

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA
COMPUTAÇÃO GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO



**ESTUDO DE ALGORITMOS PARA ANÁLISE DE SENTIMENTOS EM JOGOS
ONLINE POR TÓPICOS DE CONVERSAÇÃO**

MARCOS ANTÔNIO DA SILVA NASCIMENTO

GOIANIA
2021

MARCOS ANTÔNIO DA SILVA NASCIMENTO

**ESTUDO DE ALGORITMOS PARA ANÁLISE DE SENTIMENTOS EM JOGOS
ONLINE POR TÓPICOS DE CONVERSAÇÃO**

Trabalho de Conclusão de Curso
apresentado à Escola de Ciências Exatas e
da Computação, da Pontifícia Universidade
Católica de Goiás, como parte dos
requisitos para obtenção do título de
Bacharel em Ciência da Computação.

Orientador:

Prof. Me. Aníbal Santos Jukemura

Banca Examinadora:

Prof. Me. Fernando Gonçalves Abadia

Prof. Me. Gustavo Siqueira Vinhal

MARCOS ANTÔNIO DA SILVA NASCIMENTO

ESTUDO DE ALGORITMOS PARA ANÁLISE DE SENTIMENTOS EM JOGOS
ONLINE POR TÓPICOS DE CONVERSAÇÃO

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciências da Computação, em ____/____/____.

Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca Examinadora:

Orientador: Prof. Me. Aníbal Santos Jukemura

Prof. Me. Fernando Gonçalves Abadia

Prof. Dr. Gustavo Siqueira Vinhal GOIÂNIA 2019

AGRADECIMENTOS

Muitas coisas aconteceram no caminho até aqui, coisas boas e ruins. O que todas elas tiveram como semelhança foi o fato que todas elas me levaram adiante, buscando sempre melhorar em algum aspecto, seja ele o de filho, o de estudante ou o de amigo.

Nesse raciocínio, gostaria de agradecer primeiramente e principalmente os meus pais, eles que mesmo com todas as dificuldades deram tudo de si para criar aos meus irmãos e a mim. Eles que em todos os momentos de minha vida acadêmica buscaram me apoiar ao máximo e de todas as formas possíveis.

Gostaria de agradecer também aos meus professores que buscaram me orientar e me preparar para diversas situações, professores estes que me ajudaram a percorrer todo o caminho até aqui.

Adicionalmente, queria agradecer meu orientador, Aníbal Santos Jukemura ("Jukes"), por todas as orientações e toda a paciência, não só nos últimos 10 meses de trabalho, mas pelos últimos anos os quais ele me acompanhou no ensino acadêmico.

Por fim, gostaria de agradecer aos meus amigos, que estiveram comigo durante todo esse tempo, rindo e se desesperando pelas mais diversas situações. Em especial gostaria de agradecer aos meus amigos que caminharam comigo durante grande parte do curso, Gabriel Giani e Paulo Neto, além de todo grupo de amigos da *Ownage*.

RESUMO

Com o crescimento do mundo virtual, os jogos online aparecem como uma nova fonte de entretenimento e competição. Dentro desse mundo virtual, onde a identidade do autor pode ser facilmente ocultada, diversos atos de toxicidade começaram a se tornar comuns nas comunidades de jogos online. Considerando tal cenário, este trabalho avaliou os algoritmos de agrupamento de tópicos, *Latent Dirichlet Allocation* (LDA) e K-Means para compará-los, a fim de buscar a técnica de agrupamento mais eficiente para a categorização de toxicidade dentro dos jogos. O objeto de estudo consiste em um *dataframe* contendo registros de *chats* de um dos maiores jogos do mundo, o DotA 2 (GUERRA, 2019). Através dos resultados obtidos, é possível observar a presença de cada um dos tópicos selecionados no *dataframe*, além das vantagens e desvantagens de cada um dos algoritmos estudados.

Palavras-chave: *Mineração de Texto; Técnicas de Agrupamento; Jogos Online.*

ABSTRACT

With the growth of the virtual world, online games appear as a new source of entertainment and competition. Within this virtual world, where the author's identity can be easily hidden, several acts of toxicity have started to become common in online gaming communities. Considering this scenario, this work evaluated the topic clustering algorithms, Latent Dirichlet Allocation (LDA) and K-Means to compare them in order to find a more efficient clustering technique for categorizing toxicity within games. The object of study consists in a dataframe containing chat records from one of the biggest games in the world, DotA2. Through the results obtained, that's possible to observe the presence of each selected topics in the dataframe, besides to the advantages and disadvantages of each of the possible algorithms studied.

Keywords: *Text Mining; Clustering Techniques; Online games.*

LISTA DE FIGURAS

Figura 1. Mapa do jogo DotA 2.....	19
Figura 2. Chat de texto do jogo Dota 2.....	19
Figura 3. Chat de voz do jogo DotA 2	20
Figura 4. Chat de pings do jogo Dota 2.....	20
Figura 5. Distribuição de tópicos e frases - LDA	23
Figura 6. Separação ideal de clusters - K-Means.....	24
Figura 7. Análise de Sentimentos baseada em polaridades.	26
Figura 8. Comparação entre os tópicos mais e menos presentes para cada emoção	32
Figura 9. Exemplo de WordCloud	41
Figura 10. Retorno de True para as palavras passadas como referência.....	43
Figura 11. Palavras com maiores pesos nas frases da base de aprendizagem.	44
Figura 12. Resultados gerais dos testes do agrupamento via LDA.....	48
Figura 13. Resultados dos testes para o tópico 1 - LDA	50
Figura 14. Resultados dos testes para o tópico 2 - LDA	51
Figura 15. Dataframe com as frases e seus respectivos tópicos relacionados.	51
Figura 16. Hiperparâmetros K-Means	52
Figura 17. WordCloud do dataframe estudado.....	54
Figura 18. Palavras mais frequentes do dataframe.....	55
Figuras 19. Distribuição dos tópicos no dataframe - LDA	57
Figura 20. Distribuição dos tópicos no dataframe - K-Means.....	57

LISTA DE TABELAS

Tabela 1. Tabela da Matriz de Confusão	25
Tabela 2. Distribuição dos tópicos nos grupos	31
Tabela 3. Comparação das performances médias de grupos com/sem a prevalência de um tópico.	31
Tabela 4. Tópicos - LDA.....	45
Tabela 5. Centroides - K-Means.....	53
Tabela 6. Palavras com maior relevância em cada tópico - LDA	56
Tabela 7. Palavras com maior relevância em cada tópico - K-Means.....	56
Tabela 8. Matriz de Confusão - LDA	58
Tabela 9. Matriz de Confusão - K-Means	59

LISTA DE SIGLAS

RPG	Role Playing Game
CSV	Comma Separated Values
GPU	Graphics Processing Units
HTML	HyperText Markup Language
API	Application Programming Interface
TF-IDF	Term Frequency – Inverse Document Frequency
NLTK	Natural Language ToolKit
AGGRO	Agressividade – Ter a atenção do inimigo voltado para você.
AOE	Area of Effect
MID	Mid-laner – Jogador da rota central em MOBAS
BOT	Bot-laner – Jogador da rota inferior em MOBAS
CD	Cooldown
CDR	Cooldown Reduction

SUMÁRIO

1. Introdução.....	12
2. Conceitos teóricos relevantes.....	17
2.1. MOBA.....	17
2.2. DOTA 2	18
2.3. <i>Machine Learning</i>	20
2.3.1. Processamento de Linguagem Natural	21
2.3.3. Modelagem de Tópicos: <i>Latent Dirichlet Allocation</i> (LDA)	22
2.3.4. <i>K-Means Clustering</i>	23
2.3.5. Métricas de Avaliação	25
2.4. Análise de sentimentos	26
3. Trabalhos relacionados.....	28
3.1. Comportamento tóxico	28
3.2. <i>Machine Learning</i> para identificação do Comportamento tóxico em jogos MOBA.....	30
4. Desenvolvimento do estudo.....	34
4.1. <i>Python</i>	34
4.2. Identificação de comportamento tóxico	38
4.3. Pré-processamento	39
4.4. Processo de Amostragem de Palavras mais frequentes	40
4.5. Processo de <i>Machine Learning</i> utilizado nesse trabalho.....	41
5. Resultados alcançados.....	54
5.1. Palavras de maior relevância	54
5.2. Discussão de Resultados	57
5.3. Matriz de Confusão	58
6. Conclusão.....	60
7. BIBLIOGRAFIA.....	62

APÊNDICE A – Código das implementações.....	70
Apêndice B – Links dos Arquivos	83

1. Introdução

Atualmente, jogos *online* são considerados um meio de entretenimento e relaxamento. No Brasil, 82% das pessoas entre 13 e 59 anos jogam em pelo menos um dispositivo (NPD Group, 2015) e o mercado de jogos em si não para de crescer. Só no Brasil, o mercado de jogos *online* movimenta por volta de US\$ 1,5 bilhão por ano, sendo o 13º maior mercado mundial (LARGHI, 2019).

Nos últimos anos, o mercado de jogos deixou de ser apenas focado em entretenimento e passou a ser visto também de forma profissional. Várias companhias começaram a montar times para competição em jogos que movimentam e premiam seus jogadores com milhares de dólares anualmente. (Confederação Brasileira de eSports – CbeSports, c2017)

Guerra (2019), em seu estudo, mostra que um dos gêneros de jogos mais populares atualmente é o MOBA (*Multiplayer Online Battle Arena*), que tem como principais representantes o *League of Legends* (LoL) e o *Defense of the Ancients 2* (DotA 2). Guerra (2019) mostra ainda que ambos contam com torneios anuais e até mesmo semestrais, com premiações que passam dos US\$ 177 milhões por ano, no caso do *DotA 2*.

Diferente de alguns estilos de jogos, o gênero MOBA é altamente competitivo, uma vez que a vitória nesse estilo depende de um trabalho em equipe eficiente e de estratégias bem definidas (NETO, 2019). Para isso, os jogadores usam o *chat* de texto ou o *chat* de voz (caso o jogo dê suporte) para possibilitar vários tipos de interações entre os jogadores. Durante essas interações, expressões de sentimentos podem surgir, sendo, infelizmente, nem todas positivas.

De acordo com Joaquim Alvino de Mesquita Neto (2019):

“Insultos, provocações e culpabilizações são a forma mais clássica de comportamento tóxico, mas outros comportamentos também são considerados ofensivos por jogadores online, como perder uma partida de propósito ou abandonar uma partida repentinamente e como resultado, o humor dos

jogadores na partida (principalmente do time do autor do comportamento tóxico) ficam deteriorados negativamente, atrapalhando a experiência de jogo.”

Ainda no contexto sobre o gênero MOBA, utilizando as ações de uma pessoa durante uma partida *online* como base do estudo, a Análise de Sentimentos (*Sentiment Analysis*) pode ser uma boa opção para ajudar a identificar diversos tipos de emoções, entre elas os comportamentos tóxicos.

Uma emoção é um conjunto de respostas químicas e neurais baseadas nas memórias emocionais, e surgem quando o cérebro recebe um estímulo externo. O sentimento, por sua vez, é uma resposta à emoção e diz respeito a como a pessoa se sente diante daquela emoção (Movimento Saúde, 2019).

Dentro desse campo de estudo, a Análise de Sentimentos vem ganhando espaço durante os últimos anos, tendo como principal atividade “identificar o sentimento que os usuários apresentam a respeito de alguma entidade de interesse.” (RODRIGUES et al, 2013).

Análise de Sentimento, segundo Silva, Barbosa Pandolfi e Cazella (2017) pode ser definido como:

“A Análise de Sentimento é um campo de estudo importante para a Computação Afetiva e para a compreensão dos processos cognitivos em geral. Dois aspectos embasam essa constatação: a web ter se tornado uma arena pública para difusão de opiniões; e a comunicação, desde a mais tenra idade, ser expressa a partir de múltiplas linguagens.”

Para categorizar os sentimentos apresentados pelos jogadores durante as conversas em uma partida MOBA, com o intuito de evidenciar o comportamento tóxico dos demais, serão utilizadas técnicas referentes à Ciência de Dados (*Data Science*).

Em um alto nível, a Ciência de Dados é um conjunto de princípios fundamentais que apoiam e orientam a extração de informações e

conhecimento dos dados com base em princípios. Um dos processos aplicados em Ciência de Dados é a Mineração de Dados, que busca extrair conhecimento de dados por meio de tecnologias que incorporem esses princípios (PROVOST; FAWCETT, 2013).

De maneira resumida, em geral, a execução de um protótipo baseado em critérios estabelecidos pela Ciência de Dados pode ser representada nos seguintes passos: (ZHAO, 2018)

1. Identificar um elemento de pesquisa;
2. Obter um *dataframe* e ajustá-lo para elaboração da Análise Exploratória;
3. Executar o EDA (*Exploratory Data Analysis* - Análise Exploratória de Dados);
4. Aplicar as técnicas (Algoritmos);
5. Compartilhar os resultados.

Considerando a Ciência de Dados como fator essencial para esse trabalho, busca-se responder as seguintes questões:

1. Quais os principais tipos de comportamento e sentimentos que ocorrem durante as partidas?
2. Que tipo de diferenças de resultados serão obtidos realizando os testes com algoritmos de Clusterização (*K-Means*) e Modelos de Tópico (LDA – *Latent Dirichlet Allocation*)?
3. É possível gerar resultados satisfatórios e suficientes para que possam ser utilizados posteriormente por outros mecanismos que buscam mitigar o comportamento tóxico em jogos?

É relevante estudar esse tema, pois cada vez mais pessoas entram no mundo dos jogos, fazendo com que seja importante saber como se comportam nesse ambiente virtual. Este comportamento pode ser visualizado pela empresa desenvolvedora do jogo e pela comunidade que consome o seu produto, a fim de melhorar o ambiente do jogo, já que muitos jogadores casuais são influenciados e até mesmo atrapalhados pelos jogadores definidos como tóxicos.

Os estudos apresentados nesse trabalho serão focados principalmente em evidenciar o comportamento tóxico apresentados pelos jogadores.

Comportamento tóxico, referido por Suler (2004) como uma presença constante em jogos online, ocorrendo quando um jogador quebra regras de convivência, agindo de maneira ofensiva.

Logo, diante desse contexto, esse trabalho apresenta como objetivo principal, o uso e comparação de resultados entre técnicas de Clusterização e Modelagem de Tópicos na identificação de padrões comportamentais que são mostrados durante partidas de jogos *online* com base nos *chats* do jogo DotA 2, identificando principalmente comportamentos tóxicos.

Em decorrência desse estudo serão explorados também os seguintes objetivos específicos:

- Categorizar os sentimentos identificados nas conversas dos jogadores, em especial, os sentimentos que definem um comportamento tóxico.
- Verificar a frequência de uso de sentimentos relacionados a comportamento tóxico nos chats do jogo DoTA 2.

Para fazer tais comparações e classificações, o trabalho passou pelas seguintes etapas:

- Definição do tema exato a ser estudado em relação a área de Inteligência Artificial, levando a escolha da *Machine Learning* e da Análise de Sentimentos.
- Seleção de um *dataframe* de chats de mais de 1600 partidas do jogo DotA2. Este que está de acordo com o tema do estudo, pois é possível visualizar tudo o que é “digitado” (exposto) durante as partidas de um dos jogos mais populares do mundo atualmente.
- Definição dos algoritmos que farão o agrupamento de dados para a geração dos resultados, baseando-se na forma de funcionamento similar, mas que usam técnicas de trabalho peculiares. Foi feita uma comparação do método de trabalho de Clusterização (*Clustering*) e do Modelo de Tópico (*Topic Modeling*) com seus representantes os respectivos algoritmos, K-Means e LDA.
- Execução de testes para identificar se há Superajuste ou Sobreajuste (*Overfitting*) ou Sub-ajuste (*Underfitting*) nos modelos.

- Por fim, foi feita uma discussão sobre os resultados obtidos para a validação dos dados, podendo essa parte da metodologia ser repetida, caso seja necessário.

Espera-se que os resultados obtidos nesse trabalho possam auxiliar na percepção de diferentes sentimentos presentes durante partidas de algum jogo *online*. Com os resultados, tem-se em vista a ajuda na obtenção de dados para pesquisas em áreas relacionadas com o tema proposto, facilitando a identificação dos padrões comportamentais dos jogadores.

O restante desse trabalho está organizado como segue. O Capítulo 2 traz conceitos importantes sobre *Machine Learning* e do estilo de jogo MOBA, para o entendimento e realização do trabalho. O Capítulo 3 traz a descrição dos principais trabalhos relacionados. O Capítulo 4 demonstra toda a metodologia de pesquisa aplicada, ambiente de implementação, e a descrição da solução. O Capítulo 5 mostra os resultados alcançados através das análises realizadas. O Capítulo 6 apresenta as conclusões sobre a pesquisa e apresenta sugestões de trabalhos futuros. Por fim, o Apêndice A apresenta os a visualização da codificação desenvolvida na pesquisa e o Apêndice B apresenta o *dataframe* e o *dataset* utilizados para o estudo proposto neste trabalho.

2. Conceitos teóricos relevantes

Neste capítulo, serão apresentados conceitos fundamentais para o entendimento do trabalho. O gênero de jogo MOBA será apresentado de maneira geral. Será apresentado também o jogo DotA 2, do qual foram retirados os *chats* usados como objeto de estudo.

Adicionalmente, serão apresentados conceitos envolvendo *Machine Learning* e os algoritmos utilizados. Por fim, serão esclarecidas definições sobre a Análise de Sentimentos e a classificação utilizada para o desenvolvimento do trabalho.

2.1. MOBA

MOBA é um estilo de jogo que envolve ação, estratégia e um pouco de RPG, tudo ao mesmo tempo (CAETANO, 2019). Um MOBA conta com dois times, cada um contendo 5 jogadores que lutam em um mapa apresentado geralmente de forma simétrica em busca da destruição da base inimiga.

Sua origem se deu na criação de um mapa personalizado para o jogo *Aeon of Strife* por fãs do jogo *StarCraft*. Com o sucesso do modo de jogo criado, o jogo *Warcraft III* também ganhou um mapa customizado em 2002, mapa o qual foi responsável por estabelecer as regras atuais da modalidade (CAETANO, 2019).

Como competição nos esportes eletrônicos, o MOBA é caracterizado como o gênero mais popular. Seus dois principais representantes – *League of Legends* e DotA 2 - estão firmemente consolidados neste meio, apresentando premiações milionárias anualmente (SOUZA, 2020)

O tempo de jogo de um MOBA pode variar dependendo da situação da partida, mas suas durações costumam ser em média por volta de 30 a 45 minutos. O mapa é dividido em 3 *lanes*: A *Top Lane* (caminho superior), a *Bottom Lane* ou *Bot Lane* (caminho inferior) e a *Mid Lane* (caminho central) de cada um dos times e o tempo de jogo se divide entre o *Early*, *mid* e *late game* (Início, meio e fim de jogo). Cada etapa do jogo visa um objetivo diferente,

desde a coleta e acúmulo de recursos iniciais, a lutas por grandes objetivos no mapa buscando invadir a base inimiga para finalizar a partida.

Nesse tipo de jogo, a estratégia é algo essencial para conquistar objetivos enquanto impede o time inimigo de avançar. A comunicação é algo de extrema importância, seja por *chat* de voz ou por *chat* de texto. Atualmente o meio de comunicação mais utilizado nos MOBAs são os *chats* de texto e os *pings* (Sinalizações dentro de jogo) (NETO, 2019).

2.2. DOTA 2

O DotA 2 é um MOBA desenvolvido pela Empresa **Valve** com base no jogo *Warcraft*, e teve seu lançamento em 2013 em sua plataforma atual, a *Steam* (DABUL, 2013).

O jogo conta com diversos heróis, cada um com suas peculiaridades, além de itens com os quais o jogador monta sua *build* (conjunto dos itens) durante a partida. Cada escolha de herói, juntamente de seus itens, pode mudar o seu desempenho. A estratégia empregada no jogo, também pode fazer com que estratégias de ataque e defesa tenham mudanças constantes durante a partida.

O mapa de DotA 2, mostrado na

Figura 1, é composto de dois lados: um para a facção *Radiant* e o outro para a facção *Dire*. Para ganhar, os jogadores devem destruir o *Ancient* inimigo, que é a estrutura mais importante do jogo. Em cada uma das *lanes* estão posicionadas torres que impedem o avanço a base que elas protegem. Separando ambos os lados do mapa, existe também um rio que é cercado por altos terrenos que bloqueiam a visão dos jogadores por ambos os lados. A *Jungle* do mapa se refere a área com árvores que ficam entre cada uma das 3 *lanes*. Na *jungle* é possível encontrar monstros neutros que são abatidos para poder obter experiência para seu herói e ouro para comprar seus itens.

De acordo com o site *Esports Earnings* (2020), o DotA 2 ocupa a posição mais alta no *ranking* de jogos que mais premiam, chegando a pagar anualmente mais de US\$ 231 milhões em competições competitivas. Em 2021 o jogo chegou a registrar mais de 1 milhão de jogadores simultaneamente.

Figura 1. Mapa do jogo DotA 2



(a) Mapa Completo.

(b) Disposição das lanes.

Fonte: (a) DOTA 2 WIKI, 2018; (b) Fonte: DOTA 2, 2014.

Os jogadores se comunicam no DotA2 a partir de *chat* de texto em que os jogadores podem utilizar em forma de um bate-papo para definir estratégias. O *chat* de texto é geralmente dividido entre um *chat* de visibilidade somente para aliados (membros do seu time) e outro *chat* de visibilidade global (qualquer um que esteja na partida poderá ler) como mostrado na Figura 2 e por *chat* de voz (Figura 3) representadas por balões de fala em um dos cantos da tela e em cima do personagem do jogador que estiver falando no momento.

Figura 2. Chat de texto do jogo Dota 2



Fonte: TU, 2018.

Figura 3. Chat de voz do jogo DotA 2



Fonte: TU, 2018.

Adicionalmente, existem o *chat* de *pings* com avisos pré-definidos assim como mostra a

Figura 4. Os *chats* de voz e *pings* só podem ser usados com o próprio time do jogador.

Figura 4. Chat de pings do jogo Dota 2



Fonte: DOTA 2 WIKI, 2021.

2.3. Machine Learning

Machine Learning é um ramo da Inteligência Artificial (IA) e da Ciência da Computação que é concentrada no uso de dados e algoritmos para imitar a forma como os humanos aprendem, melhorando gradativamente sua precisão (IMB, 2020).

O processo de aprendizagem começa com observações ou dados como exemplos, experiência direta ou instruções dadas, a fim de procurar padrões nos dados e tomar as melhores decisões com base nos exemplos fornecidos (EXPERT.AI, 2020).

Uma das vertentes importantes da Ciência da Computação, que apresentam elementos de *Machine Learning* é a área de Processamento Natural de Linguagem, que será explicada a seguir.

2.3.1. Processamento de Linguagem Natural

A definição de Linguagem Natural é bem simples: trata-se da pura forma como os humanos se comunicam. Os idiomas são um ótimo exemplo de Linguagem Natural (BERTIN, 2020).

Processamento de Linguagem Natural (PLN) é uma vertente da inteligência artificial que ajuda computadores a entender, interpretar e manipular a linguagem humana. O PLN resulta de diversas disciplinas, incluindo Ciência da Computação e Linguística Computacional, que buscam preencher a lacuna entre a comunicação humana e o entendimento dos computadores. (SAS, 2021)

Utilizando técnicas dessa vertente junto a Análise de Sentimentos, serão feitos os procedimentos necessários para classificação das frases presentes no *dataframe* estudado, de forma que seja possível identificar o comportamento tóxico apresentado pelos jogadores no *chat*.

Para esse fim, serão utilizados dois algoritmos que fazem parte desta vertente, mas que trabalham com técnicas de classificação diferentes: O LDA que utiliza de Modelos de Tópicos para fazer sua classificação, e o *K-Means* que utiliza a técnica de Clusterização.

2.3.2. Métodos Supervisionados e Não Supervisionado

Os algoritmos de *Machine Learning* dentro de suas regras, podem ser divididos em duas formas principais de aprendizado: Supervisionado e Não Supervisionado.

Os algoritmos de aprendizagem supervisionada aprendem através de exemplos e depois podem fazer previsões com base em um modelo (SANTOS, 2017), ou seja, ele aprende através de exemplos inseridos previamente para

quando receber novos dados, o programa poder tomar decisões precisas (MATOS, 2015).

Por outro lado, os algoritmos de aprendizagem não supervisionada são uma forma de aprendizado em que o modelo tenta inferir uma informação com base em uma estrutura oculta ou características de dados até então não classificados de nenhuma forma (SANTOS, 2017). Essas informações inferidas pelo modelo vão automaticamente determinar padrões e relações entre o conjunto de dados estudados (MATOS, 2015).

2.3.3. Modelagem de Tópicos: *Latent Dirichlet Allocation* (LDA)

Modelagem de Tópicos (*Topic Models*) é o processo de identificações de tópicos (agrupamento de itens) em um conjunto de documentos. O modelo encontra alguns grupos naturais de tópicos mesmo quando não se tem certeza do que está procurando. (KULSHRESTHA, 2019).

O LDA é uma abordagem probabilística não supervisionada que descobre os tópicos de um conjunto de documentos, e os descreve a partir da probabilidade de eles pertencerem a algum dos tópicos descobertos. Estes tópicos são descritos como uma distribuição de palavras. Por exemplo, um tópico A pode ser representado 40% pela **palavra_1**, 10% pela **palavra_2**, 8% pela **palavra_3**, e assim por diante, para todas as palavras no vocabulário (NETO, 2019).

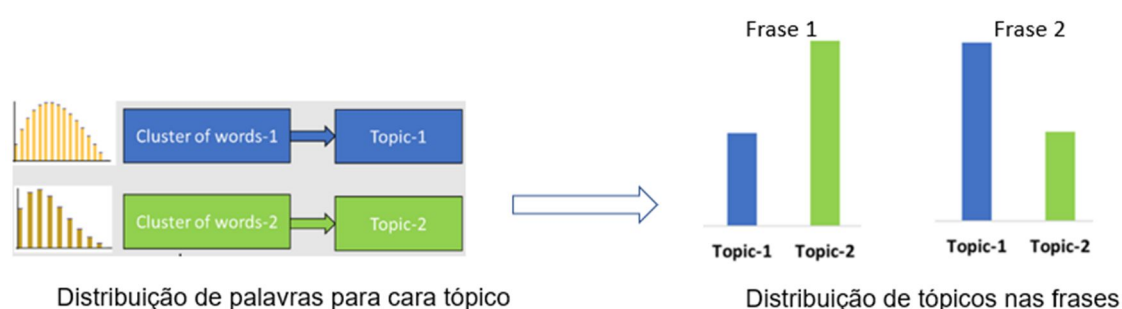
O LDA funciona da seguinte forma:

- O algoritmo percorre cada frase e atribui aleatoriamente a cada palavra, um dos k tópicos (k é pré-definido).
- Para cada frase, algoritmo:
 - A. Calcula a proporção de palavras na frase analisada que são atribuídas a cada um dos tópicos e, tenta capturar quantas palavras pertencem a cada tópico. Ao final, com exceção da palavra que está sendo analisada no momento, se várias palavras da frase pertencerem a um tópico t , as chances da palavra analisada também fazer parte deste tópico aumenta;

- B. Calcula a proporção de frases que o tópico t foi atribuído em razão da palavra analisada e assim, o algoritmo tenta capturar quantas frases estão no tópico t por causa desta palavra.

O algoritmo LDA representa um tópico é uma mistura de palavras. Da mesma ele apresenta as frases como uma mistura de tópicos (Figura 5). Se uma palavra w tem alta probabilidade de estar em um tópico t , todos os documentos com w também estarão mais fortemente associados a t . Assim, se a w tem uma baixa probabilidade de estar em t , as frases que contêm w , terão uma probabilidade muito baixa de estar em t , porque o resto das palavras na frase pertencerão a algum outro tópico e, portanto, a frase terá uma maior probabilidade para esse outro tópico. Ao final, é atualizada a probabilidade da palavra analisada pertencer ao tópico t (KULSHRESTHA, 2019).

Figura 5. Distribuição de tópicos e frases - LDA



Fonte: Grat Learning Team, 2020.

2.3.4. K-Means Clustering

A clusterização é uma das técnicas de Análise de Dados mais comumente usadas para obter uma intuição sobre a estrutura dos dados. Esta técnica pode ser definida como a tarefa de identificar subgrupos nos dados de forma que os pontos de dados no mesmo subgrupo (*cluster*) sejam semelhantes, enquanto os pontos de dados em diferentes *clusters* são diferentes. (DABBURA, 2018)

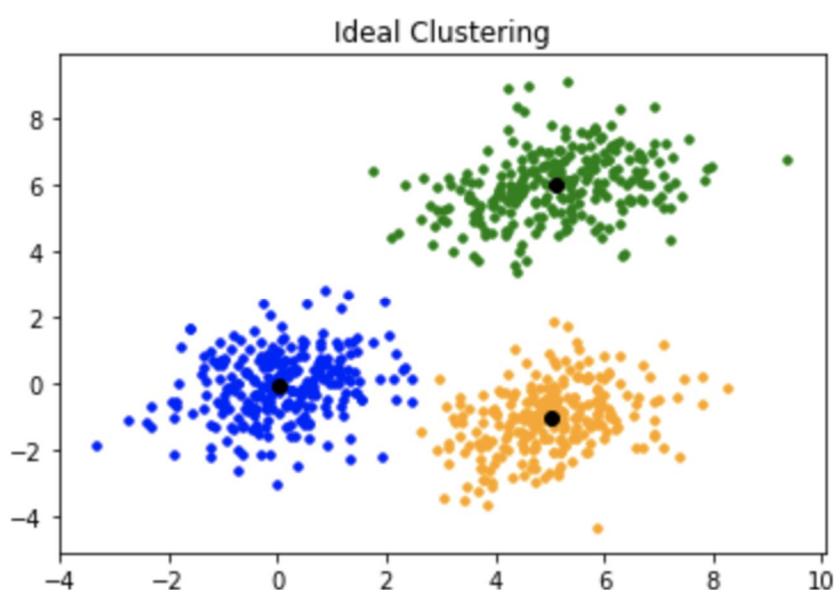
Dentre os algoritmos de clusterização mais utilizados está o *K-Means* (BROWNLEE, 2020). O algoritmo *K-Means* é um algoritmo de aprendizagem

não-supervisionado e interativo que tenta particionar os dados em subgrupos distintos, onde cada um dos pontos pertence somente a um *cluster*.

Durante este processo, o algoritmo tenta manter os pontos que fazem parte de um mesmo *cluster* o mais perto possível, e os pontos centrais de cada *cluster* (centroides) o mais distante um do outro, assim como mostra a Figura 6.

O *K-Means* recebe como entrada a quantidade de agrupamentos que ele deve encontrar, a métrica a ser utilizada para estabelecer as distâncias entre os dados, bem como os dados em forma vetorial. Após isso o algoritmo seleciona os seus centroides de acordo com a quantidade de *clusters* selecionados. Esta escolha é feita de forma aleatória dentre os dados presentes no *dataframe*.

Figura 6. Separação ideal de clusters - K-Means



Fonte: ML | *K-Means++ Algorithm*, 2020.

Com os centroides selecionados, o algoritmo atribui cada um dos pontos ao centroide mais próximo. Quando os *clusters* estiverem devidamente formados, é recalculado os centroides dos *clusters*. Este processo de agrupar os pontos aos seus centroides mais próximos e recalcular os centroides é feito diversas vezes.

A seguir serão apresentadas as métricas de avaliação, que tem como objetivo efetivar a qualidade do modelo proposto assim como seu percentual de

acerto para que seja possível visualizar se os algoritmos conseguiram atender os requisitos propostos.

2.3.5. Métricas de Avaliação

Para medir o desempenho e precisão dos modelos desenvolvidos aplicando técnicas de *Machine Learning*, é necessário usar uma ou mais métricas de avaliação para avaliar a capacidade de acerto dos modelos.

Neste trabalho são utilizadas duas métricas de avaliação: a Acurácia (*Accuracy* / Taxa de Acerto) e a Matriz de Confusão. A acurácia é uma métrica que indica a taxa percentual de acerto dos algoritmos.

Em problemas de classificação binária é utilizada uma matriz de tabulação cruzada (Matriz de Confusão) dos resultados preditos com as classes originais observadas. Essa matriz busca entender a relação entre acertos e erro que o modelo apresenta (NOGARE, 2020). A Matriz de Confusão é uma matriz 2x2 que representa categorias de classificação a partir de 4 notações, que são ilustradas na Tabela 1:

Tabela 1. Tabela da Matriz de Confusão

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: NOGARE, 2020.

- VP = Verdadeiro positivo – Significa que a classe prevista e observada originalmente faz parte da classe positiva;
- FN = Falso Negativo – Significa que a classe predita retornou positivo, mas a original observada era negativa;
- FP = Falso Positivo – Significa que os valores preditos e observados fazem parte da categoria negativa;

- VN = Verdadeiro Negativo – Significa que o valor predito resultou na classe negativa, mas o original observado era da classe positivo.

2.4. Análise de sentimentos

Análise de Sentimento (também conhecida como Mineração de Opinião) é o processo de extrair informações selecionadas de um texto e determinar a atitude que o escritor está tentando expressar através da linguagem (THOMPSON, 2017). O processo pode ser utilizado, por exemplo, para saber se um produto de uma empresa está sendo bem recebido pelos usuários.

Nesse trabalho será aplicada uma das formas mais simples de análise de sentimentos, que é a baseada em polaridades, ou seja, verificar se os sentimentos nas frases são positivos (não-tóxicos) ou negativos (tóxicos) assim como mostra a Figura 7. Essa medida pode ser tanto um número real, como um valor discreto (NETO, 2019).

Figura 7. Análise de Sentimentos baseada em polaridades.



Fonte: Binds.co, 2020.

Para realizar a classificação, seria necessário um *dataset* com frases classificadas com tais emoções relacionadas ao ambiente de jogos *online*. Porém, dadas as dificuldades para atingir tais requisitos, o *dataset* selecionado para o trabalho foi classificado de forma manual para atender a demanda da pesquisa.

No próximo capítulo, serão apresentados quatro trabalhos que exploram o tema de comportamento tóxico identificados em ambientes virtuais e em

jogos do tipo MOBA. É apresentado também, um trabalho que utiliza técnica de *Machine Learnig* para identificação e classificação de sentimentos durante as partidas.

3. Trabalhos relacionados

Neste capítulo serão abordados os trabalhos que tem como base de estudo o comportamento tóxico em ambientes online e suas diferentes classificações, assim como as causas e fins deste tipo de comportamento.

As principais bases de pesquisas utilizadas foram a *Mary Ann Liebert. Inc.*, *Google Scholar* (para pesquisa de Artigos e teses) e *Kaggle, Reddit* (para pesquisa de *dataframes*).

3.1. Comportamento tóxico

Dentre os diversos tipos de locais em que as pessoas apresentam algum tipo diferente de comportamento, em seu artigo, John Suler (2004) procura mostrar como as pessoas se comportam em redes *online*. Ele busca entender como as pessoas se comportam em um lugar onde suas identidades são inicialmente anônimas.

Mesmo que não exista um termo fixo para definir o comportamento tóxico, em seu trabalho, Suler (2004) pesquisa como o comportamento vindo de um anonimato pode se tornar tóxico, chamado por ele de **desinibição tóxica**, onde é observado uma linguagem rude, críticas severas, raiva, discursos de ódio e até mesmo ameaças. Este comportamento age em contrapartida à **desinibição benigna** que representa comportamentos onde as pessoas compartilham de emoções, medos e até mesmo podem se desviar de seu objetivo para ajudar os outros.

Suler (2004) em seu trabalho pesquisa, em elementos do ciberespaço, causas e efeitos que podem levar a essa desinibição. Além disso, mostra alguns fatores como, anonimato, invisibilidade e falta de autoridades em meios *online*, que geram o efeito dessa desinibição quando interagem entre si.

Outros trabalhos são discutidos conforme as três categorias em que a literatura pré-existente costuma entender o comportamento tóxico em ambientes *online* (FRAGOSO, 2015).

Estes comportamentos em jogos *online* são o **Spam**, que consiste em usar em quantidades abusivas recursos de comunicação dentro do jogo (como

por exemplo: *Pings* e *chats*). “**Trollagem**”, que tem como objetivo provocar um ou mais participantes de um ambiente *online* com a intenção de incitar discordâncias e confrontos através de comportamentos impertinentes, inferiorização ou ridicularização; e **Griefing**, que nas interações *online* designam jogadores (e suas ações) que perturbam os demais, tornando desagradável, dolorosa ou até traumática a experiência de jogo. (FRAGOSO, 2015. KURTZ, 2016).

Fragoso (2015) busca em seu trabalho, estudar os comportamentos na comunidade brasileira, denominado por ela como “HUEs”, através das categorias de *troll*, *spam* e *griefing*. Conforme o avanço da pesquisa foi mostrado que os brasileiros são “malvistos” em diversas partes da internet, onde se destacam por seu comportamento único. Desta forma, a comunidade brasileira busca refinar cada vez mais seus métodos de comportamento tóxico onde afirmam segundo a autora: “Nada na internet é tão sério que não possa virar motivo de riso e nada é mais risível do que as pessoas que pensam o contrário”.

Kurtz (2016) realizou uma pesquisa em que analisa seis vídeos de partidas de DotA 2 sob a ótica do *griefing*. Nos vídeos analisados, há comportamentos onde é possível identificar alguma das três categorias. Em todos os casos, houve algum tipo de comportamento que atrapalhou de alguma forma a experiência alheia no jogo.

Tais comportamentos foram vistos por Kurtz (2016) pela visão do *software*, visando a sua importância para detecção de atos como este. Após as análises, a autora pontua também a necessidade de interação humana nas análises dos dados para análises mais completas e precisas, já que ocorrem casos onde são enviadas provas de comportamento tóxico para análise e o *software* sozinho não consegue identificar tal comportamento.

Em seu trabalho, Fernandez (2017) busca compreender psicologicamente estes fenômenos. A autora tem como objetivo aumentar o conhecimento científico sobre o assunto dentro dos jogos *online*, já que os jogos atualmente ocupam uma boa parte da vida de diversas pessoas (FERNANDEZ, 2017).

Fernandez (2017) também mostra que o comportamento tóxico é recorrente nos jogos de gênero MOBA e que este tipo de comportamento cresce cada vez mais, seja pelo anonimato ou pelos fatores associados à desinibição *online*. É mostrado que dentro dessa comunidade MOBA, os jogadores apresentam cada vez mais comportamentos agressivos e desrespeitosos em exibições *online*.

Esse estudo tem como base, discussões realizadas a partir de pesquisas em cima das narrativas da literatura sobre o tema de desinibição tóxica que está presente nos meios *online* atualmente. Esse tipo de discussão, são encontradas abundantemente em jogos *online*, buscando responder o que são e o que ocasiona estes comportamentos.

Os trabalhos estudados buscam mostrar as formas de toxidade apresentadas no mundo *online* e principalmente nos jogos que fazem parte desde mundo, evidenciando as possíveis classificações para as emoções presentes no ambiente tóxico e fora dele. A partir desses estudos, os *chats* serão analisados e classificados de maneira que os algoritmos propostos possam obter um percentual satisfatório de precisão na classificação proposta.

3.2. Machine Learning para identificação do Comportamento tóxico em jogos MOBA

Neto (2019) em seu trabalho, busca caracterizar e prever comportamentos considerados tóxicos usando como base o *chat* do jogo *League of Legends* (LoL). A partir da extração de tópicos de texto de sua base de dados, ele compara os vocabulários utilizados pelos jogadores tóxicos e não-tóxicos.

Ele busca através destes tópicos encontrados, descobrir padrões utilizados por jogadores, e compreender o que ocorre em suas partidas. Com seus resultados, Neto (2019) busca diminuir o comportamento tóxico no ambiente estudado a partir de implementações de medidas que visam reduzir esse tipo de comportamento. Tais implementações seriam realizadas por empresas desenvolvedoras de jogos, buscando o controle da toxidade nos *chats* de seus jogos.

Na Tabela 2, estes padrões são encontrados em forma de tópicos de conversação entre os jogadores, que foram divididos entre aliados e inimigos. Estes tópicos variaram entre táticas, educação, bate-papo, reclamações, discussões, insultos e provocações, mostrando a distribuição dos tópicos de acordo com cada tipo de jogador, onde cada entrada indica a taxa de uso de um tópico para cada tipo de grupo. Observa-se por exemplo, que a soma total de Tópicos Negativos no estudo realizado por Neto (2019) foi de 43%, onde: 19% foram de reclamações, 13% de discussões, 7% insultos e 4% de provocações realizadas durante as partidas.

Foi observado também, as diferenças apresentadas no desempenho dos jogadores com e sem a prevalência de alguns dos tópicos. Na Tabela 3 é mostrado que o comportamento tóxico apresentado dentro do jogo afeta a desempenho geral da equipe do jogador que apresenta tal comportamento.

Tabela 2. Distribuição dos tópicos nos grupos

Tópicos	Todos	Inimigo	Não Cont.	Inimigo Cont.	Aliado	Ofensor
Tópicos Positivos	57%	81%	86%	73%	55%	35%
<i>Rel. a Táticas</i>	44%	64%	66%	55%	44%	25%
<i>Táticas</i>	21%	24%	26%	22%	22%	16%
<i>Táticas/Educação</i>	24%	40%	44%	33%	22%	9%
<i>Rel. a Humor</i>	36%	57%	58%	51%	33%	19%
<i>Bate-papo</i>	13%	17%	17%	18%	11%	10%
Tópicos Negativos	43%	18%	14%	27%	45%	65%
<i>Reclamações</i>	19%	6%	6%	8%	20%	30%
<i>Discussões</i>	13%	7%	4%	13%	16%	15%
<i>Insultos</i>	7%	3%	2%	3%	6%	13%
<i>Provocações</i>	4%	2%	1%	3%	3%	7%

Fonte: NETO, 2019.

Tabela 3. Comparação das performances médias de grupos com/sem a prevalência de um tópico.

Tópicos	Todos	Não Cont.	Inimigo Cont.	Aliado	Ofensor
Tópicos Positivos	0,104 / 0,081	0,122 / 0,109	0,113 / 0,093	0,090 / 0,077	0,089 / 0,078
<i>Rel. a Táticas</i>	0,103 / 0,087	0,122 / 0,116	0,113 / 0,102	0,089 / 0,081	0,088 / 0,080
<i>Rel. a Humor</i>	0,107 / 0,087	0,124 / 0,114	0,114 / 0,101	0,093 / 0,080	0,089 / 0,080
Tópicos Negativos	0,081 / 0,104	0,109 / 0,122	0,093 / 0,113	0,077 / 0,090	0,078 / 0,089
<i>Reclamações</i>	0,077 / 0,097	0,104 / 0,121	0,087 / 0,109	0,073 / 0,086	0,076 / 0,084
<i>Discussões</i>	0,082 / 0,095	0,113 / 0,120	0,097 / 0,109	0,078 / 0,085	0,077 / 0,082
<i>Insultos</i>	0,084 / 0,095	0,108 / 0,121	0,093 / 0,108	0,080 / 0,085	0,076 / 0,084

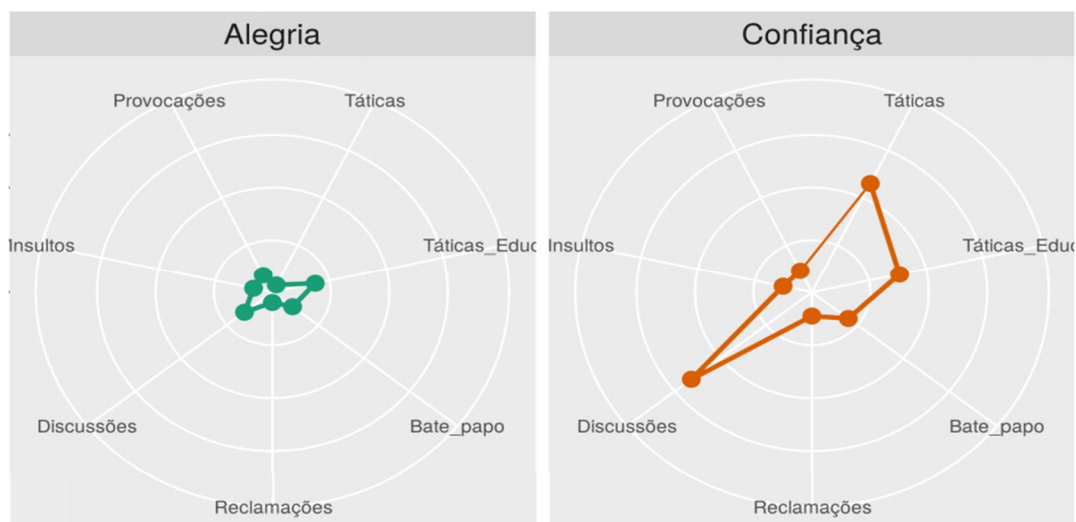
Fonte: NETO, 2019

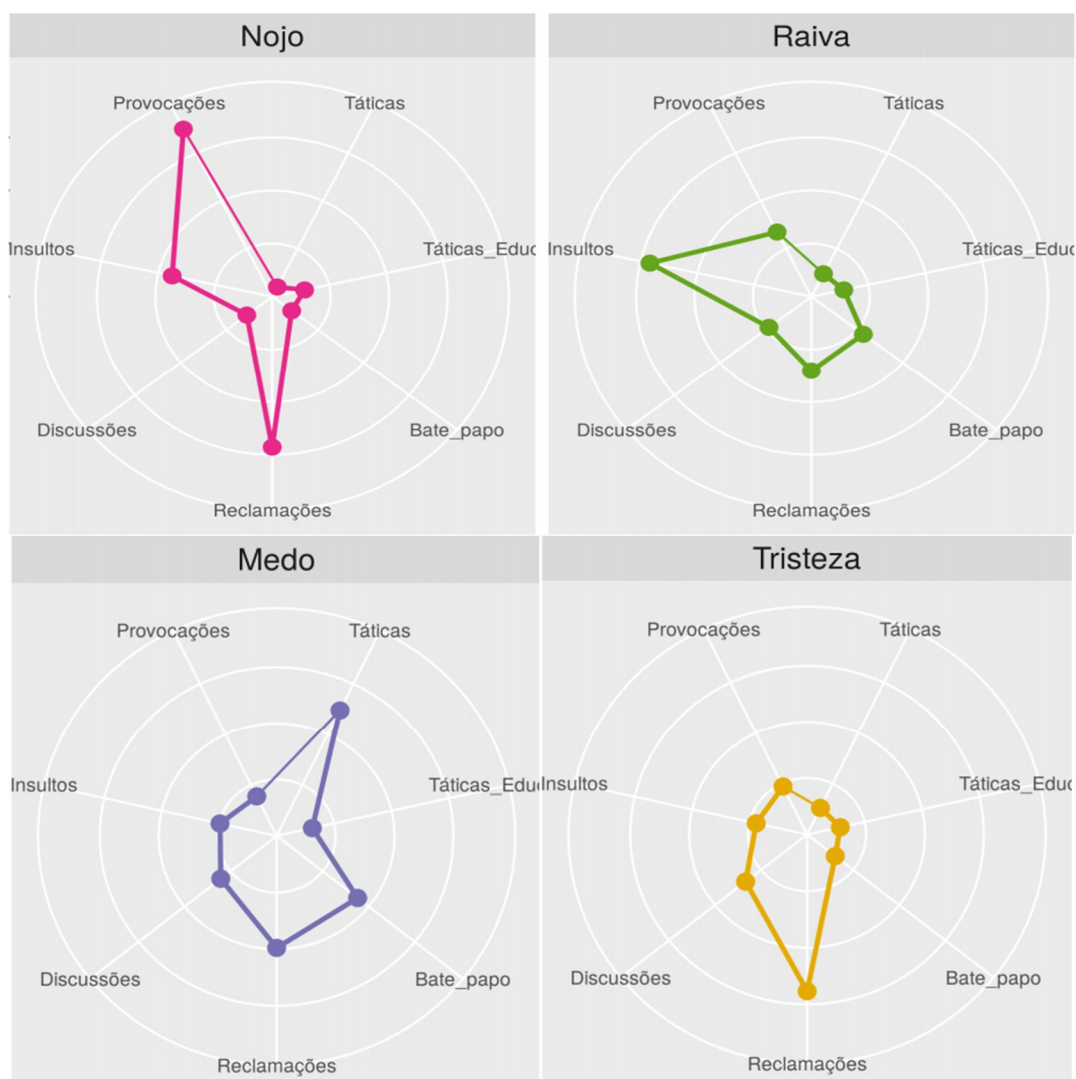
A Tabela 3 destaca, para cada tópico e tipo de grupo de jogadores, o desempenho médio considerando todos os grupos onde aquele tópico é prevalente e o desempenho médio nos outros grupos (ou seja, grupos onde outros tópicos são prevalentes). O formato de cada célula representa “desempenho médio do tópico” / “desempenho médio dos outros tópicos”. Por exemplo, a primeira célula da Tabela permite comparar o desempenho médio de todos os grupos (Todos) relacionados a um tópico positivo (0,104) com o desempenho médio de todos os outros grupos (0,081), que neste caso são os relacionados a um tópico negativo (NETO, 2019).

Em uma análise mais profunda, Neto (2019) busca desenvolver técnicas mais eficientes no combate ao comportamento tóxico observando o comportamento geral dos jogadores, de forma que os jogadores não-tóxicos não sejam prejudicados.

Finalmente Neto (2019), em suas análises dos padrões encontrados nas conversas dos jogadores, demonstra como esse comportamento tóxico afeta cada uma das equipes e como mostra a Figura 8. Ele investiga também quando cada comportamento é mais predominante, ou seja, é apresentado o índice de presença de cada tópico em cada uma das emoções estudadas por ele. Por exemplo, é possível observar que o tópico de Discussões tende a aparecer em maior frequência em emoções positivas, como alegria e confiança. Segundo Neto (2019), isso se deve ao fato de que este tópico busca conter o comportamento tóxico, possivelmente buscando resolver algum tipo de conflito.

Figura 8. Comparação entre os tópicos mais e menos presentes para cada emoção





Fonte: NETO 2019.

Por fim, Neto (2019) demonstra que há dois principais patamares nos valores de emoções dos tópicos: os positivos apareceram com menos emoções e os negativos, que apresentaram mais emoções. Cada sentimento foi caracterizado a um destes tópicos de acordo com a maneira que cada um se associa.

4. Desenvolvimento do estudo

Esse capítulo descreve todo o desenvolvimento do trabalho. São apresentados o ambiente de desenvolvimento, os algoritmos implementados, e como foi desenvolvida toda a solução do estudo feito.

4.1. *Python*

Python é uma linguagem de programação de alto nível bastante versátil, suportando a Programação Orientada a Objetos e a estruturada. Criada em 1991, o *Python* atualmente possui código aberto que é gerenciado pela *Python Software Foundation*. O *Python* é uma linguagem poderosa e simples. A linguagem contém estruturas de dados de alto nível e bastante eficientes, além de uma abordagem simples, porém eficaz para a Programação Orientada a Objetos.

Algumas de suas principais características está em sua leitura mais simples e a necessidade de poucas linhas de código quando comparada a outras linguagens.

Dentro do universo da Ciência de Dados, o *Python* é a linguagem de programação que mais cresce no mercado. De acordo com a pesquisa de Piatetsky (2016) sobre as principais ferramentas de análise em Ciência de Dados, a linguagem R ainda estava no topo da lista de ferramentas, porém a participação do *Python* no mercado aumentou em 51%, demonstrando sua influência neste campo de estudo.

Outra característica que faz o *Python* ser um diferencial na área de ciência de dados são suas bibliotecas. A linguagem fornece diversas bibliotecas e estruturas para a análise de dados. Neste trabalho as principais bibliotecas utilizadas foram: *Pandas*, *NumPy*, *Matplotlib*, *Scikit-Learn*, *pyLDAvis*, *Seaborn* e *NLTK*.

4.1.1. *Pandas*

Pandas é uma biblioteca que é utilizada como uma ferramenta para ler e gravar dados entre estruturas de dados na memória em diferentes formatos

como: Arquivos CSV, arquivos de texto, Microsoft Excel, banco de dados SQL e o formato rápido HDF5;

A biblioteca atualmente possui um alinhamento inteligente de dados e um manuseio integrado de dados perdidos, uma remodelagem de conjunto de dados, manuseio de *Data Frames* etc.

A biblioteca *Pandas* foi utilizada no trabalho para a manipulação geral do *dataframe* e do *dataset* utilizados para o estudo. Essa manipulação dos dados contou principalmente com a leitura dos arquivos CSV, alinhamento dos dados, tratamento de dados ausentes, inserção e retirada de colunas, junção de conjunto de dados e filtro de dados.

4.1.2. NumPy (Numerical Python)

O *NumPy* é um projeto de código aberto que visa habilitar a computação numérica na linguagem de programação *Python*. A biblioteca foi criada em 2005, com base no trabalho anterior das bibliotecas *Numeric* e *Numarray*.

NumPy é um pacote fundamental para a computação científica em *Python*. É uma biblioteca padrão quando se trabalha com dados numéricos em *Python* e está sempre presente em estudos científicos com *Python* (NUMPY, 2021). A biblioteca é usada extensivamente junto com a *Pandas*, *SciPy*, *matplotlib*, *Scikit-Learn*, *Scikit-image* e com a maioria dos outros pacotes de ciência de dados em *Python*.

A biblioteca fornece um objeto matriz multidimensional e estruturas de matriz de dados. Ela fornece um *ndarray*, um objeto de array n-dimensional homogêneo, com métodos para operar com eficiência nele.

Neste trabalho a biblioteca foi utilizada principalmente para apresentação de gráficos, com funções para cálculos de contagem em *array*, juntamente da biblioteca *Matplotlib*.

4.1.3. Matplotlib

A biblioteca *Matplotlib* é uma biblioteca para a criação de formas de visualizações estáticas, animadas e interativas em *Python*. A biblioteca produz

gráficos e figuras com qualidade e com uma grande variedade de formas em várias plataformas.

Utilizando o *pyplot*, o *Matplotlib* traz uma coleção de funções que o faz funcionar como o MATLAB (MATLAB, 2021). Cada função *pyplot* faz alguma alteração diferente em uma figura. Por exemplo: criar uma figura, criar uma área de plotagem em uma figura, decora o gráfico com rótulos etc.

Junto aos resultados obtidos pelo *NumPy*, o *Matplotlib* também foi utilizado para a apresentação de gráficos no trabalho, além de uma visualização geral dos dados.

4.1.4. Scikit-Learn (sk-learn)

O sk-learn é uma biblioteca de *machine learning* de código aberto e que dispõe de ferramentas para análise preditiva de dados, trabalhando juntamente com as bibliotecas numéricas e científicas do *Python*.

A sua principal aplicação é a criação de modelos de aprendizagem de máquina, sendo organizado em vários algoritmos de diversas finalidades, como por exemplo: pré-processamento, classificação, regressão, clusterização, agrupamento etc.

Durante este trabalho a biblioteca foi utilizada para a criação dos modelos do LDA e *K-Means*, além da classificação dos dados dos modelos.

4.1.5. pyLDavis

O *pyLDavis* é uma biblioteca do *Python* que é usada para visualização de modelos de tópicos interativos. O *pyLDavis* foi projetado para ajudar os usuários a interpretar os tópicos em um modelo que foi ajustado em um conjunto de dados de texto. Esta biblioteca extrai informações de um modelo de tópicos LDA ajustado para informar uma visualização interativa.

Como as versões mais novas do *pyLDavis* não funcionam bem no ambiente do *Google Colab*, foi utilizada a versão 2.1.2 neste trabalho para fins de reprodução dos gráficos.

Usado para visualização dos dados apresentados pelo modelo LDA, o *pyLDAvis* apresentou de forma dinâmica os resultados de cada um dos tópicos classificados pelo modelo.

4.1.6. Seaborn

Seaborn é outra biblioteca para visualizações de dados no *Python*. Essa, no entanto é baseada no *Matplotlib* e fornece uma interface para gráficos estatísticos, que consegue trazer de forma clara os resultados dos algoritmos estudados.

Assim como a biblioteca *Matplotlib*, o *Seaborn* foi utilizado para a apresentação de gráficos de amostra dos resultados obtidos com os cálculos realizados pelo *NumPy*.

4.1.7. NLTK

O NLTK (*Natural Language Toolkit*) é um conjunto de ferramentas para construção de programas em *Python* para trabalhar com dados de linguagem humana. Ele fornece interfaces fáceis de usar, recursos lexicais além de um pacote de bibliotecas de processamento de texto para classificação, tokenização, lematização, marcação, análise e raciocínio semântico. Foi usada no trabalho para a criação dos modelos de *Machine Learning* para a classificação e avaliação dos algoritmos implementados.

Toda implementação do trabalho foi realizada no Google *Colaboratory*, também conhecido como *Google Colab*. No *Colab* foi possível criar *notebooks*¹ para a implementação do trabalho sem nenhuma configuração anterior necessária, com acesso gratuito a GPUs dos servidores Google além de garantir um fácil compartilhamento do documento, podendo ser acessado facilmente de qualquer máquina. Os *notebooks* do *Colab* são armazenados em sua conta do Google *Drive* (nuvem), podendo ser compartilhados e acessados a qualquer momento.

¹ *Notebooks* são ambientes que permitem combinar código executável com textos, imagens, HTML etc (Google Colab, s/d).

O *Colab* vem se tornando uma API² onde é possível utilizar todas as suas bibliotecas de *Python* para a análise de dados. Para realizar estas análises, é possível fazer a importação de dados para o *notebook* e executar todo o projeto na nuvem, utilizando o *hardware* do Google independente da máquina que esteja utilizando.

Antes de implementar os algoritmos LDA e *K-Means*, é necessário esclarecer como ocorre o processo de identificação de um comportamento tóxico.

4.2. Identificação de comportamento tóxico

Foi citado inicialmente que um dos principais meios de comunicação em jogos *online* são os *chats* de texto dentro do jogo. Muitas vezes, a maneira como os jogadores estão se sentindo e seus pensamentos são expressos pela ferramenta. Em trabalhos relacionados (NETO, 2019; FRAGOSO, 2015; KURTZ, 2016) mostram que tal meio de comunicação tem alto valor quando se analisa o comportamento dos jogadores no jogo.

A toxidade na comunidade do jogo DotA 2 não é recente. Além disso, é um problema que se estende a outros jogos e que, conseguiu ganhar uma certa reputação por ele (ESTNN, 2021). Para identificação dos comportamentos tóxicos nestes *chats*, é necessário identificar antes o que seria caracterizado como comportamento tóxico e não-tóxico dentro do jogo para assim, conseguir separar bem os dois tipos de comportamentos e tentar identificá-los.

Para conseguir chegar nos resultados desejados, algumas barreiras foram encontradas, começando pelo próprio tipo de texto. As conversas de jogadores em MOBAs são cercadas por gírias, códigos, abreviações (Aggro, AoE, Mid, Bot, CD, CDR etc.) e nomes referentes a coisas do próprio jogo (como nome de campeões e itens por exemplo), o que acaba por dificultar o processamento dessas frases. Para definição de tais palavras, foi buscado em um fórum feito de jogadores para jogadores, os significados das mais comumente utilizadas nas partidas (DOTA 2, 2013).

² APIs são um conjunto de padrões que fazem parte de uma interface que permite a criação e acesso às funcionalidades de um aplicativo por um outro *software*. Desta forma dois sistemas diferentes “conversam” entre si e compartilham informações (DOCUSIGN, 2018).

Outra dificuldade encontrada ao longo do trabalho foi a falta de um base de emoções voltado para jogos, ou seja, uma base onde abreviações contidas na maioria dos jogos e até mesmo palavras de conteúdo impróprio ou ofensivo são levadas em consideração. Para contornar esta dificuldade, foi selecionada uma parcela com aproximadamente 3000 frases que estavam presentes no *dataframe* estudado. Assim, essas frases foram utilizadas, para criar o *dataset* a ser utilizado na implementação do classificador utilizado para classificação e avaliação dos modelos selecionados.

O *dataframe* selecionado precisou passar pela fase de pré-processamento de informações que será descrita a seguir.

4.3. Pré-processamento

Antes de iniciar o estudo dos dados, era necessária uma filtragem deles, pois além das gírias e códigos citados anteriormente, os dados contavam com diversos caracteres que não se fazem presentes no alfabeto latino (alfabeto utilizado na Língua Inglesa).

Para o processamento dos dados, foi utilizado como base a mesma linha de procedimentos (com algumas alterações) utilizadas por Neto (2019) em seu trabalho. Os passos a seguir descrevem todo o pré-processamento utilizado:

- **Retirada de caracteres que não fazem parte do alfabeto latino:** como tais dados dificultariam o manuseamento de todo *dataframe*, foram retirados no momento do *upload* do *dataframe* para o ambiente de trabalho.
- **Diminuição do documento trabalhado:** devido ao fato do *Colab* ser um ambiente virtual grátis para uso da comunidade, ele apresenta um limite de *hardware*. Por este limite ser atingido durante o trabalho, o *dataframe* “original” (que continha cerca de 1.4 milhões de frases) foi diminuído para cerca de 5% do seu tamanho original.
- **Transformação em *string*:** todo o *chat* do jogo foi colocado em formato *string* para melhor manuseamento dos dados.
- **Lower case:** todo o *chat* do jogo foi colocado em *lower case*, ou seja, feita a normalização em letras minúsculas.

- **Retirada de pontuação:** foi retirada toda a pontuação existente nas conversas utilizando a biblioteca *Python* de expressões regulares.
- **Stopwords:** *Stopwords* são de forma geral, palavras que podem ser consideradas irrelevantes (dependendo da busca realizada) para a obtenção de resultados (MARCEL, 2009). Foi feita a remoção das *stopwords* nas frases presentes no *dataframe* para menor custo de processamento. Foram utilizadas as *stopwords* presentes na biblioteca NLTK.
- **Stemming:** esse é um processo de remoção de radicais das palavras, utilizado para melhorar a performance de processamento.

Com o *dataframe* preparado, o próximo passo se refere à separação da amostragem por palavras mais frequentes, processo essencial para a aplicação futuro dos algoritmos.

4.4. Processo de Amostragem de Palavras mais frequentes

A amostragem se deu por meio da ferramenta *Wordcloud* para mostrar de forma mais ampla as palavras e um gráfico de barras mostrando de forma ordenada as 10 palavras mais frequentes no *dataframe* estudado.

4.4.1. Wordcloud

A *WordCloud* mostra de forma hierárquica as palavras mais frequentes de um texto, ou neste caso, de um conjunto de textos. Sua forma de mostrar as palavras é ilustrando em maior tamanho as palavras que mais aparecem no texto, e ir diminuindo o tamanho da fonte da letra quanto menos a palavra aparecer. A Figura 9 mostra um exemplo genérico de uma *Wordcloud* que ilustra as palavras mais presentes em um *dataframe* de conversas sobre a história de *Star Wars*. As palavras são colocadas de forma sortida na imagem de saída, ou seja, elas não têm uma posição fixa para aparecer. A nuvem de palavras formada no trabalho tem como quantidade máxima de palavras 5000,

para validar as informações que serão geradas pelos algoritmos de agrupamento posteriormente, seria necessário um *dataset* com frases que fazem parte do universo de vídeo games, pois como dito anteriormente, são diálogos com várias peculiaridades.

Foi neste ponto que surgiu um obstáculo, pois não há *dataset* para este tipo de estudo com fácil acesso e com classificações semelhantes às usadas no desenvolvimento do trabalho. O que dificultou ainda mais a busca por esse *dataset* foi a necessidade de ele ter frases que continham frases relacionadas à *chat* de jogos *online*.

Como solução foram retiradas 3000 frases do *dataframe* de conversas usado originalmente. Esta solução veio após a diminuição do *dataframe* original por limitações da plataforma *Google Colab*, que não suportou o processamento total do *dataframe* durante os agrupamentos feitos pelos algoritmos LDA e pelo *K-Means*.

Neste novo *dataset* gerado, todas as 3000 frases foram classificadas manualmente para gerar a base de emoções que será utilizada para verificar a precisão da classificação por agrupamento que será realizada.

Antes de gerar o classificador foram realizados mais alguns pré-processamentos adicionais necessários para melhor manuseio do *dataset*. Como estas frases já foram retiradas do *dataframe* original após passar pela primeira bateria de pré-processamento, não foi necessário repetir todo o trabalho. Os pré-processamentos adicionais foram:

- **Remoção de colunas:** as colunas de dados irrelevantes para a pesquisa do trabalho foram retiradas do *dataset*.
- **Notnull():** foi feito o mapeamento dos dados presentes no *dataset* para detecção de valores não ausentes no *dataset*. Esta função pega um objeto e indica se os valores são válidos.
- **Stemming:** utilizando a biblioteca NLTK, foi aplicado o *stemming* nas palavras presentes no *dataset*, isso é, a retirada dos radicais das palavras para conseguir melhorar o processamento do algoritmo (por exemplo: “*liked*”, “*likely*” e “*liking*” são reduzidas a “*like*”).

Continuando a geração do classificador, foi realizada a separação das frases de suas emoções (tópicos) e a separação das palavras. Desta forma, o

conjunto não faria mais parte de uma frase completa neste momento. Segue um exemplo:

`['space creat', 'topic0'] >>> ['space', 'creat']`

Realizada a listagem de todas as palavras presentes, foi feita uma busca pelas frequências das palavras que apareciam no *dataset*. Com a frequência em mãos, foi gerada uma lista de palavras únicas, de forma que cada palavra apareceria somente uma vez nesta lista.

Após ter a lista de palavras únicas, foi elaborada uma função para extrair as palavras de cada uma das frases. Desta forma, ao passar uma frase como parâmetro, para as palavras presentes na frase que forem encontradas na base de dados, assim como ilustrado na Figura 10 que mostra o exemplo da frase “*Space creat*”, será retornado um valor *True*, caso contrário, o valor *False*.

Figura 10. Retorno de *True* para as palavras passadas como referência.

```
caracteristicas_frase = extratorpalavras(['space', 'creat'])
print(caracteristicas_frase)

{'forc': False, 'space': True, 'creat': True, 'hah': False,
```

Fonte: Elaborado pelo autor

O próximo passo foi a geração de uma base de dados com a aplicação da extração das palavras, juntamente com seus respectivos sentimentos (tópicos) já classificados anteriormente. A geração dessa nova base se deu a partir da aplicação de *features* no método *classify* da biblioteca NLTK.

Para gerar o classificador, foi utilizado o *NaiveBayesClassifier* do NLTK que utilizou como parâmetro a base de dados gerada. Nesse classificador, foram utilizados dois tópicos como resultados. Cada um desses tópicos faz referência a um tipo de comportamento. São eles:

- **Topic0:** é marcado por frases NÃO-TÓXICAS, ou seja, que o algoritmo não classifica como uma frase que contém toxidade. Neste tópico, por questão de praticidade, foram incluídas todas as frases com qualquer outro

tipo de comportamento que não seja tóxico, seja este um comportamento feliz, triste e até mesmo comportamentos considerados 'neutros'.

- **Topic1:** é justamente o contrário do *Topic0*. Nesse tópico, são classificadas as frases com toxidade explícitas. Neste tópico são também reunidos os comportamentos tóxicos a partir de frases com palavras de comportamento indevido, como palavrões, racismo, pensamentos negativistas, exposição de reação excessiva sobre algo que aconteceu no jogo, além de palavras referentes a elementos do próprio contexto do jogo que possam determinar algum comportamento tóxico.

Todas as frases foram classificadas levando em consideração os pesos das palavras presentes nelas. Isso leva em conta, por exemplo, a frequência de vezes que uma palavra aparece em um dos tópicos comparado ao outro. Para se ter noção do peso de algumas das palavras, a Figura 11 mostra o quão grande é a chance de uma palavra ser de um tópico, levando em consideração as identificadas com maiores pesos na base.

Na Figura 11, é possível observar os pesos das 10 palavras com mais influência no momento de decidir o tópico de uma frase. Como exemplo uma frase tenha a palavra “*report*”, ela tem 165,9 vezes mais chance de ser considerada do tópico1.

Figura 11. Palavras com maiores pesos nas frases da base de aprendizagem.

```
print(classificador.show_most_informative_features(10))
```

Most Informative Features

report = True	topic1 : topic0 =	165.9 : 1.0
noob = True	topic1 : topic0 =	55.3 : 1.0
team = True	topic1 : topic0 =	39.9 : 1.0
shit = True	topic1 : topic0 =	33.2 : 1.0
like = True	topic1 : topic0 =	25.6 : 1.0
fuck = True	topic1 : topic0 =	24.6 : 1.0
ty = True	topic1 : topic0 =	21.1 : 1.0
pls = True	topic1 : topic0 =	18.0 : 1.0
bobo = True	topic1 : topic0 =	14.8 : 1.0
gg = True	topic0 : topic1 =	13.9 : 1.0

None

Fonte: Elaborado pelo autor

A partir deste ponto é realizada a classificação por agrupamento dos dois algoritmos que serão implementados, para **a visualização da diferença nos resultados da classificação. Foram utilizados os algoritmos LDA, e o K-Means.**

Para a criação do modelo LDA, foi necessário separar as informações de texto do *dataframe* presentes nos *chats* de conversa do jogo. Após isso, foi realizado todos os pré-processamentos citados anteriormente, mas ainda mantendo as frases separadas em formato de *dataset* (Foi criado um *dataset* separado para salvar os *chats* retirados do *dataframe* original estudado).

Como próximo passo, foi importado e iniciado o *CountVectorizer* da biblioteca *Sklearn*, que transformou os dados de texto em uma matriz de números e vetores, reduzindo os textos a componentes, pois o LDA necessita de dados numéricos como entrada.

Para gerar o modelo do LDA, foi necessário fornecer a matriz de dados gerada pelo *CountVectorizer*, além do número de tópicos: Tóxico ou Não-Tóxico. Após gerado o modelo LDA, foi feito o ajuste dos dados a partir da função *.fit*.

Infelizmente, a saída do algoritmo não é muito legível, portando uma função foi implementada para ser possível visualizar as palavras que foram agrupadas em cada um dos tópicos gerados a partir do score de similaridade do algoritmo e que são apresentadas na Tabela 4, como exemplo. Esta função faz uma leitura dos componentes do modelo utilizado para encontrar as palavras semelhantes e busca as palavras correspondentes no *CountVectorizer*.

Tabela 4. Tópicos - LDA

Topics found via LDA:

Topic #0:

gg lol ez wp haha ggwp fuck ty nice ok wtf yeah come wow min

Topic #1:

game report mid noob team end just pls dont hahaha fucking xd commend shit guys

Fonte: Elaborado pelo autor

Com o modelo treinado, foi realizada uma implementação com o *pyLDAvis* para uma melhor visualização dos tópicos gerados nos testes realizados, mostradas nas Figura 12, Para garantir uma melhor leitura pelo *TfidfVectorizer*, todos os dados da coluna referente às frases no *dataframe* foram transformadas em *Unicode Data*, ou seja, em *Strings*. Esta função faz com que qualquer tipo de símbolo presente nas frases processadas sejam transformadas em formato *Unicode*. Outros pré-processamentos feitos nos dados foram a retirada das *stopwords* do texto e a padronização dos documentos para melhorar seu processamento.

Neste ponto do trabalho, para execução do algoritmo o *K-Means* foi necessário definir a quantidade de *clusters* a serem trabalhados pelo algoritmo. Estes *clusters* nada mais são do que a quantidade de agrupamentos em que os dados serão classificados. Para manter a mesma quantidade de agrupamentos para ambos os algoritmos (LDA usa dois tópicos), foi definida a quantidade de dois *clusters*.

Figura 13 e

Figura 14, principalmente no que diz respeito a quantidade de palavras totais e presentes em cada um dos tópicos, para assim, ser possível determinar de uma maneira mais clara, as relações entre as palavras de cada tópico.

Para continuar o processamento, foi necessário atribuir um dos tópicos gerados a cada uma das frases presentes no *dataframe*. Para isso, foi implementada uma função que verifica cada frase presente no *dataframe*.

Através de um processo de contagem simples, foi possível determinar quantas palavras de cada tópico presente na frase, e atribuir o tópico pertinente àquela frase.

Para esta função foi criado um dicionário para verificar qual tópico melhor se enquadra à frase, para finalmente retornar o resultado. Esses resultados foram guardados em uma lista de tópicos que posteriormente foi adicionada a um *dataset* junto de suas frases correspondentes, para que fosse possível verificar a avaliação do algoritmo. Observe a Figura 15, com um exemplo.

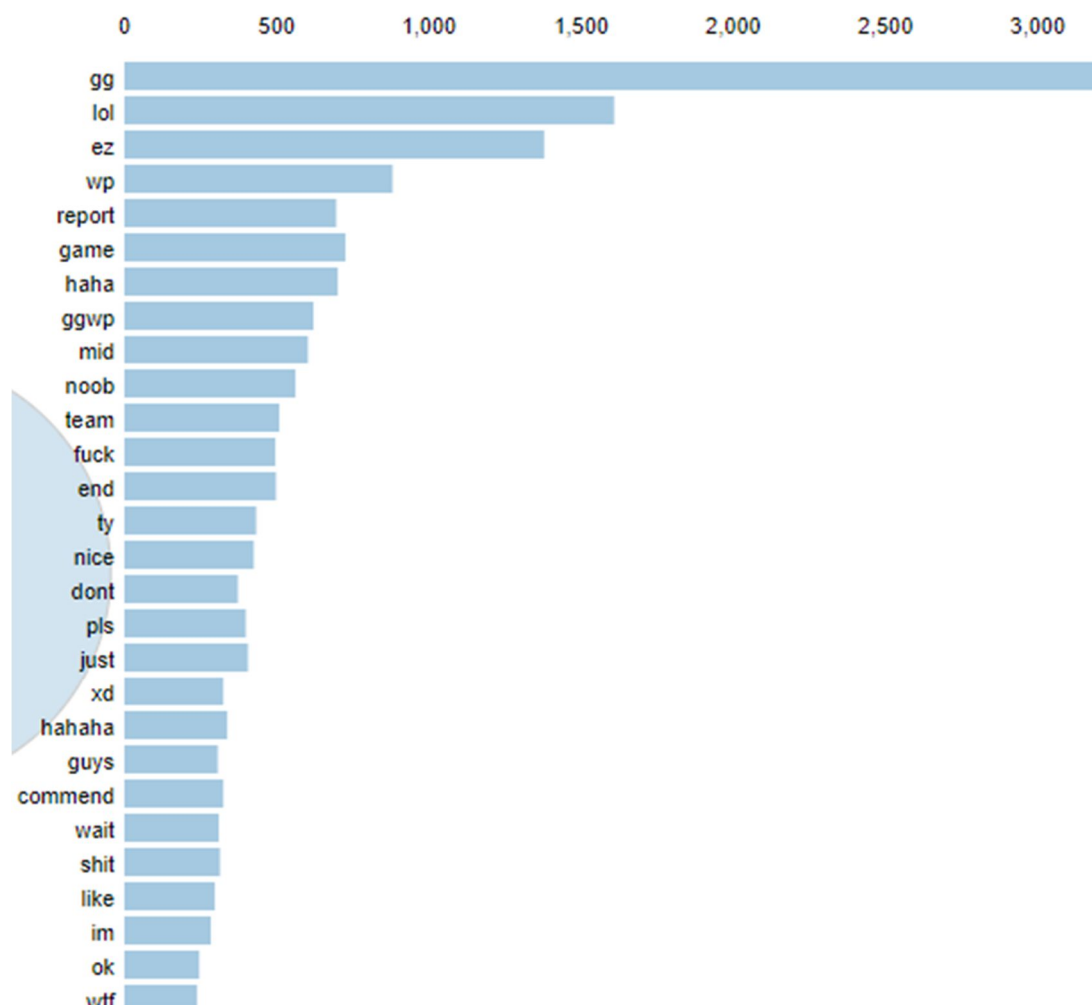
Para a criação do modelo do *K-Means*³, o começo do trabalho foi bem semelhante ao do LDA, já que a separação das frases, mantendo-as em um *dataset* separado a parte, também gerou bons resultados quanto à facilidade de manipulação dos dados.

Com os as frases relacionadas aos seus tópicos, foram importados da biblioteca *Sk-learn* o *K-Means* e o *TfidfVectorizer*, que pode converter uma coleção de documentos em uma matriz TF-IDF⁴ se assemelhando ao *CountVectorizer* em sua função.

Figura 12. Resultados gerais dos testes do agrupamento via LDA.

³ Toda a codificação do *K-Means* foi realizada em um documento diferente do LDA, então diversos procedimentos precisaram ser refeitos. Porém como toda a parte de bibliotecas e pré-processamentos foram basicamente os mesmos, o desenvolvimento continuará a partir da aplicação do modelo do *K-Means*.

⁴ TF-IDF é uma medida que tem o intuito classificar a importância de uma palavra de um documento em relação a todo o conjunto (SANTOS, 2017).

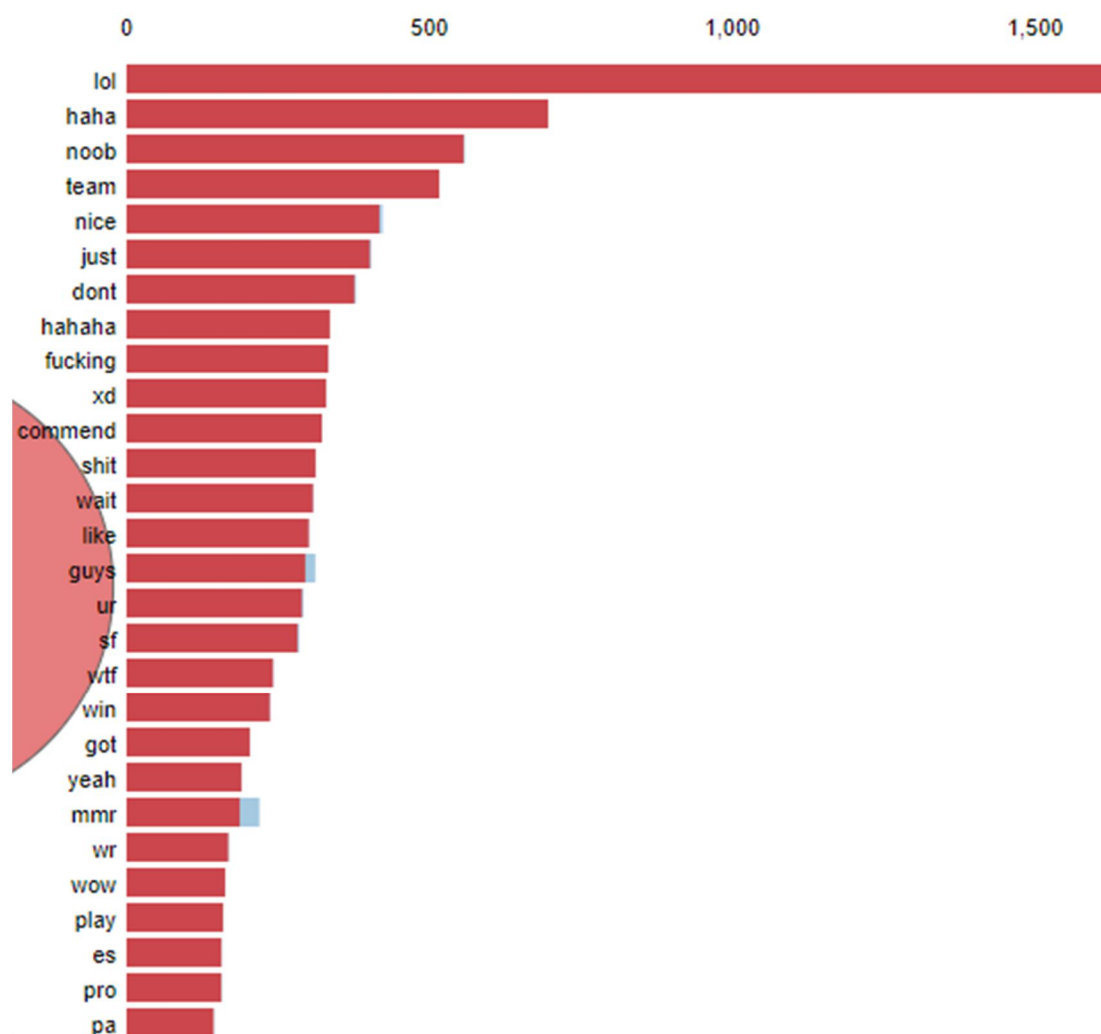


Fonte: Elaborado pelo autor

Para garantir uma melhor leitura pelo *TfidfVectorizer*, todos os dados da coluna referente às frases no *dataframe* foram transformadas em *Unicode Data*, ou seja, em *Strings*. Esta função faz com que qualquer tipo de símbolo presente nas frases processadas sejam transformadas em formato *Unicode*. Outros pré-processamentos feitos nos dados foram a retirada das *stopwords* do texto e a padronização dos documentos para melhorar seu processamento.

Neste ponto do trabalho, para execução do algoritmo o *K-Means* foi necessário definir a quantidade de *clusters* a serem trabalhados pelo algoritmo. Estes *clusters* nada mais são do que a quantidade de agrupamentos em que os dados serão classificados. Para manter a mesma quantidade de agrupamentos para ambos os algoritmos (LDA usa dois tópicos), foi definida a quantidade de dois *clusters*.

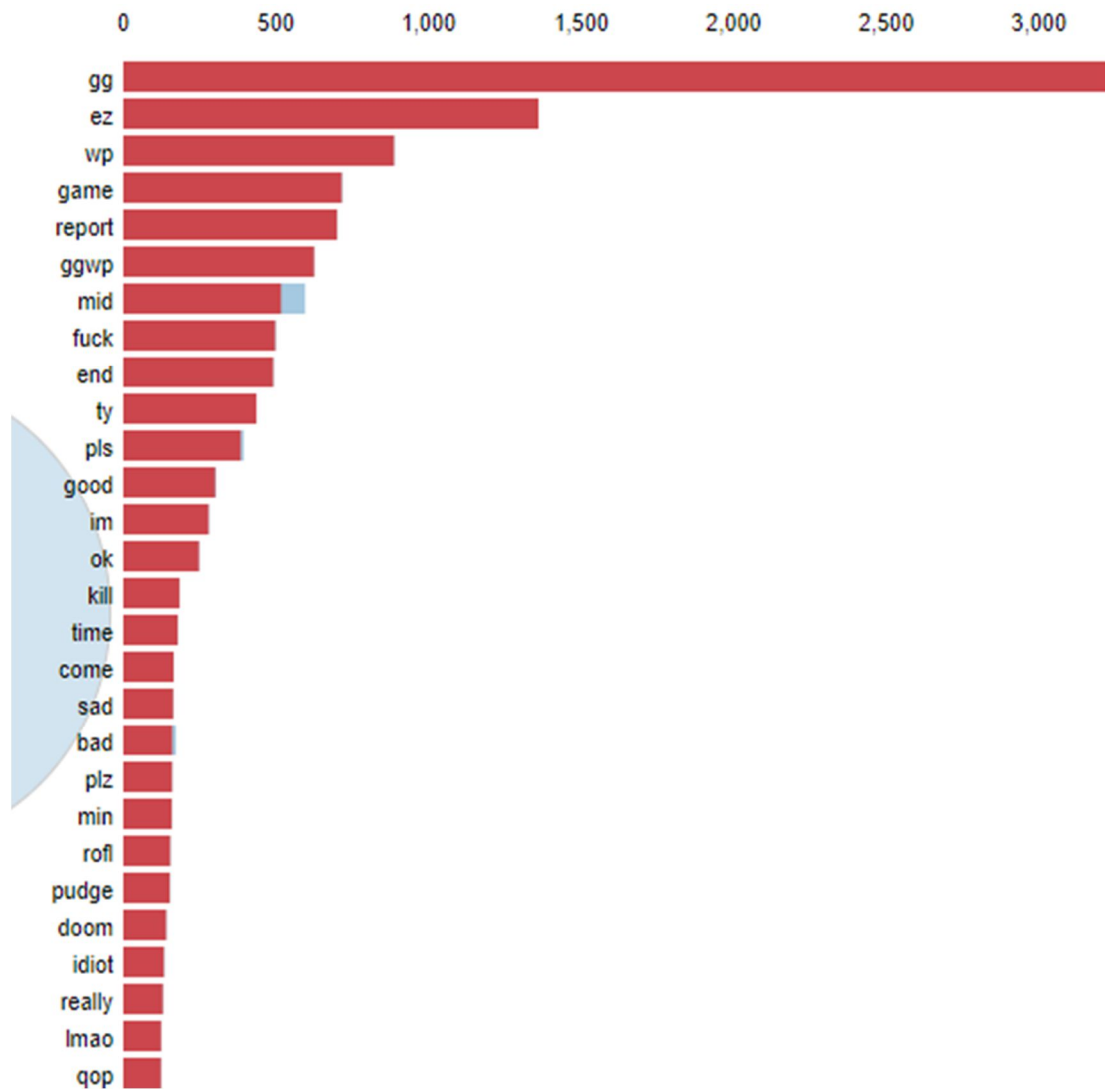
Figura 13. Resultados dos testes para o tópico 1 - LDA



Fonte: Elaborado pelo autor

Com a quantidade de *clusters* definida, foi necessário montar o modelo *K-Means* para classificação dos dados. A Figura 16 mostra os hiperparâmetros utilizados na criação do modelo do *K-Means*. Percebe-se que grande parte da configuração padrão foi utilizada:

Figura 14. Resultados dos testes para o tópico 2 - LDA



Fonte: Elaborado pelo autor

Figura 15. Dataframe com as frases e seus respectivos tópicos relacionados.

```
# Adicionando a coluna de tópicos ao dataframe
data_list_frame['topics'] = topics_list
```

```
data_list_frame
```

	text	topics
0	force it	topic0
1	space created	topic0
2	hah	topic0
3	ez	topic0
4	mvp ulti	topic0

Fonte: Elaborado pelo autor

- o número de *clusters* foi definido anteriormente como 2.
- para o parâmetro *init* foi configurado o modo *k-means++* para inicialização do algoritmo, sendo este o método padrão onde os centróides são gerados utilizando um método de convergência.
- para o *max_iter* foi dado como valor a quantidade de apenas 100 interações, já que testes com maiores quantidade de interações não mostrou diferenças relevantes aos resultados, somente gastando mais tempo de processamento.

Após criação do modelo, foi realizada uma nova padronização, desta vez nos dados do modelo gerado. Esta padronização foi realizada também usando a função “*fit*” que teve como parâmetros, os dados que foram transformados anteriormente na variável *features*.

Figura 16. Hiperparâmetros K-Means

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=100,
        n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=None, tol=0.0001, verbose=0)
```

Fonte: Elaborado pelo autor

Aqui foi novamente necessário visualizar as palavras mais presentes em cada um dos *clusters*, assim como foi feito no LDA. Para esse processo, um pequeno código (função) foi implementado após a classificação realizada pelo modelo. Nessa função as palavras mais próximas dos centros de cada *cluster* (centróides) são avaliadas. São retornadas as 30 (trinta) palavras mais próximas de cada centro um dos *clusters*, como mostra o exemplo da Tabela 5.

A partir deste ponto, o código voltou a ficar bastante parecido com o utilizado junto ao LDA. Foi criado um dicionário com as palavras mais presentes nos centróides e, após verificar qual *cluster* mais presente naquela frase, atribuiu-se o número do *cluster* àquela frase, com o retorno do resultado. Esses resultados também foram guardados em uma lista de tópicos que foi posteriormente adicionada a um *dataset* junto de suas frases correspondentes.

Tabela 5. Centroides - K-Means

Centroides:

Cluster 0:

gg, , ez, wp, haha, ggwp, report, end, game, , ty , noob,
fuck, mid, hahaha, xd, team, nice, pls, commend, wait, wtf,
ok, just, don't, shit, guys, good, sf, fucking, like, rofl

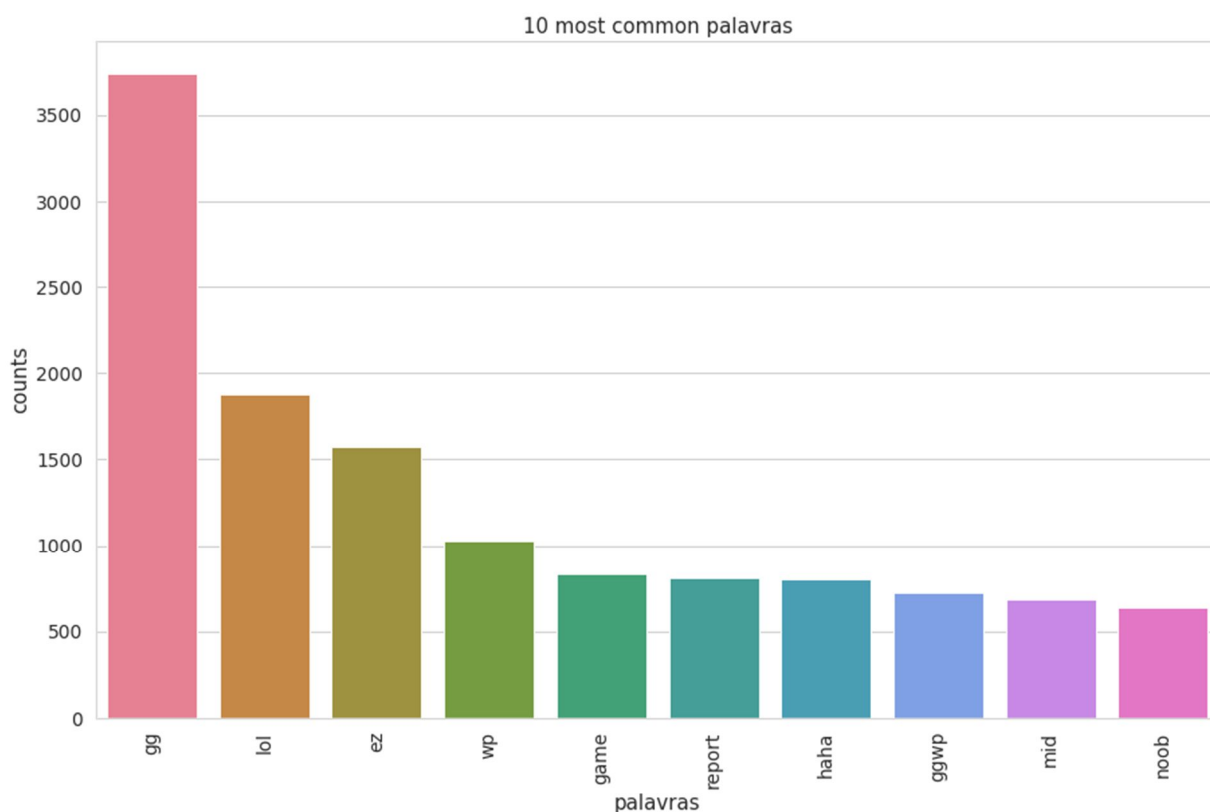
Cluster 1:

lol, gg, mid, oh, pudge, yeah, sf, wp, np, ok, noob,
hook, worth, riki, bitch, lc, need, es, bad, look, bh,
solo, lion, vs, comeback, hahah, yea, lag, support, ta

Fonte: Elaborado pelo autor

Para esse estudo, palavras não presentes na Língua Inglesa foram classificadas como palavras “Não-tóxicas”, ou seja, não representam toxicidade. Isso se deve ao fato de que são necessários estudos em diversos dialetos para o entendimento de que tipo de sentimento ou opinião está sendo expressa pelo jogador. Foram encontradas frases nos idiomas: Português, Espanhol, Russo, Chinês, Coreano e Japonês.

Figura 18. Palavras mais frequentes do dataframe



Fonte: Elaborado pelo autor

Como mostrado nas Tabela 6 e Tabela 7, foram consideradas respectivamente, as 15 palavras mais relevantes de cada tópico para o LDA e as 30 palavras mais relevantes para o *K-Means*, e que conseguiram maiores índices de precisão após passar pelo classificador.

Tabela 6. Palavras com maior relevância em cada tópico - LDA

Tópico	Palavras mais relevantes
Tópico #0 (Não-tóxico)	gg, ez, game, mid, haha, nice, fuck, end, don't, guys, commend, good, wtf, rofl, mmr
Tópico #1 (Tóxico)	lol, wp, report, team, xd, ggwp, ty, pls, noob, just, fucking, shit, sf, like, ur

Fonte: Elaborado pelo autor

Tabela 7. Palavras com maior relevância em cada tópico - K-Means.

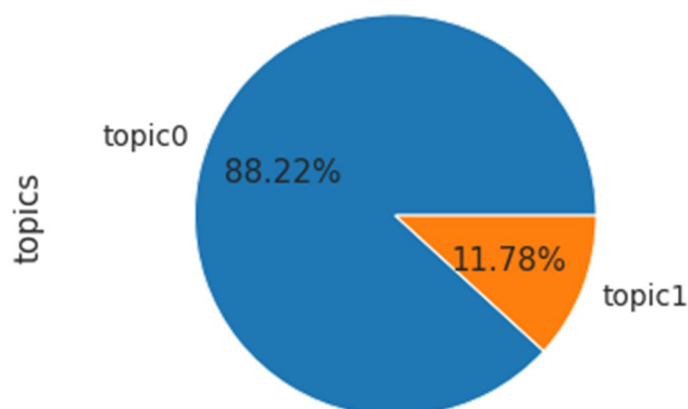
Tópico	Palavras mais relevantes
Tópico #0 (Não-tóxico)	gg, ez, wp, team, guys, mid, lol, end, noob, sf, ty, haha, ok, pa, es, pro, plz, carry, wr, time, yeah, win, commend, xd, game, gggggggggggggggg, im, pudge, hihi, hhehe
Tópico #1 (Tóxico)	lol, ez, haha, ggwp, report, end, game, ty, noob, wp, fuck, mid, hahaha, xd, nice, team, pls, commend, wait, wtf, ok, just, don't, shit, good, guys, sf, fucking, like, rofl

Fonte: Elaborado pelo autor

Na Tabela 6 pode-se observar uma quantidade menor de palavras em destaque, isso porque o aumento da quantidade de palavras não contribuiu com resultados obtidos. Já como ilustra a Tabela 7, mais palavras foram levadas em peso para o algoritmo, já que o aumento destas palavras levou a uma melhora significativa nos resultados obtidos pelo agrupamento do *K-Means*.

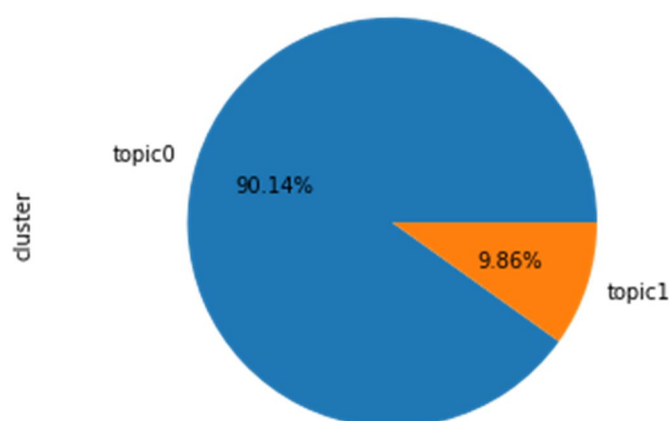
Tais palavras, assim como suas respectivas classificações nas frases, geraram resultados semelhantes na divisão de palavras tóxicas e não-tóxicas como ilustrado nas Figuras 19 e Figura 20.

Figuras 19. Distribuição dos tópicos no dataframe - LDA



Fonte: Elaborado pelo autor

Figura 20. Distribuição dos tópicos no dataframe - K-Means.



Fonte: Elaborado pelo autor

5.2. Discussão de Resultados

Após se obter os resultados da classificação de cada algoritmo, realizou-se um processo de classificação para determinar seus respectivos índices de acurácia.

Como citado no Capítulo 4, as frases que o classificador desenvolvido no trabalho utilizou para treinamento foram categorizadas manualmente devido à escassez destes tipos de dados voltados para o ambiente de jogos *online*. Após passar pelo classificador, as precisões de ambos os algoritmos foram bem parecidas, ultrapassando os 85% em ambos os casos.

Os melhores resultados obtidos chegaram a 87,69% de precisão pelo algoritmo LDA, por outro lado, o *K-Means* chegou a uma precisão de 86,77%, após diversos testes realizados com mudanças de seus hiperparâmetros.

Dessa maneira observa-se que mesmo com poucas modificações em seus hiperparâmetros e com menor quantidade de palavras relevantes, o algoritmo LDA conseguiu se sobressair ao *K-Means*, que por outro lado, precisou de mais preparações na base de dados e em seus hiperparâmetros. Tais ajustes extras realizados no algoritmo *K-Means*, evidenciam que o algoritmo LDA se mostra mais adequado a esse tipo de estudo considerando, de fato, a metodologia aplicada neste trabalho.

Outra maneira de medir os resultados encontrados pelo classificador LDA e do *K-Means* foi a partida da matriz de confusão.

5.3. Matriz de Confusão

A Matriz de Confusão mostra a quantidade de frases que os algoritmos de agrupamento classificaram em cada um dos tópicos e quantas dessas classificações foram assertivas.

A classificação da Matriz de Confusão para os algoritmos retornou os seguintes resultados:

- Para o LDA, somaram-se 44.205 classificações no tópico de frases não-tóxicas, das quais foram acertadas 41.228 frases. Com isso, a taxa de acerto das frases não-tóxicas chegou a 93%. Já as frases tóxicas classificadas somaram um total de 5.901 frases, das quais foram acertadas 2.710 frases, chegando a uma taxa de acerto de 45%, como pode ser visto na Tabela 8.

Tabela 8. Matriz de Confusão - LDA

		N		*NT = Não-tóxica
		T	T	* T = Tóxica
NT		<41228>	2977	
T		3191	<2710>	

Fonte: Elaborado pelo autor

- Para o *K-Means*, somaram-se 45.164 classificações no tópico de frases não-tóxicas, das quais foram acertadas 41.477 frases. Com isso a taxa de acerto das frases não-tóxicas chegou a 91%. Já as frases tóxicas classificadas somaram um total de 4.942 frases, das quais foram acertadas 2.000 frases, chegando a uma taxa de acerto de 40%, como pode ser visto na Tabela 9.

Tabela 9. Matriz de Confusão - K-Means

		N		*NT = Não-tóxica
		T	T	* T = Tóxica
NT		<41477>	3687	
T		2942	<2000>	

Fonte: Elaborado pelo autor

6. Conclusão

Na pesquisa realizada, foi estudado o padrão comportamental tóxico presente na comunidade de jogos *online*. Para isso, foram estudados o comportamento de jogadores a partir de um *dataframe* com aproximadamente 50 mil registros, que apresentava conversas *online* em um dos jogos mais populares mundialmente, o DotA2.

Foi estudado também, o tipo de comportamento tóxico e como ele se apresenta em comunidades *online* para serem incluídas na classificação. Foram desenvolvidas as implementações para comparação de resultados de dois algoritmos de agrupamento: o LDA, que faz agrupamento por meio de Modelos de Tópicos, e o *K-Means*, que faz agrupamentos utilizando a técnica de Clusterização.

Confirmou-se, durante a verificação de precisão dos algoritmos utilizados, que a precisão de acerto ficou acima do esperado em ambos os algoritmos, isto porque o estudo de classificação de sentimentos contém várias ferramentas linguísticas que podem não ser tão claras para o computador (a ironia, por exemplo). Estimava-se que a porcentagem de acertos ficaria entre os 70% e 80%. Isso mostrou que classificação utilizando comportamentos contraditórios (Não-tóxico e tóxico) evidenciou um bom resultado em comparação a testes realizados com mais grupos de classificação, que em alguns testes realizados mostraram resultados de 18% de precisão.

Verificou-se também, que o algoritmo LDA gerou melhores resultados em sua classificação, o que já era esperado do algoritmo, já que o LDA trabalha com classificação, extração de tópicos e agrupamento de textos, enquanto o *K-Means* representa um algoritmo de agrupamentos mais genérico, pois trabalha com textos, como também com valores numéricos.

Em sua precisão geral com 87%, o LDA foi 1,1% mais preciso que o *K-Means*. Além disso, na precisão de ambos os tópicos o LDA se mostrou superior, acertando 2,1% a mais frases Não-tóxicas e 12,5% de frases tóxicas mais frases que o *K-Means*.

Foi visto também que ainda é possível melhorar a classificação de toxidade de ambos os algoritmos, já que a taxa de precisão das frases tóxicas

ficou abaixo dos 50%. Para isso, sugere-se uma base de aprendizado mais preparada e específica para o tipo de texto que é visto no ambiente *online*.

Foi observado ainda, que a quantidade de frases tóxicas presentes nas conversas *online*, ficou abaixo do que era esperado. As ferramentas de comunicação dos próprios jogos que, mesmo criadas com o intuito de facilitar o diálogo e organização das equipes, foram apontadas como as mais utilizadas por jogadores tóxicos para projetar provocações, mensagens de ódio ou discriminações, seja por texto ou voz - respectivamente 42% e 40% das vezes (OLIVEIRA, 2020). Neste estudo a taxa de toxidade presente no *dataframe*, ficou com uma média de 10,82% considerando o resultado da classificação de ambos os algoritmos.

Mesmo estando abaixo do esperado, é um fato que a toxidade atinge os jogos *online* e que devem ser desenvolvidas ferramentas para aplicação de punições para este tipo de comportamento. Em grandes jogos existem ferramentas para combate a toxidade em suas comunidades, com punições que levam de um pequeno mute (não poder interagir) por determinado intervalo de tempo à banimentos permanentes no jogo.

Como trabalhos futuros podem ser realizados ainda, estudos mais completos do tema, abrangendo não só dois tópicos contrários um ao outro, mas também tópicos semelhantes, como por exemplo: tristeza, raiva e angústia. Um outro trabalho futuro que pode ser realizado é a implementação de uma ferramenta para aplicação de punições para jogadores que foram detectados utilizando-se de comportamentos tóxicos durante as partidas.

7. BIBLIOGRAFIA

A Biblioteca Scikit-learn – Python para machine learning. Didática Tech – Inteligência Artificial & Data Science. Disponível em: < <https://didatica.tech/a-biblioteca-scikit-learn-pyhton-para-machine-learning/> >. Acesso em: 24 de maio. 2021.

BERTIN, Adriano. O que é Processamento de Linguagem Natural? Inbenta, 2020. Disponível em: < <https://www.inbenta.com/pt/blog/o-que-e-processamento-de-linguagem-natural/> >. Acesso em: 20 de mar. 2021.

BOLLATI, Eliana. Avançando do problema do racism de DOTA 2. ESTNN, 2021. Disponível em: < <https://scikit-learn.org/stable/index.html> >. Acesso em: 22 de maio. 2021.

BROWNLEE, Jason. 10 Clustering ALgorithms With Python. Machine Learning Mastery, 2020. Disponível em: < <https://machinelearningmastery.com/clustering-algorithms-with-python/> >. Acesso em 15 maio, 2021.

CAETANO, Rafaela. O que são mobas?. ESPN, 2019. Disponível em: < <http://www.espn.com.br/infografico/a-gente-explica-e-voce-vira-fa-o-que-sao-os-mobas/> >. Acesso em: 15 de abr. 2021.

DABBURA, Imad. K-Means: Algorithm, Application, Evaluation, Methods, and Drawbacks. Towards, Data Science, 2018. Disponível em: < <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a> >. Acesso em: 22 de maio. 2021.

DABUL, Bernardo. Review DOTA 2. Techtudo, 2013. Disponível em: < <https://www.techtudo.com.br/review/dota-2.html> >. Acesso em: 23 de abr. 2021.

DOTA 2 Wiki. Gamepedia: Powered by Fandom, 2018. Mapa. Disponível em: < <https://dota2.fandom.com/pt/wiki/Mapa> >. Acesso em: 19 de mar. 2021.

DOTA 2 Wiki. Gamepedia: Powered by Fandom, 2018. Chat Wheel. Disponível em: < https://dota2.fandom.com/wiki/Chat_Wheel >. Acesso em: 19 de mar. 2021.

DOTA 2. Steam Community, 2014. Dota 2 Lane Information. Disponível em: < <https://dota2.fandom.com/pt/wiki/Mapa> >. Acesso em: 19 de mar. 2021.

DOTA 2. Steam Community, 2013. The Big Dota 2 Dictionary. Disponível em: < <https://steamcommunity.com/sharedfiles/filedetails/?id=172122376> >. Acesso em 20 de nov. 2020.

DOTA 2, Avelusca. TODAS AS FORMAS DE COMUNICAR | DOTA 2. Youtube. Disponível em: <<https://www.youtube.com/watch?v=Z40OaPSX-w0>>. Acesso em: 05 jun. 2021. 07:00.

FERNANDEZ, Amanda. Comportamento Tóxico em Jogos *Multiplayer Online: Uma Revisão Narrativa*. Monografia (Bacharelado) – Universidade Federal de Pelotas, Pelotas. 2017.

GESTÃO de APIs: veja como funciona na prática! DocuSign, 2018. Disponível em: < <https://www.docusign.com.br/blog/gestao-de-apis-veja-como-funciona-na-pratica> >. Acesso em: 06 de jun. 2021.

Great Learning Team. Understanding Latent Dirichlet Allocation (LDA). eat Learning, 2020. Disponível em: < <https://www.mygreatlearning.com/blog/understanding-latent-dirichlet-allocation/> >. Acesso em: 30 de maio. 2021.

GUERRA, Felipe. O que é MOBA? Confira significado e games de sucesso no competitivo. E-SporTV, 2019. Disponível em: <<https://sportv.globo.com/site/e->

sportv/noticia/o-que-e-moba-confira-significado-e-games-de-sucesso-no-competitivo.ghml>. Acesso em: 15 de nov, 2020.

HUNTER, John et al. Pyplot tutorial. Matplotlib, 2021. Disponível em: < <https://matplotlib.org/stable/tutorials/introductory/pyplot.html> >. Acesso em: 24 de mar. 2021.

IBM Cloud Education. Machine Learning. IBM, 2020. Disponível em: < <https://www.ibm.com/cloud/learn/machine-learning> >. Acesso em: 05 de maio. 2021.

IS Python the most popular language for data science? Maruti Techlabs, 2021. Disponível em: < <https://marutitech.com/python-data-science/> >. Acesso em: 05 de maio. 2021.

KURTZ, Gabriela. A ética dos computadores e a ética dos *griefers* nos jogos League of Legends e Dota 2. Artigo – Faculdade Federal do Rio Grande do Sul. Porto Alegre, BR-RS, 2016.

LARGHI, Nathália. Brasil é o 13º maior mercado de games do mundo e o maior da América Latina. Valor Investe, 2019. Disponível em: <<https://valorinveste.globo.com/objetivo/empreenda-se/noticia/2019/07/30/brasil-e-o-13o-maior-mercado-de-games-do-mundo-e-o-maior-da-america-latina.ghml>>. Acesso em: 16 de nov. 2020.

MABEY, Ben. pyLDAvis. pyLDAvis, 2015. Disponível em: < <https://pyldavis.readthedocs.io/en/latest/readme.html> >. Acesso em: 2 de abr. 2021.

MARCEL, Frank. Stop Words - Como funcionam palavras de parada? Agência|Mestre, 2009. Disponível em: < <https://www.agenciamestre.com/seo/stop-words-como-funcionam-palavras-de-parada/> >. Acesso em 07 jun. 2021.

MATLAB. MathWorks, 2021. Disponível em: <<https://la.mathworks.com/help/matlab/>>. Acesso em: 11 de jun. 2021.

MATOS, David. Conceitos Fundamentais de Machine Learning. Ciência e Dados: Data Science for Professionals, 2015. Disponível em: <<https://www.cienciaedados.com/conceitos-fundamentais-de-machine-learning/>>. Acesso em: 06 jun. 2021.

ML | K-Means++ Algorithm. GeeksforGeeks, 2020. Disponível em: <<https://www.geeksforgeeks.org/ml-k-means-algorithm/>>. Acesso em: 30 de maio. 2021.

Murnion, Shane et al.. Machine learning and semantic analysis of in-game chat for cyberbullying. Computers & Security, v. 36, p. 197-213. Jul. 2018. Disponível em: <<https://doi.org/10.1016/j.cose.2018.02.016>>. Acesso em: 28 out. 2020.

NETO, J. Uma análise sobre o comportamento tóxico em jogos on-line baseada em tópicos de conversa. Dissertação (Mestrado) – Faculdade Federal do Rio Grande do Sul. Porto Alegre, BR-RS, 2019.

NLTK 3.6.2 documentation. NLTK Project, 2021. Disponível em: <<https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>>. Acesso em: 19 de mar. 2021.

New Report from The NPD Group Provides In-Depth View of Brazil's Gaming Population. NPD Group, 2015. Disponível em: <<https://www.npd.com/wps/portal/npd/us/news/press-releases/2015/new-report-from-the-npd-group-provides-in-depth-view-of-brazils-gaming-population/>>. Acesso em: 13 de nov. 2020.

NOGARE, Diego. Performance de Machine Learning – Matriz de Confusão. Diego Nogare: Inteligência Artificial & Machine Learning, 2020. Disponível em:

< <http://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao> >. Acesso em: 05 de maio. 2021.

NUMPY: The absolute basics for beginners. NumPy, 2021. Disponível em: < https://numpy.org/doc/stable/user/absolute_beginners.html >. Acesso em: 2 de abr. 2021.

OLIVEIRA, Matheus. Toxidade: O mal de uma geração no esport. MGG, 2020. Disponível em: < <https://br.millenium.gg/noticias/4107.html> >. Acesso em: 23 de maio. 2021.

O que é o Colaboratory? Google Colab Research, s/d. Disponível em: < https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index >. Acesso em: 06 jun. 2021.

O que são os eSports?. CBeS – Conferência Brasileira de eSports. Disponível em: <<http://cbesports.com.br/esports/esports-o-que-sao/>>. Acesso em: 14 de nov. 2020.

PIATETSKY, Gregory. R, Python Duel As Top Analytics, Data Science software – KDnuggets 2016 Software Poll Results. KDnuggets, 2016. Disponível em: < <https://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html> >. Acesso em: 13 maio, 2021.

PROCESSAMENTO de Linguagem Natural. SAS, 2021. Disponível em: < https://www.sas.com/pt_br/insights/analytics/processamento-de-linguagem-natural.html >. Acesso em: 09 de maio. 2021.

PROVOST, F; FAWCETT, T. Data Science and its relationship to Big Data and Data-Driven decision making. Big Data. v. 1, n.1, p. 51-59. fev 2013. Disponível em: <<http://doi.org/10.1089/big.2013.1508>>. Acesso em: 27 out. 2020.

PYTHON. Wikipédia, a enciclopédia livre, 2021. Disponível em: < <https://pt.wikipedia.org/wiki/Python> >. Acesso em: 18 de mar. 2021.

RODRIGUES, C. et al. Mineração de Opinião/Análise de Sentimentos. Santa Catarina: UFSC, 2013. Disponível em: <<http://www.inf.ufsc.br/~luis.alvares/INE5644/MineracaoOpiniao.pdf>>. Acesso em 02 de abr. 2019.

SANTOS, Vinícios dos. Mineração de dados e Machine Learning: o que são Métodos supervisionados e não supervisionados. Computer Science Master, 2017. Disponível em: < <https://www.computersciencemaster.com.br/mineracao-de-dados-machine-learning-metodos-supervisionados-e-nao-supervisionados/> >. Acesso em: 06 jun. 2021.

SANTOS, Vinicius dos. Term Frequency (TF) e Term Frequency – Inverse Document Frequency (TF-IDF). Computer Science Master, 2017. Disponível em: < <https://www.computersciencemaster.com.br/term-frequency-e-term-frequency-inverse-document-frequency/> >. Acesso em: 28 de abr. 2021.

SILVA, Lucas. Análise de sentimentos em contexto: Estudo de caso em blog de empreendedorismo. Monografia (Bacharelado) – Universidade de Brasília – UnB, Brasília, 2013.

Scikit-Learn Developers. Feature extraction. Scikit-Learn, 2007-2020. Disponível em: < https://scikit-learn.org/stable/modules/feature_extraction.html > . Acesso em: 19 mar. 2021.

Scikit-Learn Developers. Naive Bayes. Scikit-Learn, 2007-2020. Disponível em: < https://scikit-learn.org/stable/modules/naive_bayes.html > . Acesso em: 19 mar. 2021.

SOUZA, Amanda. Você já conhece todos os tipos de *E-Sports*? Plataforma *e-SporTI*, 2020. Disponível em: < <https://e-sporti.com.br/noticias/2020-11-voce-ja> > .

conhece-todos-os-tipos-de-e-sports-confira-este-texto-e-descubra >. Acesso em: 25 de abr. 2021.

SULER, John. The *online* disinhibition effect. *CyberPsychology & Behavior*, v. 7, n. 3. Jul. 2004. Disponível em: <<http://doi.org/10.1089/1094931041291295>>. Acesso em: 05 nov. 2020.

SULER, John. Identity Management in Cyberspace. *Journal of Applied Psychoanalytic Studies*, v. 4, p. 455-459. out. 2002. Disponível em: <<https://doi.org/10.1023/A:1020392231924>>. Acesso em: 05 nov. 2020.

The pandas development team. About Pandas. PANDAS, 2020. Disponível em: <<https://pandas.pydata.org/about/>>. Acesso em: 18 de mar. 2021.

THE Python Tutorial. Python Software Foundation, 2001-2021. Disponível em: <<https://docs.python.org/3/tutorial/index.html>>. Acesso em: 17 de mar. 2021.

Thompson, J et al. Sentiment analysis of player chat messaging in the video game StarCraft 2: Extending a lexicon-based model. *Knowledge-Based Systems*. v.137, p.149-162. dez. 2017. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0950705117304240?via%3Dihub>>. Acesso em 28 out. 2020.

TU, Joe. A Daily Dosage of Toxicity in Dota 2. *HCIGames*, 2018. Disponível em: <<https://hcigames.com/gaming-experiences/a-daily-dosage-of-toxicity-in-dota2/>>. Acesso em: 05 de maio. 2021.

WASKON, Michael. Seaborn: statistical data visualization. Seaborn, 2012-2020. Disponível em: <<https://seaborn.pydata.org>>. Acesso em: 25 de mar. 2021.

WHAT is NumPy? NumPy, 2021. Disponível em: <<https://numpy.org/doc/stable/user/whatisnumpy.html>>. Acesso em: 2 de abr. 2021.

WORD Cloud API. Word Cloud API, s/d. Disponível em: <
<https://wordcloudapi.com> >. Acesso em 25 de nov. 2021

ZHAO, Alice, **Processamento de Linguagem Natural em Python**. YouTube.
Disponível em: <<https://youtu.be/xvqsFTUsOmc>>. Acesso em: 12 de out. 2020.
01:51:02

APÊNDICE A – Código das implementações

Abaixo é mostrado o código utilizado para o desenvolvimento do estudo apresentado no trabalho. Estes códigos estão inseridos na plataforma *Google Colab*.

Os dois algoritmos de agrupamentos propostos neste trabalho foram desenvolvidos em documentos diferentes para melhor manuseio durante o desenvolvimento do trabalho. A primeira parte é referente ao algoritmo que mostra o pré-processamento, a utilização da *WordCloud* e da utilização da técnica LDA enquanto a segunda é referente ao algoritmo que utiliza a técnica *K-Means*.

Parte 1 – Pré-processamento, *WordCloud* e LDA:

INICIO

```
#Conectar ao google drive (Testes realizados por esta forma)
1. from google.colab import drive
2. drive.mount('/content/drive')

#upload de arquivos
#from google.colab import files
#uploaded = files.upload()

#import numpy as np
3. import pandas as pd
4. import matplotlib.pyplot as plt
5. %matplotlib inline
6. import io
7. import unicodedata
8. import string
9. import nltk
10.     nltk.download('stopwords')
11.     nltk.download('wordnet')
12.     nltk.download('punkt')
13.     nltk.download('averaged_perceptron_tagger')
#Leitura do google drive (testes feitos desta maneira)
14. #data = pd.read_csv('/content/drive/My Drive/chat1.csv',
    encoding= #'utf-8', sep = ',')
```

```

#Leitura do arquivo para upload da máquina
#data = pd.read_csv(io.BytesIO(uploaded['chat1.csv']))

15.     filtro  = data['match_id'] < 1611
16.     data = data[filtro]
17.     data

#Leitura do google drive (testes feitos desta maneira)
18.     emocao = pd.read_csv('/content/drive/My Drive/base.csv',
encoding= 'utf-8', sep = ',')

#Leitura do arquivo para upload da máquina
#emocao = pd.read_csv(io.BytesIO(uploaded['emocao.csv'])
)

#Fazendo com que toda a coluna de chat fique em formato
string (mantendo o chat original p/ caso precise)
19.     data['chat'] = data['key'].apply(str)

#Lower case - letra minuscula
20.     data['chat'].str.lower()

#Carregando BIBLIOTECA DE EXPRESSÕES REGULARES para reti
rada de pontuação
21.     import re

#pontuação
22.     data['chat'] = data['chat'].map(lambda x:re.sub('[,\.\!?!]
', ' ', str(x)))
23.     data['chat'] = data['chat'].map(lambda x:re.sub(r'\s+',
' ', str(x)))
24.     data['chat'] = data['chat'].map(lambda x:re.sub('[^a-zA-
Z]', ' ', str(x)))

#criando uma lista de stopwords
25.     stopwords = nltk.corpus.stopwords.words('english')

#removendo algumas colunas
26.     data = data.drop(columns=['slot', 'time', 'unit'], axis=
1)
27.     data_list = data.chat.tolist() #transformando o datafram
e em uma lista

#Limpar tudo entre "<...>" e criar um dataframe
28.     data_list_frame = pd.DataFrame([re.sub('<.*>', ' ', str(d
ata_list[i])) for i in range(len(data_list))], columns=['text
'])

```

```

#def para remoção de stopwords
29. def removesw (frases):
30.     frase= []
31.     for palavras in frases:
32.         semsw = [p for p in palavras.split() if p not in stopwor
ds]
33.         frase.append((semsw))
34.     return frase

35.     data_list = removesw(data_list)

```

WORDCLOUD

```

#Importando biblioteca wordcloud
36. from wordcloud import WordCloud

37.     nuvem_list = ','.join(list(str(p) for p in data_list))

#Criando objeto da nuvem
38.     nuvemWC = WordCloud(background_color="white", max_words=
5000, contour_width=3, contour_color='steelblue', height= 500
, width= 1000)

#Gerando a nuvem
39.     nuvemWC.generate(nuvem_list)

#Visualizando nuvem
40.     nuvemWC.to_image()

```

AMOSTRAGEM DAS 10 PALAVRAS MAIS PRESENTES NO DATAFRAME

```

41.     sns.set_style('whitegrid')
42.     %matplotlib inline

# Devido o Plotly estar sendo utilizado no Google Collab
, precisaremos definir
# a função abaixo e chamá-
la sempre que quisermos exibir um gráfico

43.     def configure_plotly_browser_state():
44.         import IPython
45.         display(IPython.core.display.HTML(''
a. <script src="/static/components/requirejs/require.js"></
script>

```



```

b. <script>
    i. requirejs.config({
    ii. paths: {
    iii. base: '/static/base',
    iv. plotly: 'https://cdn.plot.ly/plotly-
        1.5.1.min.js?noext',
    v. },
    vi. });
c. </script>
d. ''))

#função
46. def top10palavras(count_data, count_vectorizer):
47.     import matplotlib.pyplot as plt
48.     palavras = count_vectorizer.get_feature_names()
49.     total_counts = np.zeros(len(palavras))
50.     for t in count_data:
    a. total_counts+=t.toarray()[0]

51.     count_dict = (zip(palavras, total_counts))
52.     count_dict = sorted(count_dict, key=lambda x:x[1], rever
se=True)[0:10]
53.     palavras = [w[0] for w in count_dict]
54.     counts = [w[1] for w in count_dict]
55.     x_pos = np.arange(len(palavras))

56.     plt.figure(2, figsize=(15, 15/1.6180))
57.     plt.subplot(title='10 most common palavras')
58.     sns.set_context("notebook", font_scale=1.25, rc={"lines.
linewidth": 2.5})
59.     sns.barplot(x_pos, counts, palette='husl')
60.     plt.xticks(x_pos, palavras, rotation=90)
61.     plt.xlabel('palavras')
62.     plt.ylabel('counts')
63.     plt.show()

64.     count_vectorizer = CountVectorizer(stop_words='english')
    #Ajustar e transformar os títulos processados
65.     count_data = count_vectorizer.fit_transform(data['chat']
)
    #Visualizar as 10 palavras mais frequentes
66.     top10palavras(count_data, count_vectorizer)

```

VISUALIZAÇÃO DE DADOS LDA

```

67.     import warnings
68.     warnings.simplefilter("ignore", DeprecationWarning)

```

```

#Carregando o LDA model do sk-learn
69.     from sklearn.decomposition import LatentDirichletAllocation as LDA

        #função que fará um loop sobre os componentes do modelo
        LDA para encontrar as maiores semelhanças
70.     def print_topics(model, count_vectorizer, n_top_words):
71.         words = count_vectorizer.get_feature_names()
72.         for topic_idx, topic in enumerate(model.components_):
73.             print("\nTopic #%d:" % topic_idx)
74.             print(" ".join([words[i]
75.                             for i in topic.argsort()[: -n_top_words - 1:-1]]))
        #busca as palavras correspondentes no CountVectorizer

        #Ajuste dos dois parametros
76.         number_topics = 2
77.         number_words = 15
        #Criação e ajuste do modelo
78.         lda = LDA(n_components=number_topics, n_jobs=-1)
79.         lda.fit(count_data)
        #Print dos tópicos encontrados pelo LDA
80.         print("Topics found via LDA:")
81.         print_topics(lda, count_vectorizer, number_words)

82.     from pyLDAvis import sklearn as sklearn_lda
83.     import pickle
84.     import pyLDAvis
85.     import os
86.     import base64
87.     LDAvis_data_filepath = os.path.join('./ldavis_prepared_'
        + str(number_topics))

        #Como demora um tempo de execução, mudar o if pra false
88.     if 1 == 1:
89.         LDAvis_prepared = sklearn_lda.prepare(lda, count_data, c
        ount_vectorizer)
90.         with open(LDAvis_data_filepath, 'wb') as f:
91.             pickle.dump(LDAvis_prepared, f)

        # carregar os dados pyLDAvis pré-processados do disco
92.         with open(LDAvis_data_filepath, 'rb') as f:
93.             LDAvis_prepared = pickle.load(f)

94.         pyLDAvis.save_html(LDAvis_prepared, './ldavis_prepared_'
        + str(number_topics) + '.html')

95.         pyLDAvis.display(LDAvis_prepared)

```

GRAFICOS LDA

```

#Inserir as palavras mais relevantes de cada tópico
gerados pelo modelo LDA
96. topicos = {'topic0':['gg', 'ez', 'game', 'mid', 'haha',
'nice', 'fuck', 'end', 'dont', 'guys', 'commend', 'good', 'wt
f', 'rofl', 'mmr'], 'topic1':['lol', 'wp', 'report', 'team',
'xd', 'ggwp', 'ty', 'pls', 'noob', 'just', 'fucking', 'shit',
'sf', 'like', 'ur']}

#função para contar palavras
97. def count_word(word, text):
98.     counter = 0
99.     for word in word:
100.         for p in text.split():
101.             if word == p:
102.                 counter+=1
103.         return counter

# Função para classificar cada frase em um tópico, verif
icando o tópico que contém mais palavras.
# Para cada frase, cria um dicionário {topic: qtd words
da lista} e depois verifica qual é a chave no dicionário
com o maior número, que será o número do tópico a ser ad
icionado a uma lista e devolvido.

104. def assign_topic(topic, text):
105.     assign = []
106.     for row in text:
107.         topic_counts = {}
108.         for key, value in topic.items():
109.             topic_counts[key] = count_word(value, row) #use function
previously created
110.         assign.append(max(topic_counts, key=lambda k: topic_coun
ts[k]))
111.     return assign

# A lista de tópicos que irão no dataframe
112. topics_list = assign_topic(topicos, data_list_frame.text
)

# Adicionando a coluna de tópicos ao dataframe
113. data_list_frame['topics'] = topics_list

114. import plotly.express as px
115. import plotly.offline as py
116. import plotly.graph_objs as go

```

```

117.     data_list_frame['topics'].value_counts()

118.     data_list_frame.topics.value_counts().plot(kind='pie', a
        utopct='%.2f%%')
119.     plt.axis('equal', )    #Para deixar o gráfico

```

APRENDIZADO

```

        # remoção de células vazias NaN
120.     emocao = emocao[emocao['text'].notnull()]
121.     emocao = emocao[emocao['topics'].notnull()]
        #removendo algumas colunas
122.     emocao = emocao.drop(columns=['Unnamed: 0'], axis=1)

123.     base = emocao.values.tolist()

124.     from nltk.stem import PorterStemmer
125.     from nltk.tokenize import word_tokenize
126.     from nltk.stem.snowball import SnowballStemmer

        #Aplicando stemmer
127.     def aplicastemmer(texto):
128.         stemmer =SnowballStemmer("english")
129.         frasesstemming = []
130.         for (palavras, emocao) in texto:
131.             comstemming = [str(stemmer.stem(p)) for p in palavras.sp
                lit() if p not in stopwords]
132.             frasesstemming.append((comstemming, emocao))
133.         return frasesstemming

134.     frases_com_stemming_treinamento = aplicastemmer(base)
        #print(frases_com_stemming_treinamento)

        #Buscando palavras da base
135.     def buscapalavras(frases):
136.         todas_palavras = []
137.         for (palavras, emocao) in frases:
138.             todas_palavras.extend(palavras)
139.         return todas_palavras

140.     palavras = buscapalavras(frases_com_stemming_treinamento
        )

        #Buscando palavras únicas

```

```

141.     def busca_freq(palavras):
142.         palavras = nltk.FreqDist(palavras)
143.         return palavras

144.     frequencia = busca_freq(palavras)
145.     #print(frequencia.most_common(50))

146.     def busca_unicas(frequencia):
147.         freq = frequencia.keys()
148.         return freq

149.     palavras_unicas = busca_unicas(frequencia)
150.     #print(palavras_unicas)

151.     #extração de palavras das frases
152.     def extratorpalavras(documento):
153.         doc = set(documento)
154.         caracteristicas = {}
155.         for palavras in palavras_unicas:
156.             caracteristicas['%s' % palavras] = (palavras in doc)
157.         return caracteristicas

158.     caracteristicas_frase = extratorpalavras(['fuck', 'haha'
159. ]) #teste
160.     print(caracteristicas_frase)

161.     #base com extração de caracteristicas
162.     base_completa = nltk.classify.apply_features(extratorpal
163. avras, frases_com_stemming_treinamento)
164.     #base_completa

```

CLASSIFICADOR

```

165.     #constroi a tabela de probabilidade
166.     classificador = nltk.NaiveBayesClassifier.train(base_com
167. pleta)

168.     print(classificador.show_most_informative_features(10))

```

AVALIANDO O CLASSIFICADOR

```

169.     data_list_frame = data_list_frame[data_list_frame['text'
170. ].notnull()]
171.     data_list_frame = data_list_frame[data_list_frame['clust
172. er'].notnull()]

```

```

162.     base_teste = data_list_frame.values.tolist()
        #base_teste

163.     frases_com_stemming_teste = aplicastemmer(base_teste)
164.     palavras_teste = buscapalavras(frases_com_stemming_teste
        )
165.     frequencia_teste = busca_freq(palavras_teste)
166.     palavras_unicas_teste = busca_unicas(frequencia_teste)
167.     base_completa_teste = nltk.classify.apply_features(extra
        torpalavras, frases_com_stemming_teste)

168.     print(nltk.classify.accuracy(classificador, base_complet
        a_teste))

        #MATRIZ DE CONFUSÃO
169.     from nltk.metrics import ConfusionMatrix

170.     esperado = []
171.     previsto = []
172.     for (frase, topic) in base_completa_teste:
173.         resultado = classificador.classify(frase)
174.         previsto.append(resultado)
175.         esperado.append(topic)

176.     matriz_confusao = ConfusionMatrix(esperado, previsto)
177.     print(matriz_confusao)

```

Parte 2 – K-Means:

```

1. from sklearn.feature_extraction.text import TfidfVectorizer
2. from sklearn.cluster import KMeans

3. kmeans = KMeans(n_clusters=2, init='random')
4. chat = data['key'].values.astype("U") #converte os dados para
    Unicode (string)

5. vectorizer = TfidfVectorizer(stop_words='english')
6. features = vectorizer.fit_transform(chat)

7. k=2
8. modelo = KMeans(n_clusters=k, init='k-means++', max_iter=100)
9. modelo.fit(features)

10.     print("Centroides: \n")
11.     centroides = modelo.cluster_centers_.argsort()[:, :-1]
12.     terms = vectorizer.get_feature_names()

```

```

13.     for i in range(k):
14.         print("Cluster %d:" % i)
15.         for j in centroides[i, :30]: #printa 15 termos de cada c
            luster
            a. print (' %s' % terms[j])
16.         print('-----')

            #Inserir os termos de cada tópico
17.         topicos = {'topic0':['gg', 'ez', 'wp', 'team', 'guys', '
            mid', 'lol', 'end', 'noob', 'sf', 'ty', 'haha', 'ok', 'pa', '
            es', 'pro', 'plz', 'carry', 'wr', 'time', 'yeah', 'win', 'com
            mend', 'xd', 'game', 'ggggggggggggggg', 'im', 'pudge', 'hihi',
            'hhehe'], 'topic1':['lol', 'ez', 'haha', 'ggwp', 'report', '
            end', 'game', 'ty', 'noob', 'wp', 'fuck', 'mid', 'hahaha', 'x
            d', 'nice', 'team', 'pls', 'commend', 'wait', 'wtf', 'ok', 'j
            ust', 'dont', 'shit', 'good', 'guys', 'sf', 'fucking', 'like'
            , 'rofl']}]

            #função para contar palavras
18.         def count_word(word, text):
19.             counter = 0
20.             for word in word:
21.                 for p in text.split():
22.                     if word == p:
23.                         counter+=1
24.             return counter

25.         def assign_topic(topic, text):
26.             assign = []
27.             for row in text:
                a. topic_counts = {}
                b. for key, value in topic.items():
                    i. topic_counts[key] = count_word(value, row) #use a
                        função previamente criada
                c. assign.append(max(topic_counts, key=lambda k: topic_coun
                    ts[k]))
28.             return assign

29.         topics_list = assign_topic(topicos, data_list_frame.text
            )
30.         data_list_frame['cluster'] = topics_list
31.         data_list_frame.cluster.value_counts().plot(kind='pie',
            autopct='%.2f%%')
32.         plt.axis('equal', )

```

APRENDIZADO

```
# remoção de células vazias NaN
```

```

33.     emocao = emocao[emocao['text'].notnull()]
34.     emocao = emocao[emocao['topics'].notnull()]
        #removendo algumas colunas
35.     emocao = emocao.drop(columns=['Unnamed: 0'], axis=1)

36.     base = emocao.values.tolist()
37.     from nltk.stem import PorterStemmer
38.     from nltk.tokenize import word_tokenize
39.     from nltk.stem.snowball import SnowballStemmer

        #Aplicando stemmer
40.     def aplicastemmer(texto):
41.         stemmer =SnowballStemmer("english")
42.         frasesstemming = []
43.         for (palavras, emocao) in texto:
44.             comstemming = [str(stemmer.stem(p)) for p in palavras.sp
lit() if p not in stopwords]
45.             frasesstemming.append((comstemming, emocao))
46.         return frasesstemming
47.     frases_com_stemming_treinamento = aplicastemmer(base)
        #print(frases_com_stemming_treinamento)

        #Buscando palavras da base
48.     def buscapalavras(frases):
49.         todas_palavras = []
50.         for (palavras, emocao) in frases:
51.             todas_palavras.extend(palavras)
52.         return todas_palavras

53.     palavras = buscapalavras(frases_com_stemming_treinamento
)

        #Buscando palavras únicas
54.     def busca_freq(palavras):
55.         palavras = nltk.FreqDist(palavras)
56.         return palavras

57.     frequencia = busca_freq(palavras)
        #print(frequencia.most_common(50))

58.     def busca_unicas(frequencia):
59.         freq = frequencia.keys()
60.         return freq

61.     palavras_unicas = busca_unicas(frequencia)
        #print(palavras_unicas)

        #extração de palavras das frases

```



```

62.     def extratorpalavras(documento):
63.         doc = set(documento)
64.         caracteristicas = {}
65.         for palavras in palavras_unicas:
66.             caracteristicas['%s' % palavras] = (palavras in doc)
67.         return caracteristicas

68.     caracteristicas_frase = extratorpalavras(['fuck', 'haha'
69.         ]) #teste
70.     print(caracteristicas_frase)

    #base com extração de caracteristicas
70.     base_completa = nltk.classify.apply_features(extratorpalavras, frases_com_stemming_treinamento)
    #base_completa

```

CLASSIFICADOR

```

    #constroi a tabela de probabilidade
71.     classificador = nltk.NaiveBayesClassifier.train(base_completa)

72.     print(classificador.show_most_informative_features(10))

```

AVALIANDO O CLASSIFICADOR

```

73.     data_list_frame = data_list_frame[data_list_frame['text']
74.         ].notnull()]
75.     data_list_frame = data_list_frame[data_list_frame['cluster']
76.         ].notnull()]

75.     base_teste = data_list_frame.values.tolist()
    #base_teste

76.     frases_com_stemming_teste = aplicastemmer(base_teste)
77.     palavras_teste = buscapalavras(frases_com_stemming_teste)
78.     frequencia_teste = busca_freq(palavras_teste)
79.     palavras_unicas_teste = busca_unicas(frequencia_teste)
80.     base_completa_teste = nltk.classify.apply_features(extratorpalavras, frases_com_stemming_teste)

81.     print(nltk.classify.accuracy(classificador, base_completa_teste))

```

```
#MATRIZ DE CONFUSÃO
82. from nltk.metrics import ConfusionMatrix

83. esperado = []
84. previsto = []
85. for (frase, topic) in base_completa_teste:
86.     resultado = classificador.classify(frase)
87.     previsto.append(resultado)
88.     esperado.append(topic)

89. matriz_confusao = ConfusionMatrix(esperado, previsto)
90. print(matriz_confusao)
```

Apêndice B – Links dos Arquivos

Abaixo são disponibilizados os links para download dos *dataframes* utilizados durante a pesquisa, ambos ainda sem nenhum tipo de modificação ou limpeza de dados.

Para melhor visualização dos resultados, recomenda-se utilização dos arquivos no *Google Drive* ou seu upload direto a partir da plataforma do *Google Colab*. No Link 1 é apresentado o *dataframe* da base completa de chats do jogo DotA 2 utilizada para realização dos estudos. No Link 2, é apresentado o *dataframe* da base utilizada para o aprendizado do modelo para validação da performance dos algoritmos de agrupamento.

Link 1 - Github

<https://github.com/yBlade/Data-frames-TCC>

Link 2 - *GoogleDrive*

<https://drive.google.com/file/d/1P8Gh1TRu62rwLoWaFFqEA0FRBH8anPIk/view?usp=sharing>



**PUC
GOIÁS**

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Marcos Antônio da Silva Nascimento
do Curso de Ciência da Computação, matrícula 2014.1.0028.0448-7,
telefone: 62 99658-6433 e-mail marcos.nascimento0321@gmail.com, na qualidade de titular dos
direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor),
autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o
Trabalho de Conclusão de Curso intitulado
Estudo de Algoritmos para Análise de Sentimentos em Jogos Online por Tópicos
de Conversação, gratuitamente, sem ressarcimento dos direitos autorais, por 5
(cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial
de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som
(WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da
área; para fins de leitura e/ou impressão pela internet, a título de divulgação da
produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 09 de junho de 2021.

Assinatura do(s) autor(es):

Marcos Antônio

Nome completo do autor: MARCOS ANTONIO DA SILVA NASCIMENTO

Assinatura do professor-orientador:

[Assinatura]

Nome completo do professor-orientador:

ANIBAL SANTOS JUKEMURA