

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

Guilherme Magno

**PREDIÇÃO DO RESULTADO UTILIZANDO KNN:
ANÁLISE DO JOGO *LEAGUE OF LEGENDS*.**

BambuÍ, MG
2018

Guilherme Magno

PREDIÇÃO DO RESULTADO UTILIZANDO KNN: ANALISE DO JOGO *LEAGUE OF LEGENDS*.

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

Predição do resultado utilizando KNN: Análise do jogo *League of Legends*. / por Guilherme Magno. – 2018.

40 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. Predição do resultado utilizando KNN: Análise do jogo *League of Legends*. .

Guilherme Magno

PREDIÇÃO DO RESULTADO UTILIZANDO KNN: ANALISE DO JOGO *LEAGUE OF LEGENDS*.

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

PREDIÇÃO DO RESULTADO UTILIZANDO KNN: ANALISE DO JOGO *LEAGUE OF LEGENDS*.

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.....	17
	(a) Grafo binário com quatro vértices e quatro arestas.....	17
	(b) Grafo não-binário com quatro vértices e três arestas.	17
	(c) Grafo não-binário e orientado.	17
	(d) Exemplo de vizinhança de um vértice de um grafo.	17
2	Variação do p sem alterar o numero de vértices $v = 20$ e $k = 4$	18
3	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.	20
4	Coeficiente de aglomeração.	21
	(a)	21
	(b)	21
5	Mapa do jogo de <i>League of Legends</i>	22
6	Coeficiente de aglomeração.	23
	(a)	23
	(b)	23
7	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.....	26
8	Esquema usado do Banco de Dados.	28
9	Exemplo de uso do <i>web service</i> para visão geral dos dados.	33
	(a) Filtro dos dados para campeões da mesma equipe.	33
	(b) Filtro dos dados para adversários.	33
	(c) Filtro para escolher quais campeões podem participar (<i>picks</i>) e quais nao podem (<i>bans</i>).....	33
	(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 círculo é a frequência daquela ligação e a cor é a força da ligação.	33
10	Visualização da topologia final da Rede Complexa.	34
11	Erro do KNN em variação ao K.	35
12	Exemplo de uso do <i>webservice</i> predição da partida.....	36
	(a) Área para escolher o time 0.	36
	(b) Área para escolher o time 1.	36
	(c) Exemplo de resposta do <i>webservice</i>	36
	(d) Tela toda da parte de predição.	36

LISTA DE TABELAS

1	Exemplo de <i>subset</i> salvo no banco de dados	32
2	Matriz de confusão do KNN	35

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>)
MLP	Perceptron Multicamadas (do inglês <i>Multilayer Perceptron</i>)
LOL	Liga das Lendas (do inglês <i>League of Legends</i>)
IFMG	Instituto Federal de Minas Gerais

SUMÁRIO

1	INTRODUÇÃO.....	13
1.1	CONTEXTUALIZAÇÃO	13
1.2	OBJETIVOS	14
1.2.1	Objetivos Gerais.....	14
1.2.2	Objetivos Específicos e Resultados Esperados.....	14
1.3	JUSTIFICATIVA	14
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	GRAFOS	16
2.2	PROPRIEDADE TOPOLÓGICAS	17
2.3	REDES COMPLEXAS	18
2.3.1	<i>Redes Small-Worlds</i>	18
2.3.2	Redes Livres de Escala	19
2.3.3	Redes Aleatórias.....	19
2.3.4	Coeficiente de Aglomeração	20
2.4	<i>LEAGUE OF LEGENDS</i>	21
2.5	D3.JS	23
2.6	ESTADO DA ARTE	23
3	METODOLOGIA	25
3.1	API	25
3.2	PERSISTÊNCIA DOS DADOS	27
3.3	AQUISIÇÃO DOS DADOS	28
3.4	VISUALIZAÇÃO DOS DADOS	29
3.5	CLASSIFICAÇÃO DA TOPOLOGIA	30
3.6	PREDIÇÃO DOS DADOS	30
4	RESULTADOS	32
5	CONCLUSÃO	37
	REFERÊNCIAS	39

1 INTRODUÇÃO

Muitos pesquisadores tem tido interesse em estudar sobre grafos, pois estes estão sendo desenvolvidos para modelizar sistemas (JOACHIMS et al., 2009), como, por exemplo, rede de energia de Nova York e a rede alimentar de Little Rock Lake (STROGATZ, 2001). Os objetivos desses pesquisadores são variados, que podem ser desde classificação de grafos (XIANG; NEVILLE; ROGATI, 2010), à predição (YANG; HARRISON; ROBERTS, 2014) e para uma compreensão melhor do mesmo (ROSVALL; BERGSTROM, 2008).

Alguns exemplos de modelação das pesquisas são redes sociais (XIANG; NEVILLE; ROGATI, 2010), estruturas químicas (ROSVALL; BERGSTROM, 2008) e jogos (YANG; HARRISON; ROBERTS, 2014). Em nosso trabalho, trabalharemos na de jogos, pois ainda no meio acadêmico são novidades quando se trata de modelação em grafos. Jogos que tem conquistado a atenção e o tempo de muitas pessoas, movimentando mais receita do que a indústria de filmes Hollywood¹.

Com a popularização do *League of Legends*², também conhecido como LOL, um jogo da categoria de MOBA (**M**ultiplayer **O**nline **B**attle **A**rena, ou, pela tradução, "Arena de batalha online de multijogadores"), e sua crescente premiação em campeonatos³, jovens e adultos tem se tornado *pro players* (de tradução: jogadores profissionais) e dedicado sua vida ao mundo dos jogos⁴. Esse fato viabiliza o uso de ferramentas para ajudá-los na análise dos jogos e de estratégias. Devido a popularidade do jogo e a facilidade de obtenção dos dados devido a API (**A**pplication **P**rogramming **I**nterface ou, da tradução, Interface de Programação de Aplicações) do LOL, ele é escolhido como foco do trabalho.

Nesse contexto, o presente trabalho propõe a modelagem do grafo das partidas do jogo *League of Legends*, a sua visualização, a classificação do mesmo e a predição de equipes vencedoras utilizando MLP (**M**ultilayer **P**erceptron, ou, da tradução, Perceptron Multicamadas), no formato de um *webservice* (da tradução, serviço da internet).

¹ <http://www.webnoticias.fic.ufg.br/n/68881-industria-de-games-supera-o-faturamento-de-hollywood>

² https://play.br.leagueoflegends.com/pt_BR

³ http://www.espn.com.br/noticia/736116_premiacao-do-mundial-de-league-of-legends-ultrapassa-us-4-milhoes

⁴ <https://oglobo.globo.com/esportes/de-nerds-ciberatletas-crescimento-exponencial-do-sports-21233721>

1.1 CONTEXTUALIZAÇÃO

Na matriz curricular do curso de Engenharia de Computação do Instituto Federal de Minas Gerais - *campus* Bambuí, existem várias disciplinas que foram utilizadas no presente trabalho (como Programação Orientada a Objetos, Banco de Dados, Programação Web dentre outras) e com o estudo mais aprofundado sobre grafos, foi possível desenvolver o *webservice* pois a API do LOL é fácil de ser usada e facilita a obtenção dos dados.

A proposta deste trabalho será o desenvolvimento de um *webservice* que indicará uma possível equipe vencedora, de acordo pela escolha dos campeões (personagens jogáveis do jogo, em que cada um, individualmente, tem atributos e habilidades diferentes) dos jogadores, e, também, mostrar a sinergia entre os campeões e o seus *counter picks* (da tradução, "contra-escolha", que seria uma expressão usada nos jogos de MOBA para indicar quando um campeão específico tem, por si só, uma vantagem sobre outro campeão, também específico).

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

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

1.2.2 Objetivos Específicos e Resultados Esperados

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

1.3 JUSTIFICATIVA

Devido ao grande crescimento e a popularização do jogo *League of Legends*, tem surgido muitos sites de análises do mesmo que vem auxiliando tanto os jogadores esporádicos, tanto quanto os *pro players*. Porém esses sites, apenas mostram o que os jogadores profissionais fazem, ou algumas escolhas em comum com a comunidade, como quais equipamentos comprar, ou habilidades para priorizar. Este trabalho propõe desenvolver um modelo computacional para que usuários, *coach*, ou analistas possam ter uma ferramenta para obtenção de informações ou para predição do resultado de um jogo.

A utilização da rede complexa para modelar os dados, foi escolhida, pois ela possibilita representar quantitativamente as conexões de cada herói, sendo conexões aliadas ou adversárias.

Assim, este trabalho se justifica principalmente pela análise da rede complexa obtida, pela visualização gráfica e pela predição de partidas para que estas informações adquiridas se tornem acessíveis. Para execução desse trabalho é necessário o uso de várias áreas do conhecimento estudadas no curso de Engenharia de Computação, principalmente computação.

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.

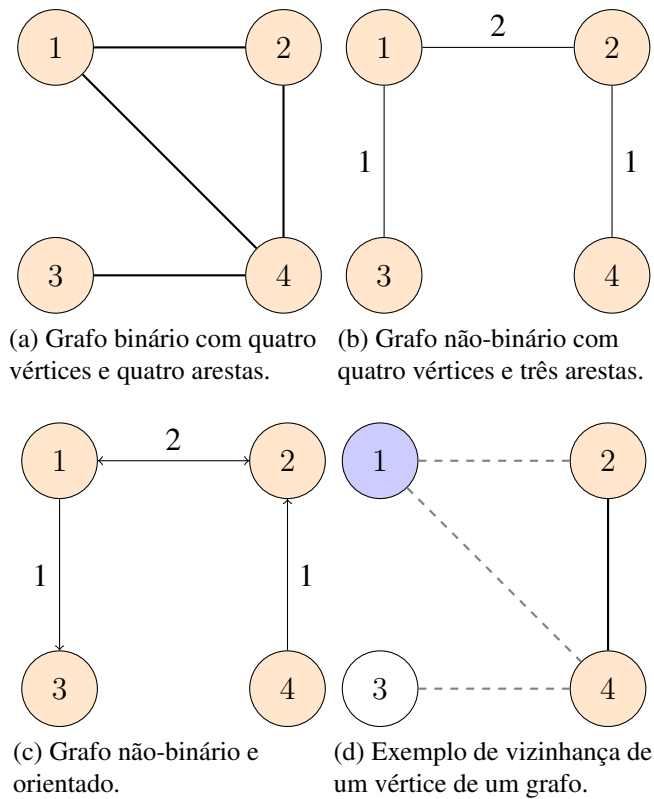
2.1 GRAFOS

De uma maneira resumida e simples, poderia-se dizer que os grafos são a abstração matemática que descreve uma situação utilizando pontos e uma ligação entre alguns destes (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 os vértices $(u, v) \in E$, então eles u e v são conectados pela aresta e e são considerados como vértices vizinhos ou adjacentes (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011; VIANA, 2007).

Os grafos também podem ter suas arestas classificadas como binárias ou ponderadas, sendo os binários suas arestas representadas apenas por 1 (se existem) ou 0 (se não existem) e as ponderadas quando existe um valor no conjunto W , o qual informa a intensidade da interação de uma aresta entre dois vértices (VIANA, 2007; LEISERSON et al., 2002). Na Figura 1a temos um grafo binário, em que 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 valor da aresta no conjunto W informando o peso.

É chamado de grafo orientado quando $(u, v) \in E$ é diferente de $(v, u) \in E$. Neste caso, quando um vértice é ligado ao outro, não necessariamente esse segundo vértice é conectado ao primeiro, como um exemplo mostrado na Figura 1c. FEOFILOFF; KOHAYAKAWA; WAKABAYASHI (2011) afirmam 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 1d.

Figura 1 – Exemplo de grafos.



Fonte: Autor.

2.2 PROPRIEDADE TOPOLÓGICAS

O grau de um vértice $v, g(v)$, é definido sendo a quantidade de arestas que chegam nele. Por exemplo, na Figura 1a vemos que o grau do vértice 1, $g(1)$, é igual à 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 igual ao do vértice que tem o maior grau presente nesse grafo $V(G)$, podendo ser expresso por $\Delta(G) = \max\{g(v) : v \in V(G)\}$. O grau mínimo de um grafo, por outro lado, é o grau do vértice com o qual poderíamos escrever por: $\delta(G) = \min\{g(v) : v \in V(G)\}$. um grafo é considerado regular caso $\delta(G) = \Delta(G)$ e é k -regular se $\delta(G) = \Delta(G) = k$ (FEOFILOFF; KOHAYAKAWA; WAKABAYASHI, 2011).

Os grafos que são modelamos a partir de um sistema real são classificados como redes complexas, as quais veremos na seção seguinte.

2.3 REDES COMPLEXAS

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

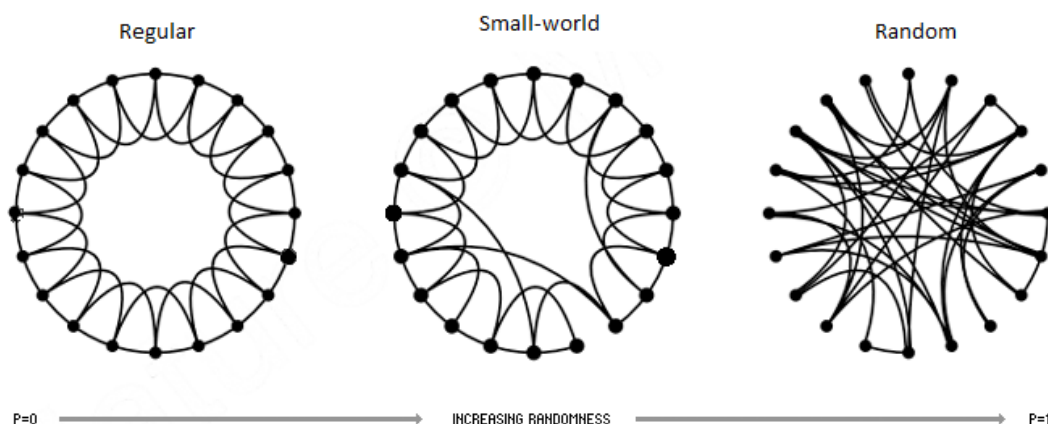
Para que seja possível classificar o nosso modelo adequadamente, serão apresentados três modelos que, segundo ALBERT; BARABÁSI (2002) destacam entre os pesquisadores as redes *small-worlds*, as redes livres de escala e as redes aleatórias.

2.3.1 Redes Small-Worlds

As redes *small-worlds* são um modelo de rede complexas que 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).

Vamos supor que exista um grafo regular com v vértices e k arestas ligando os vizinhos mais próximos e, após de ser formado, cada uma de suas arestas têm uma probabilidade p de se reconectar com outro vértice aleatório. Isto permite que a geração do grafo seja controlada para uma rede regular com $p \approx 0$ ou uma rede aleatória com $p \approx 1$, e, também, a formação de uma rede com topologia intermediária $0 < p < 1$ (LOPES, 2011).

Figura 2 – 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 nós possuem probabilidade $P(k)$ de possuir k arestas obedecendo a lei da potência $P(k) \sim k^{-\gamma}$, em que γ determina o decaimento exponencial (ANTIQUERA et al., 2005; LOPES, 2011). Segundo VIANA (2007), nas redes livres de escala possuem muitos nós com poucas arestas e poucos nós que 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, um novo vértice v é adicionado na rede a cada instante de tempo t , o número de arestas desse vértice segue uma preferência linear de ligação da fórmula

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

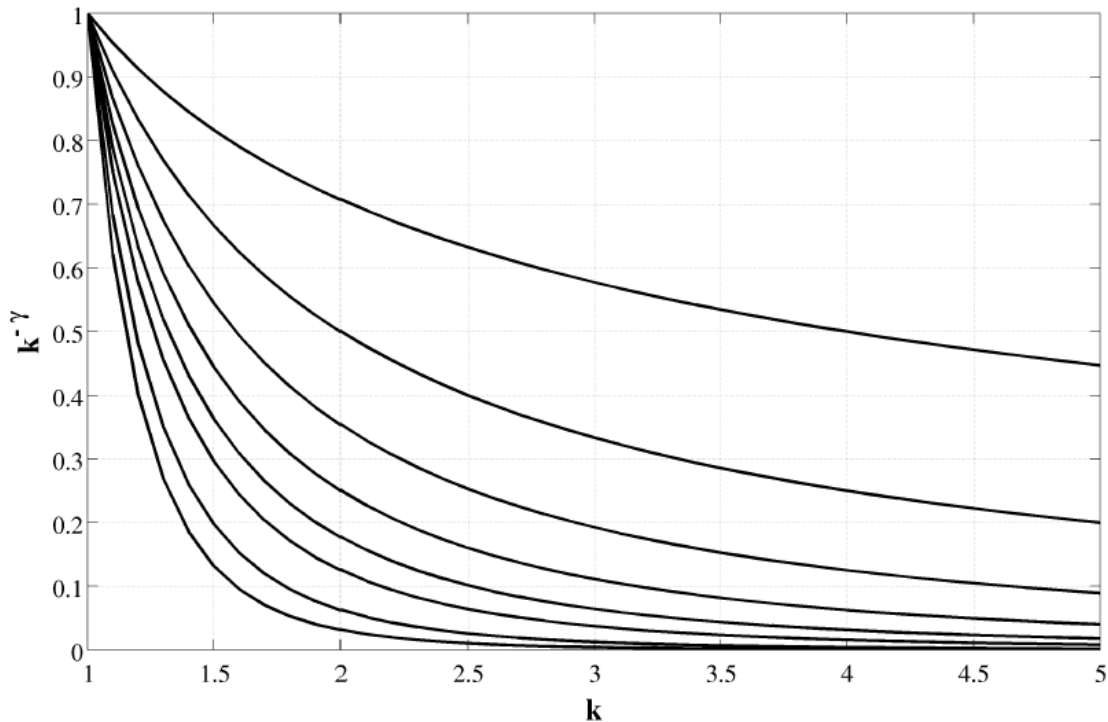
Sendo P a probabilidade de um vértice v_i ser conectado ao novo vértice v_j é linearmente proporcional ao grau k_j do vértice v_j LOPES (2011).

É possível ver na Figura 3 que quanto maior o γ , menor a probabilidade de existir vértices com grande 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 que são dispostos inicialmente os vértices e depois aleatoriamente as arestas são distribuídas com probabilidade igual entre os vértices, evitando auto-relacionamentos e conexões múltiplas entre dois vértices.

Figura 3 – 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).

2.3.4 Coeficiente de Aglomeração

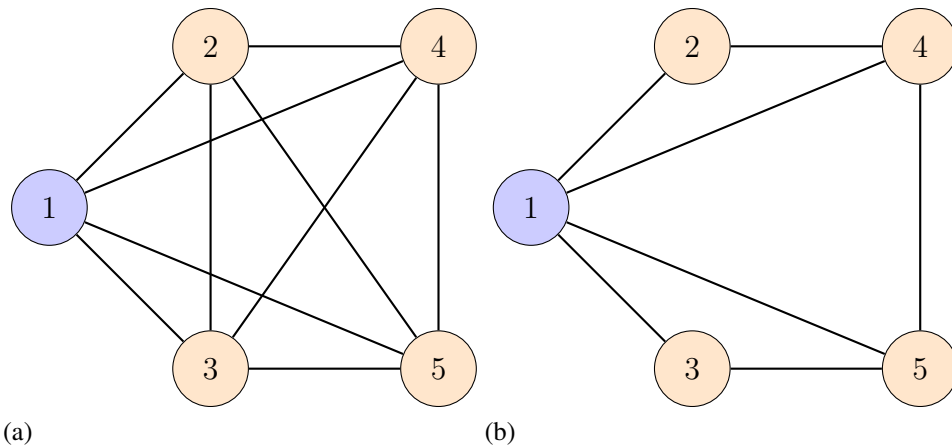
Segundo VIANA (2007), o coeficiente de aglomeração mede o quão conectado estão os nós da rede ou do grafo. ANTQUEIRA et al. (2005) definem o coeficiente de aglomeração sendo

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

ANTQUEIRA 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 4a, tem-se que o peso de todas as arestas é 1 e, por isso, o coeficiente de aglomeração do vértice em azul é 1. Já na Figura 4b, o coeficiente do vértice em azul, segundo a fórmula é $\frac{1}{2}$.

Figura 4 – Coeficiente de aglomeração.



Fonte: Autor.

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 em que duas equipes se enfrentam virtualmente em um campo de batalha e cada jogador 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 na qual 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 5 – Mapa do jogo de *League of Legends*



Fonte: Autor (2018).

A Figura 5 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.

No início do jogo cada jogador escolhe um herói diferente, em que cada herói tem um conjunto de características únicas, como habilidades especiais que causa uma influência individual no jogo e, indiretamente, na escolha e nos heróis dos outros jogadores, tanto na equipe aliada quanto na equipe inimiga.

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

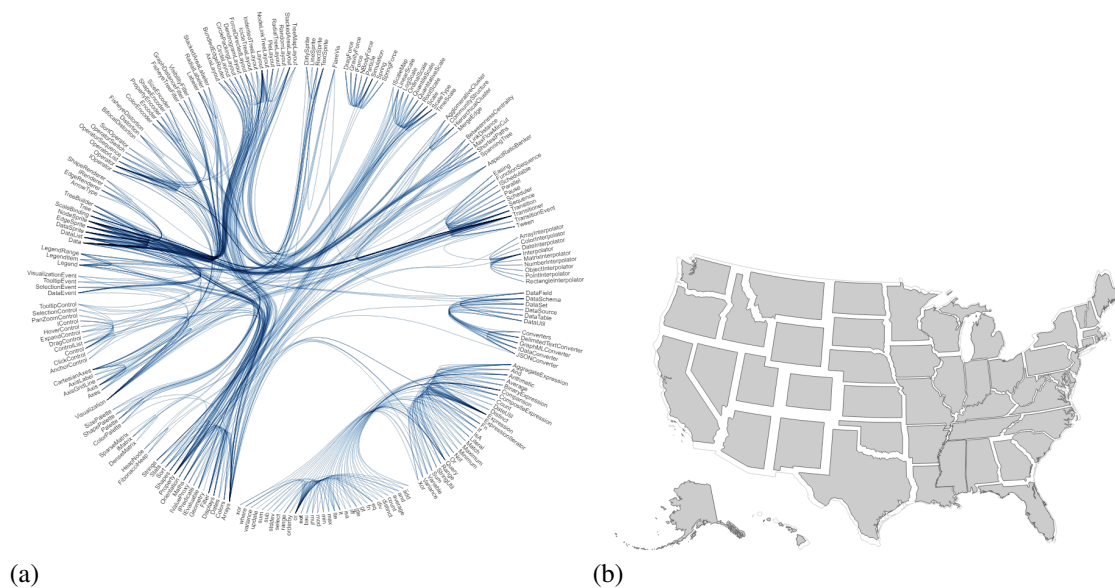
2.5 D3.JS

A biblioteca D3.js é uma biblioteca em JavaScript especializada em deixar os dados fáceis de serem lidos e entendidos. Segundo ZHU (2013): "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 6 que possui duas das diversas maneiras de se exibir informações com o D3.js . Com o D3, foi gerado o grafo do *web service*, programa que será explorado na seção seguinte.

Figura 6 – Coeficiente de aglomeração.



Fonte: (BOSTOCK, 2016).

2.6 ESTADO DA ARTE

XIANG; NEVILLE; ROGATI (2010) fizeram uma modelação de redes sociais, em que, dados eram adquiridos das redes sociais, e eram modelados em forma de rede complexa. Cada rede complexa depois, passa por processamento de informação, onde são retiradas medidas para

estimar a "força" de uma amizade naquela rede social.

BACKSTROM et al. (2006) também fizeram um trabalho com redes sociais, onde estudam as estruturas das redes sociais MySpace e LiveJournal, e mostraram que até homofilia pode ser usado para melhorar os modelos de predição em um grupo de pessoas.

YANG; HARRISON; ROBERTS (2014) com um tema de trabalho também sobre jogos MOBA, fizeram a aquisição de partidas de Dota 2, um jogo da Steam⁵, onde modelam o posicionamento dos jogadores em uma batalha virtual em uma rede complexa, e predizem o vencedor daquela batalha, apenas pelo posicionamento dos jogadores.

⁵ <https://store.steampowered.com/?l=portuguese>

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 quais informações eles carregam. A segunda clarifica como estes dados serão armazenadas e 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 URL⁶, 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 7 resume um exemplo de uso da função acessando a URL⁷, acessada em 21 de março de 2018 pelo autor, sendo que *minhachave* tem que ser substituída por uma chave privada válida.

⁶ <http://developer.riotgames.com>

⁷ https://br1.api.riotgames.com/lol/match/v3/matches/1381102031?api_key=minhachave

Figura 7 – 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:

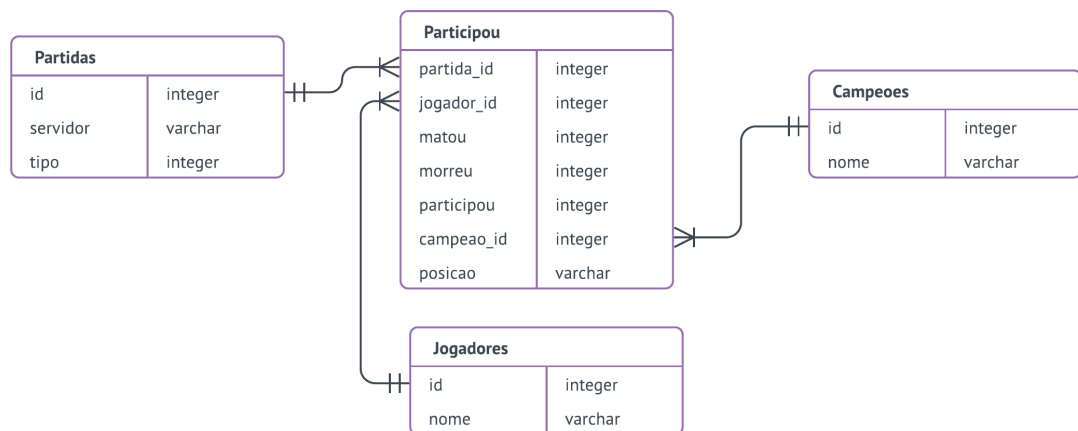
1. *gameId*. Identificador único da partida;
2. *kills*, *deaths* e *assists*. Informações que falam, respectivamente, quantas vezes esse jogador matou, morreu e participou na morte de outrem;
3. *win*. Se ele ganhou;
4. *championId*. Qual campeão ele escolheu;
5. *lane*. Em qual posição ele tava jogando;
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 3.3 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 8, 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 8 – 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* público⁸. Ambos apresentam os mesmos conjuntos de informações em formatos diferentes. Além destes, será adquirido outros dados pela API para agilização do trabalho.

A base de dados, na sua versão 9, conta com aproximadamente 183 mil partidas ranqueadas (partidas que, em poucas palavras, servem para classificar as habilidades do jogador) diferentes, dando uma incrementada no banco de dados inicialmente. Estes que precisam ser convertidos para o esquema do banco de dados usados no trabalho. Em seguida, 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.

⁸ <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
    última 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

```

Decidiu-se 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. Na etapa seguinte são contabilizados quantas vezes cada campeão específico apareceu em uma mesma partida com outro campeão específico, seja no mesmo time ou no time adversário, calculando quantas vitórias em oposição e quantas vitórias em junção.

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 explanada na próxima seção.

3.4 VISUALIZAÇÃO DOS DADOS

O *web service* será 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 deles podem participar da pesquisa e quais não podem. Também será exequível a predição de uma partida, na qual se seleciona os campeões

de cada equipe e o sistema tenta antever quem vencerá.

3.5 CLASSIFICAÇÃO DA TOPOLOGIA

Esperando o artigo.

3.6 PREDIÇÃO DOS DADOS

Para a parte de predição dos dados, serão considerados as conexões e os pesos entre os campeões da partida obtidas através do grafo obtido na Seção 3.3. Os dados de cada partida foram transformados em 3 informações de entrada e a saída:

- Sinergia Time 0;
- Sinergia Time 1;
- Entropia Somada;
- Time vencedor.

Sendo que as duas primeiras informações se referem a somatória dos pesos da aresta que diz o quão bom os campeões de um time vencem juntos, dados 0 a 1. Para tanto, vamos considerar que p_1, p_2, p_3, p_4, p_5 são do time 0, $p_6, p_7, p_8, p_9, p_{10}$ do time 1, e $w(u, v)$ sendo o peso da arestas de porcentagem de partidas ganhas juntas de u e v , de 0 a 1. Então pode se dizer que:

$$SinergiaTime0 = \sum_{\substack{1 \leq i \leq 5 \\ i < j \leq 5}} w(p_i, p_j),$$

e, também:

$$SinergiaTime1 = \sum_{\substack{6 \leq i \leq 10 \\ i < j \leq 10}} w(p_i, p_j).$$

A entropia somada, foi definida como a soma de quantas partidas um campeão c_1 ganhou de um campeão c_2 , sendo que este valor também vai de 0 a 1. A definição matemática pode ser descrita como, sendo $w_e(c_1, c_2)$ a porcentagem que o c_1 ganhou do c_2 :

$$EntropiaSomada = \sum_{\substack{1 \leq i \leq 5 \\ 6 \leq j \leq 10}} w_e(c_i, c_j)$$

A entropia do time 1 para o time 0 não foi feita, porque a correlação entre as duas entropias seria alta, uma vez que $w_e(p_i, p_j) = 1 - w_e(p_j, p_i)$ e elas informariam a mesma coisa.

Então, por eficiência, e para aumentar a velocidade de treino do modelo, ela foi retirada. O time vencedor apenas diz qual time venceu, informando se foi o time 0 ou o time 1.

Então com esses valores, é criado um classificador KNN, em que, com as três informações de entrada, ele tentará classificar o time vencedor. Os valores que foram armazenados, são separados em dados de treino e de teste, sendo que o teste representam 20% dos dados totais e todos os dados são re-escalados no intervalo de 0 a 1, para que o KNN possa ser treinado adequadamente. Com o modelo final treinado e testado, o *webservice* será capaz de comunicar com o KNN para responder às entradas do usuário, indicando um time vencedor.

4 RESULTADOS

O trabalho ficou quatro meses adquirindo partidas da API do LOL, sendo que apenas 14 mil partidas aproximadamente eram válidas para os filtros descritos na Seção 3.3 que dizem que as partidas devem ser ranqueadas e não terem os dados incompletos. Por este algoritmo, foram armazenadas 230000 jogos válidos diferentes, com esse escopo e o número de partidas usados foram ao todo 197000 contando com o *dataset* público. 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.

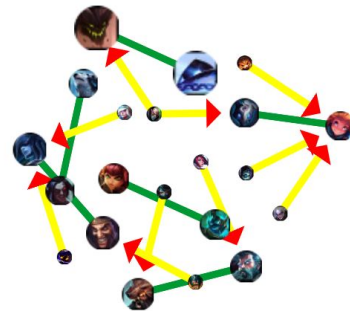
Com os dados armazenados, e o processamento deles executado, foi feito o *webservice* que consegue exibir para o usuário as informações sobre a topologia do grafo, qual campeão é adequado contra um outro específico. Na Figura 9 é possível ver um exemplo de uso do serviço para uma visão geral das informações e na Figura 9d mostra quem tem uma sinergia com quem (arestas verdes) e quem é um *counter pick* do outro (arestas com uma seta vermelha orientando o grafo). O tamanho das arestas dos campeões indica a quantidade de vezes que aquele determinado herói jogou com os filtros escolhidos pelo usuário.

Figura 9 – 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="Wendel"/>
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="Wendel"/>
<input type="button" value="Search"/>	

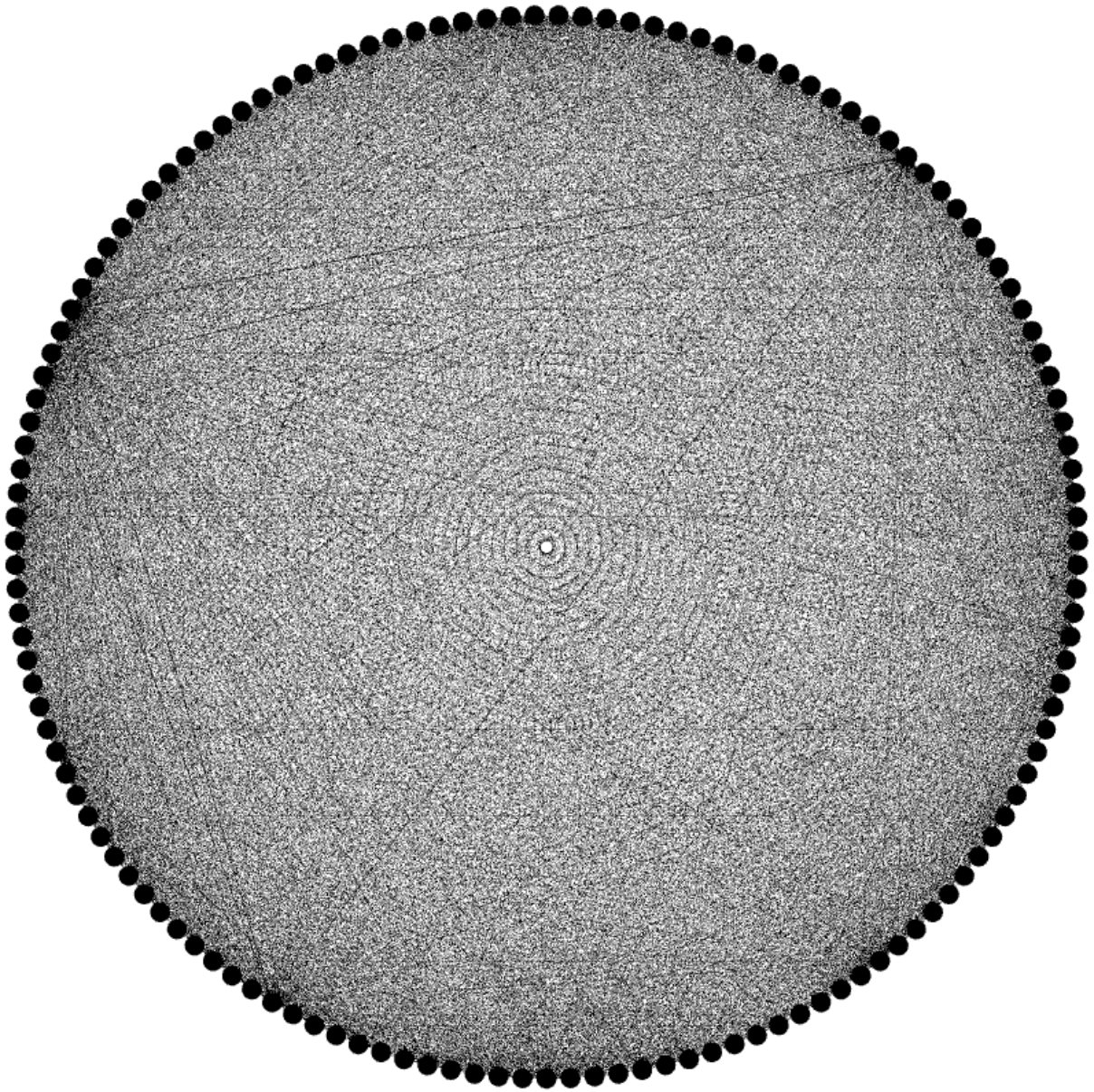


(c) Filtro para escolher quais campeões podem participar (*picks*) e quais não podem (*bans*). (d) Exemplo de grafo exibido pelo *web service*, sendo as setas vermelhas mostrando quem é bom contra quem, e as linhas continuas quem é bom com quem. O tamanho do círculo é a frequência daquela ligação e a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

Fonte: Autor.

É possível ver topologia da rede complexa final na Figura 10, que consta com 139 vértices diferentes (todos os heróis lançados até a aquisição dos dados) e com 19180 arestas já que estas são orientadas, e que se aproxima de uma rede mundo pequeno, pois faltam duas arestas para se tornar uma rede regular.

Figura 10 – Visualização da topologia final da Rede Complexa.



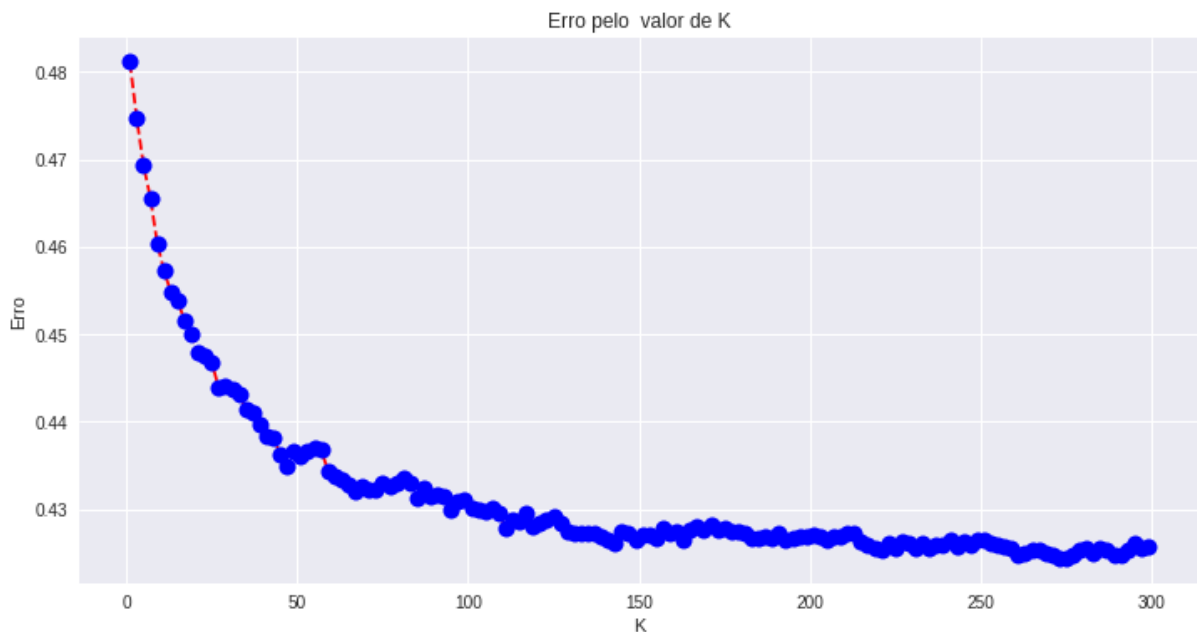
Fonte: Autor.

Tabela 2 – Matriz de confusão do KNN

Real x Predição	time 0	time 1
time 0	10909	6993
time 1	7848	9107

Fonte: Autor.

Figura 11 – Erro do KNN em variação ao K.



Fonte: Autor.

O Classificador KNN ficou com uma precisão de 58% com a melhor calibração conseguida, com o $k = 243$, na Figura 11 é possível ver a precisão em relação ao k . E com o modelo preparado é possível entrar no *webservice* e seleccionar os campeões de cada time e conferir o resultado da predição. É possível ver um exemplo de uso do *webservice* na Figura 12 onde possível ver a sinergia dos dois times e a entropia da partida.

Com a matriz de confusão é possível ver as classificações corretas e as erradas por resultado, ou seja, na Tabela 2 é possível ver quantas vezes o KNN classificou sendo vencedor o time 0, porém era vencedor o time 1 ou qualquer outro cenário.

Figura 12 – Exemplo de uso do *webservice* predição da partida

Time 0

Player 1

Sion

Player 2

Lee Sin

Player 3

Viktor

Player 4

Caitlyn

Player 5

Sonna

Time 1

Player 6

Tryndamare

Player 7

Jarvan IV

Player 8

Xerath

Player 9

Lucian

Player 10

Lulu

Sinergia time 0: 5.626

Sinergia time 1: 7.005

Entropia do jogo: 12.599

Vencedor: Time 1

Time 0

Player 1

Sion

Player 2

Lee Sin

Player 3

Viktor

Player 4

Caitlyn

Player 5

Sonna

Time 1

Player 6

Tryndamare

Player 7

Jarvan IV

Player 8

Xerath

Player 9

Lucian

Player 10

Lulu

Sinergia time 0: 5.626

Sinergia time 1: 7.005

Entropia do jogo: 12.599

Vencedor: Time 1

(a) Área para escolher o time 0.

(b) Área para escolher o time 1.

(c) Exemplo de resposta do *webservice*.

(d) Tela toda da parte de predição.

Fonte: Autor.

5 CONCLUSÃO

Os resultados foram bons, levando em conta que o algoritmo adquiriu partidas ranqueadas de pessoas iniciantes a pessoas profissionais, e para predição da partida quando o usuário utiliza o sistema, também não diferencia o nível de habilidade dos jogadores. A estratégia de modelar em rede complexa e classificar com KNN poderia obter resultados melhores se analisasse o escopo de habilidade das partidas, e também do jogo em análise.

O *webservice* consegue mostrar informações para auxiliar aos usuários, mostrando quem pode ser uma escolha viável para jogar com quem ou qual herói é bom para jogar contra a escolha do adversário, mesmo que eles não se vejam ou enfrentem uma grande parte do jogo.

Os algoritmos em *Python* para aquisição das partidas e os arquivos do *webservice* encontram no Github⁹ para trabalhos futuros.

⁹ <https://github.com/Rout222/lol>

REFERÊNCIAS

- ALBERT, R.; BARABÁSI, A.-L. Statistical mechanics of complex networks. **Reviews of modern physics**, [S.l.], v.74, n.1, p.47, 2002.
- ANTIQUEIRA, 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.
- BACKSTROM, L. et al. Group formation in large social networks: membership, growth, and evolution. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 12. **Proceedings...** [S.l.: s.n.], 2006. p.44–54.
- 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.
- ERDDS, P.; R&WI, A. On random graphs I. **Publ. Math. Debrecen**, [S.l.], v.6, p.290–297, 1959.
- FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma introdução sucinta à teoria dos grafos. , [S.l.], 2011.
- JOACHIMS, T. et al. Predicting structured objects with support vector machines. **Communications of the ACM**, [S.l.], v.52, n.11, p.97–104, 2009.
- 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.

ROSVALL, M.; BERGSTROM, C. T. Maps of random walks on complex networks reveal community structure. **Proceedings of the National Academy of Sciences**, [S.l.], v.105, n.4, p.1118–1123, 2008.

STROGATZ, S. H. Exploring complex networks. **nature**, [S.l.], v.410, n.6825, p.268, 2001.

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.

XIANG, R.; NEVILLE, J.; ROGATI, M. Modeling relationship strength in online social networks. In: WORLD WIDE WEB, 19. **Proceedings...** [S.l.: s.n.], 2010. p.981–990.

YANG, P.; HARRISON, B. E.; ROBERTS, D. L. Identifying patterns in combat that are predictive of success in MOBA games. In: FDG. **Anais...** [S.l.: s.n.], 2014.

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