

INSTITUTO FEDERAL DE MINAS GERAIS
CENTRO DE TECNOLOGIA
PROGRAMA DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Guilherme Magno

**VISUALIZAÇÃO E PREDIÇÃO DO COMPORTAMENTO DE
PARTIDA DO JOGO LEAGUE OF LEGENDS UTILIZANDO
GRAFOS.**

BambuÍ, MG
2018

Guilherme Magno

**VISUALIZAÇÃO E PREDIÇÃO DO COMPORTAMENTO DE PARTIDA DO JOGO
LEAGUE OF LEGENDS UTILIZANDO GRAFOS.**

Trabalho de Conclusão de Curso apresentado
ao Bacharelado em Engenharia de Computação
do Instituto Federal de Minas Gerais (IFMG),
como requisito parcial para a obtenção do grau
de **Bacharel em Engenharia de Computação**

Orientador: Prof. Dr. Laerte Mateus

BambuÍ, MG

2018

Magno, Guilherme

Visualização e predição do comportamento de partida do jogo League of Legends utilizando grafos. / por Guilherme Magno. – 2018.
38 f.: il.; 30 cm.

Orientador: Nome do orientador TEM Q CONFERIR ISSO
Trabalho de Conclusão de Curso - Instituto Federal de Minas Gerais, Centro de Tecnologia, Bacharelado em Engenharia de Computação, RS, 2018.

1. Modelo. 2. Latex. 3. Tcc. 4. Graduação. I. TEM Q CONFERIR ISSO, Nome do orientador. II. Visualização e predição do comportamento de partida do jogo League of Legends utilizando grafos. .

Guilherme Magno

**VISUALIZAÇÃO E PREDIÇÃO DO COMPORTAMENTO DE PARTIDA DO JOGO
LEAGUE OF LEGENDS UTILIZANDO GRAFOS.**

Trabalho de Conclusão de Curso apresentado
ao Bacharelado em Engenharia de Computação
do Instituto Federal de Minas Gerais (IFMG),
como requisito parcial para a obtenção do grau
de **Bacharel em Engenharia de Computação**

Aprovado em dia de mês de 2018:

Nome do orientador TEM Q CONFERIR ISSO, Dr. (UFSM)
(Presidente/Orientador)

Nome membro banca Sobre nome, Me. (UFSM)

Nome membro banca Sobre nome, Tecg. (UFSM)

BambuÍ, MG

2018

DEDICATÓRIA

Dedico este trabalho a

AGRADECIMENTOS

Agradeço

“O valor de uma coisa depende da maneira como a abordamos mentalmente e não da coisa em si”

(JIGORO KANO)

RESUMO

VISUALIZAÇÃO E PREDIÇÃO DO COMPORTAMENTO DE PARTIDA DO JOGO LEAGUE OF LEGENDS UTILIZANDO GRAFOS.

AUTOR: GUILHERME MAGNO

ORIENTADOR: NOME DO ORIENTADOR TEM Q CONFERIR ISSO

Aqui você escreve o resumo. Lembrando no máximo 250 palavras para tcc e 500 palavras para tese ou dissertação.

Palavras-chave: Modelo. latex. tcc. graduação.

ABSTRACT

ABSTRACT TITLE

AUTHOR: GUILHERME MAGNO

ADVISOR: NOME DO ORIENTADOR TEM Q CONFERIR ISSO

Here you write the summary. Remembering a maximum of 250 words for tcc and 500 words for thesis or dissertation.

Keywords: Model. latex. tcc. graduation.

LISTA DE FIGURAS

1	Exemplo de grafos.....	15
	(a) Grafo binário com quatro vértices e quatro arestas.....	15
	(b) Grafo não-binário com quatro vértices e três arestas.	15
2	Exemplo de vizinhança de um vértice de um grafo.	16
3	Variação do p sem alterar o numero de vértices $v = 20$ e $k = 4$	17
4	Lei de potência do número de conexões k . De cima pra baixo, as curvas de lei de potência foram obtidas com o parâmetro γ definido como: 0.5 , 1 , 1.5 , 2, 2.5, 3, 4 e 5.	19
5	Coeficiente de aglomeração.	20
	(a)	20
	(b)	20
6	Mapa do jogo de <i>League of Legends</i>	21
7	Exemplo de visualização de dados utilizando o D3.js.	22
8	Exemplo de retorno do uso da função <i>MATCH-V3</i> . Sendo as informações divididas em informações da partida, informações dos times e informações dos participantes.....	24
9	Esquema usado do Banco de Dados.	26
10	Exemplo de uso do <i>web service</i> para visão geral dos dados.	29
	(a) Filtro dos dados para campeões da mesma equipe.	29
	(b) Filtro dos dados para adversários.	29
	(c) Filtro para escolher quais campeões podem participar (<i>picks</i>) e quais nao podem (<i>bans</i>).....	29
	(d) Exemplo de grafo exibido pelo <i>web service</i> , sendo as setas vermelhas mostrando quem é bom contra quem, e as linhas continuas quem é bom com quem. O tamanho do circulo é a frequência daquela ligação e a cor é a força da ligação.	29
11	Exemplo de uso do <i>web service</i> predição da partida	31
	(a) Esperando ser feito	31
	(b) Esperando ser feito	31
	(c) Esperando ser feito	31
	(d) Esperando ser feito	31

LISTA DE TABELAS

1	Exemplo de <i>subset</i> salvo no banco de dados	27
---	--	----

LISTA DE ABREVIATURAS E SIGLAS

MOBA	Arena de batalha online de multijogadores (do inglês <i>Multiplayer Online Battle Arena</i>)
API	Interface de Programação de Aplicativos (do inglês <i>Application Programming Interface</i>)
JSON	Notação de Objeto JavaScript (do inglês <i>JavaScript Object Notation</i>)
URL	Localizador Uniforme de Recursos (do inglês <i>Uniform Resource Locator</i>)
LOL	Liga das Lendas (do inglês <i>League of Legends</i>)

SUMÁRIO

1	INTRODUÇÃO	13
1.1	OBJETIVOS	13
1.1.1	Objetivos Gerais	13
1.1.2	Objetivos Específicos e Resultados Esperados	13
1.2	JUSTIFICATIVA	14
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	GRAFOS	15
2.2	PROPRIEDADE TOPOLÓGICAS	16
2.3	REDES COMPLEXAS	16
2.3.1	<i>Redes Small-Worlds</i>	17
2.3.2	Redes Livres de Escala	18
2.3.3	Redes Aleatórias	18
2.3.4	Coeficiente de Aglomeração	18
2.4	<i>LEAGUE OF LEGENDS</i>	19
2.5	D3.JS	22
2.6	ESTADO DA ARTE	22
3	METODOLOGIA	23
3.1	API	23
3.2	PERSISTÊNCIA DOS DADOS	25
3.3	AQUISIÇÃO DOS DADOS	26
3.4	VISUALIZAÇÃO DOS DADOS	28
3.5	CLASSIFICAÇÃO DA TOPOLOGIA	29
3.6	PREDIÇÃO DOS DADOS	29
4	CONCLUSÃO	31
	REFERÊNCIAS	33
	APÊNDICES	35
	ANEXOS	37

1 INTRODUÇÃO

Atualmente muitos jogos tem conquistado a atenção e o tempo de muitas pessoas, com o surgimento da nova categoria, chamada *e-sports*, muitas pessoas têm dedicado o seu tempo aos jogos como forma de trabalho, já que a premiação só vem aumentando com o passar dos anos.

Premiação essa que, segundo uma pesquisa realizada pela ESPN (2017), teve um total no Mundial de *League of Legends* de 2017 ultrapassou 4 milhões de dólares. Mostrando que os jogos já estão tendo premiações consideráveis.

E como todos os jogos que existem equipes adversárias, no *League of Legends*, existem vencedores e perdedores. E este trabalho vem com a intenção de prever os vencedores de uma partida desse jogo utilizando grafos.

Quando descrevemos uma situação utilizando pontos e algumas ligações entre esses pontos, estamos modelando este cenário em forma de grafos. E quando algo está modelado em forma de grafos, podemos usar tecnologias que são aplicáveis em grafos.

1.1 OBJETIVOS

Os objetivos desse trabalho podem ser divididos em objetivos gerais e específicos, que podem ser encontrados nas próximas subseções, junto também com os resultados esperados.

1.1.1 Objetivos Gerais

Criar um modelo computacional capaz de exibir resumos dos dados armazenados e prever um vencedor no jogo *League of Legends* baseando-se num histórico de partidas usando redes complexas.

1.1.2 Objetivos Específicos e Resultados Esperados

1. Construir um *webservice* para visualização;
2. Calcular deterministicamente quantas vezes cada herói ganhou contra/com outro herói;
3. Testar o resultado obtido em outra amostra de dados;
4. Documentar os resultados.

1.2 JUSTIFICATIVA

Classificar uma rede complexa é de grande importância, (?) diz

2 FUNDAMENTAÇÃO TEÓRICA

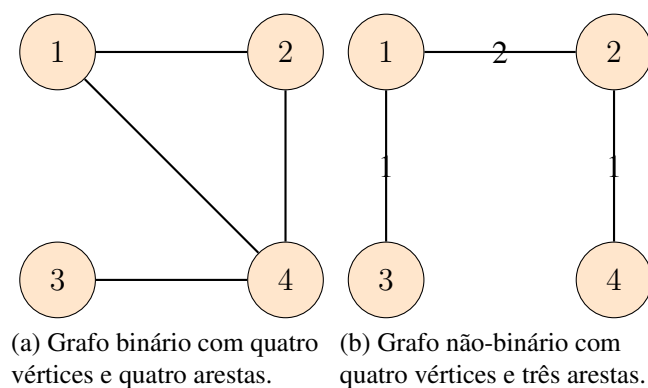
Nessa parte será apresentados conceitos para melhor entendimento deste trabalho e da modelagem utilizada. A próxima seção elucida o que são grafos e depois é explicado a variação de grafo utilizado para modelar o problema proposto. Logo em seguida, é esclarecido o ambiente que será abstraído, explicando algumas regras básicas do jogo.

2.1 GRAFOS

Quando descrevemos uma situação utilizando pontos e ligações entre algum desses pontos, a abstração matemática desse tipo dá lugar ao conceito de grafo (LUCCHESI, 1979). Um grafo $G(V, E)$ pode ser definido como um conjunto de vértices V e seu conjunto de arestas E sendo que cada elemento $e \in E$, associa dois vértices do conjunto V . Assim, se $(u, v) \in E$, então os vértices u e v são conectados pela aresta e . Referindo-se a eles como vértices vizinhos ou adjacentes (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011; VIANA, 2007).

Os grafos também podem ter suas arestas classificadas como binárias e ponderados, sendo os binários suas arestas representadas apenas por 1 (se existem) ou 0 (se não existem) e os ponderados quando existe um conjunto W onde informa a intensidade da interação de uma aresta entre dois vértices (VIANA, 2007; LEISERSON et al., 2002). Na Figura 1a temos um grafo binário onde se $(u, v) \in E$ então ela está representada no grafo, já na Figura 1b além da existência da conexão entre os vértices, está representado o conjunto W informando o peso daquela aresta e .

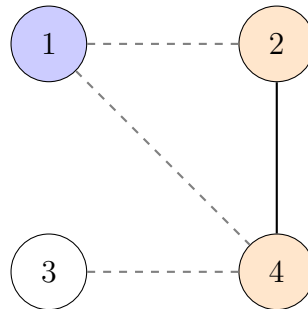
Figura 1 – Exemplo de grafos.



Fonte: Autor.

FEOFILOFF; KOHAYAKAWA; WAKABAYASHI (2011) afirma sobre a vizinhança de um grafo que "O conjunto de vértices X de um grafo G é o conjunto de todos os vértices que tem algum vizinho em X ", e diz que esse conjunto de vértices pode ser chamado de $\Gamma_G(X)$. Exemplificando, o conjunto $\Gamma_G(1)$ da Figura 1a são os vértices 2 e 4 e a aresta $(2, 4)$ sendo representados na Figura 2.

Figura 2 – Exemplo de vizinhança de um vértice de um grafo.



Fonte: Autor.

2.2 PROPRIEDADE TOPOLÓGICAS

O grau de um vértice v é definido sendo a quantidade de arestas que chegam no vértice v , sendo denotado por $g(v)$. Continuando ainda no exemplo da Figura 1a o $g(1)$ é 2 pois apenas as arestas $(1, 2)$ e $(1, 4)$ incidem no vértice 1 (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011).

O grau máximo de um grafo G é o grau do vértice que tem o maior grau presente nesse grafo $V(G)$, ou seja $\Delta(G) = \max\{g(v) : v \in V(G)\}$. E o grau mínimo de um grafo G é o grau do vértice com menor grau, $\delta(G) = \min\{g(v) : v \in V(G)\}$. E ainda, um grafo é regular se $\delta(G) = \Delta(G)$ e é k -regular se $\delta(G) = \Delta(G) = k$ (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011).

Quando modelamos um sistema real em forma de grafos vamos ver na seção seguinte que esse grafo recebe um nome especial, sendo chamado de rede complexa.

2.3 REDES COMPLEXAS

Ainda sobre grafos VIANA (2007) afirma que é utilizado o termo redes complexas quando um grafo representa um sistema físico real, então um grafo do jogo LOL pode ser considerado como uma rede complexa.

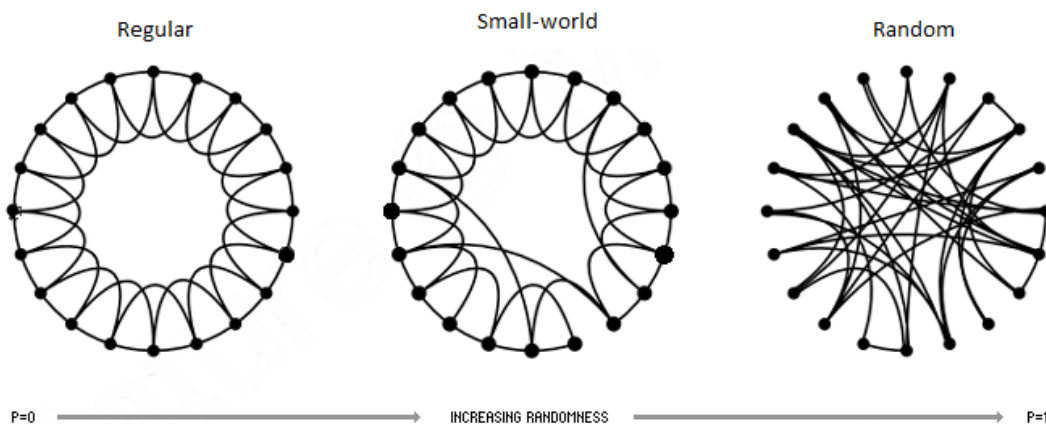
Para que seja possível classificar o nosso modelo adequadamente, serão apresentados três modelos que se destacam entre os pesquisadores segundo ALBERT; BARABÁSI (2002): As redes *small-worlds*, as redes livres de escala e as redes aleatórias. E também será explicado sobre o coeficiente de aglomeração para melhor caracterização do modelo adquirido.

2.3.1 Redes Small-Worlds

As redes *small-worlds* são um modelo de rede complexas e este tipo de rede complexa representa uma alternativa ao modelo aleatório, assumindo como hipótese que redes complexas do mundo real presente no mundo animal, biológico e químico não são completamente aleatórias (WATTS; STROGATZ, 1998; LOPES, 2011).

Tendo um grafo regular com v vértices e k arestas ligando os vizinhos mais próximos, depois de formado o grafo cada aresta tem uma probabilidade p de se reconectar com outro vértice aleatório. Permitindo que a geração do grafo seja controlado para uma rede regular com $p \approx 0$ ou uma rede aleatória com $p \approx 1$, e também permitindo uma rede com topologia intermediária $0 < p < 1$ (LOPES, 2011).

Figura 3 – Variação do p sem alterar o numero de vértices $v = 20$ e $k = 4$.



Fonte: WATTS; STROGATZ (1998).

As redes *small-worlds* são caracterizadas com duas principais medidas: O tamanho do caminho $L(p)$ e o coeficiente de agrupamento ou coeficiente de aglomeração $C(p)$. $L(p)$ é medido como a média do caminho mais curto de todos pares de vértices (LOPES, 2011). O coeficiente de aglomeração será melhor definido na seção 2.3.4.

2.3.2 Redes Livres de Escala

As redes livres de escala, foram propostas por BARABÁSI; ALBERT (1999), nelas um nó tem a probabilidade $P(k)$ de possuir k arestas obedecendo a lei da potência $P(k) \sim k^{-\gamma}$, onde γ determina o decaimento exponencial (ANTIQUERA et al., 2005; LOPES, 2011). Segundo VIANA (2007) nas redes livres de escala muitos nós tem poucas arestas e poucos nós se ligam a muitos.

A rede livre de escala é baseado em duas regras: crescimento e preferência linear de ligação. A rede é gerada com a inclusão de $n_0 < n$ vértices aleatoriamente conectados sendo estas inserções antes do crescimento da rede. Na etapa de crescimento da rede, a cada instante de tempo t um novo vértice v é adicionado na rede, o número de arestas desse vértice segue uma preferência linear de ligação da fórmula:

$$P(v_i \leftrightarrow v_j) = \frac{k_j}{\sum_u k_u}, \forall v_u \in V$$

Sendo P a probabilidade de um vértice v_i ser conectado ao novo vértice v_j é linearmente proporcional ao grau k_j do vértice v_j LOPES (2011).

É possível ver na Figura 4 que quanto maior o γ menos existem vértices com maior número de arestas. O paradigma de ligação dos novos vértices também é conhecido como "o rico fica mais rico" (COSTA et al., 2007).

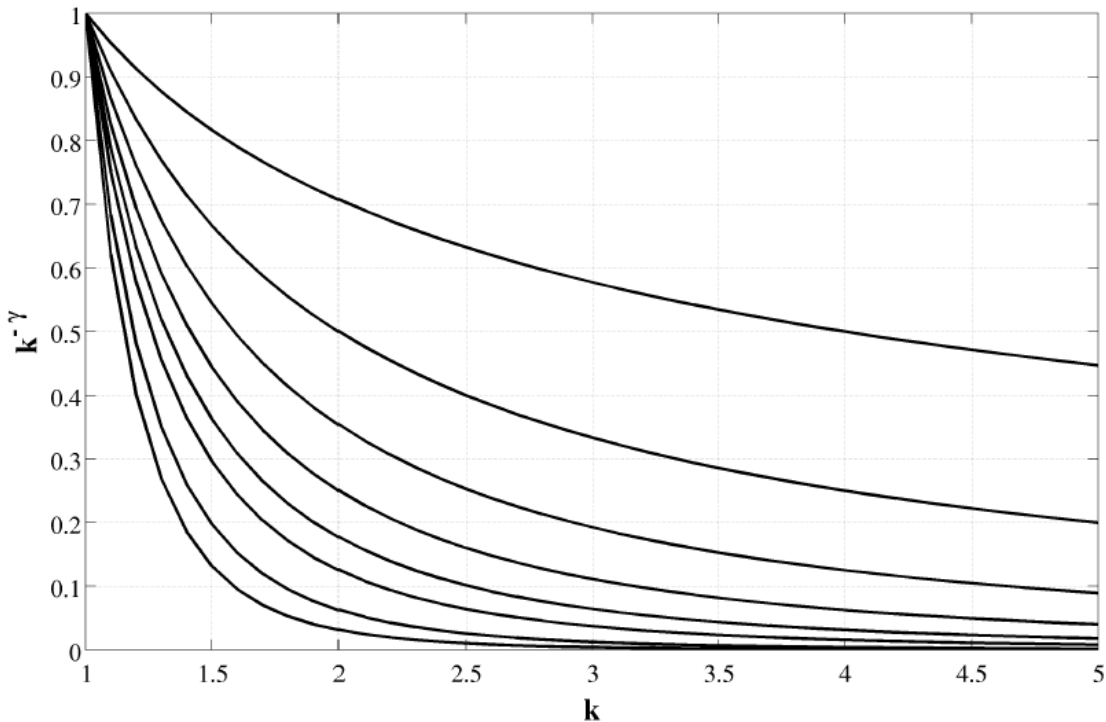
2.3.3 Redes Aleatórias

As redes aleatórias, foram inicialmente propostas em 1959 e podem ser consideradas o modelo mais elementar de rede complexas (LOPES, 2011; ERDDS; R&WI, 1959). Segundo VIANA (2007) e LOPES (2011) as redes aleatórias são um sistema formado por E arestas e N vértices, inicialmente são dispostos os vértices, e depois aleatoriamente as arestas são distribuídas com probabilidade igual entre os vértices, evitando auto-relacionamentos e conexões múltiplas entre dois vértices.

2.3.4 Coeficiente de Aglomeração

Segundo VIANA (2007) o coeficiente de aglomeração mede o quão conectado estão os nós da rede ou do grafo. ANTIQUERA et al. (2005) define o coeficiente de aglomeração

Figura 4 – Lei de potência do número de conexões k . De cima pra baixo, as curvas de lei de potência foram obtidas com o parâmetro γ definido como: 0.5 , 1 , 1.5 , 2, 2.5, 3, 4 e 5.



Fonte: LOPES (2011).

sendo:

$$CA_i = \frac{E_i}{k_i(k_i - 1)}$$

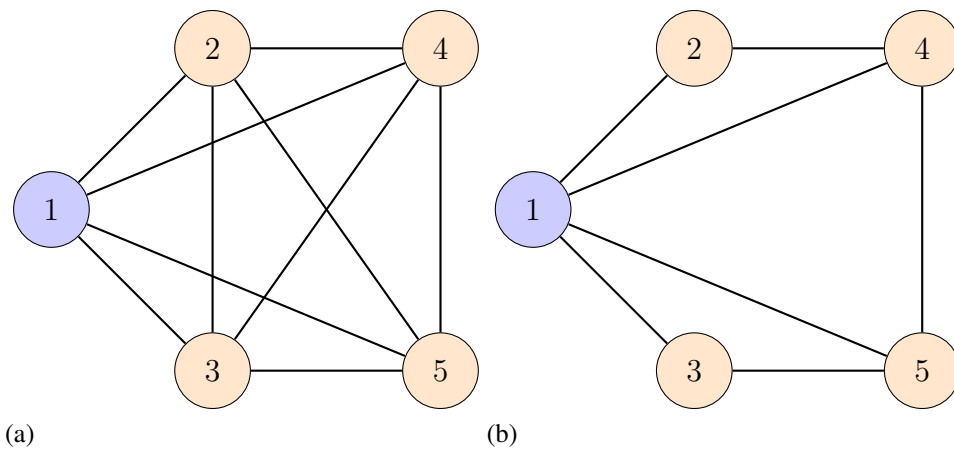
ANTIQUEIRA et al. (2005) continua: “Sendo para cada vértice i existe k_i arestas, que os ligam a outros k_i vértices. Se esses k_i vértices estivessem ligados diretamente à todos os outros vértices do conjunto, haveriam $k_i(k_i - 1)$ arestas entre eles. E assumindo E_i o número de arestas que existentes entre os k_i vértices. O coeficiente da rede inteira é a média de todos CA_i ”.

Na Figura 5a, assumindo o peso de todas as arestas como 1, o coeficiente de aglomeração do vértice em azul é 1, já na Figura 5b o coeficiente do vértice em azul segundo a fórmula é $\frac{1}{2}$.

2.4 LEAGUE OF LEGENDS

O jogo *League of Legends* é um jogo classificado como arena de batalha online de multi jogadores (do inglês *Multiplayer Online Battle Arena*) ou conhecido também como MOBA, que é um estilo de jogo onde duas equipes se enfrentam em um campo de batalha e cada joga-

Figura 5 – Coeficiente de aglomeração.



Fonte: Autor.

dor controla o seu personagem, mais chamado de herói ou campeão. O objetivo do MOBA é derrotar a equipe adversária destruindo a construção principal da equipe inimiga.

A arena onde acontece o jogo é uma arena onde normalmente o mapa inicialmente é espelhado, ou seja, o lado que cada time está não oferece vantagens exclusivas. O mapa é composto de três caminhos até a base inimiga.

Figura 6 – Mapa do jogo de *League of Legends*



Fonte: Autor (2018).

A Figura 6 mostra como é realmente o mapa do jogo, sendo que os círculos mostram a localização da construção principal, e os triângulos as construções de suporte de cada equipe, sendo azul uma equipe e vermelho a outra.

Com o início do jogo cada jogador escolhe um herói diferente, onde cada herói tem um conjunto de características únicas, como habilidades especiais, seu impacto no jogo, na equipe adversária e na equipe aliada.

Depois de escolher os heróis de cada equipe, cada jogador deve procurar adquirir recursos no jogo e objetivos para conseguir vantagens. Os recursos são limitados por equipe e por tempo, ou seja, deve ser bem escolhido quem ficará com a maior parte dos recursos da equipe.

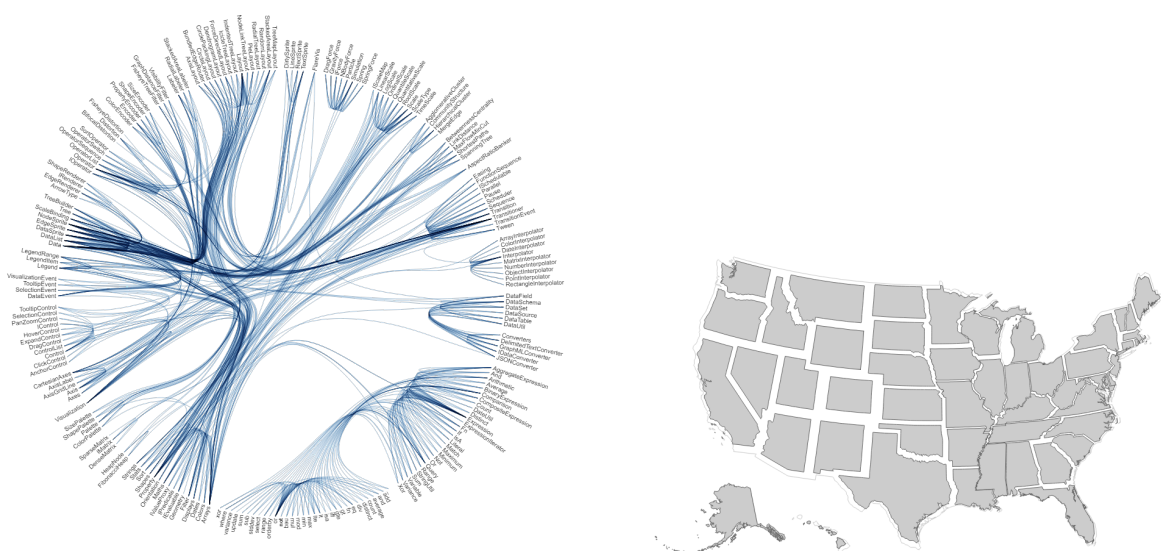
2.5 D3.JS

A biblioteca D3.js é uma biblioteca em JavaScript especializada em dar vida aos dados. Segundo ZHU (2013, tradução do autor) "Em certo sentido, o D3 é uma biblioteca JavaScript especializada que permite criar incríveis visualizações de dados usando uma abordagem mais simples (baseada em dados), aproveitando os padrões da *Web* existentes", e a organização oficial continua :

O D3.js é uma biblioteca em JavaScript para manipulação de documentos por dados. D3 te ajuda a trazer vida para os dados utilizando HTML, SVG e CSS. A ênfase da biblioteca D3 nos padrões da *web* dá-lhe todos os recursos dos navegadores modernos sem te prender a um *framework* proprietário, combinando poderosos componentes de visualização e uma aproximação orientada por dados da manipulação do DOM. (BOSTOCK, 2016, tradução do autor)

Com essa biblioteca é possível dar vida à informações e dados, como por exemplo a Figura 7 que possui duas das diversas maneiras de se exibir informações com o D3.js . E com o D3 foi gerado o grafo do *web service*, programa este que será clarificado na seção seguinte.

Figura 7 – Exemplo de visualização de dados utilizando o D3.js.



Fonte: (BOSTOCK, 2016).

2.6 ESTADO DA ARTE

VOU REFAZER

3 METODOLOGIA

Este capítulo apresenta as ferramentas e algoritmos utilizados neste projeto tanto no processo de extração do conhecimento, quanto na visualização dos dados. A primeira seção irá explicar como os dados estão disponíveis na API da Riot Games e de que informações eles carregam. A segunda clarifica como estes dados serão armazenadas. A terceira fala de que maneira se processa os dados, verifica se os dados estão completos e armazena-os em um banco de dados.

3.1 API

A Riot Games, desenvolvedora e dono do jogo *League of Legends*, fornece uma Interface de Programação de Aplicativos (do inglês *Application Programming Interface*) ou chamado de API, para que terceiros consigam acessar dados sobre os jogos.

A Riot games disponibiliza o acesso as informações à partir da URL¹ de modo que é gerado uma chave válida por 1 ano para projetos cadastrados. O acesso à essa API é por URL utilizando uma função disponível pela Riot Games. Neste trabalho, usaremos apenas a função *MATCH-V3* que é uma função que retorna os dados de uma partida já terminada.

A função *MATCH-V3* é disponibilizada publicamente pela Riot Games, retornando um conjunto de informações no formato JSON sobre a partida passada por parâmetro, quando essa partida existe. A Figura 8 resume um exemplo de uso da função acessando a URL², acessada em 21 de março de 2018 pelo autor, sendo que *minhachave* tem que ser substituída por uma chave privada válida.

¹ <http://developer.riotgames.com>

² https://br1.api.riotgames.com/lol/match/v3/matches/1381102031?api_key=minhachave

Figura 8 – Exemplo de retorno do uso da função *MATCH-V3*. Sendo as informações divididas em informações da partida, informações dos times e informações dos participantes.

```

"gameId":1381102031,
"platformId":"BR1",
"gameCreation":1526653171409,
"gameDuration":1533,
"queueId":420,
"mapId":11,
"seasonId":11,
"gameVersion":"8.10.229.7328",
"gameMode":"CLASSIC",
"gameType":"MATCHED_GAME",
"teams":[
  {
    "teamId":100,
    "win":"Win",
    "firstBlood":true,
    "firstTower":true,
    "firstInhibitor":true,
    "firstBaron":true,
    "firstDragon":true,
    "firstRiftHerald":true,
    "towerKills":10,
    "inhibitorKills":2,
    "baronKills":1,
    "dragonKills":2,
    "vilemawKills":0,
    "riftHeraldKills":1,
    "dominionVictoryScore":0,
    "bans":[]
  },
  {
  },
],
"participants":[
  {
    "participantId":1,
    "teamId":100,
    "championId":420,
    "spell1Id":12,
    "spell2Id":4,
    "highestAchievedSeasonTier":"PLATINUM",
    "stats":{
    },
    "timeline":{
    },
  },
  {
  },

```

De todas essas informações retornadas pela função, as que foram armazenadas para o projeto por participante são:

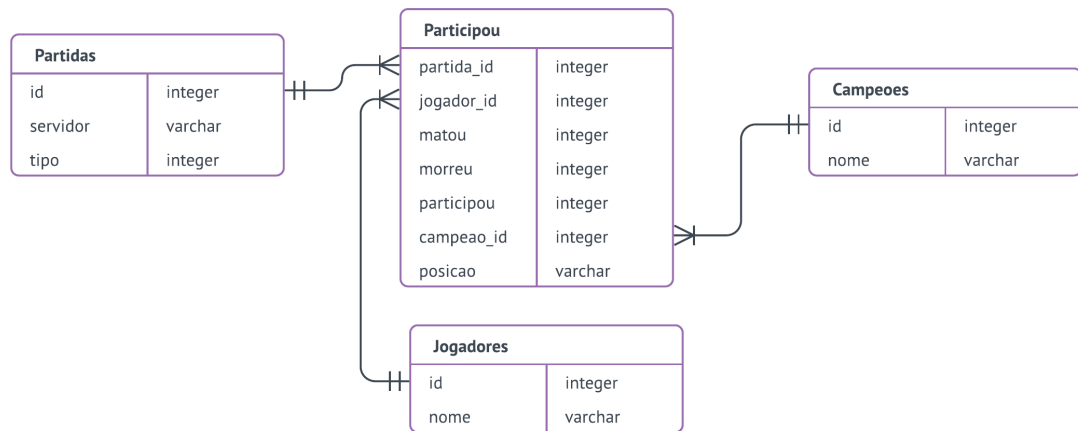
1. *gameId*. Identificador único da partida;
2. *kills*, *deaths* e *assists*. Informações que falam, respectivamente, quantas vezes esse jogador matou, morreu e participou na morte de outrem;
3. *win*. Se ele ganhou;
4. *championId*. Qual campeão ele escolheu;
5. *lane*. Em qual posição ele tava jogando;
6. *platformId*. Em qual servidor ele jogava;
7. *queueId*. E qual o tipo de partida.

Sabendo qual informação vamos armazenar, ficou escolhido a linguagem *Python* para ser desenvolvido o algoritmo no qual obterá as informações de forma não manual e o sistema de gerenciamento de banco de dados MySQL. Na próxima seção será apresentado sobre banco de dados e na seção ?? será esclarecido como foram obtidos o *dataset* das partidas.

3.2 PERSISTÊNCIA DOS DADOS

Os dados adquiridos serão armazenados em um banco de dados MySQL, cujo o esquema das tabelas podem ser vistas na Figura 9 onde é possível ver quais foram as informações salvas. Com o esquema do banco de dados pronto, já é possível passar para a aquisição dos dados.

Figura 9 – Esquema usado do Banco de Dados.



Fonte: Autor.

3.3 AQUISIÇÃO DOS DADOS

A aquisição dos dados será dada à partir de duas fontes: A API da Riot Games e de um *dataset* público³. Ambos apresentam os mesmos conjuntos de informações em formatos diferentes. Será adquirido outros dados além destes que serão adquiridos pela API para agilização do trabalho.

A base de dados, na sua versão 9, conta com aproximadamente 183 mil partidas ranqueadas diferentes, dando uma incrementada no banco de dados inicialmente. Dados estes que precisam ser convertidos para o esquema do banco de dados usados no trabalho.

Depois, será usado um algoritmo para a aquisição dos dados de partidas pela API do *League of Legends* em *Python*, que usa a biblioteca *requests* para conseguir acessar a API por URL's, verificando se a partida requisitada existe e se os dados necessários estão completos, salvando no banco de dados apenas as partidas válidas. O pseudo-código se encontra no Algoritmo 1.

³ <https://www.kaggle.com/paololol/league-of-legends-ranked-matches>

Algoritmo 1: AQUISIÇÃO DOS DADOS DAS PARTIDAS

```

início
  i = 1000000000; // Esta é a id de uma partida aleatória na
    ultima versão do jogo
  chave = minha chave privada de acesso;
  se Existe arquivo com ultimo partida lido então
    | i = ultima partida lida;
  repita
    pedido = requisita URL Da API(i, chave);
    se pedido.status == 200 então ( // Se a partida existe
      | JSON = carrega JSON Do Pedido(pedido);
      | se JSON é válido então
        | Armazena os dados em um banco de dados;
        | i = i + 1;
      | fim
    até Até ser pausado;
    Salva i no arquivo;
  fim
fim

```

Onde com esse algoritmo foram armazenadas 28597 jogos válidos diferentes, e foi decidido usar apenas as partidas ranqueadas 5 contra 5, já que estas são as partidas responsáveis pela classificação dos jogadores dentro dos jogos, e as partidas não ranqueadas são consideradas como amistosas. Com esse escopo, o número de partidas usados foram ao todo 11717 partidas ranqueadas diferentes. Na tabela 1 é possível ver uma pequena fatia dos dados salvos.

Tabela 1 – Exemplo de *subset* salvo no banco de dados

match_id	kills	deaths	assists	win	champ_id	lane	player_id	platform	type
1282000002	11	10	7	0	67	BOTTOM	16112724	BR1	420
1282000002	2	8	15	0	412	BOTTOM	18604874	BR1	420
1282000002	7	4	7	0	34	MIDDLE	19281809	BR1	420
1282000002	7	6	4	0	5	JUNGLE	21304223	BR1	420
1282000002	5	7	12	0	98	TOP	651014	BR1	420
1282000002	0	6	23	1	44	BOTTOM	18592234	BR1	420
1282000002	19	5	5	1	222	BOTTOM	7170345	BR1	420

Fonte: Autor.

Na etapa seguinte são contabilizados quantas vezes cada campeão jogou com outro campeão, seja no mesmo time ou no time adversário, calculando quantas vitórias em oposição e quantas vitórias em junção. E com as informações já processadas foi montado um *web service*, que será explicado na seção 3.4, utilizando uma biblioteca chamada D3.js para uma melhor visualização dos dados obtidos, biblioteca essa que será explanado na próxima seção.

3.4 VISUALIZAÇÃO DOS DADOS

O *web service* foi criado utilizando o *framework* Flask, que como RONACHER (2010, tradução do autor) diz "Flask é um micro *framework* para Python baseado em Werkzeug, Jinja 2 e em boas intenções."

Com esse serviço será possível um relatório geral onde o usuário será capaz de ver a topologia do modelo, vendo quais são os campeões melhores contra os outros e quais são as duplas mais favoráveis, podendo filtrar quais campeões podem participar da pesquisa e quais não podem. Também será exequível a predição de uma partida, onde se seleciona os campeões de cada equipe e o sistema tenta antever quem vencerá.

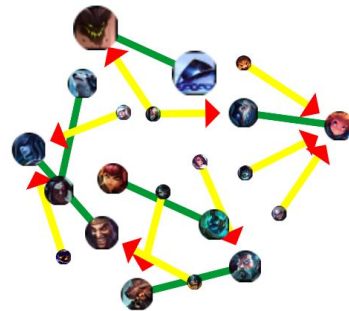
Na Figura 10 é possível ver um exemplo de uso do serviço para uma visão geral das informações. Na seção 3.6 será esclarecido a maneira que essa predição da partida é feita.

Figura 10 – Exemplo de uso do *web service* para visão geral dos dados.

Mesmo Time	Times diferentes
Número Mínimo de partidas	Número Mínimo de partidas
Número Máximo de partidas	Número Máximo de partidas
Mínimo de % de vitória	Mínimo de % de vitória

(a) Filtro dos dados para campeões da mesma equipe. (b) Filtro dos dados para adversários.

Bans	<input type="text" value="MonkeyKing"/> <input type="text" value="Jax"/> <input type="text" value="Fiddlesticks"/> <input type="text" value="Shaco"/> <input type="text" value="Wendell"/>
Picks	<input type="text" value="MonkeyKing"/> <input type="text" value="Jax"/> <input type="text" value="Fiddlesticks"/> <input type="text" value="Shaco"/> <input type="text" value="Wendell"/>
<input type="button" value="Search"/>	



(c) Filtro para escolher quais campeões podem participar (*picks*) e quais não podem (*bans*). (d) Exemplo de grafo exibido pelo *web service*, sendo as setas vermelhas mostrando quem é bom contra quem, e as linhas contínuas quem é bom com quem. O tamanho do círculo é a frequência daquela ligação e a cor é a força da ligação.

Fonte: Autor.

3.5 CLASSIFICAÇÃO DA TOPOLOGIA

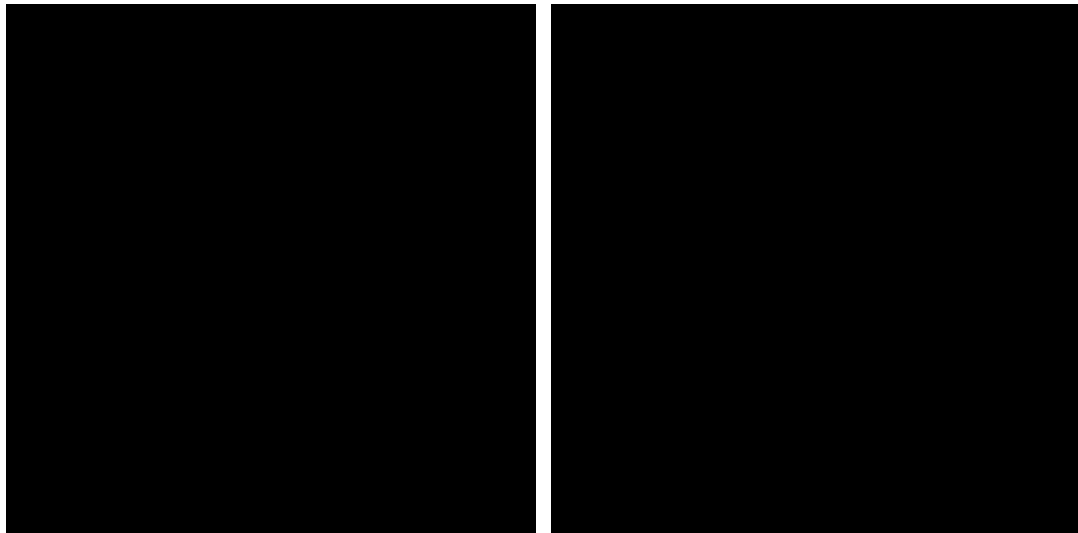
Esperando o artigo.

3.6 PREDIÇÃO DOS DADOS

Esperando as referencias.

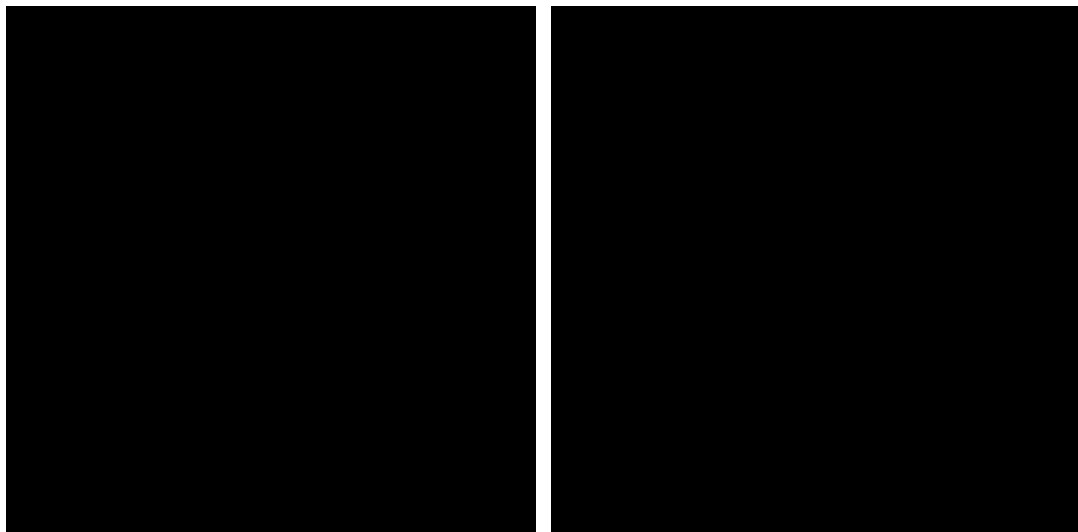
4 CONCLUSÃO

Figura 11 – Exemplo de uso do *web service* predição da partida



(a) Esperando ser feito

(b) Esperando ser feito



(c) Esperando ser feito

(d) Esperando ser feito

Fonte: Autor.

REFERÊNCIAS

- ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. **Reviews of modern physics**, [S.l.], v.74, n.1, p.47, 2002.
- ANTIQUÊIRA, L. et al. Modelando textos como redes complexas. In: III WORKSHOP EM TECNOLOGIA DA INFORMAÇÃO E DA LINGUAGEM HUMANA. **Anais...** [S.l.: s.n.], 2005. p.22–26.
- BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. **science**, [S.l.], v.286, n.5439, p.509–512, 1999.
- BOSTOCK, M. D3. js-data-driven documents (2016). **URL: <https://d3js.org>**, [S.l.], 2016.
- COSTA, L. d. F. et al. Characterization of complex networks: a survey of measurements. **Advances in physics**, [S.l.], v.56, n.1, p.167–242, 2007.
- ERDŐS, P.; R&WI, A. On random graphs I. **Publ. Math. Debrecen**, [S.l.], v.6, p.290–297, 1959.
- ESPN. **Premiação do Mundial de League of Legends ultrapassa US 4 milhões**. 2017.
- FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma introdução sucinta à teoria dos grafos. , [S.l.], 2011.
- LEISERSON, C. E. et al. Algoritmos: teoria e prática. **Campus, ed**, [S.l.], v.1, 2002.
- LOPES, F. M. **Redes complexas de expressão gênica: síntese, identificação, análise e aplicações**. 2011. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.
- LUCCHESI, C. L. **Introdução à teoria dos grafos**. [S.l.]: IMPA, 1979.
- RONACHER, A. Welcome—flask (a python microframework). **URL: <http://flask.pocoo.org/>**(visited on 02/04/2018), [S.l.], p.38, 2010.
- VIANA, M. P. **Metodologia das redes complexas para caracterização do sistema de Havers**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo.
- WATTS, D. J.; STROGATZ, S. H. Collective dynamics of ‘small-world’ networks. **nature**, [S.l.], v.393, n.6684, p.440, 1998.

ZHU, N. Q. **Data visualization with D3.js cookbook**. [S.l.]: Packt Publishing Ltd, 2013.

APÊNDICES

.1 TESTE

ANEXOS

ANEXO A – Título do Anexo

Este é o anexo A