UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

VICTOR HUGO PENHALVES MARTINS

Algoritmo paralelo para análise comportamental de usuários de mídias sociais na detecção de distúrbios mentais

São José do Rio Preto
2020

VICTOR HUGO PENHALVES MARTINS

Algoritmo paralelo para análise comportamental de usuários de mídias sociais na detecção de distúrbios mentais

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiador: Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq. Proc. 134172/2017-5.

Prof. Dr. Carlos Roberto Valêncio
Orientador

São José do Rio Preto
2020

M386a

Martins, Victor Hugo Penhalves

Algoritmo paralelo para análise comportamental de usuários de mídias sociais na detecção de distúrbios mentais / Victor Hugo Penhalves Martins. -- São José do Rio Preto, 2020

72 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Carlos Roberto Valêncio

1. Ciência da computação. 2. Processamento de textos (Computação). 3. Processamento paralelo (Computadores). 4. Big data. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

VICTOR HUGO PENHALVES MARTINS

Algoritmo paralelo para análise comportamental de usuários de mídias sociais na detecção de distúrbios mentais

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiador: Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq. Proc. 134172/2017-5.

Comissão examinadora:

Prof. Dr. Carlos Roberto Valêncio
UNESP – São José do Rio Preto - SP
Orientador

Prof. Dr. Geraldo Francisco Donegá Zafalon
UNESP – São José do Rio Preto - SP

Prof. Dr. Angelo Cesar Colombini
Universidade Federal Fluminense – Niterói - RJ

São José do Rio Preto
28 de fevereiro de 2020

Agradecimentos

Agradeço a todos que de alguma forma me ajudaram e estiveram ao meu lado durante essa jornada. Em especial, agradeço:

Aos meus pais, Raquel e Sinval, por todo amor, pela educação que me proporcionaram e por tornarem esse momento possível. Sem vocês não chegaria até aqui.

Aos meus familiares, que sempre me incentivaram e me motivaram. Vô Ulisses e Tia Neide (*in memorian*).

À minha namorada e companheira de vida, Nayara, por estar sempre ao meu lado, nos momentos bons e ruins. Você me fez crescer muito, obrigado.

Ao Prof. Dr. Carlos Roberto Valêncio, meu orientador, pela confiança, paciência e pelos ensinamentos. Sou muito grato pelas oportunidades e por permitir a execução desse trabalho. O que me tornei hoje, tanto no profissional quanto no pessoal, devo muito ao senhor.

Ao Prof. Dr. Geraldo Francisco Donegá Zafalon e Prof. Dr. Angelo César Colombini, obrigado pelas contribuições dadas no decorrer deste projeto.

Aos amigos do Grupo de Banco de Dados - GBD, dos mais antigos aos mais novos, obrigado pelos ensinamentos, amizade, companheirismo e ajuda para o meu desenvolvimento.

Aos amigos do Futsal, obrigado pela amizade, vocês tornaram os meus dias na UNESP mais felizes.

Ao Conselho Nacional de Pesquisa e Desenvolvimento Técnico Científico – CNPQ, pela concessão da bolsa de pesquisa, sob o nº 134172/2017-5.

"*O que realmente importa na vida é o que se faz com o tempo que nos é dado*"

J. R. R. Tolkien (1954, p. 44)

Resumo

A quantidade de dados tem crescido significativamente nos últimos anos, principalmente em formatos de textos e não estruturados, com a colaboração efetiva das mídias sociais. Tais plataformas podem ser definidas como aplicativos de internet que podem ser web ou mobile e permitem a criação, acesso e a troca de conteúdos criados por usuários. Com isso, o conjunto de dados produzidos por essas mídias podem ser chamados de *Big Data* e são especialmente importantes para pesquisas computacionais de extração de conhecimento. O termo *Big Data* pode ser definido como um grande volume de dados complexos provenientes de múltiplas fontes que desafiam a capacidade de armazenamento e processamento dos computadores com as tecnologias atuais. Com isso, as técnicas de programação distribuída e paralela têm sido amplamente utilizadas a fim de retornar em tempo hábil os resultados dos algoritmos de extração de conhecimento em dados de mídias sociais. Tendo em vista as características dos dados criados nas mídias sociais e o aumento de pessoas no mundo com problemas relacionados a transtornos de saúde, ferramentas que analisam esses dados para encontrar correlações podem contribuir para o cenário atual. Dessa forma, a contribuição científica deste trabalho está no desenvolvimento de algoritmos paralelos para prospecção de conhecimento em dados textuais, com foco em mídias sociais, que permita a classificação dos indivíduos em classes comuns e que considera o contexto inserido. Os resultados de desempenho indicam que a ferramenta com abordagem paralela desenvolvida foi capaz de reduzir em cerca de 11 vezes o tempo de pré-processamento, extração de características e classificação.

Palavras-chave: Ciência da computação. Processamento de textos (Computação). Processamento paralelo (Computadores). Big data.

Abstract

The amount of data has grown significantly in recent years with the effective collaboration of social media. Such platforms can be defined as internet applications that can be web or mobile and allow the creation, access and exchange of user-created content. With this, the data set produced by these media can be called Big Data and are especially important for computational searches of knowledge extraction. The term Big Data can be defined as a large volume of complex data from multiple sources that challenge the storage and processing capacity of computers with today's technologies. In this sense, the techniques of framework Apache Spark and its parallelized implementation have been widely used to return in a timely manner the results of the algorithms of knowledge extraction in social media data. Given the large amount of data generated is social media and the increase of people in the world with problems related to health disorders, tools that analyze these data to find correlations can contribute to the current scenario. Thus, the scientific contribution of this work is in the development of parallel algorithms for prospecting knowledge in textual data, with a focus on social media, which allows the classification of individuals in common classes and considering the inserted context. The performance results indicate that the tool with a parallel approach developed was able to reduce the pre-processing time, extraction of characteristics and classification by approximately 11 times.

Keywords: Computer science. Text mining. Parallel processing. Big data.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
AWS	*Amazon Web Service*
CSV	*Comma-Separeted Values*
GBD	Grupo de Banco de Dados
HABM	Hospital Dr. Adolfo Bezerra de Menezes
HDFS	*Hadoop Distributed File System*
HTML	*HyperText Markup Language*
JSON	*JavaScript Object Notation*
KDT	*Knowledg Discovery in Texts*
NLP	*Natural Language Processing*
NoSQL	*Not Only* SQL
OMS	Organização Mundial da Saúde
SQL	*Structured Query Language*
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency – Inverse Document Frequency*
WHO	*World Health Organization*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Com a evolução das tecnologias dos *smartphones*, a velocidade e disponibilidade das redes móveis, cada vez mais pessoas estão conectadas na internet. Muitas dessas pessoas são ativas e estão frequentemente conectadas em mídias sociais, e estima-se que existam mais de 1 bilhão de usuários em todo o mundo (GOLE; TIDKE, 2015).

Os sites de redes sociais, do inglês *social networking sites* - SNSs, de fato, se tornaram um meio integral de comunicação na vida cotidiana das pessoas (FOX; MORELAND, 2015), em que são gerados uma enorme quantidade de dados e que aguardam para serem explorados (GOLE; TIDKE, 2015).

O Facebook[1] é o SNS mais abrangente, com cerca de 845 milhões de usuários ativos, que permite às pessoas produzirem, espalharem e trocarem informações sobre diversos assuntos. O Twitter[2], por sua vez, restringe o tamanho da mensagem em 280 caracteres, o que permite uma comunicação nítida e direcionada, além disso, são produzidos cerca de 500 milhões de *tweets* por dia nessa plataforma (KRIKORIAN, 2013) (WU et al., 2016).

Tendo isso em vista, o termo *Big Data* é usado para se referir a volumes de dados que ultrapassam a casa dos exabytes (10^{18}), quantidades estas que excedem a capacidade dos sistemas comuns de armazenamento e de processamento. O grande volume de dados é apenas uma das características do *Big Data*, sendo também destacadas a velocidade em que os dados são criados, as variedades de fontes e formatos, valores e complexidades (KAISLER et al., 2013).

[1] https://www.facebook.com/
[2] https://twitter.com/

O processamento dos dados para conjuntos *Big Data*, demandam técnicas eficientes para realizar a extração de conhecimento. Nos últimos anos, sistemas baseados em *MapReduce* surgiram como uma solução escalável e eficaz para o processamento de dados massivamente paralelo, o que possibilita suportar grandes análises de dados (DEAN; GHEMAWAT, 2008), sua versão de código aberto mais usada é o Apache Hadoop[3] (AJI et al., 2013). Porém, com os avanços das tecnologias surgiu o Apache Spark[4], que tem se mostrado uma boa abordagem para ser utilizada juntamente com mídias sociais (SHORO; SOOMRO, 2015).

De acordo com a Organização Mundial da Saúde - OMS, a saúde é um estado de completo bem-estar físico, mental e social e não apenas a ausência de doença ou enfermidade. Portanto, a saúde mental, pode ser afetada por uma série de fatores socioeconômicos que necessitam de abordagens com estratégias abrangentes de promoção, prevenção, tratamento e recuperação pelos responsáveis pela saúde (WORLD HEALTH ORGANIZATION, 2013).

Estima-se que no Brasil, 23 milhões de pessoas, aproximadamente 11% da população, necessitam de algum tipo de atendimento em saúde mental e pelo menos 5 milhões de brasileiros sofrem com transtornos mentais mais graves, segundo a Associação Brasileira de Psiquiatria - ABP[5].

De acordo com o relatório digital mundial de 2018 (*Global Digital Report*)[6], 139,1 milhões de habitantes do Brasil possuem internet, desses, aproximadamente 93% são ativos em redes sociais, e em um ano houve um crescimento de 7% desse número. Com base no crescente uso das redes sociais e o valor que essas mídias podem conter, estudos sobre esses dados podem ter uma importante contribuição para a área da saúde (BOSAK; PARK, 2018).

Com base nos desafios apresentados, têm-se nas mídias sociais uma ferramenta potencial para detectar e classificar distúrbios mentais em indivíduos (DE CHOUDHURY et al., 2013). Uma abordagem eficiente é a utilização de técnicas de aprendizado de máquina, que desempenham um papel fundamental na análise destes dados (MELETHADATHIL et al, 2017).

1.1 Motivação e escopo

Tendo em vista a grande quantidade de dados gerados todos os dias nas mídias sociais, bem como o possível valor desses dados produzidos, atrelado ao grande número de pessoas que

[3] <http://hadoop.apache.org/>
[4] <https://spark.apache.org/>
[5] <http://www.abp.org.br/portal/>
[6] <https://digitalreport.wearesocial.com/>

sofrem de algum tipo de transtorno mental faz com que a análise dos dados neste escopo, seja imprescindível (GOLE; TIDKE, 2015).

Por outro lado, os dados de mídias sociais contam com características únicas de linguagem e expressões, com o uso de dialetos, gírias, emojis, o que dificulta os processos convencionais de análise. Dessa forma, para alcançar os resultados esperados, os algoritmos necessitam de adaptações.

Portanto, para produzir resultados significativos e com melhorias de desempenho faz-se necessário o uso de técnicas de prospecção de conhecimento sobre dados textuais com uma abordagem paralela e consideração de contexto.

1.2 Objetivos

Com o crescente uso de mídias sociais torna-se fundamental integrar, armazenar, processar e analisar o conjunto de dados gerados por estas plataformas (GOLE; TIDKE, 2015). Com isso, o presente trabalho disponibiliza recursos que permitem integrar dados de diferentes mídias sociais, armazenar os dados de maneira eficiente e processar os dados paralelamente.

Para tal, são implementadas técnicas de pré-processamento de textos, extração de características, desenvolvimento lexical e classificação de características com o uso de programação paralela e distribuída.

Dessa forma, foi disponibilizado um ferramental eficiente para prospecção de conhecimento em postagens de redes sociais ou dados textuais diversos, com foco na área da saúde.

1.3 Contribuições

A contribuição principal do trabalho está na abordagem paralela dos algoritmos desenvolvidos e o ganho de desempenho verificado com os testes da ferramenta. Além disso, dispõe um ferramental que trata todas as etapas do processo de extração de conhecimento sobre dados textuais de forma genérica, desde o pré-processamento, até a classificação dos elementos de estudo.

Como contribuições secundárias, temos os mecanismos de tratamento de dados para a fase de pré-processamento para dados originários de mídias sociais, como a substituição de palavras que não estão em sua forma normal, tratamentos de endereços de sites, emojis, hashtags, em que se tem como produto um dicionário para o contexto de redes sociais.

1.4 Organização da dissertação

O conteúdo do trabalho é apresentado da seguinte forma: na Seção 1 foi dada a visão geral sobre as mídias sociais, a complexidade e também a necessidade de extrair conhecimento sobre esses dados, bem como a motivação, os objetivos e as contribuições do trabalho; na Seção 2 é apresentada a fundamentação teórica e o estado da arte relacionado ao ambiente de mídias sociais, *Big Data* e o processo de extração de conhecimento sobre dados textuais, assim como estratégias de desempenho e os trabalhos correlatos; na Seção 3 é apresentado o trabalho desenvolvido com detalhes de implementação e seu funcionamento; na Seção 4 são apresentados os experimentos realizados e as discussões dos resultados a fim de validar o trabalho; na Seção 5 são apresentadas as conclusões do trabalho, as contribuições e direcionamento para possíveis trabalhos futuros na área.

2 Fundamentação teórica

Os dados provenientes de mídias sociais são um bom exemplo para descrever o contexto Big Data e a mudança de paradigma que acontece atualmente na área da computação. Essa quebra de paradigma representa a necessidade de algoritmos e abordagens mais eficazes para o tratamento de grandes volumes de dados e que contam com características diferentes das convencionais, ou seja, formato, valor, origem. Desta forma, tecnologias de armazenamento e novas estratégias de desempenho são necessários para suportar os obstáculos supracitados.

Além disso, a população atual tem sofrido cada vez mais de doenças e transtornos mentais, e por outro lado, passa cada vez mais tempo conectado à internet e principalmente em redes sociais.

Por fim, é apresentado o levantamento bibliográfico a respeito de cada tema pertinente para o desenvolvimento do trabalho.

2.1 Big Data e mídias sociais

O conceito de *Big Data* tem sido endêmico na área da ciência da computação, e foi definido originalmente como volume de dados que não poderiam ser processados de maneira eficiente por métodos e ferramentas tradicionais de banco de dados (KAISLER et al., 2013).

De acordo com o grupo Gartner[7], o termo *Big Data* é usado para se referir ao grande volume, grande velocidade e grande variedade de ativos de informação que demandam formas eficientes e inovadoras de processamento das informações para uma melhor compreensão e tomada de decisões (SICULAR, 2013). A primeira parte da definição se refere às três

[7] https://www.gartner.com/technology/home.jsp

características principais e iniciais do *Big Data* segundo Laney (LANEY, 2001), são comumente chamadas de os 3 Vs, são elas:

a) Volume – se refere à quantidade de dados existente e que continua crescendo, porém, as ferramentas atuais não são capazes de processá-los, o que excede a capacidade das técnicas de armazenamento e de análise atuais (KAISLER et al., 2013) (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013).
b) Velocidade – fluxo contínuo na criação dos dados com o interesse de processamento de informação útil em tempo real (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013).
c) Variedade – os dados produzidos vêm de várias fontes e não são de uma única categoria, por exemplo, texto, imagem, vídeo, áudio, etc. (KAISLER et al., 2013) (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013) (WU, 2016).

Com base no levantamento bibliográfico é possível encontrar outras características do *Big Data*, em que são adicionados novos V's ao contexto, são eles: variabilidade, veracidade e valor, que são definidas abaixo:

d) Variabilidade – inconsistências no fluxo de dados. Com o uso das mídias sociais atrelada à certos eventos, há a geração de picos na geração de dados, com isso as cargas dos dados tornam-se difíceis de serem mantidas (KATAL; WAZID; GOUDAR, 2013).
e) Veracidade – o uso de diferentes fontes de dados pode apresentar qualidades diferentes, o que pode acarretar dificuldades de compreensão, precisão e atualização dos dados fornecidos (DONG; SRIVASTAVA, 2013) (LUKOIANOVA; RUBIN, 2014).
f) Valor – mede a utilidade dos dados na tomada de decisões, ou seja, pode-se adquirir vantagem competitiva a partir de análises e técnicas sobre dados que até então não eram conhecidos. Os resultados ajudam a encontrar tendências nos negócios, o que possibilita mudar estratégias (KAISLER et al., 2013) (KATAL; WAZID; GOUDAR, 2013).

Tendo isso em vista, um grande exemplo de aplicabilidade dos conceitos de Big Data são as mídias sociais. Tais recursos produzem um grande volume de dados diariamente, como por exemplo o Twitter, que produz cerca de 500 milhões de *tweets* por dia (KRIKORIAN, 2013).

As mídias sociais são definidas como aplicativos de internet que podem ser web ou mobile que permitem a criação, acesso e a troca de conteúdos criados por usuários que são acessíveis de forma ubíqua (KAPLAN; HAENLEIN, 2010). Dessa forma, ela é importante para pesquisas computacionais em ciências da sociedade que podem ter como objetivo de análise, por exemplo, determinar tendências de comportamento, opiniões públicas ou em questões ligadas à saúde do usuário (ATIG et al, 2014) (BOSAK; PARK, 2018).

O crescente uso das mídias sociais, por exemplo, Facebook e Twitter, servem como uma poderosa plataforma online de comunicação que permitem que milhões de usuários produzam, espalhem, compartilhem ou troquem informações a qualquer momento e em qualquer lugar. Tais informações normalmente incluem conteúdo multimídia, como texto, imagem e vídeo (WU, 2016), tais dados são denominados não estruturados (KAISLER et al., 2013).

Os quatro formatos mais comuns para disponibilizar os dados são HTML, XML, JSON E CSV, que são definidos a seguir (BATRINCA; TRELEAVEN, 2015):

a) *HyperText Markup Language* (HTML), é conhecida como uma linguagem de marcação para páginas web e outras informações que podem ser visualizadas em um navegador. Ela consiste em elementos HTML, o que inclui tags (por exemplo, ...).
b) *Extensible Markup Language* (XML) é uma linguagem de marcação para estruturar dados textuais, usa tags para definir as marcações.
c) *JavaScript Object Notation* (JSON) é um padrão aberto baseado em texto projetado para ser legível por humanos e é derivado da linguagem JavaScript.
d) *Comma-Separeted Values* (CSV) é um arquivo de valores separados por vírgula em uma tabela com uma série de linhas de texto ASCII em que os valores das colunas são separados por vírgulas e em cada linha está contido um registro.

Tendo em vista as características individuais do *Big Data* atrelada aos dados de mídias sociais, é necessário o uso de um banco de dados que suporte as operações necessárias da aplicação. Atualmente, os bancos de dados NoSQL têm sido potencialmente utilizados para aplicações em escala *web* devido aos seus suportes de escalabilidade e para esquemas dinâmicos e flexíveis (BANERJEE; SARKAR, 2016). Além do banco de dados, o processamento dos dados para conjuntos *Big Data* deve ser levado em consideração, pois estas aplicações demandam técnicas eficientes para realizar a extração de conhecimento (KAISLER et al., 2013).

2.2 Distúrbios da saúde

Os distúrbios mentais, também chamados de transtornos mentais, doenças mentais ou transtornos psiquiátricos, são um padrão comportamental ou mental que pode causar sofrimento ou afetar a capacidade de viver. Tais características podem ser persistentes, recorrentes, remitentes ou ocorrer como um único episódio (WORLD HEALTH ORGANIZATION, 2014).

A classificação internacional de doenças, chamada de CID 10, foi criada para padronizar e catalogar doenças e problemas relacionados à saúde tendo como referência a Nomenclatura Internacional de Doenças, estabelecida pela OMS. Segundo a classificação, os transtornos mentais e comportamentais são classificados com os códigos de F00 a F99 e são especificados a seguir:

a) Transtornos mentais orgânicos, inclusive os sintomáticos (F00 – F09);
b) Transtornos mentais e comportamentais devidos ao uso de substância psicoativa (F10 – F19);
c) Esquizofrenia, transtornos esquizotípicos e transtornos delirantes (F20 – F29);
d) Transtornos de humor (F30 – F39);
e) Transtornos neuróticos, transtornos relacionados com o stress e transtornos somatoformes (F40 – F48);
f) Síndromes comportamentais associadas a disfunções fisiológicas e a fatores físicos (F50 – F59);
g) Transtornos da personalidade e do comportamento do adulto (F60 – F69);
h) Retardo mental (F70 – F79);
i) Transtornos do desenvolvimento psicológico (F80 – F89);
j) Transtornos do comportamento e transtornos emocionais que aparecem habitualmente durante a infância ou a adolescência (F90 – F98)
k) Transtorno mental não especificado (F99 – F99).

De acordo com a OMS, pessoas com transtornos mentais correm maiores riscos de incapacidade e mortalidade. Por exemplo, pessoas com depressão profunda e esquizofrenia têm uma chance de 40% a 60% maior de morrer prematuramente do que a população em geral, devido a problemas de saúde que são deixados sem acompanhamento, como cânceres, doenças cardiovasculares, diabetes e infecção por HIV, ou suicídio. O suicídio é a segunda maior causa de morte entre os jovens do mundo (WORLD HEALTH ORGANIZATION, 2013).

Segundo estudos, a taxa de transtornos mentais comuns, como depressão e ansiedade, em 27 unidades de saúde da família de quatro capitais brasileiras supera os 50%. Os resultados

apontam 51,9% no Rio de Janeiro, 53,3% em São Paulo, 64,3% em Fortaleza e 57,7% em Porto Alegre (GONÇALVES et al., 2014).

Muitos fatores de risco podem influenciar os transtornos mentais, entre eles o baixo nível socioeconômico, o consumo de álcool e o estresse. Também há correlação entre os distúrbios e o uso de substâncias psicoativas. Em conjunto, os distúrbios mentais, neurológicos e de uso de substâncias psicoativas representam 13% do total de doenças em 2004. Somente a depressão representa 4,3% da carga global de doenças e está entre as maiores causas de incapacidade do mundo (WORLD HEALTH ORGANIZATION, 2013).

Além dos prejuízos para a pessoa doente, os transtornos mentais também afetam negativamente a economia: um estudo recente estimou que o impacto global acumulado em termos de perda de produção econômica superará US$16,3 milhões entre 2011 e 2030 (WORLD HEALTH ORGANIZATION, 2013).

Tendo em vista as altas taxas de pessoas com transtornos mentais, o alto índice de suicídios entre jovens, as graves consequências econômicas, ferramentas que possam ajudar em políticas de prevenção, diagnóstico e tratamento tornam-se muito importantes para a saúde pública.

2.3 Extração de conhecimento em textos

Nos últimos anos, minerar conteúdo textual para extrair informações relevantes e conhecimentos úteis tem se tornado uma tarefa importante. Os dados podem ser extraídos de textos de várias fontes, por exemplo: notícias, blogs, redes sociais, e geralmente são armazenados em bancos de dados de forma estruturada (MEJRI; AKAICHI, 2016). Mesmo que os dados produzidos nas redes sociais sejam não estruturados, por exemplo: imagens, vídeos ou GIFS e textos, há a possibilidade que eles sejam coletados e analisados, especialmente os textos. Além disso, o uso da linguagem natural torna o entendimento e interpretação do seu conteúdo uma tarefa difícil para as máquinas (BALDWIN, 2012).

Tendo isso em vista, são necessárias técnicas e abordagens eficientes para processar e extrair conhecimento destes dados, este processo é denominado de Descoberta de Conhecimento em Textos, do inglês *Knowledge Discovery in Texts* – KDT (FELDMAN; DAGAN, 1995).

O KDT é composto pelas etapas de coleta e aquisição dos dados, pré-processamento, mineração e análise de resultados. O processo se inicia com uma coleção de dados ou documentos, geralmente em formato não estruturado. Em seguida, é feita a aquisição dos dados

e a padronização dos mesmos de acordo com técnicas de pré-processamento. Após a aquisição e pré-processamento os dados estão prontos para serem analisados, nesta etapa é realizada a mineração dos textos com o objetivo de agregar valor aos dados. Por fim, os resultados são armazenados e são apresentados os resultados encontrados (FAN et al, 2006). Na Figura 1 são apresentadas as etapas KDT.

Figura 1 - Processo de Descoberta de Conhecimento em Textos - KDT.

Fonte: Adaptado de (FAN et al, 2006).

2.4 Pré-processamento de dados textuais

A etapa de pré-processamento tem como objetivo a preparação e padronização dos dados obtidos para a aplicação da mineração. Esta etapa é a mais importante para o processo de descoberta de conhecimento, pois os dados podem originar de diversas fontes e podem conter tanto informações valiosas quanto informações inúteis (NOUREEN; QAMAR; ALI, 2017).

Além disso, o pré-processamento busca diminuir a quantidade de palavras de um documento de modo a manter a qualidade e o sentido do mesmo, e transformar o dado textual original em uma representação adequada para os algoritmos de mineração e classificação (AKAICHI; DHOUIOUI; PÉREZ, 2013, REZAEIAN; MONTAZERI; LOONEN, 2017).

O pré-processamento normalmente envolve três sub-etapas: identificação de palavras (*tokens*), remoção de palavras de parada (*stop words*) e radicalização (*stemming*).

2.4.1 Identificação de palavras

A identificação de palavras, identificação de tokens ou tokenização é uma sub-tarefa do processo KDT que quebra as sequências de *strings* em palavras, palavras-chave, símbolos, frases ou outros elementos significativos. O objetivo principal da tokenização é a identificação de palavras significativas. Para tal, cada sentença é convertida em uma forma de árvore e são

descartados símbolos, verbos, advérbios e outras palavras consideras irrelevantes para o contexto de aplicação (ALAM; YAO, 2018).

2.4.2 Remoção de palavras de parada

Após as etapas de identificação de palavras, o texto passa por um método chamado de remoção de palavras de parada, em que são removidas palavras com baixo valor semântico e que não exercem influência significativa em seu sentido. Em sua maioria, as palavras de parada aparecem com mais frequência no texto, são elas: o, a, para, um, uma, entre outras (MUNKOVÁ; MUNK; VOZÁR, 2014).

Segundo (VIJAYARANI; ILAMATHI; NITHYA, 2015) (NOUREEN; QAMAR; ALI, 2017) existem três maneiras de remover as palavras de parada:

- Remoção baseada em informações visionárias – palavras com pouca relevância são removidas independentemente do número de ocorrências;
- Remoção de acordo com uma lista predefinida – utilização de uma lista com cerca de 545 palavras de parada. Podem fazer parte desta lista artigos, preposições, conjunções, entre outros;
- Remoção baseada em ocorrência de palavras – baseada na frequência em que as palavras de parada aparecem. Primeiro, as frequências das palavras são calculadas, em seguida, determinados pesos são atribuídos às palavras. Pode ser feito por meio do TF-IDF (abreviação do inglês *term frequency-inverse document frequency*, que significa frequência do termo-inverso da frequência nos documentos) e lei de Zipf.

2.4.3 Radicalização

Nesta etapa são utilizados métodos para padronizar os termos de um documento e melhorar a indexação de seu conjunto. A radicalização visa reduzir as palavras de um texto a uma raiz ou radical (*stem*), sem perda de sentido. Esse processo é feito com a remoção de prefixos e sufixos, e, portanto, haverá uma redução no número de caracteres do texto, o que proporciona uma redução significativa de tamanho e complexidade dos dados, bem como rapidez no processo de classificação (PORTER, 1980) (NOUREEN; QAMAR; ALI, 2017).

De acordo com o foco do trabalho, foram encontrados na literatura três algoritmos de radicalização para a língua portuguesa que podem ser destacados. O primeiro e mais famoso de radicalização é algoritmo de Porter, nomeado *Snowball* (PORTER, 2001), temos também o

algoritmo RSLP (ORENGO; HUYCK, 2001) e a abordagem menos custosa, *Savoy* para o português (SAVOY, 2006). A seguir cada algoritmo é apresentado com mais detalhes.

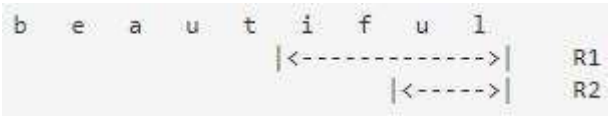
2.4.4 Algoritmo de Porter

O algoritmo original de Porter foi desenvolvido em 1980 (PORTER, 1980), e é considerado um dos algoritmos de radicalização mais famosos. A partir dele, foi desenvolvida uma versão do algoritmo para a língua portuguesa. O algoritmo funciona com a divisão da palavra em duas regiões, R1 e R2, porém, também é utilizada uma outra região RV para alguns casos. R1, R2 e RV podem ser definidas da seguinte forma:

- R1 – região que se encontra após a primeira letra não vogal (consoante ou letras com acento) depois de uma vogal;
- R2 – é a região após a primeira não vogal após uma vogal em R1, ou é a região nula no final da palavra se não houver vogal.
- RV – se a segunda letra é uma consoante, RV é a região após a próxima vogal seguinte, ou se as duas primeiras letras são vogais, RV é a região após a próxima consoante, caso contrário (consoante-vogal) RV é a região após a terceira letra. Mas RV é o fim da palavra se essas condições não forem satisfeitas.

Na Figura 2 é apresentado um exemplo em inglês das regiões R1 e R2 da palavra *beautiful*.

Figura 2 - Regiões de R1 e R2 definidas por Porter.



Fonte: (PORTER, 2001).

A seguir são apresentados os passos para o processo de radicalização segundo Porter[8].

- As letras *ã* e *õ* devem ser tratadas como uma vogal seguida por consoante. As letras *ã* e *õ* devem ser substituídas por *a~* e *o~*;
- Passo 1 – remoção de sufixos.
    - Procurar o sufixo mais longo da palavra dentro da lista de sufixos. Se o sufixo estiver em R2 deletar;
    - Se encontrar *logia* ou *logias* em R2 substituir por *log*;

[8] <http://snowballstem.org/algorithms/portuguese/stemmer.html>

    - Se encontrar *ução ou uções* em R2 substituir por *u*;
    - Se encontrar *ência* ou *ências* em R2 substituir por *ente*;
    - Se encontrar *amente* em R1, remover;
    - Se precedido por *iv*, deletar se estiver em R2 (e se for precedido por *at*, deletar se estiver em R2), caso contrário, se precedido por *os*, *ic* ou *ad,* deletar se em R2;
    - Se encontrar *mente* deletar se em R2;
    - Se precedido por *ante*, *avel* ou *ível*, deletar se em R2;
    - Se encontrar *idade*, *idades*, deletar se em R2;
    - Se precedido por *abil*, *ic* ou *iv*, deletar se em R2;
    - Se encontrar *iva*, *ivo*, *ivas*, *ivos*, deletar se em R2;
    - Se precedido por *at*, deletar se em R2;
    - Se encontrar *ira*, *iras*, substituir por *ir* se estiver em RV e precedido por *e*.
- Passo 2 – remoção de sufixos verbais. Procurar o sufixo mais longo em RV se encontrar delete. De acordo com a lista de sufixos verbais encontrados na língua portuguesa (entre eles, *ada*, *ida*, *erei*, *erdes*, *eres*, *ávamos*, *emos*, *ássemos*, etc).
- Se o passo 1 ou passo 2 alterar a palavra faça o passo 3;
- Passo 3 – remover o sufixo *i* se estiver em RV e precedido por *c*;
- Se o passo 1 ou passo 2 não alterar a palavra faça o passo 4;
- Passo 4 – sufixo residual. Se a palavra terminar com *os*, *a*, *i*, *o*, *á*, *í*, *ó* em RV, remover.
- Passo 5 – se a palavra terminar com *e*, *é*, *ê* em RV, remover, e se precedido por *gu* (ou *ci*) com *u* (ou *i*) em RV remover o *u* (ou *i*). Ou se terminar com ç remover a cedilha.
- Volte *a~* e *o~* para *ã* e *õ*.

Na Tabela 1 são apresentados alguns exemplos de radicalização de palavras segundo o algoritmo de Porter.

Tabela 1 - Exemplos de radicalização de palavras em português segundo Porter.

Palavra	**Radical**
boa	*boa*
boate	*boat*

bagagem	*bagag*
quietos	*quiet*
quimioterapia	*quimioterap*
quiosque	*quiosq*

2.4.5 Algoritmo RSLP

O algoritmo Removedor de Sufixos da Língua Portuguesa – RSLP (ORENGO; HUYCK, 2001) é um dos radicalizadores mais utilizados para língua portuguesa. O RSLP conta com 199 regras, divididas em 8 passos, conta com lista de exceções e contém um conjunto maior de regras referentes à língua portuguesa.

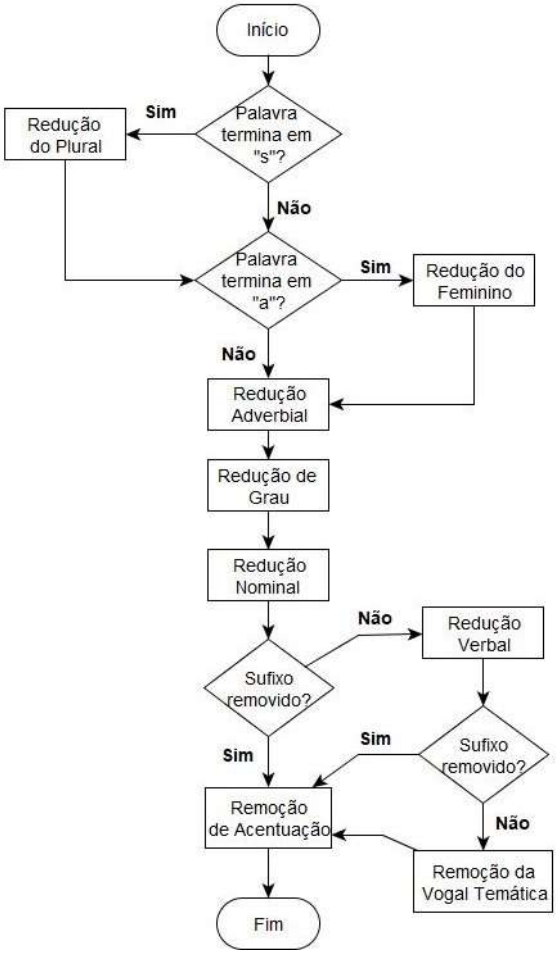
Segundo o autor, os passos são definidos da seguinte forma:

- Passo 1 – remoção de plural;
- Passo 2 – mudança de gênero do feminino para o masculino;
- Passo 3 – redução adverbial;
- Passo 4 – redução de grau, aumentativo, diminutivo ou superlativo;
- Passo 5 – redução nominal, redução de substantivos e adjetivos;
- Passo 6 – redução verbal, se a palavra não se encaixar nos passos 4 e 5. Os sufixos dos verbos são removidos até a palavra chegar em sua forma normal;
- Passo 7 – redução da vogal temática;
- Passo 8 – redução de acento;

Os passos presentes no algoritmo podem ser ilustrados por meio de um fluxograma, que é apresentado na Figura 3.

Em uma comparação entre o algoritmo RSLP e Porter, foi constatado que o RSLP diminui o tamanho do vocabulário em média 51%, enquanto o algoritmo proposto por Porter reduz em 44%. Essa diferença ocorre por conta da utilização dos termos presentes na lista de exceções no RSLP (XAVIER; GOMES; SILVA, 2013).

Figura 3 - Fluxograma de passos para o algoritmo RSLP.



Fonte: Adaptado de (ORENGO; HUYCK, 2001).

2.4.6 Algoritmo de Savoy

O algoritmo de Savoy foi escrito originalmente para a língua francesa e sua adaptação para o português se deu pela simplicidade em seu entendimento e concepção, e pela sua leveza, uma vez que a quantidade de regras e passos necessários para realizar a radicalização é muito menor que a dos algoritmos de Porter e RSLP.

Para sua execução são necessários apenas quatro passos, são eles:

- Passo 1 – remoção do plural e do sufixo *mente*, composto por 10 regras;
- Passo 2 – mudança de gênero do masculino para o feminino, composto por 13 regras;
- Passo 3 – remoção de vogal temática;
- Passo 4 – remoção de acentos.

Apesar de ser um algoritmo mais leve e conter poucas regras, este algoritmo apresenta diversos erros de *understemming*, ou seja, os sufixos não foram removidos de forma suficiente para que as palavras fossem agrupadas. No entanto é considerado o mais rápido dentre os algoritmos apresentados (XAVIER; GOMES; SILVA, 2013).

2.4.7 Desenvolvimento do léxico

Conforme o contexto das mídias sociais, se torna importante o tratamento da informalidade da linguagem adotada nesses meios. Com isso, devem ser implementadas estratégias para absorver emoções, hashtags, emojis, interjeições e acrônimos. Muitos pesquisadores criam os seus próprios léxicos (AKAICHI; DHOUIOUI; PÉREZ, 2013).

a) Acrônimos - são palavras que se formam pela junção das letras ou sílabas iniciais de palavras ou de uma expressão, por exemplo: SQN (Só Que Não), LOL (*Laughing Out Loud*), OMG (*Oh My God*), GG (*Good Game*), etc. Que apesar de alguns acrônimos serem da língua inglesa são muito utilizados na língua portuguesa.
b) Interjeições - são palavras que podem expressar sentimentos, estados emocionais ou de espírito e emoções. Alguns exemplos são: uau, ufa, hahaha, hum, uh, oh.
c) Emojis - muito utilizado nas redes sociais, também tem como função expressar emoções e estados de espírito. Os emojis podem ser definidos como ideogramas *smileys*. Eles existem em diversos gêneros, entre eles: expressões faciais, objetos, lugares, animais e tipos de clima.
d) Hashtags - é uma palavras-chave antecedida pela cerquilha (#) que os usuários utilizam para identificar o tema do conteúdo compartilhado.

2.4.8 Extração de características

A extração de características é a principal etapa deste processo, pois por meio dela são obtidos os meios para realizar os experimentos. Neste contexto, devem ser extraídas características que descrevam o estado emocional do usuário (NOUREEN; QAMAR; ALI, 2017). Na literatura, foram encontradas três abordagens para realizar a extração de características dos textos, que são apresentadas a seguir:

a) Rótulos de Humor – os rótulos de humor são comumente utilizados no Facebook. A mídia permite postagens com rótulos de acordo com uma lista de

emoções disponível. Por exemplo: Sentindo-se animado, feliz, triste, sozinho, cansado, entre outros;

b) Hashtags – as hashtags são comumente utilizadas pelos usuários nas redes sociais para representar diferentes acontecimentos ao seu redor. Também podem ser utilizados para classificação de comportamento;
c) Características semânticas – extrair características por meio análises linguísticas, ou seja, padrões de escrita, uso frequente de palavras e análise de sentimento.

2.4.9 Seleção de características

Após a etapa de extração de características dos textos, torna-se necessário selecionar os melhores atributos para análise, dessa forma o processamento não será desperdiçado com características que não contenham informação útil.

Esse processo pode ser realizado com métodos encontrados na literatura, entre eles: ponderação booleana (*boolean weighting*), frequência de documentos, frequência do termo inverso da frequência nos documentos (*Term Frequency Inverse Document Frequency* – TFIDF), ganho de informação e laço (*lasso*).

a) Ponderação booleana –a presença e ausência de determinada palavra depende da frequência em que essa palavra ocorre em cada classe. Esse método retorna apenas dois resultados, se a palavra estiver presente pelo menos uma vez, é marcada como 1, caso contrário 0;
b) TFIDF – é uma medida estatística que tem como objetivo indicar a relevância de uma palavra de um documento em relação à uma coleção de documentos;
c) Ganho de informação – quanto maior o número de palavras presentes nos textos mais problemas o processo de classificação pode trazer à aplicação. Portanto, esse método realiza uma predição se determinada palavra irá aparecer ou não. Torna o processo de seleção eficiente, mas não é um processo trivial. (DIVYA; NANDA KUMAR, 2015);
d) Laço – é um modelo que utiliza regressão logística e seleciona a característica concorrentemente. Assim, deve ter cuidado com o desempenho da aplicação.

2.5 Classificação

Esta etapa é uma das mais importante do processo de descoberta de conhecimento sobre textos, pois por meio dela é possível determinar quais textos pertencem a determinadas classes. Além disso, são importantes pela sua capacidade de organizar grandes quantidades de dados que são apresentadas de forma não estruturada (ALTINEL; GANIZ, 2018).

Para a classificação podemos encontrar três classes de algoritmos: supervisionado, não supervisionado ou semi supervisionado. Esses tipos de algoritmos podem ser executados do início ao fim com uma entrada de dados de treinamento, sem a entrada de dados de treinamento ou com entrada parcial de dados de treinamento.

De acordo com a literatura, os algoritmos mais utilizados para classificação, especialmente para classificar comportamentos, entre eles é possível destacar os algoritmos Naive Bayes, Máquina de Vetores de Suporte (*Support Vector Machine* - SVM) e K-Means (NOUREEN; QAMAR; ALI, 2017).

2.5.1 Naive Bayes

O algoritmo de classificação Naive Bayes é provavelmente o mais simples e também o mais comum classificador genérico utilizado. Ele também é extensivamente utilizado em abordagens de aprendizado de máquina para categorizar textos de acordo com a frequência das palavras usadas e análise de sentimentos (AGGARWAL; ZHAI, 2012) (NOUREEN; QAMAR; ALI, 2017).

O seu funcionamento é baseado no teorema de Bayes, que utiliza dados de treinamento para formar um modelo probabilístico de acordo com as características dos dados. Além disso, ele parte do pressuposto que há uma independência entre os atributos do modelo, ou seja, que os atributos não têm relação entre eles (AGGARWAL; ZHAI, 2012) (ALTINEL; GANIZ, 2018).

A probabilidade condicional é a probabilidade que o evento X ocorra, dada a evidência Y, e é escrito como P (X | Y). O teorema de Bayes permite determinar essa probabilidade de acordo com a Equação 1.

$$P(X|Y) = \frac{P(X) * P(Y|X)}{P(Y)} \quad (1)$$

Para tal, devem ser criadas tabelas para as palavras em suas respectivas classes, uma tabela de frequência das palavras e suas classes e uma tabela de probabilidades. A seguir é apresentado um exemplo prático para a aplicação do algoritmo de Naive Bayes e a construção

das tabelas de palavras e suas classes, frequência de palavras de acordo com a classe e a tabela de probabilidade na Tabela 2, Tabela 3 e Tabela 4 respectivamente (TROUSSAS et al, 2013).

Tabela 2 - Tabela de palavras e suas classes

Palavra	Classe
cachorro	positiva
amor	negativa
mau	negativa
amor	positiva
gato	positiva
amor	positiva
lar	positiva
gato	negativa
amor	positiva
amor	negativa

Tabela 3 - Tabela de frequência das palavras e suas classes

Palavra	Positiva	Negativa
cachorro	1	
amor	3	2
mau		1
gato	1	1
lar	1	
Total	6	4

Tabela 4 - Tabela de probabilidades das palavras

Palavra	Positiva	Negativa	Probabilidade
cachorro	1		1/10 = 0.1
amor	3	2	5/10 = 0.5
mau		1	1/10 = 0.1
gato	1	1	2/10 = 0.2
lar	1		1/10 = 0.1
Total	6/10 = 0.6	4/10 = 0.4	

De acordo com exemplo é possível calcular a probabilidade da palavra amor ser positiva ou negativa da seguinte forma:

$$P(positiva \mid amor) = \frac{P\ (amor \mid positiva) * P(positiva)}{P(amor)} = \frac{0.5 * 0.6}{0.5} = 0.6$$

$$P(negativa \mid amor) = \frac{P\ (amor \mid negativa) * P(negativa)}{P(amor)} = \frac{0.5 * 0.4}{0.5} = 0.4$$

No exemplo apresentado a probabilidade da palavra amor ser positiva é maior do que a probabilidade de ser negativa, isso ocorre devido aos itens que foram utilizados para o treinamento na aplicação.

2.5.2 Máquina de Vetores de Suporte

O algoritmo de classificação Máquina de Vetores de Suporte, SVM, também é muito utilizado e é muito eficiente para classificações, além disso, seus resultados são melhores que o algoritmo de Naive Bayes (AKAICHI; DHOUIOUI; PÉREZ, 2013).

O SVM é um algoritmo de aprendizado de máquina supervisionado em que os dados são plotados como um ponto no espaço n-dimensional, em que n é o número de atributos e com o valor de cada atributo sendo determinada pela coordenada espacial. A classificação é realizada ao encontrar o hiperplano que diferencia muito bem as classes. Normalmente é utilizado para classificação de sentimentos (AKAICHI; DHOUIOUI; PÉREZ, 2013).

Um ponto negativo deste algoritmo é que ele funciona apenas para classificação binária ou em pares, além disso não é aconselhado para grandes conjuntos de dados (NOUREEN; QAMAR; ALI, 2017).

2.5.3 K-Means

O algoritmo K-Means é popular e muito utilizado devido a sua facilidade de implementação e sua ordem de complexidade O(n). Ele é um algoritmo de agrupamento (*clustering*) em que cria grupos de acordo com a similaridade entre os atributos selecionados para análise (FONSECA; BELTRAME, 2010).

O seu funcionamento se dá por meio da utilização de centroides como representantes dos grupos, em que o centroide representa o centro de um grupo, que é a média de todos os objetos do grupo.

Segundo Fontana e Naldi (FONTANA; NALDI, 2009) o algoritmo pode ser descrito pelos seguintes passos:

1) São atribuídos valores iniciais para os protótipos de acordo com algum critério, por exemplo, sorteio aleatório dentro dos limites de domínio de cada atributo;
2) Cada objeto é atribuído ao grupo cujo protótipo apresente maior similaridade;
3) O valor do centroide (protótipo) de cada grupo é recalculado sendo a média dos objetos atuais do grupo;
4) Os passos 2 e 3 são repetidos até que os grupos se estabilizem.

É uma abordagem de aprendizado automático não supervisionado, ou seja, sem inserção de dados de treinamento para pré-classificação. Seu funcionamento é baseado na análise e comparações entre os valores numéricos dos dados.

2.6 Estratégias de desempenho

Atualmente, diversas estratégias de otimização de desempenho de algoritmos são utilizadas, entre elas, o *MapReduce* e Spark se destacam como boas soluções para processamento de conjuntos *Big Data* (AJI et al., 2013) (SINGH; REDDY, 2015). Esses dois *frameworks* escondem a complexidade do paralelismo de tarefas e da tolerância de falhas, e disponibilizam uma API de programação simples para os usuários (SHI et al, 2015). A seguir, são apresentadas as duas abordagens com mais detalhes.

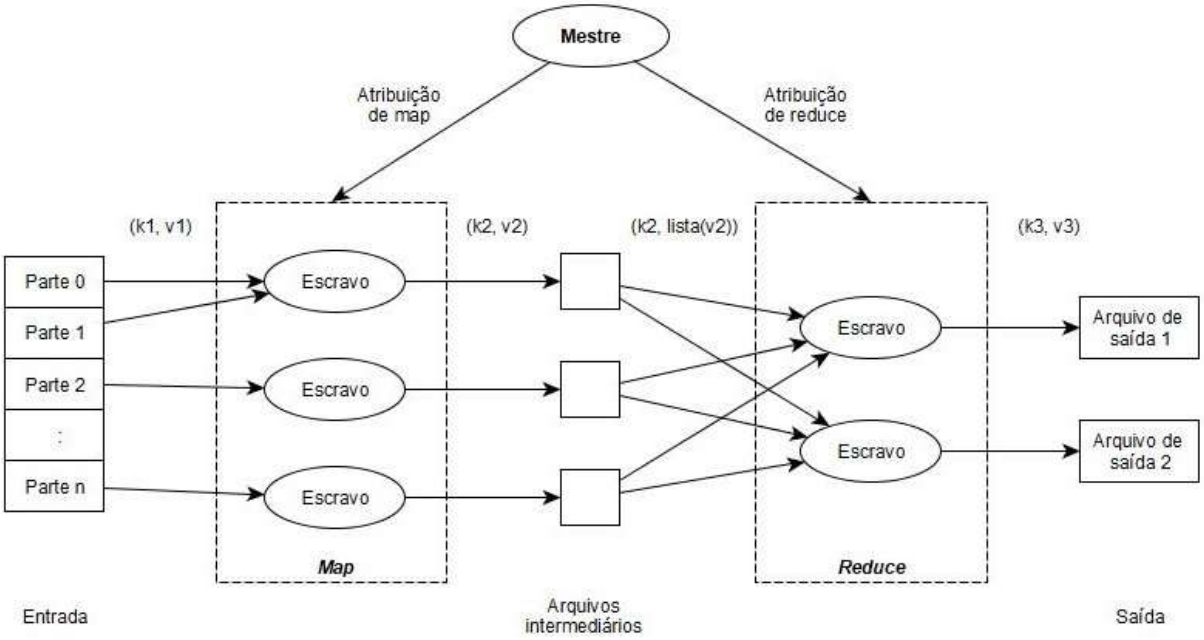
2.6.1 Apache Hadoop

O MapReduce é um modelo de programação proposto por engenheiros da Google em 2004 e tem como finalidade suportar computações paralelas em grandes coleções de dados em clusters de computadores (DEAN; GHEMAWAT, 2008).

A arquitetura do MapReduce é composta por um conjunto de máquinas conectadas em rede, em que um nó é definido como mestre e os outros como escravos. A função do mestre é controlar a aplicação e dividir as tarefas entre os escravos, enquanto a função dos escravos é realizar o processamento dos dados (WHITE, 2012).

O fluxo de execução do programa é composto por duas funções principais: *map* e *reduce*. No início da execução do programa, a função *map* é responsável por ler os dados de entrada do HDFS, processar esses dados e gerar um conjunto de dados intermediários compostos de pares chave/valor. Por fim, a função *reduce* é aplicada aos valores que possuem a mesma chave, com o objetivo de combinar os dados derivados de forma correta (LEE et al, 2012) (WHITE, 2012). Na Figura 4 é apresentado o modelo de programação MapReduce.

Figura 4 - Modelo de execução MapReduce.



Fonte: Adaptado de (DEAN; GHEMAWAT, 2008).

Conforme apresentado na Figura 4, para cada entrada *(k1, v1)*, a função *map* gera uma lista intermediária como saída *(k2, v2)*. Em seguida, na etapa intermediária, também conhecida como embaralhamento, do inglês *shuffle*, os pares intermediários são agrupados como *(k2, lista(v2))* e enviados para a função *reduce*. Por fim, a função *reduce*, gera a lista de pares de saída final *(k3, v3)* (DEAN; GHEMAWAT, 2008).

A implementação de MapReduce mais utilizada atualmente é o Apache Hadoop (AJI et al., 2013) (SHI et al, 2015). O Apache Hadoop é um *framework* de código aberto que tem o objetivo de armazenar e processar grandes conjuntos de dados com a utilização de clusters, ele foi projetado para ser escalável, comportar centenas e até milhares de nós e ser tolerante a falhas (SINGH; REDDY, 2015).

Além de oferecer recursos para processamento distribuído, ele também conta com componentes importantes que podem ser utilizados no contexto *Big Data* (WHITE, 2012), o mais importante é o *Hadoop Distributed File System* - HDFS (BORTHAKUR et al., 2008), que é um sistema de arquivos distribuídos utilizado para armazenar dados em um cluster e é o responsável por oferecer alta disponibilidade e tolerância a falhas (SINGH; REDDY, 2015). Resumidamente, o Hadoop é formado por quatro módulos:

a) *Hadoop Common* – utilitários comuns que suportam outros módulos do Hadoop;
b) *Hadoop Distributed File System* (HDFS) – sistema de arquivos distribuídos;

c) *Hadoop YARN* – estrutura para escalonamento de tarefas e gerenciamento de recursos do *cluster*;
d) *Hadoop MapReduce* – sistema baseado no YARN para processamento paralelo e distribuído de grandes conjuntos de dados.

Porém, o modelo Apache Hadoop tem limitações e uma das principais desvantagens é a sua ineficiência na execução de algoritmos iterativos, pois realiza muitas operações de entrada e saída no disco, o que sobrecarrega o HDFS e gera um gargalo que degrada significativamente o desempenho do sistema (SINGH; REDDY, 2015).

2.6.2 Apache Spark

O Apache Spark é um *framework* de processamento de dados para aplicações *Big Data*, ele foi inicialmente desenvolvido por pesquisadores da Universidade da Califórnia em Berkeley (ZAHARIA et al., 2012). Ele é uma alternativa ao Hadoop e foi projetado para superar as limitações de entrada e saída de disco e melhorar a performance dos sistemas anteriores (SINGH; REDDY, 2015).

Para superar essas limitações, o Spark conta com uma abstração chamada de conjuntos de dados distribuídos resilientes, *Resilient Distributed Datasets* - RDDs, em que são coleções de objetos somente leitura dividida em um conjunto de máquinas que podem ser reconstruídas se uma partição for perdida (ZAHARIA et al, 2012). Para certas tarefas, o Spark pode superar o Hadoop em 100 vezes quando os dados couberem na memória e até 10 vezes quando os dados estiverem em disco (SINGH; REDDY, 2015).

O Spark é um *framework* de código aberto para processamento de dados em grande escala e pode ser implementado nas linguagens Java, Scala e Python (SINGH; REDDY, 2015). Além disso, conta com componentes para diversos tipos de processamento, entre eles:

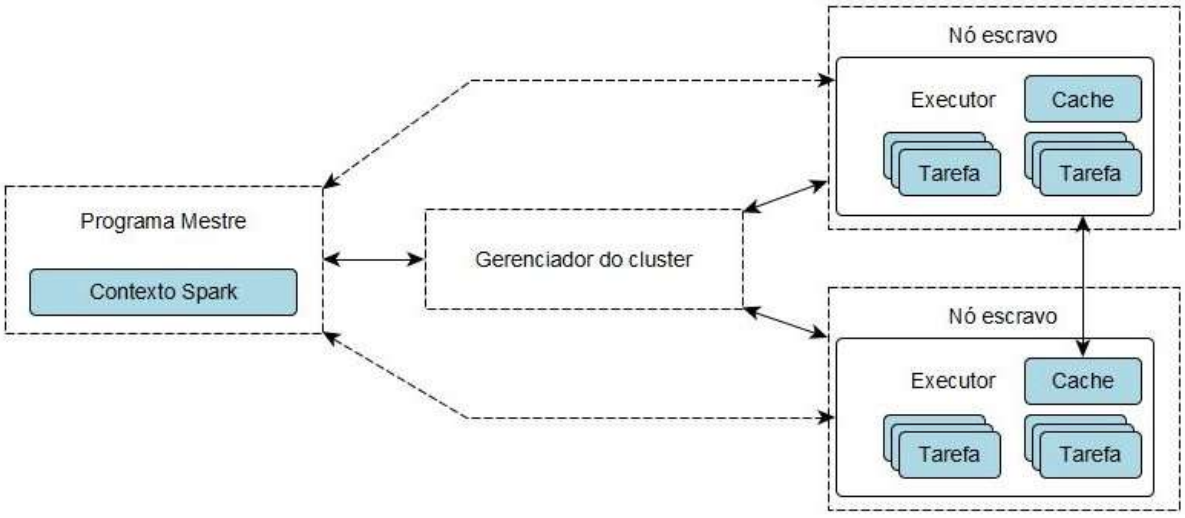
a) Spark Streamming – componente responsável pelo processamento de fluxos de dados em tempo real;
b) GraphX – processamento de dados sobre grafos e visualização de resultados;
c) SparkSQL – interface que permite executar consultas em SQL sobre os dados do Spark;
d) MLlib – biblioteca de aprendizado de máquina.

As aplicações Spark são formadas por um programa mestre, um gerenciador de cluster e os trabalhadores ou escravos. O programa mestre é a aplicação principal que gerencia a criação e a execução do processamento definido pelos programadores. O gerenciador do *cluster*

é um componente opcional, que só é necessário se o Spark for executado de forma distribuída. Ele é o responsável por administrar as máquinas que serão utilizadas como trabalhadores/escravos. O sistema suporta três tipos de *cluster*, *Standalone*, Apache Mesos e Hadoop YARN. Os trabalhadores ou escravos são as máquinas que realmente executarão as tarefas enviadas pelo programa mestre. Se o Spark for executado de forma local, no modo *Standalone*, a máquina desempenhará tanto o papel de programa mestre quanto o de escravo.

Na Figura 5 é apresentada a arquitetura do Spark e seus principais componentes.

Figura 5 - Arquitetura do Spark.



Fonte: Adaptado de (APACHE SPARK, 2020).

Além da arquitetura de uma aplicação Spark, é importante conhecer os principais componentes desse modelo de programação. A seguir são descritos os três conceitos fundamentais que são utilizados em todas as aplicações desenvolvidas com o *framework* Spark:

a) RDDs – abstraem um conjunto de objetos distribuídos no *cluster*, geralmente executados na memória principal. Os conjuntos de objetos podem estar armazenados em sistemas de arquivo tradicional, no HDFS e em Bancos de Dados NoSQL, como Cassandra e HBase. Os RDDs são os principais objetos do modelo de programação Spark, pois são nesses objetos que são executados os processamentos de dados;

b) Operações – representam as transformações, que podem ser agrupamentos, filtros, mapeamentos entre os dados ou ações como contagens e persistências. Essas ações são realizadas em um RDD. Um programa Spark normalmente é

definido como uma sequência de transformações ou ações que são realizadas sobre um conjunto de dados.

c) Contexto Spark – o contexto é o objeto que conecta o Spark ao programa que está sendo desenvolvido. Ele pode ser acessado como uma variável em um programa para utilizar os seus recursos.

De acordo com a documentação oficial do Spark (APACHE SPARK, 2020), na Tabela 5 e Tabela 6 são apresentadas as principais transformações e ações do *framework* respectivamente.

Tabela 5 - Operações para transformações suportadas pelo Spark

Transformação	Descrição
map(*f*)	Retorna um novo RDD formado pelo processamento de cada elemento da fonte por meio de uma função *f*
filter(*f*)	Retorna um novo RDD formado pela seleção de elementos da origem que satisfazem a condição da função *f*
flatMap(*f*)	Semelhante ao **map**, mas cada item de entrada pode ser mapeado para 0 ou mais itens de saída (a função *f* retorna uma sequência em vez de um único item)
mapPartitions(*f*)	Semelhante ao **map**, mas executa separadamente em cada partição do RDD. *f* precisa ser do tipo Iterator<T> => Iterator<U> quando executado em um RDD do tipo T.
mapPartitionsWithIndex(*f*)	Semelhante ao **mapPartitions**, mas também fornece *f* com um valor inteiro que representa o índice da partição. *f* precisa ser do tipo (Int, Iterator<T>) => Iterator<U> quando executado em um RDD do tipo T.
sample(*withReplacemente, fraction, seed*)	Retorna uma fração do conjunto de dados, com ou sem substituição, com uso de um número aleatório como semente.
union(*otherDataset*)	Retorna um novo conjunto de dados, composto pela união dos RDDs.
intersection(*otherDataset*)	Retorna um novo conjunto de dados, composto pela intersecção dos RDDs.
distinct(*otherDataset*)	Retorna um novo conjunto de dados que contém os elementos distintos dos RDDs.

groupByKey([*numPartitions*])	Quando invocado sobre uma coleção de pares (k, v) retorna um par (k, iterable<v>)
reduceByKey(*f*, [*num Partitions*])	Semelhante ao **groupByKey()**, porém os valores são processados por uma função *f*.
sortByKey([asc/desc], [*numPartitions*])	Retorna um novo conjunto de dados ordenado de modo ascendente ou decrescente
join(*otherDataset*, [*numPartitions*])	Retorna a combinação de elementos de dois RDDs a partir de uma chave K. (K, V) e (K, W) => (K, (V, W))

Tabela 6 - Operações para ações suportadas pelo Spark

Ação	Descrição
reduce(*f*)	Agrega os elementos de um RDD por meio de uma função *f*. A função *f* deve ser comutativa e associativa.
collect()	Retorna todos os elementos do RDD.
count()	Retorna a quantidade de elementos do RDD.
first()	Retorna o primeiro elemento do RDD.
take(n)	Retorna a quantidade *n* de elementos do RDD.
takeSample(*withReplacement*, num, *seed*)	Retorna um subconjunto aleatório do RDD.
saveAsTextFile(c)	Escreve todos os elementos do RDD em um arquivo texto e armazena no caminho *c*.
saveAsSequenceFile(c)	Escreve todos os elementos do RDD em um arquivo sequencial e armazena no caminho *c*.
countByKey()	Conta os elementos de acordo com uma chave K.
foreach(*f*)	Executa uma função *f* sobre cada elemento do RDD.

2.6.3 Comparação entre Hadoop e Spark

Nesta seção é apresentado um estudo comparativo entre os dois *frameworks* de programação *MapReduce*, representado pelo Apache Hadoop e Apache Spark, bem como os pontos positivos e negativos de cada abordagem.

No artigo (SHI et al, 2015) foram avaliados os principais componentes arquiteturais dos *frameworks*, entre eles: embaralhamento (*shuffle*), modelo de execução e armazenamento em

cache, os testes foram feitos com a utilização de um conjunto de cargas de trabalho importantes para a análise.

Para conduzir uma análise detalhada, os autores desenvolveram duas ferramentas de criação de perfil:

a) O plano de execução de tarefas foi correlacionado com a utilização de recursos de *MapReduce* e Spark, e a correlação é apresentada visualmente;
b) Os tempos de execução da tarefa são fornecidos para uma análise detalhada.

Por meio dos experimentos, as diferenças de desempenho entre o *MapReduce* e Spark foram quantificadas, além disso, essas diferenças foram atribuídas a componentes que são arquitetados de maneira diferente nas duas abordagens.

A simplicidade do *MapReduce* é atrativa para os usuários, mas o *framework* tem diversas limitações. Aplicações que utilizam aprendizado de máquina e análise de grafos processam seus dados forma iterativa, o que significa que são feitas múltiplas rodadas de computação sobre os mesmos dados. Pois para cada trabalho os dados de entrada são lidos, processados e em seguida são escritos novamente no HDFS. Para que o próximo trabalho utilize a saída do trabalho executado anteriormente, ele deve repetir o ciclo de leitura, processamento e gravação. Portanto, para algoritmos iterativos, o modelo *MapReduce* apresenta uma sobrecarga significativa (SHI et al, 2015).

Com o uso dos RDDs, o Spark supera as limitações do *MapReduce* de leitura e gravação dos dados, em que implementam estruturas de dados na memória para armazenar dados intermediários em um conjunto de nós. Como os RDDs podem ser mantidos na memória, os algoritmos podem iterar dados RDD muitas vezes de maneira mais eficiente (SHI et al, 2015).

O estudo dos três componentes arquiteturais por meio de experimentos abrange a maioria das diferenças entre os dois *frameworks*.

a) *Shuffle*: o componente *shuffle* é responsável pela troca de dados intermediários entre dois estágios computacionais[9]. Por exemplo, no *MapReduce* os dados são embaralhados entre os estágios de *map* e de *reduce* para a sincronização em massa. Esse componente geralmente afeta a escalabilidade do *framework*. Frequentemente, uma operação de ordenação é executada durante o estágio de *shuffle*. Um algoritmo externo de ordenação, como o *merge sort*, geralmente é necessário para manipular dados muito grandes que não se encaixam na memória

[9] Para o *MapReduce* existem dois estágios: *map* e *reduce*. Para o Spark podem existir muitos estágios, em que são construídos em dependências aleatórias.

principal. Além disso, a agregação e a combinação são executadas durante o *shuffle*.

b) Modelo de execução: esse componente determina como as funções definidas pelo usuário são convertidas em um plano de execução física. O modelo de execução geralmente afeta a utilização de recursos para execução de tarefas paralelas. Com o interesse em paralelismo entre tarefas, sobreposição de estágios computacionais e pipeline de dados entre estágios computacionais.

c) Armazenamento em cache: o componente de armazenamento em cache permite que os dados intermediários sejam reutilizados em vários estágios. O armazenamento em cache efetivo acelera os algoritmos iterativos ao custo de espaço adicional na memória ou no disco. São avaliados diferentes níveis de armazenamento em cache, cache de buffer do sistema operacional, armazenamento em cache do HDFS, cache de Tachyon (LI et al, 2014) e RDD.

Para os experimentos os autores selecionaram cinco abordagens, que incluem contagem de palavras, ordenação, algoritmo k-means, regressão linear e o algoritmo PageRank. Essas abordagens foram escolhidas porque coletivamente abrangem as principais abordagens analíticas que normalmente são executadas no MapReduce e no Spark (SHI et al, 2015).

Na Tabela 7 são apresentadas as cargas de trabalho utilizadas e suas respectivas características acerca de tipo de trabalho, seletividade de *shuffle* (relação do tamanho da saída do *map* e o tamanho da entrada de trabalho, que representa a quantidade de E/S do disco e da rede para um *shuffle*) e da seletividade de iteração (a proporção do tamanho da saída para o tamanho da entrada para cada iteração, que representa a quantidade de dados intermediários trocados entre iterações).

Tabela 7 - Características das cargas de trabalho selecionadas. Extraído de (SHI et al, 2015).

		Contagem de palavras	Ordenação	K-Means (Regressão Linear)	PageRank
Tipo de trabalho	Uma passagem	X	X		
	Iterativo			X	X
Seletividade de *shuffle*	Alto		X		
	Médio				X
	Baixo	X		X	

Seletividade de iterações	Alto		X		
	Médio				X
	Baixo	X		X	

Além disso, na Tabela 8 são apresentadas as características dos componentes arquiteturais das cargas de trabalhos utilizadas. Conforme apresentado, para o componente de *shuffle*, foi avaliada a estrutura de agregação, a ordenação externa e transferência de dados intermediários. Para o componente do modelo de execução, foram avaliadas como as funções definidas pelo usuário são convertidas em um plano de execução física, com o foco no paralelismo de tarefas, na sobreposição de etapas e no pipeline de dados. Para o componente de armazenamento em cache, foram avaliadas a eficácia do armazenamento em cache disponível em diferentes níveis para o armazenamento em cache de dados de entrada e intermediários.

Tabela 8 - Componentes arquiteturais de interesse. Extraído de (SHI et al, 2015).

		Contagem de palavras	Ordenação	K-Means (Regressão Linear)	PageRank
Shuffle	Agregação	X		X	X
	Ordenação externa		X		
	Transferência de dados		X		X
Modelo de execução	Paralelismo de tarefas	X	X	X	X
	Sobreposição de estágios		X		
	Pipeline de dados				X
Armazenamento em cache	Entrada			X	X
	Dados intermediários				X

Por meio dos experimentos realizados, os autores concluíram que, de forma geral, o Spark se mostra mais rápido que MapReduce 2.5x, 5x e 5x para Contagem de Palavras (*Word*

Count), K-Means e PageRank respectivamente. A seguir, as vantagens de desempenho do Spark são atribuídas a uma série de diferenças estruturais em comparação com o MapReduce:

a) Para operações de contagem de palavras e aplicações semelhantes, em que a seletividade do *map* pode ser significativamente reduzida por meio da agregação com base em *hash* no Spark é mais eficiente comparado com a agregação baseada em ordenação do MapReduce. Com isso, os resultados para o tempo de execução medidos indicam que o *framework* baseado em *hash* contribui aproximadamente 39% para a melhoria geral para o Spark;
b) Para algoritmos iterativos como K-Means e PageRank, ao deixar as entradas em cache RDD é possível reduzir tanto a sobrecarga de CPU (conversão de texto para objetos) quanto a sobrecarga de entrada e saída em disco para as iterações subsequentes. É importante ressaltar que a sobrecarga de CPU é muitas vezes o gargalo em cenários em que as iterações subsequentes não usam o cache RDD. Portanto, o cache RDD é muito mais eficiente comparado a outras abordagens de cache de baixo nível, como cache de buffer do sistema operacional e cache em HDFS, que são capazes de reduzir apenas a entrada e saída de disco. Por meio dos experimentos apresentados, foi verificado que reduzir a sobrecarga da CPU contribuiu em mais de 90% no *speed-up* geral para as iterações subsequentes no K-Means;
c) Uma vez que o Spark habilita o pipeline de dados entre estágios, evita sobrecarga da materialização na saída dos dados no HDFS (serialização, E/S de disco e E/S em rede) entre iterações nos algoritmos de análise de grafo como o PageRank. Uma exceção a vantagem de desempenho do Spark sobre o MapReduce é para aplicações do tipo de ordenação, no qual o MapReduce é 2x mais rápido que o Spark. Isto ocorre, pois, o MapReduce pode sobrepor a fase de *shuffle* com a fase de *map*, o que efetivamente esconde a sobrecarga da rede, que muitas vezes é um gargalo para a fase de *reduce*.

2.7 Armazenamento não convencional

O termo NoSQL, do inglês *Not Only Structured Query Language*, é utilizado para uma classe definida de bancos de dados não relacionais. Esses bancos de dados emergiram como complemento dos bancos de dados relacionais e solução das crescentes necessidades de escalabilidade, flexibilidade do esquema, capacidade de armazenar grandes volumes de dados

e desempenho para aplicações intensivas de dados (ABRAMOVA; BERNARDINO; FURTADO, 2014) (MOHAN, 2013).

A seguir são definidas as características dos bancos de dados NoSQL e os benefícios sobre os bancos de dados relacionais[10]:

e) Dinâmica e ágil – pode crescer dinamicamente de acordo com o tempo e mudanças de requisitos, além de ser capaz de lidar com dados estruturados, semiestruturados ou não estruturados.
f) Escalável horizontalmente – diferente dos bancos de dados relacionais que escalam verticalmente, o NoSQL escala horizontalmente com a adição de mais servidores.
g) Melhor desempenho – todos os bancos de dados NoSQL alegam oferecer desempenho melhor em comparação com os relacionais (ABRAMOVA; BERNARDINO; FURTADO, 2014).

As limitações nesses tipos de bancos de dados diferem entre si, já que NoSQL é um conjunto de bancos de dados, em que cada um tem a sua força e sua fraqueza.

Diante de dados estruturados, as técnicas de processamento para extração de informação útil mais utilizadas são as de *Data Mining*, que consistem em extrair resultados não triviais implícitos no banco de dados (HAN; PEI; KAMBER, 2011). Porém a extração de conhecimento para conjuntos *Big Data* deve ter uma abordagem diferente da utilizada para dados não estruturados, o que necessita de algoritmos capazes de processar um grande conjunto de dados de maneira eficiente (GOLE; TIDKE, 2015).

2.8 Trabalhos correlatos

A seguir são apresentados os principais trabalhos encontrados na literatura sobre extração de conhecimento em textos, análise de dados de mídias sociais, mineração de textos e *Big Data*.

2.8.1 Reconhecimento de traços de personalidade com o uso de mídias sociais

[10] http://www.tutorialspoint.com/articles/what-is-nosql-and-is-it-the-next-big-trend-in-databases

O trabalho "*Recognising Personality Traits using Social Media*" (VARSHNEY et al, 2017) utilizou dados de mídias sociais para analisar as personalidades dos usuários. Os dados foram coletados das redes sociais Twitter e Facebook.

Como metodologia de trabalho, foram realizadas as seguintes etapas: aquisição dos dados, pré-processamento dos dados, processo de classificação e saída. No pré-processamento foram utilizadas as etapas de limpeza do texto, identificação de palavras, remoção de *stop words*, radicalização, rotulação de palavras do texto, atribuição de pesos, redução da frequência dos documentos e seleção de características.

Para o processo de classificação foram utilizados os algoritmos *Multinomial Naive Bayes* - MNB, *K-Nearest Neighbors* - KNN e *Support Vector Machine* SVM. Como treinamento, foram utilizadas diversas publicações do usuário como um documento único, em que os algoritmos mediram a frequência em que as palavras aparecem e atribuição de pesos para as palavras.

As personalidades foram classificadas de acordo com as cinco grandes dimensões de personalidade, neuroticismo, extroversão, abertura à novas experiências, simpatia e conscienciosidade.

Algumas das limitações encontradas foram que normalmente os usuários omitem ou deturpam informações inseridas nas mídias sociais, o que pode levar a resultados imprecisos, além da utilização de caracteres especiais, gírias e a linguagem utilizada nas redes sociais.

O trabalho concluiu que o algoritmo MNB apresentou os melhores resultados, com 60% de acurácia. Os outros dois algoritmos apresentaram desempenhos inferiores devido à dificuldade em separar os dados entre duas classes. Além disso, para o KNN, ocorreu dificuldade em calcular o valor de K, o que resultou em uma grande desvantagem.

2.8.2 Predição de depressão via Mídias Sociais

O trabalho "*Predicting Depression via Social Media*" (DE CHOUDHURY et al., 2013) foi desenvolvido por pesquisadores da Microsoft em 2013 e foi apresentado uma abordagem para realizar a predição de pessoas que sofrem de depressão maior. Para isso, foi explorado o potencial das mídias sociais para detectar e diagnosticar transtornos depressivos nos usuários por meio de suas postagens.

O estudo foi realizado com usuários do Twitter, em que a amostra de treinamento foi realizada com usuários que relataram ter diagnóstico de depressão clínica com base em um instrumento psicométrico padrão.

Os pesquisadores se basearam na hipótese de que ocorrem mudanças na linguagem, nas atividades e nos vínculos sociais para construção dos modelos estatísticos. Foram analisadas postagens na mídia social no período de um ano antes do diagnóstico da doença, e foram medidos atributos comportamentais relacionados ao engajamento social, emoção, linguagem e estilos linguísticos, ego e menções a medicamentos antidepressivos. Com essas dicas comportamentais, foi criado um classificador estatístico que fornece estimativas do risco de depressão, antes do início diagnosticado.

Segundo os autores, as mídias sociais são fontes de sinais úteis que podem caracterizar o aparecimento da depressão em indivíduos conforme medido pela diminuição da atividade social, efeito negativo, aumento de preocupações de relacionamento e com relação a medicações, além de maior expressão de envolvimento religioso.

2.8.3 Classificação de personalidade de usuários de mídias sociais com uso de Linguística Computacional

O trabalho "*Social Media User Personality Classification using Computacional Linguistic*" (LUKITO et al., 2016) explorou o Twitter como fonte de dados para classificar os usuários com base no vocabulário utilizado. Foram analisados e comparados três modelos estatísticos diferentes, com isso foi possível encontrar correlações sobre quais traços de personalidade estão relacionados com o comportamento linguístico.

As personalidades analisadas são divididas em quatro classes, introvertido ou extrovertido (I/E), intuitivo ou sensitiva (N/S), pensativo ou emotivo (T/F) e julgadora ou perceptiva (J/P).

A análise foi realizada com o classificador Naive Bayes com uma abordagem de *machine learning* e a utilização da ferramenta *Waikato Environment for Knowledge Analysis* (WEKA[11]), em que foram utilizados quatro arquivos de texto com palavras, um para cada classe, como treinamento.

Segundo os autores, o classificador Naive Bayes apresentou os melhores resultados, 80% de acurácia para traços de personalidade I/E, 60% para S/N, T/F e J/P e mostrou melhor desempenho em termos de velocidade na classificação dos usuários. Além disso, o trabalho gerou como resultado as 30 palavras mais utilizadas por cada personalidade.

2.8.4 Análise *Big Data*: uma perspectiva Apache Spark

[11] https://www.cs.waikato.ac.nz/ml/weka/

O trabalho "*Big Data Analysis: Apache Spark Perspective*" (SHORO; SOOMRO, 2015) tem como objetivo apresentar o conceito de análise *Big Data* e reconhecer informações significativas de dados do Twitter com a utilização da ferramenta Apache Spark.

A abordagem utilizada pelo trabalho para realização dos experimentos foi a análise de *tweets* em tempo real, em que foram medidos as 10 palavras mais utilizadas, as 10 línguas mais utilizadas e o número de vezes em que uma palavra particular foi utilizada, tudo em período de 10 minutos.

Na primeira parte da análise foram verificados 23865 *tweets*, 160989 palavras, das quais 77548 são palavras únicas, em 10 minutos de *streaming* de dados. Na segunda parte do experimento foram analisados 23311 *tweets* e foram identificadas 42 línguas diferentes, também em 10 minutos. No terceiro experimento a palavra "mtvstars" foi encontrada em 41806 *tweets* de um total de 42119 em 10 minutos.

O trabalho apresentado mostra a capacidade de aplicação do *framework* Apache Spark em um contexto de streaming de dados, bem como a sua utilização para grandes quantidades de dados de mídias sociais.

2.9 Considerações

Com o levantamento bibliográfico dos trabalhos correlatos foi possível verificar quais são os métodos mais utilizados pelos trabalhos recentes na área de extração de conhecimento sobre dados textuais, especialmente com dados de origem de mídias sociais.

Além disso, foi possível identificar as contribuições e os gargalos dos trabalhos com relação ao desempenho e resultados dos processos de análise. Na Tabela 9 são apresentadas as características dos trabalhos correlatos estudados.

Tabela 9 - Comparativo entre os trabalhos correlatos

	(VARSHNEY et al., 2017)	(DE CHOUDHURY et al., 2013)	(LUKITO et al., 2016)	(SHORO; SOOMRO, 2015)
Estratégias propostas ao cenário *Big Data*	✔	✔	✔	✔
Origem de dados de Mídias Sociais	✔	✔	✔	✔
Pré-processamento dos dados	✔	✘	✘	✘
Extração de características	✔	✔	✘	✘

Classificação	✔	✔	✔	✘
Buscas por palavras específicas	✘	✔	✘	✔
Contexto de mídias sociais na análise	✘	✘	✘	✘
Análise comportamental	✔	✔	✘	✘
Estratégias de desempenho paralela e escalável	✘	✘	✘	✔

Fonte: Elaborado pelo autor.

3 Trabalho desenvolvido

Neste capítulo é apresentada a ferramenta desenvolvida, desde a sua elaboração até os aspectos de sua implementação. A ferramenta desenvolvida teve como base os trabalhos pertinentes mais recentes encontrados na literatura e, assim, centrado no propósito de contribuir para as áreas de processamento de textos e classificação de usuários de acordo com as características extraídas. O desempenho alcançado pela ferramenta deu-se pelo uso de técnicas de processamento paralelo e distribuído.

3.1 Detalhes da Ferramenta

A motivação para o desenvolvimento deste trabalho é a tentativa de contribuir com as áreas de *Big Data*, análise de dados de mídias sociais e com a saúde mental dos usuários de redes sociais, visto que é uma área promissora e relativamente nova. Além disso, é de conhecimento comum que as doenças e transtornos mentais têm afetado a população brasileira, e cerca de 23 milhões de pessoas necessitam de algum tipo de atendimento médico.

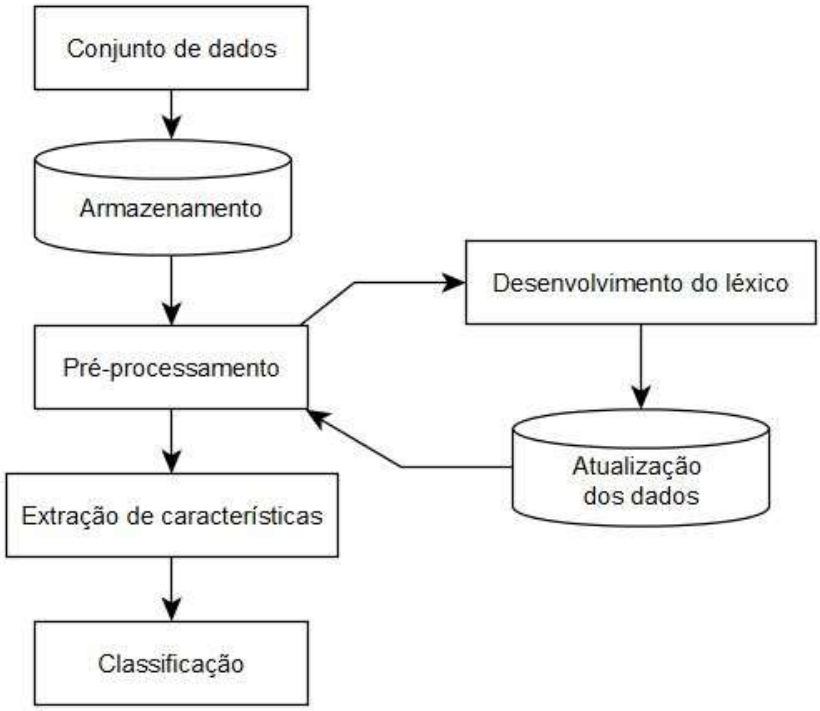
Dessa forma, as redes sociais são locais que podem conter dados preciosos para a análise de dados, visto que em média o brasileiro passa 225 minutos conectado por dia contra 150 minutos da média global, segundo uma pesquisa da GlobalWebIndex[12].

A ferramenta desenvolvida tem como objetivo analisar os textos de modo a classificar o usuário em uma classe conhecida, com o foco na saúde mental. Para tal, são utilizadas técnicas do processo de descobertas de conhecimento sobre textos. Além disso, a análise deve ser feita

[12] <https://www.globalwebindex.com/>

de forma eficiente e que permita escalabilidade de acordo com o contexto e requisitos da análise. A arquitetura da ferramenta é apresentada na Figura 6.

Figura 6 - Arquitetura da ferramenta



Fonte: Elaborada pelo autor.

De acordo com a imagem apresentada, a ferramenta executa os seguintes passos:

a) A entrada de dados pode ser de diferentes fontes de dados: texto (arquivo .csv), banco de dados PostgreSQL ou banco de dados NoSQL Cassandra. É importante mencionar que o objeto de análise é um dado textual;

b) A ferramenta é interativa, ou seja, apresenta as análises pré-cadastradas, busca as possíveis tabelas de análise e os atributos textuais disponíveis na tabela. Com isso o usuário deve escolher em qual tabela e qual atributo textual deve iniciar a análise;

c) É feita uma consulta sobre os dados alvo da análise e são carregados com o uso da estrutura de dados `JavaRDD<Row>`, em que os dados são conduzidos diretamente para a memória principal e utilizados para transformações e ações do *framework*;

d) Com o uso da transformação `map()` sobre a estrutura de dados `JavaRDD<Row>` é possível percorrer todos os elementos carregados em

memória, dividir o registro em tokens e realizar as substituições do desenvolvimento lexical e as etapas do processo de pré-processamento. Após essa etapa, a nova estrutura `JavaRDD<Row>` é carregada, em que fica diferente da inicial;

e) Após as fases de pré-processamento, são calculadas as frequências das palavras em relação aos documentos. Nesta etapa também é utilizada a transformação `map()` sobre o novo `JavaRDD<Row>`;
f) Nesse momento, a ferramenta faz análise de sentimento do texto. Essa etapa necessita que o analista complemente as informações que não estão no banco de dados se pertencem a classe positiva ou negativa;
g) Após a extração de características o algoritmo Naive Bayes é executado com dados de treinamento para selecionar as postagens semelhantes pertencentes a mesma classe.

A ferramenta pode ser configurada para receber dados de entrada já pré-processados ou com a etapa de extração de características já realizada. Dessa forma, se o analista já conta com arquivos prontos para análise não é necessário realizar as etapas anteriores. Além disso, pode ser configurado para salvar os dados de saída de cada etapa em arquivo, conforme pré-configuração da ferramenta.

É importante salientar que com a leitura e escrita dos resultados em arquivo padrão, sem a utilização do HDFS, o desempenho do sistema é deteriorado.

Nas próximas seções são apresentadas as etapas presentes na ferramenta com mais detalhes.

3.2 Banco de Dados

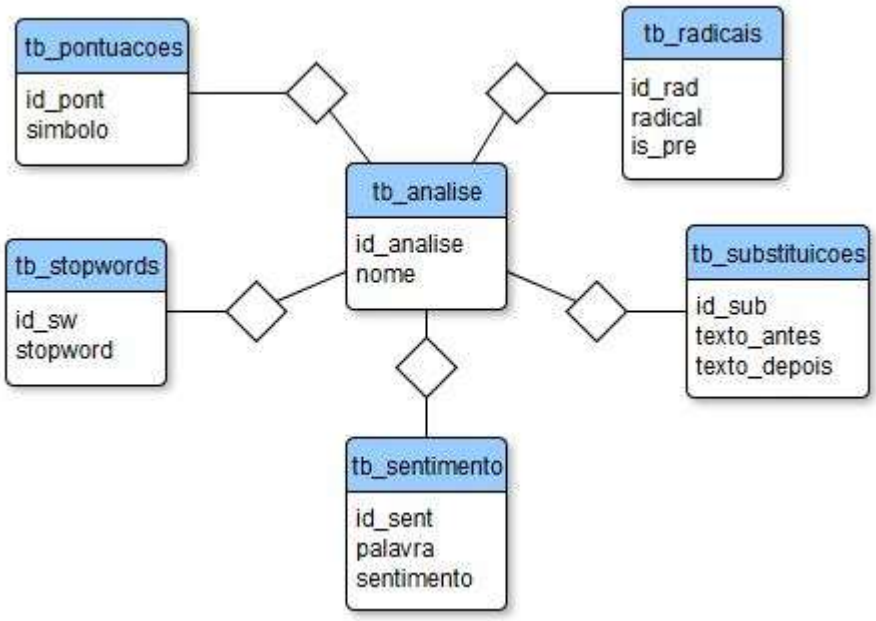
Com o intuito de abranger diversas formas de entradas de dados, a ferramenta foi construída para suportar dados textuais oriundos de arquivos texto (.csv), bancos de dados PostgreSQL, que são comumente utilizados em aplicações convencionais, e bancos de dados Cassandra, geralmente utilizados em aplicações Big Data e que buscam características de armazenamento distribuído.

Os dados armazenados no banco de dados Cassandra, que é altamente recomendado para utilização em conjunto com o Spark (APACHE SPARK, 2020), além disso, por ser NoSQL apresenta características compatíveis para o contexto de dados de mídias sociais em relação aos dados não estruturados e velocidade de recuperação de informações.

3.3 Preparação do ambiente

De forma a facilitar a utilização da ferramenta e permitir o seu correto funcionamento, foram criadas tabelas e suas associações afim de manter a análise confiável. Na Figura 7 são apresentadas as tabelas criadas para o funcionamento da ferramenta, sem considerar os dados de análise.

Figura 7 - Diagrama Entidade-Relacionamento da ferramenta



Fonte: Elaborada pelo autor.

Desta forma, antes de iniciar a etapa de pré-processamento, é necessário dar um nome para a análise, especificar quais pontuações devem ser removidas, especificar quais palavras de parada devem ser removidas e indicar quais sufixos e prefixos devem ser removidos. Além disso, foi criada uma lista de exceções para que algumas palavras não fossem removidas na etapa de pré-processamento.

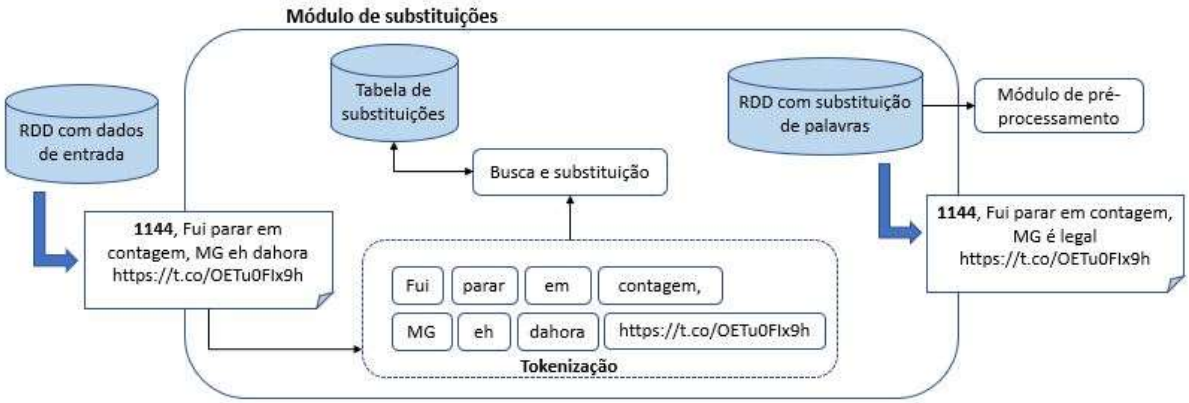
Com essa estruturação, a ferramenta permite a análise em diferentes contextos e até mesmo em idiomas diferentes. Porém, o analista deve ter conhecimento do contexto para preparar o ambiente corretamente.

3.4 Pré-processamento

Na etapa de pré-processamento os dados de entrada são carregados na estrutura RDD e passados como parâmetro para o módulo de pré-processamento. A primeira etapa do processo é a identificação de palavras, ou seja, dividir o texto em tokens. Após a tokenização o conjunto

é enriquecido por meio das substituições, o que procura minimizar a linguagem usada em mídias sociais. Na Figura 8 é apresentado um exemplo de tokenização e substituição de palavras. Com as palavras tokenizadas e substituídas, elas passam para as etapas de remoções.

Figura 8 - Exemplo de funcionamento do módulo de tokenização e substituição



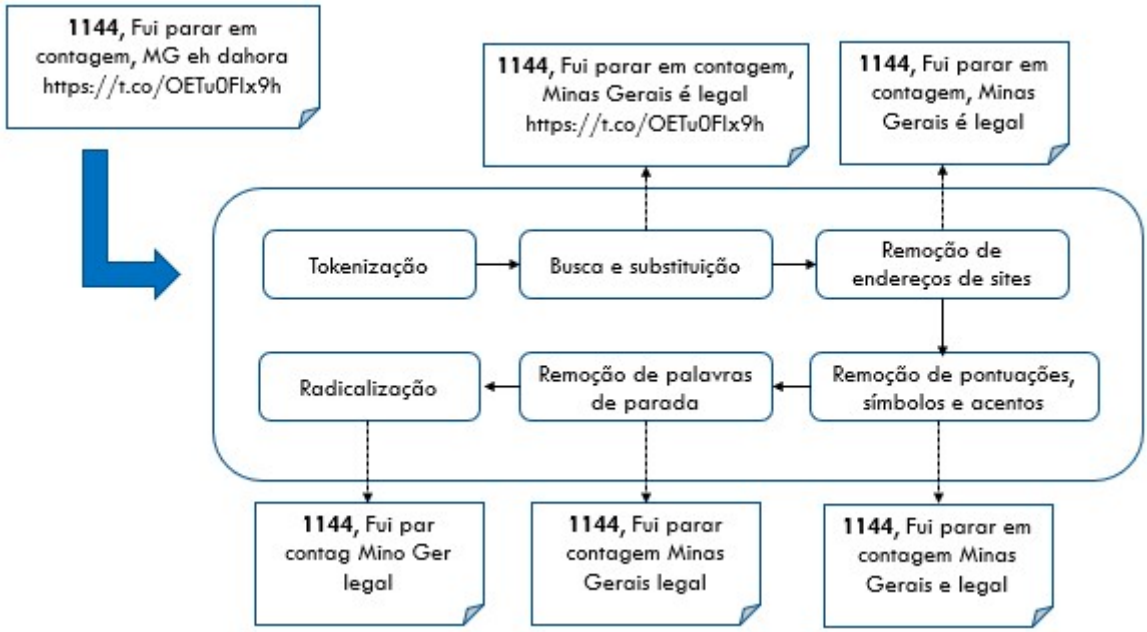
Fonte: Elaborada pelo autor.

Após a substituição de palavras do contexto são executadas as etapas de remoção de endereços de sites, remoção de palavras de parada e radicalização, seguindo o algoritmo RSLP (ORENGO; HUYCK, 2001) com a implementação paralela. As fases de pré-processamento são descritas a seguir:

- Identificação de palavras – as palavras do texto são identificadas e são divididos em tokens;
- Substituição – palavras abreviadas e gírias são substituídas por palavras em sua forma padrão, o que possibilita a remoção se for uma palavra de parada ou radicalização. Além disso, também são feitas as substituições de emojis por palavras;
- Remoção de endereços de sites – os endereços são removidos por meio de expressão regular;
- Remoção de pontuações – remoção de pontuações do texto, símbolos não identificados cadastradas previamente na ferramenta e acentos;
- Remoção de marcações – são removidas as marcações a outros usuários;
- Remoção de palavras de parada – palavras de parada são removidas, por meio desta fase o texto fica menor e conta apenas com palavras que apresentam algum valor semântico;
- Radicalização – por meio desta fase, são removidos os sufixos das palavras para deixa-las em sua forma padrão. Para tal, foi utilizado o algoritmo RSLP.

Na Figura 9 é apresentado um exemplo com a sequência das etapas do pré-processamento e suas saídas.

Figura 9 - Módulo de pré-processamento dos dados



Fonte: Elaborada pelo autor.

No contexto de redes sociais, surgem novas características que devem ser contempladas pela ferramenta e que não estão descritas no processo de análise textual convencional, são elas: não remoção de emojis, não remoção do símbolo cerquilha (#) que é conhecido como hashtag e é acompanhado por uma palavra que permite a classificação da postagem.

No caso dos nomes de usuários, que são formados pelo símbolo "@" + palavra não foram levados em consideração no presente trabalho e, portanto, foram removidos do conjunto de dados.

3.5 Extração de características e classificação

Quanto melhor é a etapa de pré-processamento, menos processamento será feito na etapa de extração de características. Isto ocorre pois o texto é diminuído drasticamente com a remoção de palavras sem valor agregado, por exemplo, palavras de parada.

Por meio desta etapa são extraídas as características de frequência no uso das palavras e a análise de sentimento do texto. Essas técnicas são descritas nas próximas seções.

3.5.1 Frequência do termo inverso da frequência nos documentos

Frequência do termo inverso da frequência nos documentos, do inglês *term frequency – inverse document frequency* – TF-IDF é uma medida estatística amplamente utilizado na mineração de texto para refletir a importância de um termo para um documento no corpus. Tomamos um termo por *t*, um documento por *d* e o corpus por *D*, a frequência do termo *TF(t, d)* é o número de vezes que o termo *t* aparece no documento *d*, enquanto a frequência do documento *TF(t, D)* é o número de documentos que contém o termo *t*. A frequência inversa do documento é uma medida numérica da quantidade de informação que um termo fornece. Em que a equação simplificada é apresentada a seguir:

$$TFIDF(t, d, D) = TF(t, d) * IDF\ (t, D)$$

$$TF(t, d) = \frac{Número\ de\ vezes\ que\ \boldsymbol{t}\ aparece\ em\ \boldsymbol{d}}{Número\ total\ de\ termos\ de\ um\ documento}$$

$$IDF(t, D) = \log\left(\frac{Tamanho\ total\ de\ documentos}{Número\ de\ documentos\ com\ \boldsymbol{t}}\right)$$

Neste contexto, o documento é o conjunto de postagens de um mesmo usuário e o corpus o conjunto de postagens utilizadas na análise. Com isso, possibilita saber a importância de determinadas palavras utilizadas por um usuário, que pelo TFIDF apresenta maior valor, ou seja, maior relevância do que palavras comuns utilizadas que podem ser encontradas em outros documentos.

Este método retorna um vetor com os valores numéricos TFIDF de cada palavra do registro em que serão utilizados na etapa de classificação.

3.5.2 Análise de sentimento

A análise de sentimento das postagens é o processo realizado após a etapa de pré-processamento, em que restam apenas as raízes das palavras, os emojis e as hashtags. Com isso, o analista insere no sistema os dados desconhecidos sobre a classe em que a palavra pertence atribuindo pontuações como -1 para palavras negativas e 1 para palavras positivas.

Dessa forma a categoria do registro é dada pelo somatório dos pontos atribuídos para cada token. Com a utilização da ferramenta para diferentes contextos são criados dicionários de sentimentos que reúnem as palavras utilizadas em um determinado idioma.

Uma mesma palavra pode ser classificada tanto positiva em um determinado contexto quanto negativa em outro contexto. Um exemplo com o texto original e pré-processado foi retirado do conjunto de dados de análise é apresentado a seguir:

1) @baekliar o governo de minas dá ate estojo???????socorro
    a. Govern mino da estoj socorr
2) Calamidade em Minas, o estado pede socorro
    a. Calam mino est ped socorr

De acordo com os exemplos apresentados, a palavra "socorro" apresenta variações linguísticas e conta com 2 classificações de sentimento. No primeiro exemplo a palavra é utilizada como interjeição, ou seja, elas exprimem estados emocionais, com o significado de surpresa, portanto positivo. Já no segundo exemplo a palavra "socorro" á apresentada como substantivo masculino e exprime o sentido de pedido de ajuda, portanto negativa.

Os dois exemplos caracterizam muito bem o contexto de mídias sociais, em que palavras comumente utilizadas na língua são usadas de diferentes formas, essas variações são conhecidas como variações linguísticas.

Desta forma, para a definição do valor do sentimento atrelada à palavra "socorro", o cálculo é feito por meio do algoritmo Naive Bayes e a determinação da probabilidade condicional (NOUREEN; QAMAR; ALI, 2017), além disso precisa-se criar as tabelas de palavras e suas classes, frequência de palavras de acordo com a classe e a tabela de probabilidade que são apresentadas na Tabela 10 e na Tabela 11.

Tabela 10 - Tabela de palavras e suas classes

Palavra	Classe
govern	negativa
mino	positiva
da	positiva
estoj	positiva
socorr	positiva
calam	negativa
mino	positiva
est	positiva
ped	negativa
socorr	negativa

Tabela 11 - Tabela de frequência das palavras, suas classes e suas probabilidades

Palavra	Positiva	Negativa	Probabilidades
govern		1	1/10 = 0,1
mino	2		2/10 = 0,2
da	1		1/10 = 0,1
estoj	1		1/10 = 0,1
socorr	1	1	2/10 = 0,2
calam		1	1/10 = 0,1
est	1		1/10 = 0,1
ped		1	1/10 = 0,1
Total	6	4	
	6/10 = 0,6	4/10 = 0,4	

Para o exemplo apresentado, o cálculo do sentimento da palavra “socorro” é feito da seguinte forma:

$$P(socorro \mid positiva) = 0{,}1 * 0{,}6 = 0{,}06$$

$$P(positiva \mid socorro) = \frac{P\ (socorro \mid positiva) * P(positiva)}{P(socorro)} = \frac{0.06 * 0.6}{0.2} = 0.18$$

$$P(socorro \mid negativa) = 0{,}1 * 0{,}4 = 0{,}04$$

$$P(negativa \mid socorro) = \frac{P\ (socorro \mid negativa) * P(negativa)}{P(socorro)} = \frac{0.04 * 0.4}{0.2} = 0.08$$

Normalizando as probabilidades, temos que a probabilidade da palavra “socorro” ser positiva é de aproximadamente 69%, enquanto a probabilidade de ser negativa é de aproximadamente 31%.

Portanto, conforme apresentado, a probabilidade da palavra socorro apresentar sentimento positivo é maior do que apresentar sentimento negativo no contexto do exemplo. Essas probabilidades condicionais foram realizadas para determinar o sentimento das palavras no conjunto de análise tratado no trabalho.

4 Testes e resultados

Neste capítulo é apresentada a metodologia de experimentação para avaliação da ferramenta de extração de conhecimentos sobre dados textuais com algoritmos paralelos, bem como os resultados comparativos e a discussão dos experimentos realizados por meio do desenvolvimento do trabalho.

4.1 Metodologia dos experimentos

Os experimentos realizados têm como objetivo validar o funcionamento da ferramenta com o contexto de dados textuais de diferentes origens, conexão entre etapas do processo de extração de conhecimento sobre textos e desempenho dos algoritmos paralelos. Portanto para as etapas de pré-processamento, extração de características e classificação os algoritmos foram desenvolvidos tanto em sua forma sequencial quanto em sua forma paralela com o uso do *framework* Apache Spark. A etapa de classificação foi executada juntamente com a etapa de extração de características, em que é construída a estrutura de dados utilizada e são feitos os cálculos de probabilidade condicional do sentimento das palavras. Com isso valida o ganho de desempenho com a abordagem desenvolvida no trabalho.

As etapas de processamento foram executadas cinco vezes e os resultados apresentados correspondem à média das execuções. Além disso, o desvio padrão dos valores medidos também são apresentados.

Na Tabela 12 são apresentadas as configurações de software e hardware utilizadas para a realização dos testes.

Tabela 12 - Configurações da máquina utilizada para testes

Atributo	**Configuração**
Processador	4 núcleos Intel Core i7-7500U a 2.90 GHz
Memória	4GB
Disco rígido	1TB
Sistema Operacional	Ubuntu 16.04
Apache Spark	2.4.3
Java	jdk_1.8.0_181

4.2 Dados de mídias sociais

A partir de 2018 as empresas Facebook e Twitter restringiram o acesso à dados de usuários e endureceram as políticas de privacidade. Isso ocorreu por conta de um vazamento de dados de aproximadamente 50 milhões de usuários para a empresa de marketing político Cambridge Analytica, conforme noticiado pela empresa de tecnologia Bloomberg[13]. O vazamento influenciou na campanha eleitoral de Donald Trump nos Estados Unidos e na campanha para a saída do Reino Unido da União Europeia.

Com isso, o acesso à dados de mídias sociais tornou-se muito difícil, principalmente pelo seu uso indevido.

Contudo, para a realização dos experimentos foi possível coletar dados de postagens reais de brasileiros no Twitter no ano de 2017 que totalizaram 8.199 registros. Esses registros foram disponibilizados no formato JSON e contam com 9 atributos, entre eles: identificador, texto, latitude, longitude, localização, nome de usuário, apelido e contador de *retweets*.

Para as análises executadas no trabalho, foram considerados o identificador, texto da postagem e o nome do usuário, a fim de conseguir agrupar postagens das mesmas pessoas.

4.3 Testes de qualidade

Os testes de qualidade consistem em executar a ferramenta com os dados de postagens de usuários e analisar as saídas de cada etapa presente no processo de extração de conhecimento em textos, ou seja, pré-processamento, desenvolvimento léxico, extração de características e classificação.

[13] < https://www.bloomberg.com/news/articles/2018-03-21/cambridge-analytica-s-brazil-partner-suspends-deal-amid-scandal>

O conjunto de dados inicial conta com 8.199 registros dos quais possuem 132.310 tokens e 1.093.728 caracteres, uma média de aproximadamente 16 tokens e 133 caracteres por publicação.

Antes de iniciar o processo de descoberta de conhecimento sobre textos, o contexto dos dados de análise deve ser trabalhado e estudado para criação dos dicionários de substituições.

Após a etapa de pré-processamento a quantidade de tokens do conjunto de dados diminuiu para 89.823 tokens e 520.875 caracteres. O que representa uma redução de aproximadamente 32% de tokens e 52% no número de caracteres. Essa redução na quantidade de tokens e caracteres é importante, pois retira da amostra tokens sem valor semântico ou reduz o token de uma forma que o seu entendimento não é comprometido.

O desenvolvimento da etapa de pré-processamento com a abordagem paralela e o algoritmo de radicalização RSLP (ORENGO; HUYCK, 2001) foram implementados de modo a preservar a lógica do algoritmo original sequencial, tendo como saída os mesmos resultados.

Como apresentado anteriormente, com o algoritmo TF-IDF é possível identificar as palavras mais relevantes para o documento, que podem ser ditas como o seu assunto. Com isso, foi possível identificar postagens que continham palavras como: “financeiro”, “assassinato”, “esperança”, “criminal”, “drogas”, “inacreditável”, “vingança”, “presos”, etc. A palavra “financeiro” apareceu com maior valor de TF-IDF em postagens de 146 usuários, enquanto a palavra “drogas” apareceu com maior valor TF-IDF em postagens de 38 usuários.

A classificação de sentimento dos dados de análise foram a seguinte: 3.300 postagens com sentimento positivo, 2.446 postagens com sentimento negativo e 2.453 postagens com sentimento neutro. O sentimento neutro foi verificado em postagens com o teor informativo, em postagens em que o usuário não expressa nenhum sentimento ou em postagens que o somatório de palavras positivas e negativas são as mesmas.

Além disso, o programa pode ser configurado para gerar saída de cada etapa em arquivo, dessa forma, possibilita ao analista verificar se as saídas estão de acordo com a teoria de cada etapa e se os cálculos estão corretos.

Com o eficiente pré-processamento dos dados, extração de características por TF-IDF e análise de sentimento de postagens o trabalho demonstra que pode ser utilizado em contextos de mídias sociais e que as características extraídas podem ser utilizadas na classificação de usuários com um determinado comportamento de escrita.

4.4 Testes de desempenho

Esta seção tem o objetivo de apresentar os resultados comparativos da execução dos algoritmos em sua forma sequencial e em sua forma paralela, a fim de comprovar o ganho de desempenho com as alterações realizadas e sem perder a qualidade, conforme apresentado na seção anterior.

Como dito anteriormente, as medições apresentadas são as médias de cinco execuções e o desvio padrão segue em parênteses. Na Tabela 13 são apresentados os tempos de execução nos módulos de pré-processamento, extração de características e classificação, distinguindo a abordagem de programação. Os resultados de tempo são apresentados em segundos e entre parênteses é apresentado o desvio padrão observado nas medições.

Tabela 13 – Comparação de resultados entre os modelos de programação nos módulos da ferramenta

	Tempo de processamento em segundos e desvio padrão		
	Pré-processamento	Ext. de características e Classificação	Total
Sequencial	14,7628 (3,640)	99,3274 (10,154)	114,0902 (12,484)
Paralelo	0,1050 (0,001)	9,8208 (0,291)	9,9258 (0,291)

Fonte: Elaborado pelo autor.

Conforme apresentado na tabela, é possível verificar o ganho de desempenho alcançado pela abordagem paralela desenvolvida em todos os módulos da ferramenta.

No módulo de pré-processamento, a execução paralela foi superior à sequencial em aproximadamente 140 vezes. Esse ganho de desempenho se dá por conta das características desse módulo, que conta com operações que podem ser feitas diretamente dentro da transformação `map()`. Dessa forma, não conta com laços de repetições entre todos os elementos e operações custosas, que vimos que pode prejudicar o desempenho do Spark, conforme apresentado na Seção 2.

No módulo de extração de características e classificação, a execução paralela foi superior à sequencial em aproximadamente 10 vezes. O ganho de desempenho ainda é significativo, porém, menor do que o observado pelo módulo de pré-processamento. Nesse módulo, é feita a operação de `map()` duas vezes, conforme apresentado na Seção 3, e pela necessidade do cálculo de frequência de cada token no documento completo, o desempenho é prejudicado.

O Spark é conhecido por contar com o processamento preguiçoso, do inglês *Lazy Evaluation*, que consiste em realizar o processamento de fato apenas quando uma

transformação nos dados, neste caso os RDDs, acontece. Portanto, as etapas do pré-processamento são efetuadas pelo programa quando realiza as operações para extração de características. Dessa forma, conforme apresentado na Tabela 13, é esperado um tempo pequeno na etapa de pré-processamento pois os dados são apenas carregados na estrutura de dados.

Para uma análise mais precisa, foi feita a soma do tempo gasto no pré-processamento, extração de características e classificação. O ganho de desempenho verificado foi de aproximadamente 11 vezes, ou seja, mesmo com o processamento dos dados de forma "preguiçosa", o algoritmo desenvolvido apresenta uma diminuição significativa no tempo de execução.

O cálculo do ganho de desempenho é feito por meio do *speedup*. O *speedup* representa quão mais rápida determinada tarefa pode ser executada em um programa que recebeu modificações e melhorias, comparado ao programa inicial. Em que t_{sm} é o tempo de execução do programa sem melhorias e t_{cm} é o tempo de execução do programa com melhorias, conforme apresentado abaixo.

$$speedup = \frac{t_{sm}}{t_{cm}}$$

Portanto, o valor de *speedup* verificado para os aprimoramentos é dado da seguinte forma:

$$speedup = \frac{114{,}0902}{9{,}9258} = 11{,}4943$$

Com o intuito de validar a escalabilidade da ferramenta, foram realizados testes com maiores quantidades de dados. Os dados utilizados para estes testes foram com as mesmas características do teste anterior, ou seja, de origem de mídias sociais, em forma de texto, com emojis e endereços de sites.

Os testes consistiram em executar a ferramenta desenvolvida com as quantidades de 10, 20, 30, 40 e 50 mil registros. Os resultados de tempo são apresentados na Tabela 14, entre parênteses é apresentado o valor do desvio padrão observado nas medições.

Por meio dos tempos apresentados, é possível verificar que o crescimento de tempo até 30 mil registros é quase linear, ou seja, apresenta um bom desempenho mesmo com maior quantidade de dados. Porém, a partir dos 40 mil registros o crescimento do tempo começa a aumentar perdendo a linearidade.

Tabela 14 – Tempos de execução de 10 mil a 50 mil registros

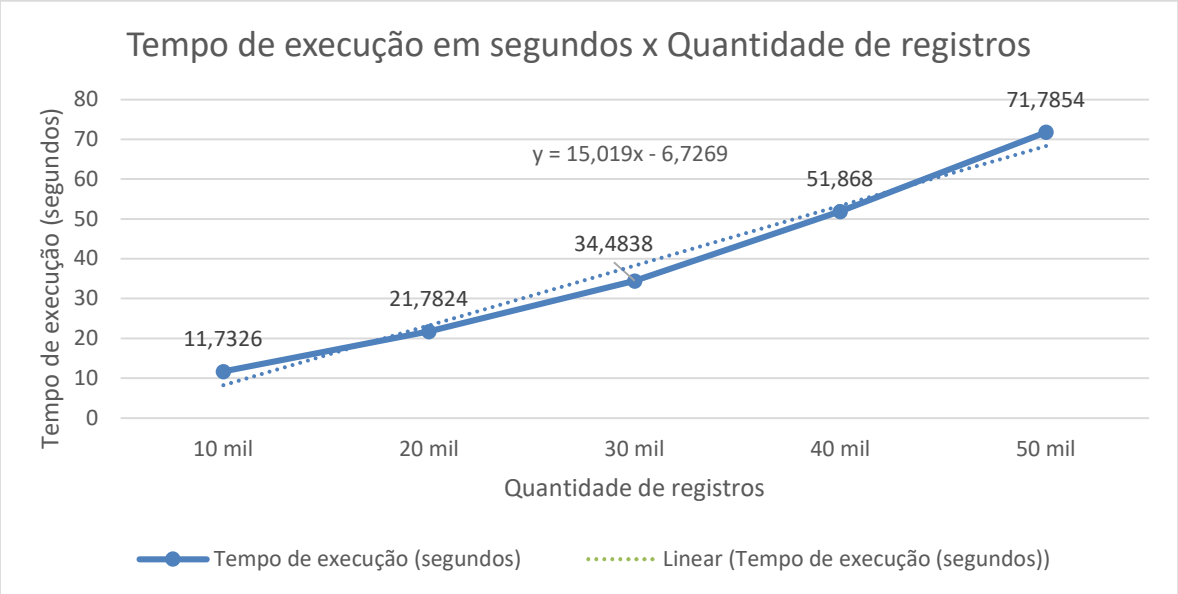
Execuções com abordagem paralela	
Quantidade de registros de entrada	Tempo de execução em segundos e desvio padrão
10 mil	11,7326 (0,499)
20 mil	21,7824 (0,664)
30 mil	34,4838 (1,953)
40 mil	51,868 (3,008)
50 mil	71,7854 (0,984)

Fonte: Elaborada pelo autor.

No Gráfico 1 é apresentado o crescimento de tempo em relação ao aumento do número de registros como entrada. A equação gerada por meio dos resultados é y = 15,019x – 6,7269, em que x é dado a cada 10 mil registros. Dessa forma é possível fazer uma estimativa do tempo de execução para a quantidade de registros de entrada.

Porém, seriam necessários mais testes para definir com maior precisão uma equação para o crescimento de tempo em relação ao tamanho de entrada.

Gráfico 1 - Gráfico do tempo de execução por quantidade de registros



4.5 Testes complementares

Com o objetivo de apresentar a possibilidade do uso da ferramenta em outros contextos, foram realizados testes com uma maior quantidade de dados textuais e que fazem parte de dados

referentes à saúde mental. Dessa forma, os testes foram aferidos com a média de cinco execuções e o cálculo do desvio padrão.

A base de dados utilizada conta com 152.688 registros de prontuários médicos, 4.509.918 tokens e 33.105.402 caracteres. Essa base de dados conta com anotações realizadas por médicos sobre os pacientes, portanto conta com características diferentes dos registros de postagens de mídias sociais, portanto, para este contexto não foi realizada a etapa de classificação de sentimentos.

Esse contexto conta com textos pequenos de duas ou três palavras, até textos mais completos com relatos do estado em que o paciente se encontra no momento do atendimento. Nos textos temos informações de doenças com código CID, aferimento de pressão e sinais vitais, estado mental e se o paciente apresentou melhora ou piora com o tratamento.

O algoritmo foi configurado para fazer substituições em relação ao código de doença CID, remover pontuações e palavras de parada, e radicalizar as palavras. Dessa forma, sendo possível realizar a extração das características com TF-IDF.

Após a etapa de pré-processamento, o conjunto de dados foi reduzido para 4.132.256 tokens e 25.919.547 caracteres, o que reduziu a quantidade de tokens em aproximadamente 8,5% e de caracteres em aproximadamente 22%.

Com a etapa de TF-IDF foi possível verificar que o algoritmo a seleção de palavras como: “ansioso”, “agitado”, “calmo”, “bipolar”, “drogas”, “insônia”, “hidratado”, “confuso”, etc.

Os tempos de processamento da ferramenta para este contexto de dados, que apresenta maior quantidade de tokens e caracteres por registro, apresentou tempo de processamento em média de 576,405 segundos com um desvio padrão de 9,461. O tempo maior em relação à equação de crescimento apresentado na seção anterior deve-se principalmente pelo maior volume de texto encontrado nesse conjunto de dados, em que para o cálculo de TF-IDF o algoritmo precisa verificar se o token de análise está presente em outros registros e em quantos registros.

Dessa forma, salienta-se que para o bom funcionamento da ferramenta, é necessário que o analista tenha informações sobre o contexto de estudo. Pois nesse caso, verificou-se o frequento uso de termos médicos e notações comuns para aferimentos de temperatura, pressão e sinais vitais. E com as corretas substituições o algoritmo apresenta um melhor desempenho e melhora a qualidade das palavras frequentemente utilizadas em cada registro.

4.6 Considerações finais

Conforme apresentado nas subseções anteriores a abordagem paralela para execução do programa para extração de conhecimentos em textos de mídias sociais apresentou um ganho de desempenho de 11 vezes em relação à abordagem sequencial. Além disso, apresentou uma boa escalabilidade para maiores quantidades de dados e foi verificada possibilidade de ser utilizada em outros contextos de análises.

5 Conclusão

Com a popularização dos sites de redes sociais, entre eles Facebook e Twitter, a quantidade de dados gerados pelos usuários crescem de maneira muito rápida. Essa grande quantidade de dados e sua complexidade apresenta desafios para as ferramentas convencionais de análise e armazenamento de dados. Juntamente com essa dificuldade, têm-se nesses dados uma fonte de informação interessante.

Além disso, segundo relatórios da Organização Mundial da Saúde um dos maiores motivos de incapacidade das pessoas é a saúde mental, em que afetam diretamente na capacidade de viver, convivência em sociedade e de relacionamento, tendo também uma série consequência para a economia do país.

Tendo isso em vista, o trabalho desenvolvido busca contribuir com as áreas de análise de dados de redes sociais por meio da disponibilização de um ambiente em que considera o contexto dos dados e com o desenvolvimento de algoritmos de pré-processamento, extração de características e classificação de sentimentos paralelos. Com as abordagens apresentadas espera-se contribuir para a área do trabalho e oferecer uma ferramenta capaz de suportar o desenvolvimento de novos algoritmos e compará-los com o estado da arte.

5.1 Contribuições

Por meio dos experimentos realizados, foi possível observar o ganho de desempenho nas etapas de extração de conhecimento sobre textos, desde o pré-processamento até a classificação. Além disso, o desenvolvimento do trabalho apresentou tratamentos específicos para o conjunto de dados com origem em redes sociais, como hashtags, emojis, lista de exceções e substituição de palavras.

Na Tabela 15 é apresentada a tabela com os atributos da ferramenta desenvolvida em comparação com os trabalhos correlatos encontrados na literatura.

Tabela 15 - Comparativo entre os trabalhos correlatos e este trabalho

	(VARSHNEY et al., 2017)	(DE CHOUDHURY et al., 2013)	(LUKITO et al., 2016)	(SHORO; SOOMRO, 2015)	Este trabalho
Estratégias propostas ao cenário *Big Data*	✓	✓	✓	✓	✓
Origem de dados de Mídias Sociais	✓	✓	✓	✓	✓
Pré-processamento dos dados	✓	✗	✗	✗	✓
Extração de características	✓	✓	✗	✗	✓
Classificação	✓	✓	✓	✗	✓
Buscas por palavras específicas	✗	✓	✗	✓	✓
Considera o contexto de mídias sociais na análise	✗	✗	✗	✗	✓
Análise comportamental	✓	✓	✗	✗	✗
Estratégias de desempenho paralela e escalável	✗	✗	✗	✓	✓

Portanto, a ferramenta desenvolvida é a principal contribuição e os mecanismos de tratamento de dados de redes sociais são as contribuições secundárias que podem ser usadas para comparações em novos trabalhos desenvolvidos nesta área.

Além disso, ela pode ser utilizada para analisar outros contextos com dados textuais, o que foi apresentado como teste complementar. Salienta-se que para o seu bom funcionamento as configurações devem ser feitas para alavancar o seu desempenho.

5.2 Trabalhos futuros

Com a finalização do trabalho, foi possível verificar áreas relacionadas que podem ser exploradas com o objetivo de contribuir com o estado da arte, são elas:

- Desenvolvimento de novas técnicas de extração de características por meio de Processamento de Linguagem Natural para dados de redes sociais;
- Trabalho de coleta de dados de pessoas que sofrem de algum tipo de transtorno mental para treinamento do conjunto de dados;
- Recursos de streaming do *framework* Apache Spark para processamento de dados em tempo real;

- Permissões para acessar dados de postagens de redes sociais – submeter a ferramenta para análise da companhia de rede social;
- Fazer uso de atributos não textuais das postagens, por exemplo: evolução temporal, geolocalização, compartilhamentos, etc;
- Integrar dados de diferentes fontes de um mesmo indivíduo para análise.

Referências

ABRAMOVA, Veronika; BERNARDINO, Jorge; FURTADO, Pedro. Which nosql database? a performance overview. **Open Journal of Databases (OJDB)**, v. 1, n. 2, p. 17-24, 2014.

AGGARWAL, Charu C.; ZHAI, ChengXiang (Ed.). **Mining text data**. Springer Science & Business Media, 2012.

AJI, Ablimit et al. Hadoop gis: a high performance spatial data warehousing system over mapreduce. **Proceedings of the VLDB Endowment**, v. 6, n. 11, p. 1009-1020, 2013.

AKAICHI, Jalel; DHOUIOUI, Zeineb; PÉREZ, Maria José López-Huertas. Text mining facebook status updates for sentiment classification. In: **System Theory, Control and Computing (ICSTCC), 2013 17th International Conference**. IEEE, 2013. p. 640-645.

ALAM, Saqib; YAO, Nianmin. Big Data Analytics, Text Mining and Modern English Language. **Journal of Grid Computing**. 2018. p. 1-10.

ALTINEL, Berna; GANIZ, Murat Can. Semantic text classification: A survey of past and recent advances. **Information Processing & Management**, v. 54, n. 6, p. 1129-1153, 2018.

ATIG, Mohamed Faouzi et al. Activity profiles in online social media. In: **Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining**. IEEE Press, 2014. p. 850-855.

APACHE SPARK, Spark. Disponível em: https://spark.apache.org/. Acesso em 20 de Janeiro de 2020.

BALDWIN, Timothy. Social media: friend or foe of natural language processing?. In: **PACLIC**. 2012. p. 58-59.

BANERJEE, Shreya; SARKAR, Anirban. Logical level design of NoSQL databases. In: **Region 10 Conference (TENCON), 2016 IEEE**. IEEE, 2016. p. 2360-2365.

BATRINCA, Bogdan; TRELEAVEN, Philip C. Social media analytics: a survey of techniques, tools and platforms. **AI & SOCIETY**, v. 30, n. 1, p. 89-116, 2015.

BORTHAKUR, Dhruba et al. HDFS architecture guide. **Hadoop Apache Project**, v. 53, 2008.

BOSAK, Kelly; PARK, Shin Hye. Characteristics of Adults' Use of Facebook and the Potential Impact on Health Behavior: Secondary Data Analysis. **Interactive journal of medical research**, v. 7, n. 1, 2018.

DEAN, Jeffrey; GHEMAWAT, Sanjay. MapReduce: simplified data processing on large clusters. **Communications of the ACM**, v. 51, n. 1, p. 107-113, 2008.

DE CHOUDHURY, Munmun et al. Predicting Depression via Social Media. **ICWSM**, v. 13, p. 1-10, 2013.

DIVYA, P; NANDA KUMAR, G.S. "Study on Feature Selection Methods for Text Mining" **Proceeding of International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)** Vol. 2, Jan 2015.

DONG, Xin Luna; SRIVASTAVA, Divesh. Big data integration. In: **Data Engineering (ICDE), 2013 IEEE 29th International Conference on**. IEEE, 2013. p. 1245-1248.

FAN, Weiguo et al. Tapping the power of text mining. **Communications of the ACM**, v. 49, n. 9, p. 76-82, 2006.

FELDMAN, Ronen; DAGAN, Ido. Knowledge Discovery in Textual Databases (KDT). In: **KDD**. 1995. p. 112-117.

FONSECA, Felipe Cesar Stanzani; BELTRAME, Walber Antônio Ramos. Aplicações Práticas dos Algoritmos de Clusterização K-means e Bisecting K-means. **Universidade Federal do Espírito Santo (UFES)**. Vitória-ES, 2011.

FONTANA, André; NALDI, Murilo Coelho. Estudo e comparação de métodos para estimação de números de grupos em problemas de agrupamento de dados. **Universidade de São Paulo (USP)**, 2009.

FOX, Jesse; MORELAND, Jennifer J. The dark side of social networking sites: An exploration of the relational and psychological stressors associated with Facebook use and affordances. **Computers in Human Behavior**, v. 45, p. 168-176, 2015.

GOLE, Sheela; TIDKE, Bharat. A survey of big data in social media using data mining techniques. In: **Advanced Computing and Communication Systems, 2015 International Conference on**. IEEE, 2015. p. 1-6.

GOLE, Sheela; TIDKE, Bharat. Frequent Itemset Mining for Big Data in social media using ClustBigFIM algorithm. In: **Pervasive Computing (ICPC), 2015 International Conference on**. IEEE, 2015. p. 1-6.

HAN, Jiawei; PEI, Jian; KAMBER, Micheline. **Data mining: concepts and techniques**. Elsevier, 2011.

HARRATHI, Farah. **Extraction de concepts et de relations entre concepts à partir des documents multilingues: approche statistique et ontologique**. 2009. Tese de Doutorado. Villeurbanne, INSA.

TOLKIEN, John Ronald Reuel. **The Lord of the Rings – PART 1. The Fellowship of the Ring.** George Allen & Unwin Ltd, 1954.

KAISLER, Stephen et al. Big data: Issues and challenges moving forward. In: **System Sciences (HICSS), 2013 46th Hawaii International Conference on**. IEEE, 2013. p. 995-1004.

KAPLAN, Andreas M.; HAENLEIN, Michael. Users of the world, unite! The challenges and opportunities of Social Media. **Business horizons**, v. 53, n. 1, p. 59-68, 2010.

KATAL, Avita; WAZID, Mohammad; GOUDAR, R. H. Big data: issues, challenges, tools and good practices. In: **Contemporary Computing (IC3), 2013 Sixth International Conference on**. IEEE, 2013. p. 404-409.

KRIKORIAN, Raffi. New Tweets per second record, and how. **Twitter Engineering Blog**, v. 16, 2013.

LANEY, Doug. 3D data management: Controlling data volume, velocity and variety. **META Group Research Note**, v. 6, p. 70, 2001.

LAZER, David et al. Life in the network: the coming age of computational social science. **Science (New York, NY)**, v. 323, n. 5915, p. 721, 2009.

LEE, Kyong-Ha et al. Parallel data processing with MapReduce: a survey. **AcM sIGMoD Record**, v. 40, n. 4, p. 11-20, 2012.

LI, Haoyuan et al. Tachyon: Reliable, memory speed storage for cluster computing frameworks. In: **Proceedings of the ACM Symposium on Cloud Computing**. ACM, 2014. p. 1-15.

LUKITO, Louis Christy et al. Social media user personality classification using computational linguistic. In: **Information Technology and Electrical Engineering (ICITEE), 2016 8th International Conference on**. IEEE, 2016. p. 1-6.

LUKOIANOVA, Tatiana; RUBIN, Victoria L. Veracity roadmap: Is big data objective, truthful and credible?. 2014.

MEJRI, Mohamed; AKAICHI, Jalel. A Survey of Textual Event Extraction from Social Networks.

MELETHADATHIL, Nidheesh et al. Assessing short-term social media marketing outreach of a healthcare organization using machine learning. In: **Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on**. IEEE, 2017. p. 387-392.

MOHAN, C. History repeats itself: sensible and NonsenSQL aspects of the NoSQL hoopla. In: **Proceedings of the 16th International Conference on Extending Database Technology**. ACM, 2013. p. 11-16.

MUNKOVÁ, Daša; MUNK, Michal; VOZÁR, Martin. Influence of stop-words removal on sequence patterns identification within comparable corpora. In: **ICT innovations 2013**. Springer, Heidelberg, 2014. p. 67-76.

NOUREEN, Amna; QAMAR, Usman; ALI, Mubashir. Semantic analysis of social media and associated psychotic behavior. In: **2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)**. IEEE, 2017. p. 1621-1630.

ORENGO, Viviane; HUYCK, Christian. A stemming algorithmm for the portuguese language. In: **spire**. IEEE, 2001. p. 0186.

PORTER, Martin F. An algorithm for suffix stripping. **Program**, v. 14, n. 3, p. 130-137, 1980.

PORTER, Martin F. Snowball: A language for stemming algorithms. 2001.

REZAEIAN, Mina; MONTAZERI, Hamid; LOONEN, Roel C. G. M. Science foresight using lifecycle analysis, text mining and clustering: A case study on natural ventilation. **Technological Forecasting and Social Change**. Elsevier, v. 118, p. 270-280, 21 fev. 2017.

SAGIROGLU, Seref; SINANC, Duygu. Big data: A review. In: **Collaboration Technologies and Systems (CTS), 2013 International Conference on**. IEEE, 2013. p. 42-47.

SAVOY, Jacques. Light stemming approaches for the French, Portuguese, German and Hungarian languages. In: **Proceedings of the 2006 ACM symposium on Applied computing**. ACM, 2006. p. 1031-1035.

SHI, Juwei et al. Clash of the titans: Mapreduce vs. spark for large scale data analytics. **Proceedings of the VLDB Endowment**, v. 8, n. 13, p. 2110-2121, 2015.

SHORO, Abdul Ghaffar; SOOMRO, Tariq Rahim. Big data analysis: Apache spark perspective. **Global Journal of Computer Science and Technology**, v. 15, n. 1, 2015.

SICULAR, Svetlana.; Gartner's Big Data Definition Consists of Three Parts, Not to Be Confused with Three 'V's. Disponível em: https://www.forbes.com/sites/gartnergroup/2013/03/27/gartners-big-data-definition-consists-of-three-parts-not-to-be-confused-with-three-vs/#68f1014442f6. Publicado em: 27 de março de 2013. Acesso em: 10 de outubro de 2017.

SINGH, Dilpreet; REDDY, Chandan K. A survey on platforms for big data analytics. **Journal of Big Data**, v. 2, n. 1, p. 8, 2015.

TROUSSAS, Christos et al. Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning. In: **Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on**. IEEE, 2013. p. 1-6.

VARSHNEY, Vanshika et al. Recognising personality traits using social media. In: **2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)**. IEEE, 2017. p. 2876-2881.

VIJAYARANI, S.; ILAMATHI, Ms J.; NITHYA, Ms. Preprocessing techniques for text mining-an overview. **International Journal of Computer Science & Communication Networks**, v. 5, n. 1, p. 7-16, 2015.

XAVIER, Bruno Missi; GOMES, Georgia Regina Rodrigues; SILVA, Alcione Dias. Análise comparativa de algoritmos de redução de radicais e sua importância para a mineração de texto. **Pesquisa Operacional para o Desen-volvimento**, v. 5, n. 1, p. 84-99, 2013.

WHITE, Tom. **Hadoop: The definitive guide**. " O'Reilly Media, Inc.", 2012.

WORLD HEALTH ORGANIZATION. Mental Health Action Plan 2013-2020. **WHO Library Cataloguing-in-Publication DataLibrary Cataloguing-in-Publication Data**, 1–44, 2013. https://doi.org/ISBN 978 92 4 150602 1.

WORLD HEALTH ORGANIZATION. Mental Disorders. Fact sheet nº 396. Publicado em out/2014. Disponível em: https://web.archive.org/web/20150518090215/http://www.who.int/mediacentre/factsheets/fs396/en/. Acesso em: 14 de novembro de 2017.

WU, Yingcai et al. A survey on visual analytics of social media data. **IEEE Transactions on Multimedia**, v. 18, n. 11, p. 2135-2148, 2016.

ZAHARIA, Matei et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: **Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. USENIX Association, 2012. p. 2-2.

TERMO DE REPRODUÇÃO XEROGRÁFICA

Autorizo a reprodução xerográfica do presente Trabalho de Conclusão, na íntegra ou em partes, para fins de pesquisa.

São José do Rio Preto, 16 de março de 2020

Victor Hugo Penhalves Martins

Victor Hugo Penhalves Martins