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

Guilherme Magno

VISUALIZAÇÃO E CLASSIFICAÇÃO DE UMA REDE COMPLEXA, E PREDIÇÃO DO RESULTADO UTILIZANDO MLP: ANALISE DO JOGO *LEAGUE OF LEGENDS*.

Bambuí, MG
2018

Guilherme Magno

VISUALIZAÇÃO E CLASSIFICAÇÃO DE UMA REDE COMPLEXA, E PREDIÇÃO DO RESULTADO UTILIZANDO MLP: 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

Visualização e Classificação de uma rede complexa, e predição do resultado utilizando MLP: Analise do jogo *League of Legends*. / por Guilherme Magno. – 2018.

41 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 Classificação de uma rede complexa, e predição do resultado utilizando MLP: Analise do jogo *League of Legends*. .

E-mail: Routmagno@gmail.com

Guilherme Magno

VISUALIZAÇÃO E CLASSIFICAÇÃO DE UMA REDE COMPLEXA, E PREDIÇÃO DO RESULTADO UTILIZANDO MLP: 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 menbro banca Sobre nome, Me. (UFSM)

Nome menbro 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 CLASSIFICAÇÃO DE UMA REDE COMPLEXA, E PREDIÇÃO DO RESULTADO UTILIZANDO MLP: 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

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MOBA	Arena de batalha online de multijogadores (do inglês *Multiplayer Online Battle Arena*)
API	Interface de Programação de Aplicativos (do inglês *Application Programming Interface*)
JSON	Notação de Objeto JavaScript (do inglês *JavaScript Object Notation*)
URL	Localizador Uniforme de Recursos (do inglês *Uniform Resource Locator*)
MLP	Perceptron Multicamadas (do inglês *Multilayer Perceptron*)
LOL	Liga das Lendas (do inglês *League of Legends*)
IFMG	Instituto Federal de Minas Gerais

SUMÁRIO

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 industria de filmes Hollywood[1].

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 campeonatos[3], jovens e adultos tem se tornado *pro players* (de tradução: jogadores profissionais) e dedicado sua vida ao mundo dos jogos[4]. 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 (Multilayer Perceptron, 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 sí só, uma vantagem sobre outro campeão, também específico). Também será feito a classificação da modelação do grafo obtido dos dados do jogo para fins acadêmicos.

1.2 OBJETIVOS

1.2.1 Objetivos Gerais

Criar um modelo computacional capaz de exibir resumos da rede complexa gerada dos dados armazenados e predizer 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 quantificadamente 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.

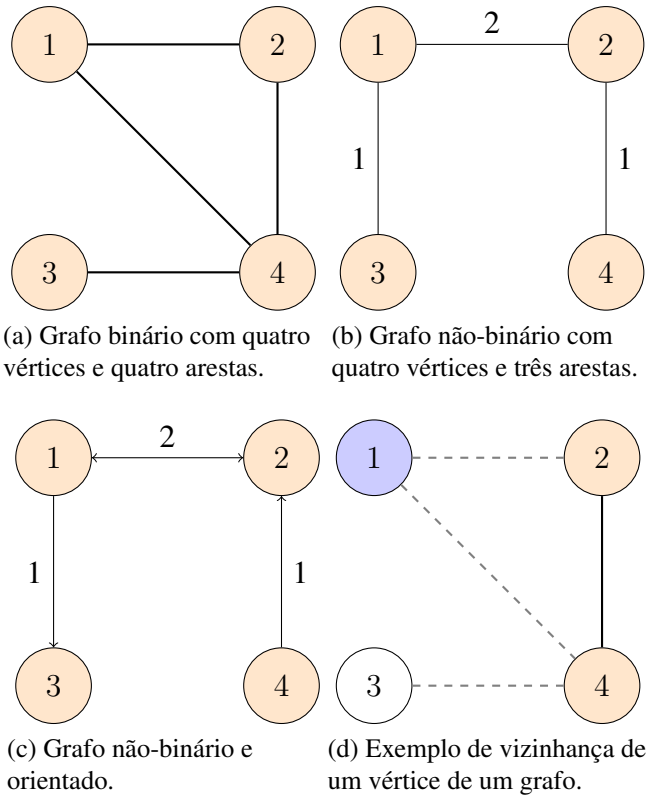
2.1 GRAFOS

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 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 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 valor da aresta no conjunto W informando o peso.

É chamado de grafo orientado quando $(u, v) \in E$ é diferente de $(v, u) \in E$, ou seja, quando um vértice é ligado ao outro, não necessariamente esse segundo vértice é conectado ao primeiro, um exemplo está na 1c. 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 1d.

Figura 1 – Exemplo de grafos.



(a) Grafo binário com quatro vértices e quatro arestas.

(b) Grafo não-binário com quatro vértices e três arestas.

(c) Grafo não-binário e orientado.

(d) 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.

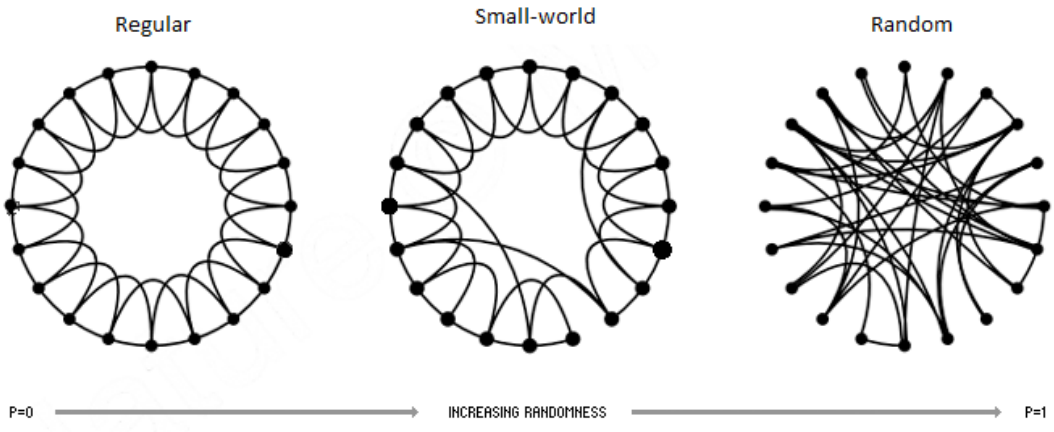
Para que seja possível classificar o nosso modelo adequadamente, serão apresentados três modelos que se destacam entre os pesquisadores segundo ALBERT; BARABÁSI (2002): As redes *small-worlds*, as redes livres de escala e as redes aleatórias. E também será explicado sobre o coeficiente de aglomeração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

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

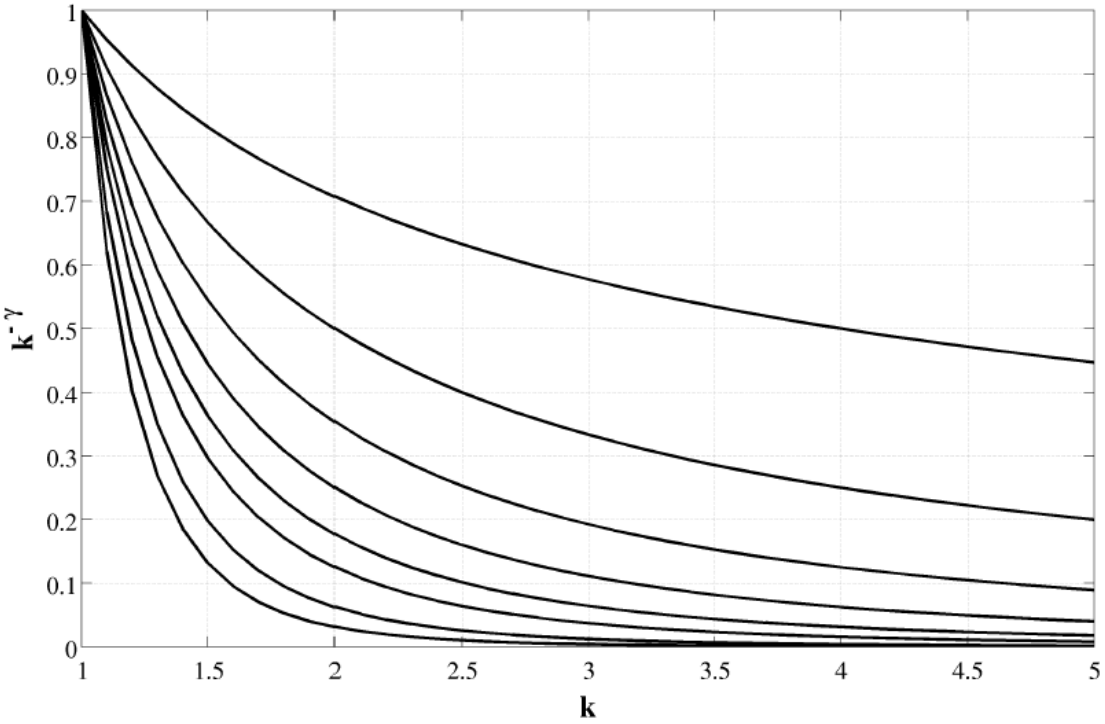
Sendo P a probabilidade de um vértice v_i ser conectado ao novo vértice v_j é linearmente proporcional ao grau k_j do vértice v_j LOPES (2011).

É possível ver na Figura 3 que quanto maior o γ 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.

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

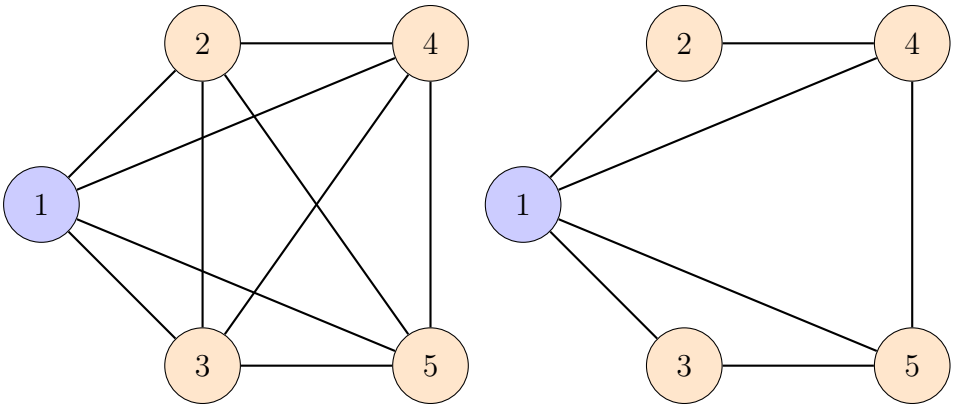
Segundo VIANA (2007) o coeficiente de aglomeração mede o quão conectado estão os nós da rede ou do grafo. ANTIQUEIRA et al. (2005) define o coeficiente de aglomeração sendo:

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

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 4a, assumindo o peso de todas as arestas como 1, 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.



(a) (b)

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 onde duas equipes se enfrentam 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 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 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.

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.

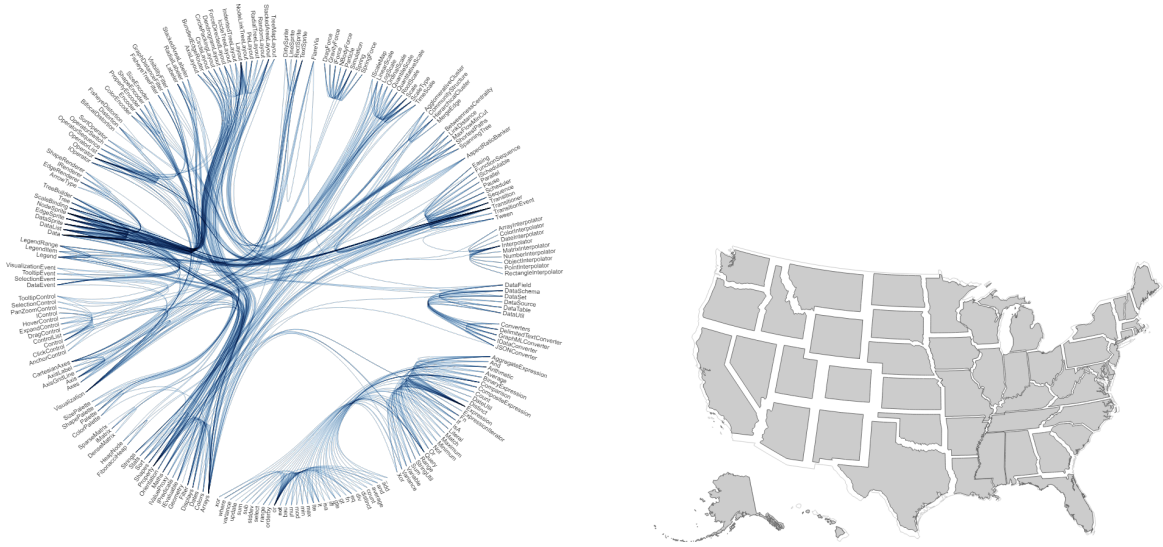
2.5 D3.JS

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 6 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 6 – 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 patir da URL[5] 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 URL[6], acessada em 21 de março de 2018 pelo autor, sendo que $minhachave$ tem que ser substituída por uma chave privada válida.

[5] http://developer.riotgames.com
[6] 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":{ ⊞ }
   },
   { ⊞ },
```


Fonte: Autor.

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

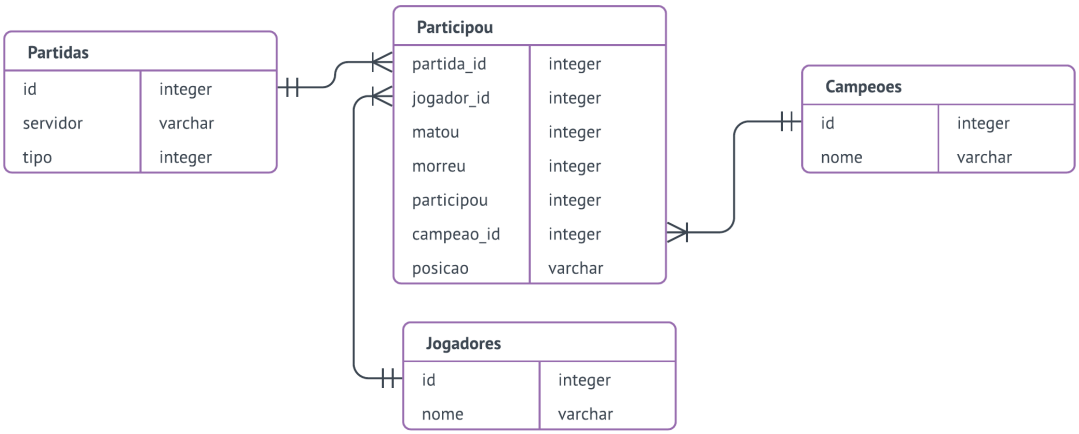
1. *gameId*. Identificador único da partida;
2. *kills*, *deaths* e *assists*. Informações que falam, respectivamente, quantas vezes esse jogador matou, morreu e participou na morte de outrem;
3. *win*. Se ele ganhou;
4. *championId*. Qual campeão ele escolheu;
5. *lane*. Em qual posição ele tava jogando;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

[7] 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
```

Foi decidido usar apenas as partidas ranqueadas 5 contra 5, já que estas são as partidas responsáveis pela classificação dos jogadores dentro dos jogo, e as partidas não ranqueadas são consideradas como amistosas. 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.

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* 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 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.

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, que em cada partida, foram transformados os dados 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, de 0 a 1. Pode se dizer que, sendo p_1, p_2, p_3, p_4, p_5 os heróis 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:

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

Logo também:

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

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 $w_e(p_1, p_2)$ a porcentagem que o p_1 ganhou do p_2 :

$$EntropiaSomada = \sum_{\substack{1<=i<=5 \\ 6<=j<=10}} w_e(p_i, p_j)$$

A entropia do time 1 para o time 0 não foi feita, porque a correlação entre as duas entropias seria alta pois, $w_e(p_i, p_j) = 1 - w_e(p_j, p_i)$ 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 onde 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. Onde com esse algoritmo foram armazenadas 230000 jogos válidos diferentes, com esse escopo, 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.

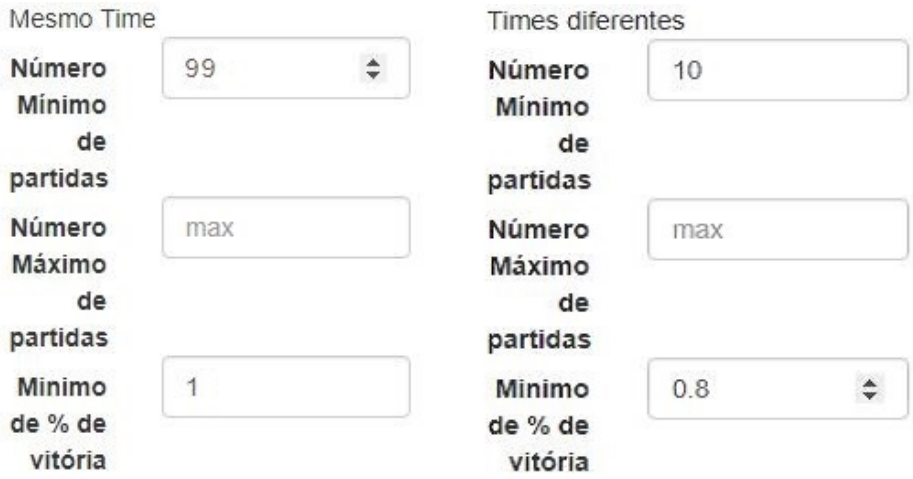
Tabela 1 – Exemplo de *subset* salvo no banco de dados

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.

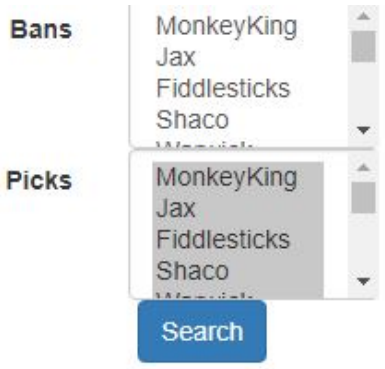
Com os dados armazenados, e o processamento deles executado, foi feito o *webservice* que consegue exibir para o usuário informações sobre a topologia do grafo, como quem é adequado contra um campeão específico, ou com o campeão. Na Figura 9 é possível ver um exemplo de uso do serviço para uma visão geral das informações. Sendo que 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 que aquele determinado herói jogou com os filtros escolhidos pelo usuário.

Figura 9 – Exemplo de uso do *web service* para visão geral dos dados.

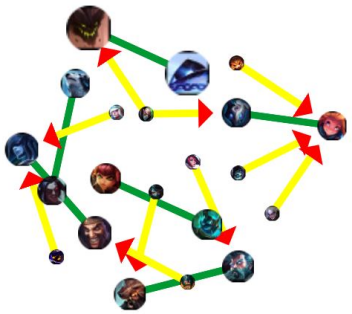


(a) Filtro dos dados para campeões da mesma equipe.

(b) Filtro dos dados para adversários.



(c) Filtro para escolher quais campeões podem participar (*picks*) e quais nao podem (*bans*).

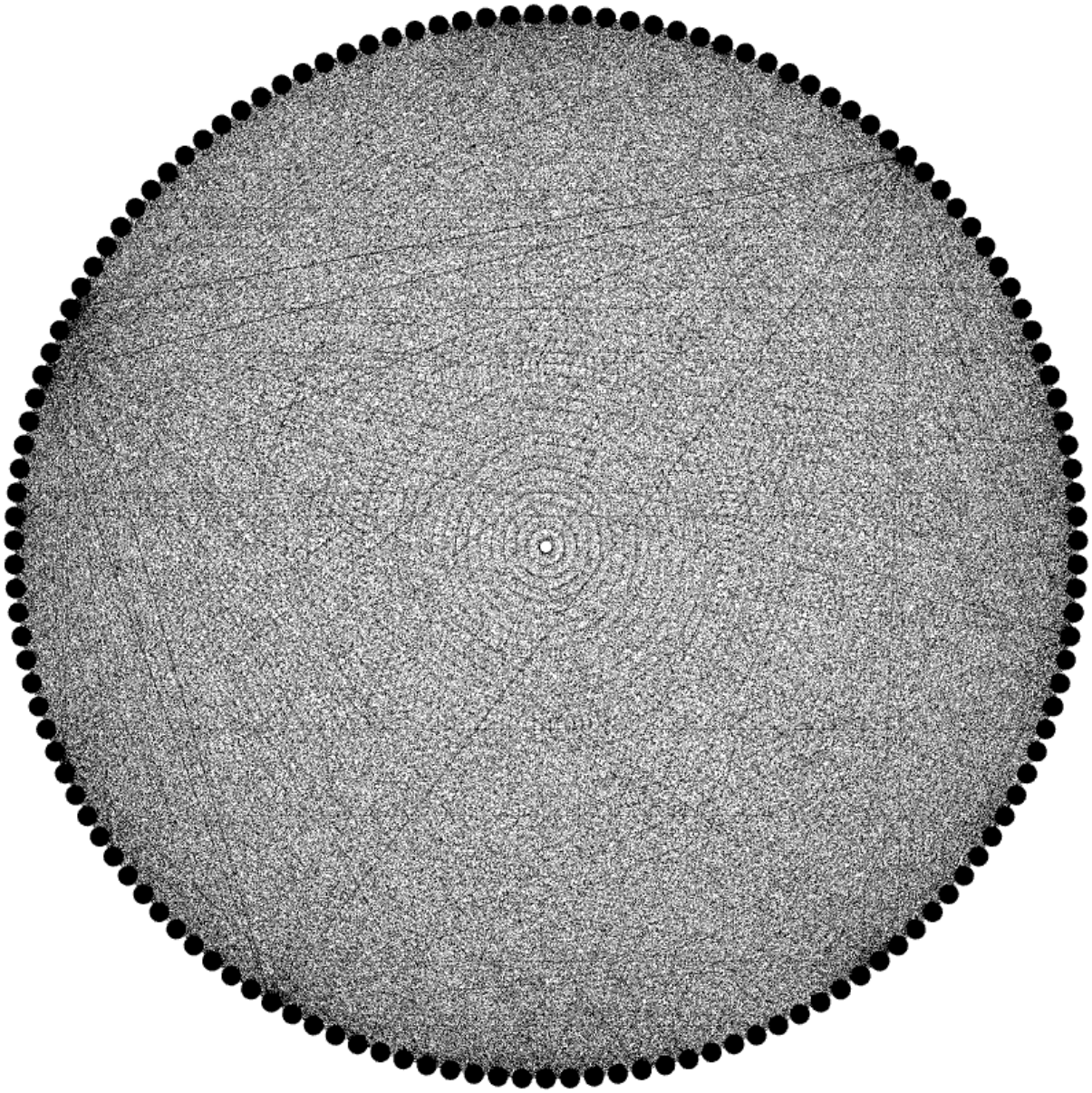


(d) Exemplo de grafo exibido pelo *web service*, sendo as setas vermelhas mostrando quem é bom contra quem, e as linhas continuas quem é bom com quem. O tamanho do circulo é a frequência daquela ligação e a cor é a força da ligação.

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.

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.

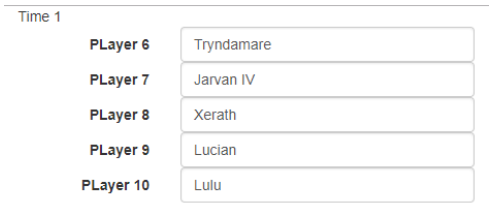
Fonte: Autor.

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 selecionar 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.

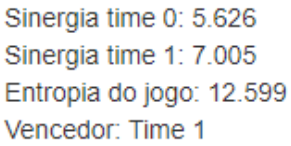
Com a matriz de confusão é possível ver as classificações corretas e as erradas por resultado, ou seja, na Tabela 2 é possível ver quantas vezes o KNN classificou sendo vencedor o time 0, porém era vencedor o time 1 ou qualquer outro cenário.

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

(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*se encontram no Github[8] para trabalhos futuros.

[8] 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.

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.