systems*, n. 10, p. 1–40.

Fisher, R. A., (1936), "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, v. 7, n. 2 (set.), p. 179–188.

Gomes, C., Schneider, D., Moraes, K., de Souza, J., (2012), "Crowdsourcing for music: Survey and taxonomy". In: *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, p. 832–839

Han, J., Kamber, M., Pei, J., (2011), *Data Mining: Concepts and Techniques: Concepts and Techniques*. Edição: 3 ed. Morgan Kaufmann.

Healey, C. G., (2015). Perception in Visualization. Disponível em: http://www.csc.ncsu.edu/faculty/healey/PP/. Acesso em: 1 set 2015.

Healey, C. G., Booth, K. S., Enns, J. T., (1996), "High-speed Visual Estimation Using Preattentive Processing", *ACM Trans. Comput.-Hum. Interact.*, v. 3, n. 2 (jun.), p. 107–135.

Howe, J., (2006). Crowdsourcing: Crowdsourcing: A Definition. Disponível em: http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html. Acesso em: 31 ago 2015.

HRTALENT, (2014). DuoLingo's Crowdsourcing Business Model. *HRTALENT*. Disponível em: http://www.hrtalentmanagement.com/2014/02/13/duolingos-crowdsourcing-business-model/. Acesso em: 2 set 2015.

Jovian, L. T., Amprimo, O., (2011), "OCR Correction via Human Computational Game". In: *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, p. 1–10, Washington, DC, USA.

Kawrykow, A., Roumanis, G., Kam, A., Kwak, D., Leung, C., Wu, C., Zarour, E., Sarmenta, L., Blanchette, M., "Phylo: a citizen science approach for improving multiple sequence alignment."

Keim, D. A., Kriegel, H.-P., (1996), "Visualization techniques for mining large databases: a comparison", *IEEE Transactions on Knowledge and Data Engineering*, v. 8, n. 6 (dez.), p. 923–938.

KNIME, (2015). Distance Measure. Disponível em: http://tech.knime.org/wiki/distance-measure. Acesso em: 1 set 2015.

Kubrusly, C. S., (2001), *Elements of Operator Theory*. Birkhäuser, Boston.,

Lichman, M., (2013). UCI Machine Learning Repository. Disponível em: http://archive.ics.uci.edu/ml. Acesso em: 2 set 2015.

Lima, A. S., (2013), *Modelo de desenvolvimento de jogos com propósito baseado em mecânicas tradicionais de jogos.* Dissertação de Mestrado, Universidade Federal do Rio de Janeiro

Maimon, O. Z., Rokach, L., (2005), *Data Mining and Knowledge Discovery Handbook*. Springer Science & Business Media.

Quinn, A. J., Bederson, B. B., (2011), "Human Computation: A Survey and Taxonomy of a Growing Field". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, p. 1403–1412, New York, NY, USA.

Schaeffer, S. E., (2007), "Survey: Graph Clustering", *Comput. Sci. Rev.*, v. 1, n. 1 (ago.), p. 27–64.

Schuler, D., (1994), "Social Computing", *Commun. ACM*, v. 37, n. 1 (jan.), p. 28–29.

SIGMAZONE, (2015). Understanding Binomial Confidence Intervals. Disponível em: http://www.sigmazone.com/binomial_confidence_interval.htm. Acesso em: 1 set 2015.

Surowiecki, J., (2005), *The Wisdom of Crowds*. Knopf Doubleday Publishing Group.

Treisman, A., (1985), "Preattentive Processing in Vision", *Comput. Vision Graph. Image Process.*, v. 31, n. 2 (ago.), p. 156–177.