Fatec Cruzeiro Prof. Waldomiro May
Revista Científica H-TEC Humanidades & Tecnologia

UMA APLICAÇÃO DE BIG DATA PARA CLASSIFICAÇÃO DE SENTENÇAS DEPRESSIVAS DO TWITTER

Autores
José Walmir Gonçalves Duque[1]
Abner Lucas Raymundo[2]
Pedro Ferreira Neto[3]

Resumo
Com o crescente número de casos de depressão, foi pensada uma aplicação de Inteligência Artificial que, utilizando-se Processamento de Linguagem Natural, realizasse Análise de Sentimentos em publicações da rede social Twitter e, então, identificasse se um texto contém uma mensagem depressiva. Portanto, este trabalho tem por objetivo desenvolver um protótipo de software que possa ser um classificador que, ao receber um texto livre, seja capaz de identificar padrões depressivos. Para a realização deste trabalho, foi utilizada a linguagem Python com a biblioteca NLTK, a qual é alicerçada no teorema de Naive Bayes. Para criar a base de dados de teste, foram utilizados 1200 tweets coletados de Big Data de terceiros, que foram classificados manualmente. Em seguida foi realizado um tratamento dessa lista, composto pelas fases de remoção de stopwords, palavras que não possuem peso na análise, e de extração de radical, para melhor aproveitamento do vocabulário. Com a base de treinamento já tratada, foi realizado o treinamento do algoritmo. Para validar o algoritmo, foi utilizada uma base de testes com 120 tweets, quando foi realizada comparação entre resultados já esperados da base de testes e os gerados pela inteligência - foi alcançado índice de 75% de acerto. Assim, comprovou-se a possibilidade de identificar pessoas que publicam frequentemente textos depressivos em uma rede social por meio de algoritmo inteligente.
Palavras-chaves: 1. Big Data; 2. Processamento de Linguagem Natural; 3. Análise de Sentimento; 4. Mineração de Texto.

INTRODUÇÃO

A situação atual mostra que o volume de informações que são geradas diariamente na internet cresceu tanto que se tornou necessário construir métodos, técnicas e ferramentas para conseguir minimamente aproveitar essa riqueza. Graças à internet e às redes sociais, as pessoas podem se expressar de uma maneira mais livre com imagens, textos e vídeos alcançando um número maior de pessoas, as quais compõem a maior parte do fluxo de dados que circula pela internet.

Por meio de algoritmos inteligentes, em particular na Mineração de Dados (*Data Mining*), é possível filtrar e fazer uma breve análise de informações relevantes para um objetivo

[1] Docente no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na FATEC
[2] Graduando no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na FATEC
[3] Graduando no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na FATEC

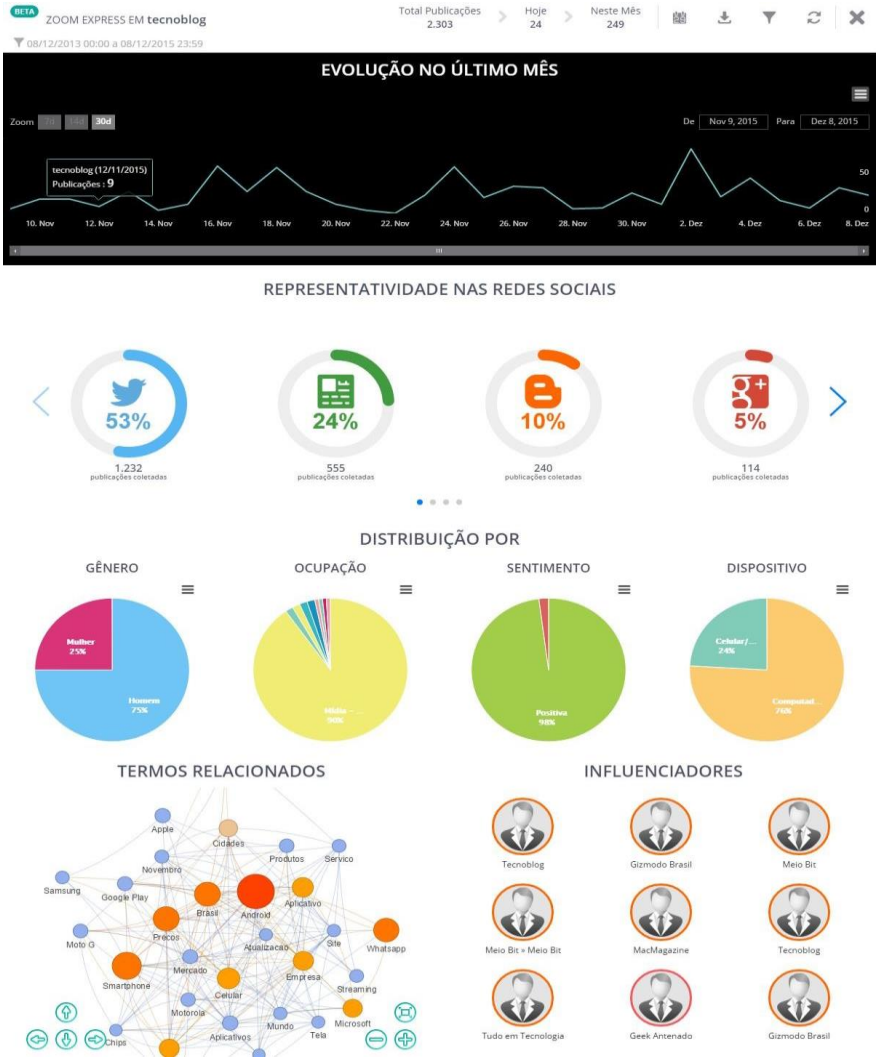
específico dentro desses terabytes diários que são gerados.

Esses dados trazem informações muito valiosas que podem ser trabalhadas de diversas maneiras: muitas empresas já utilizam essas informações para se beneficiar comercialmente, porém o uso delas para outras áreas pode ser muito bem aproveitado também, com um cunho social ou médico, por exemplo. Para esse tipo de análise, faz-se necessário um processamento mais complexo, pois, para uma máquina realizar interpretações de sentimentos humanos, é necessário uma IA (Inteligência Artificial) bem treinada para isso.

Nesse cenário de aplicações, percebe-se um crescente número de casos de depressão no mundo inteiro, o que preocupa a OMS (Organização Mundial da Saúde), que divulgou, em 2017, uma estatística do número de casos de depressão, o qual tem crescido cada vez mais, superando doenças como o câncer, por exemplo.

A área de IA tem apresentado avanços no sentido de análise de dados, em particular textos advindos de redes sociais. Na Figura 1 é mostrado um gráfico gerado pela ferramenta Stilingue que usa IA para monitoramento de redes sociais:

Figura 1: Gráficos de monitoramento de redes sociais do tecnoblog utilizando a ferramenta Stilingue



Fonte: https://tecnoblog.net/189060/stilingue-resumo-internet-ia/

Para que uma IA tenha a capacidade esperada, é necessária uma base de treinamento bem ampla: como neste trabalho se trata de PLN (Processamento de Linguagem Natural), quanto maior o vocabulário base de treinamento de uma IA, melhores os resultados alcançados.

A área de PLN, com base em IA, portanto, uma vez treinada, pode ser capaz de reconhecer e classificar perfis de pessoas depressivas. Com base nessas informações, especialistas na área (psicólogos e psiquiatras) poderiam desenvolver tratamentos específicos para cada perfil de depressão reconhecido pela IA. Pretende-se também, por meio deste estudo, testar até onde uma IA pode entender e reconhecer reações, diálogos e debates humanos.

Percebe-se a análise de sentimento como uma área de IA amplamente aplicada em redes sociais, a despeito de análise de veracidade dos posts, portanto, da qualidade dos dados analisados, um fonte rica de informação a respeito de sinais comportamentais das pessoas, marcas psicológicas a serem percebidas de forma marcante.

Entende-se, portanto, que resultados possam ser interessantes ao aplicar a técnica em diversas situações de observação de comportamento online nas redes sociais: em particular, com o foco em casos de depressão, pode ser possível encontrar maneiras de prevenir suicídios com apoio virtual ou, ao menos, apontar indicadores que possam apoiar trabalhos de voluntários no acompanhamento de pessoas com perfil suicida.

Em face a esse cenário supracitado, este trabalho tem por objetivo desenvolver um protótipo de aplicação que possa ser um classificador que, ao receber um texto livre, seja capaz de identificar se o texto representa sentido de depressão (padrões depressivos), a fim de testar a aplicação de métodos, técnicas e ferramentas adequadas.

Para o atendimento do objetivo geral, a seguir apresentam-se os seguintes objetivos específicos:

- Construir 2 bases de dados, uma base de treinamento com tweets de sentimentos depressivos e outra com tweets genéricos que não sejam depressivos;
- Implementar o algoritmo de PLN, em particular Naive Bayes, customizando os parâmetros para o atendimento do contexto desejado;
- Treinar o algoritmo de forma que ele possa reconhecer os padrões depressivos;
- Validar a base treinada no algoritmo de acordo com um conjunto de dados de testes;
- Por fim, pretende-se estudar os benefícios do PLN e o quão amplas são suas possibilidades de aplicação e uso.

1. FUNDAMENTOS DE BIG DATA, PLN E ANÁLISE DE SENTIMENTO

1.1.Inteligência Artificial

A Inteligência Artificial (IA) é um campo de estudo acadêmico recente em que se desenvolve e estuda máquinas e algoritmos capazes de tomar decisões. Segundo Russel e Norvig (2010, p.3), a IA além de aprender, também pode criar outras inteligências artificiais, sendo uma das ciências mais recentes, com origem em 1956, logo após a Segunda Guerra Mundial.

Em seguida, Russel e Norvig (2010, p. 3) comenta sobre o quão dinâmica a IA é, portanto, aplicável em qualquer outro campo de estudo ou trabalho, desde jogos à medicina. A

IA pode ser definida em 4 categorias como mostrado na Figura 2, de acordo com Russel e Norvig (2010):

Figura 2 - As 4 Categorias de IA definidas por Russel e Norvig (2010)

Pensando como um humano	**Pensando racionalmente**
"O novo e interessante esforço para fazer os computadores pensarem (...) *máquinas com mentes*, no sentido total e literal." (Haugeland, 1985) "[Automatização de] atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado..." (Bellman, 1978)	"O estudo das faculdades mentais pelo uso de modelos computacionais." (Charniak e McDermott, 1985) "O estudo das computações que tornam possível perceber, raciocinar e agir." (Winston, 1992)
Agindo como seres humanos	**Agindo racionalmente**
"A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas." (Kurzweil, 1990) "O estudo de como os computadores podem fazer tarefas que hoje são melhor desempenhadas pelas pessoas." (Rich and Knight, 1991)	"Inteligência Computacional é o estudo do projeto de agentes inteligentes." (Poole *et al.*, 1998) "AI... está relacionada a um desempenho inteligente de artefatos." (Nilsson, 1998)

Fonte: (RUSSEL, NORVIG, 2010)

Dentro dessas quatro categorias, pode-se observar que no primeiro bloco em "Pensando como um humano", Bellman (1978) apresenta a automatização de tarefas realizadas por humanos com a IA. Hoje se pode ver algumas inteligências artificiais que funcionam a partir de treinamentos como, por exemplo, os teclados de smartphones que, com base em tudo que é digitado, passa a recomendar palavras que são mais utilizadas pelo usuário em determinados contextos em que ela foi utilizada.

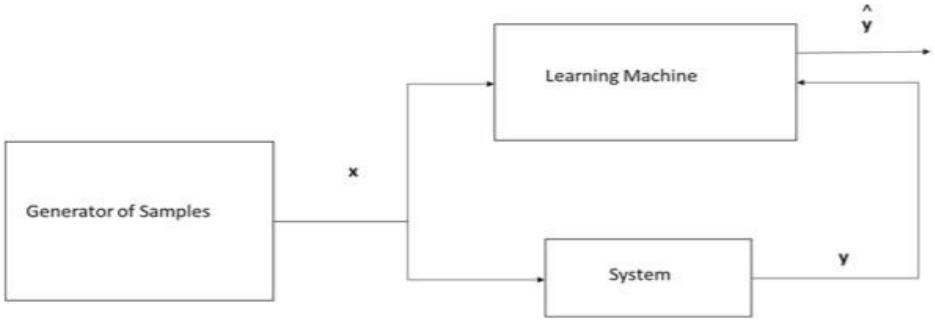
1.2. Aprendizado de Máquina

O Aprendizado de Máquina (*Machine Learning*) é uma subárea da IA que foca em desenvolvimento de algoritmos inteligentes que aprendam com seus dados e possam buscar soluções melhores com o que for aprendido.

Segundo Lampropoulos e Tsihrintzis (2015, p.32), o Aprendizado de Máquina é responsável pela maneira que o computador irá aprender o que for proposto a ele, logo, como a máquina irá aprender e se projetar para melhorias com o que for absorvido na aprendizagem, os projetistas da máquina precisam apenas cuidar do que for necessário para que ela funcione dessa maneira.

Existem três tipos de técnicas de treinamento em Aprendizado de Máquina: aprendizado supervisionado, aprendizado não-supervisionado e aprendizado por reforço. Neste projeto serão utilizadas técnicas de Processamento de Linguagem Natural com aprendizado supervisionado, como apresentado na Figura 3:

Figura 3 - Aprendizado de máquina a partir de exemplos



Fonte: Lampropoulos e Tsihrintzis (2015)

1.3. Mineração de Dados

Granatyr (2017) afirma que existem duas abordagens sobre a mineração de dados, a Estatística e a de Processamento de Linguagem Natural. Na primeira será relevante a frequência em que os termos aparecem e partir disso gerar alguma informação, ignorando a semântica e sintática do texto. Já na segunda, a interpretação sintática e semântica dos textos é mais relevante, fazendo o computador entender e aprender textos em linguagem humana. Neste projeto os dois tipos de mineração serão abordadas cada uma em um processo.

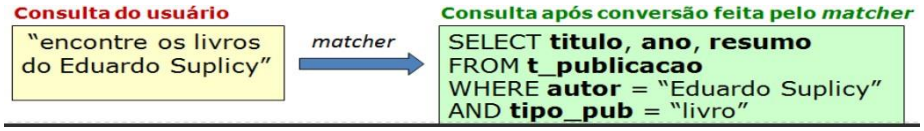
1.4. Processamento de Linguagem Natural

A partir do Aprendizado de Máquina é possível entender melhor como funciona o Processamento de Linguagem Natural, que é a interpretação sintática e semântica dos textos, fazendo o computador entender e aprender textos em linguagem humana.

Conforme Liddy (2001), Processamento da Linguagem Natural é um conjunto de técnicas computacionais que analisam e representam textos em vários níveis de análises linguísticas diferentes com o objetivo de executar tarefas, aplicações em variedades de idiomas diferentes a partir do processamento do linguajar humano.

Uma aplicação de PLN, citada por Granatyr (2017), seria uma busca em uma base da dados por voz: uma pessoa poderia falar o que procura com as próprias palavras, as quais seriam transformadas em texto livre – por meio de um algoritmo de mineração de texto e PLN, assim, seria extraído desse texto o que interessa para a busca e retornaria para pessoa a informação solicitada, como exemplificado na Figura 4:

Figura 4 - Casamento de esquemas



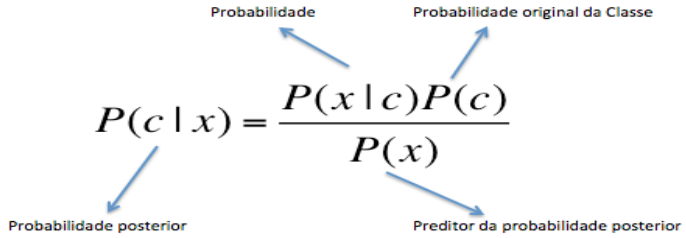
Fonte: Granatyr (2017)

1.5. Algoritmo de Naive Bayes

Para funcionar a interpretação sintática e semântica dos textos, neste projeto será aplicada a teoria de Naive Bayes com NLTK. De acordo com Granatyr (2017), o algoritmo de Naive Bayes é um algoritmo probabilístico que e pode ser usado em qualquer mineração de textos: o algoritmo faz um treinamento e aprendizado de uma base de dados de textos, retornando probabilidades de ocorrências de classes com esses dados.

Por meio da frequência das palavras e as classes às quais elas estão relacionadas, a máquina consegue entender e interpretar os textos que foram entradas para o algoritmo. Ray (2017) explica a equação base do teorema de Bayes, conforme a Figura 5:

Figura 5 - Equação Bayesiana



Fonte: https://www.vooo.pro/insights/6-passos-faceis-para-aprender-o-algoritmo-naive-bayes-com-o-codigo-em-python/

- P (c | x) é a probabilidade posterior da classe (c, alvo) dada preditor (x, atributos).
- P (c) é a probabilidade original da classe.
- P (x | c) é a probabilidade que representa a probabilidade de preditor dada a classe.
- P (x) é a probabilidade original do preditor.

Segundo também Ray (2017), o teorema de Bayes é um classificador que pode ser entendido pelo seguinte exemplo: na Figura 6 apresenta-se um conjunto de dados de treinamento com clima e o objetivo "play", parâmetro que vai "ensinar" o algoritmo a definir se em um determinado clima meteorológico é possível jogar ou não. Nessa mesma figura, percebe-se que, em seguida, os dados foram convertidos em uma tabela de frequência em que cada conclusão (objetivo) será atingida (por exemplo, tempo "nublado" tem o valor "sim" por quatro vezes). Por fim, uma última tabela foi criada com o cálculo das probabilidades de cada valor objetivo, tal como para tempo "nublado".

Figura 6 - Tabela de Probabilidade e Frequências

TEMPO	"PLAY"
Sol	Não
Nublado	Sim
Chuva	Sim
Sol	Sim
Sol	Sim
Nublado	Sim
Chuva	Não
Chuva	Não
Sol	Sim
Chuva	Sim
Sol	Não
Nublado	Sim
Nublado	Sim
Chuva	Não

Tabela de frequência		
Clima	Não	Sim
Nublado	0	4
Sol	3	2
Chuva	2	3
Total	5	9

Tabela de probabilidade				
Clima	Não	Sim		
Nublado	0	4	=4/14	0,29
Sol	3	2	=5/14	0,36
Chuva	2	3	=5/14	0,36
Total	5	9		
	=5/14	=9/14		
	0,36	0,64		

Fonte: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Em seguida, a equação bayesiana demonstrada na Figura 5 deve ser aplicada com os dados das tabelas: a classe com maior probabilidade posterior é a resolução do problema.

Problema: Os jogadores irão jogar se o tempo está ensolarado. Esta afirmação está correta?
Podemos resolver isso usando o método discutido acima de probabilidade posterior.

- P (Sim | Sol) = P (Sol | Sim) * P (Sim) / P (Sol)
- Aqui temos P (Sol | Sim) = 3/9 = 0,33, P (Sol) = 5/14 = 0,36, P (Sim) = 9/14 = 0,64
- Agora, P (Sim | Sol) = 0,33 * 0,64 / 0,36 = 0,60, que tem maior probabilidade.

1.6. Análise de Sentimento

Segundo Russel e Norvig (2010), a Análise de Sentimento é uma tarefa de classificação, que é realizada por meio de algoritmos e Aprendizado de Máquina. Nesse projeto será trabalhada com mineração de emoção, um Aprendizado de Máquina com base em um algoritmo de classificação junto com Naive Bayes, usando um algoritmo que funciona da seguinte maneira: a partir dos atributos previsores de um determinado relatório, uma base histórica de dados e uma classe supervisora que anota resultados específicos da base de dados histórica; futuros relatórios com os mesmos atributos previsores podem ser automaticamente avaliados para gerar resultados com os dados já avaliados no atributo supervisor do primeiro relatório, preenchidos manualmente pelo supervisor responsável.

Para que o algoritmo de classificação aplique uma análise de sentimento, a classificação é feita em polaridade, também chamado de valência em 3 pontos (positivo, neutro e negativo) - cada registro possui uma classe e um conjunto de atributos previsores, o objetivo é descobrir um relacionamento entre os atributos previsores e o atributo classe, o valor do atributo classe são os dados de aprendizado supervisionado.

1.7. Natural Language Toolkit (NLTK)

Para trabalhar melhor com o tratamento de textos e a análise de sentimento, foi utilizada

a NLTK, que é uma biblioteca em Python de código aberto e licença livre.

Segundo o site oficial do NLTK, ela disponibiliza bibliotecas de PLN para se trabalhar de várias maneiras diferentes, desde uma abordagem mais estatística, com bibliotecas de processamento de texto, classificação, tokenização, análise, marcação, derivação até raciocínio semântico e léxico, permitindo trabalhar com PLN.

No site oficial do NLTK é possível encontrar todo material e apoio para usar suas funcionalidades, tornando bem mais simples a curva de aprendizagem necessária para se utilizar esta plataforma. Toda a parte de treinamento e validação da base de dados desse protótipo foi desenvolvida com o NLTK junto com o teorema de Naive Bayes.

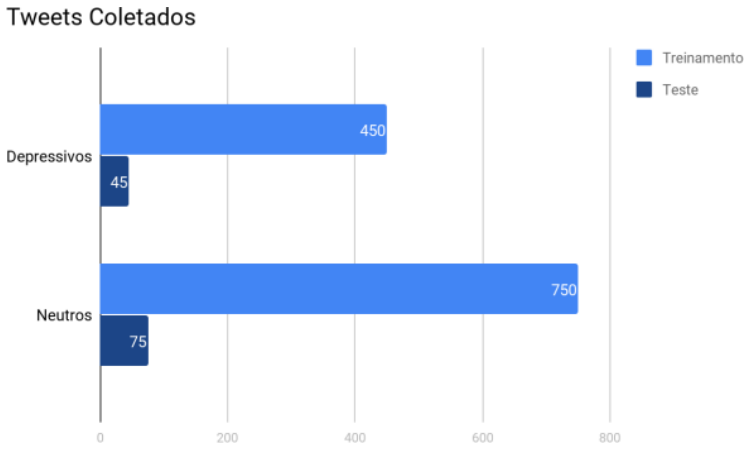
2. MATERIAIS E MÉTODOS

O projeto foi realizado por etapas de aplicação de técnicas empíricas: na primeira etapa, foi realizada a classificação dos tweets fornecidos de Big Data de terceiros, quando foram classificados um total de 1320 tweets: 825 neutros (tudo que não é depressivo) e, desses 825 tweets, 750 foram separados para treinar a o algoritmo inteligente e 75 para testes. Os 495 tweets restantes foram separados em 450 tweets depressivos para treinamento e 45 para testes. A classificação foi realizada manualmente utilizando dicionários do Python.

Em seguida, em uma segunda etapa, foi realizado o tratamento das bases removendo stopwords (palavras que não têm relevância na análise), e extração dos radicais das palavras dos tweets utilizando a biblioteca NLTK em Python. Para o treinamento do algoritmo inteligente, a extração dos radicais é importante, pois facilita o reconhecimento das palavras como por exemplo “triste”, extraindo o radical “trist”, pode-se obter as palavras entristecer, tristeza, tristonho, tristinho – tais palavras, na base de dados, são classificadas como depressivas. No entanto, isso ainda pode gerar pequenos enganos em algumas palavras!

Após o treinamento do algoritmo inteligente, foi utilizada a base de testes para comparar os erros e acertos e medir a precisão que foi adquirida com a base de treinamento. Nesse projeto foi alcançado a precisão de 75% de acerto e 25% de erros: com o algoritmo NLTK, foi criada uma lista para que fossem enviadas todas as frases classificadas de maneira errada e comparasse com o que foi classificado manualmente. Com essa lista, é possível gerar uma matriz e ver a quantidade de frases que estão classificadas erradas. O classificador marcou 15 dos 75 tweets neutros como depressivos e 15 dos 45 depressivos como neutros. A partir dessa lista foi possível observar as frases que foram classificadas erradas e aprimorar a base de dados para poder contornar esses erros.como demonstrado na Figura 7:

Figura 7 - Quantidade de Tweets Coletados



Fonte: Os Autores

3. RESULTADOS E DISCUSSÃO DA APLICAÇÃO DE PLN E ANÁLISE DE SENTIMENTO

Nesta seção apresenta-se a aplicação do processo de PLN, com base no algoritmo de Naive Bayes, no contexto de Análise de Redes Sociais, em particular o Twitter. Como aplicação das técnicas, tomou-se o tema "Depressão" para a construção das classes norteadores do processo de PLN.

3.1. Fase de Treinamento

Primeiramente foi necessário criar duas bases de dados, uma para treinamento e outra para testes. A partir de uma coleta de tweets feita por terceiros, foi realizada a classificação desses tweets para alimentar as bases – a construção da lista de classes foi feita manualmente.

A base de treinamento contém um total de 1200 tweets, com 750 neutros (tudo que não carrega uma mensagem depressiva) e 450 depressivos. A base de testes tinha o número de tweets no valor de 10% da quantidade da base de treinamento, ou seja, 120 no total, sendo 75 neutros e 45 depressivos.

Em seguida, um tratamento foi realizado nas bases: na primeira fase desse processo foi executada a remoção das "stopwords", que são basicamente palavras que não agregam valor para a análise (como por exemplo as palavras "algum", "agora", "aquele", "de", "e", "a", "ou", "que", "talvez", dentre outros). Na Figura 8, a seguir, apresenta-se o formato dos tweets e classes bem como a definição da remoção das stopwords, em seguida:

Figura 8 - Classificação de tweets em dicionário

```
import nltk

depressionteste = [('Precisando fazer levantamento de autoestima e de ego, para ver se pelo menos isso cresce na minha vida',
'depressivo'),
                   ('eu nao sei pq eu choro com tudo meu Deus me ajuda', 'depressivo'),
                   ('Se for assim todas as semanas consigo bem conciliar com os estudos', 'neutro'),
                   ('claro que sim po vc quer se formar em que? se for algo que envolva tecnologia tu pode trabalhar na minha
                   empresa vlh', 'neutro'),]

stopwordsnltk = nltk.corpus.stopwords.words('portuguese')
```

Fonte: Os autores

Em seguida, foi necessário extrair o radical das palavras, para evitar sentidos duplicados de um grupo de palavras com o mesmo radical na tabela probabilística gerada a partir do treinamento, como por exemplo a palavra “triste”: se admitir apenas o radical que seria “trist” é possível analisar tristeza, triste, tristemente, tristonho, entristecer, entristecido e etc. Na Figura 9, apresenta-se a aplicação do “stemmer” para a extração do radical das palavras:

Figura 9 - Extração de radicais

```
def aplicaStemmer(texto):
    stemmer = nltk.stem.RSLPStemmer()
    frasesStemming = []
    for (palavras, emocao) in texto:
        comStemming = [str(stemmer.stem(p)) for p in palavras.split() if
                       p not in stopwordsnltk]
        frasesStemming.append((comStemming, emocao))
    return frasesStemming

frasesStemming = aplicaStemmer(depressionteste)
```

Fonte: Os autores

Para melhor visualização das modificações feitas, foi implementada uma função para retornar todas as palavras da base de treinamento já tratadas. Uma outra função foi implementada para visualizar a frequência de que cada radical aparece na base de dados: a partir dessa função, foram exibidos todos os radicais sem repetições. Por fim, foi criada a função que gera a tabela probabilística do Naive Bayes, que é o treinamento propriamente dito, conforme apresentado a seguir na Figura 10:

Figura 10 - Treinamento da IA

```
def buscaPalavras(frases):
    todasPalavras = []
    for (palavras, emocao) in frases:
        todasPalavras.extend(palavras)
    return todasPalavras

palavras = buscaPalavras(frasesStemming)

def buscaFrequencia(palavras):
    palavras = nltk.FreqDist(palavras)
    return palavras

frequencia = buscaFrequencia(palavras)

def buscaPalavrasUnicas(frequencia):
    freq = frequencia.keys()
    return freq

palavrasUnicas = buscaPalavrasUnicas(frequencia)

def extratorpalavras(documento):
    doc = set(documento)
    caracteristicas = {}
    for palavras in palavrasUnicas:
        caracteristicas['%s' % palavras] = palavras in doc
    return caracteristicas
```

Fonte: Os autores

3.2. Fase de Validação

Após o treinamento, é necessário validar o algoritmo. Nesse processo foi realizada uma comparação entre os resultados apresentados pelo algoritmo e os dados reais da base testada, criando assim uma lista contendo os textos classificados de forma equivocada pelo algoritmo, demonstrado na Figura 11. Com essa lista criada e a função de classificação *accuracy* do NLTK, é possível apresentar o resultado de precisão do algoritmo, que, no caso, foi de 75%.

Figura 11 - Lista de erros

```
erros = []

for (frase, classe) in depressionteste:
    resultado = classificador.classify(frase)
    if resultado != classe:
        erros.append((classe, resultado, frase))

nltk.classify.accuracy(classificador, basecompletateste)
```

Fonte: Os autores

Assim, para poder visualizar os resultados de acertos e erros do algoritmo, foi criada uma matriz, comparando dois tipos de dados: os resultados esperados, pertencentes à base de teste, previamente conhecidos, e os resultados previstos, realizados pelo classificador, mostrados na na Figura 12, a seguir. A matriz pode ser observada mais adiante, em que os resultados acertados pelo classificador se encontram entre o sinal '<>' na Figura 13.

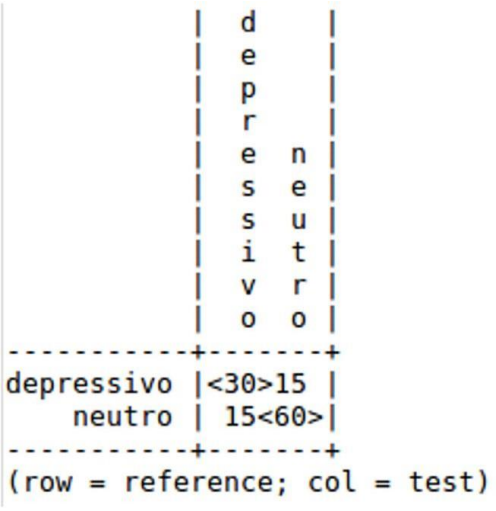
Figura 12 - Criação da matriz de erros e acertos

```
from nltk.metrics import ConfusionMatrix
esperado = []
previsto = []
for (frase, classe) in depressionteste:
    resultado = classificador.classify(frase)
    previsto.append(resultado)
    esperado.append(classe)

matriz = ConfusionMatrix(esperado, previsto)
print(matriz)
```

Fonte: Os autores

Figura 13 - Matriz de erros e acertos do classificador



Fonte: Os Autores

O algoritmo treina a base de acordo com o exemplo abaixo de aprendizagem supervisionada, no qual são extraídas todas as palavras e classes da base, relacionada cada palavra da frase com a respectiva classe, gerando, assim, o classificador. Em seguida a base de teste é analisada pelo classificador.

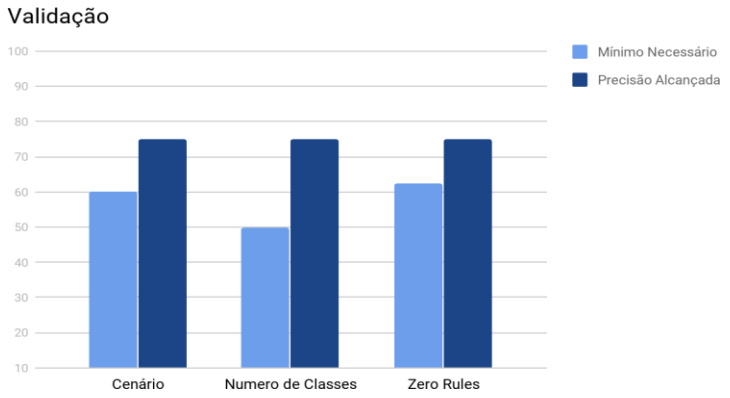
Com o teste do algoritmo feito, foi realizada uma avaliação para decidir se o algoritmo é utilizável em alguma aplicação ou não, para isso ele deve passar em 3 condições: cenário, número de classes e Zero Rules.

Na condição de cenário deve-se levar em conta em que situação está sendo implementado o algoritmo. Como o cenário do presente trabalho se apresenta como acadêmico, um resultado superior a 60% de acerto do algoritmo já o valida nesse requisito.

Na próximo condição, deve-se considerar o número de classes, que, no caso, foram 2 (depressivo e neutro), na quais são divididas por 100%, assim, o algoritmo precisaria possuir uma porcentagem de acerto maior que 50%, caso contrário, se apresentaria irrelevante, pois um gerador de resultados randômicos teria mais credibilidade do que a inteligência.

E, na última condição, o Zero Rules, foi obtido o número da quantidade de frases da maior classe (neutro) e dividido pelo número total de frases da base de dados, multiplicando em seguida por 100, para obter a percentagem. O algoritmo deveria apresentar um percentual acima de 62,5% de acerto para ser validado nessa condição, fato que também ocorreu. Nesse caso, se a inteligência não ultrapassasse o percentual de Zero Rules, o resultado da análise sempre seria a da classe de maior volume de dados, o que pode ser observado na Figura 14 a seguir:

Figura 14 - Comparação com as Condições de Validação



Fonte: Os Autores

CONSIDERAÇÕES FINAIS

É possível concluir ao final desse trabalho que quanto maior a base de dados de treinamento, quanto mais amplo for o vocabulário das frases contidas no treinamento, melhor será a precisão do classificador. O ideal seria que o classificador acrescentasse à sua base de dados todas as frases novas que forem analisadas para expandir automaticamente o vocabulário de sua base e consequentemente a precisão de acertos.

Na implementação do algoritmo inteligente, em todas as etapas, foi utilizada a biblioteca NLTK do Python, seja na parte de preparação da base de dados, que engloba as fases de remoção de stopwords, palavras que não agregam peso à classificação, e extração de radical, para melhor aproveitamento do vocabulário, seja no treinamento da inteligência, propriamente dito. Em ambas as etapas, a biblioteca mostrou-se bem satisfatória na implementação do algoritmo, pois detinha diversas funções que facilitaram todo o processo. Algo que se tornou um tanto quanto trabalhoso foi a classificação da base de dados para o treinamento da inteligência, por ter sido toda feita manualmente, etapa essa que a biblioteca não apresentou ferramentas que auxiliassem no procedimento.

Vale ressaltar, também, que existem alguns classificadores dessa espécie, porém em sua maioria se encontram na língua inglesa, portanto, escasso na língua portuguesa, sendo um diferencial a mais para o algoritmo inteligente do presente estudo.

Com uma precisão maior da base de dados em reconhecimento de publicações depressivas em redes sociais, é possível prevenir suicídios e elaborar planos de tratamentos sociais para as pessoas que estiverem publicando textos depressivos com frequência, direcionando coisas mais positivas para a rede social dessa pessoa para que ela valorize mais a própria vida, ou ainda profissionais que possuem conhecimento de como agir em situações desse tipo, tenham a oportunidade de conversar com essa pessoa por meio da sua rede social e demonstrar que pessoas se importam com ela.

Futuramente, o ideal seria que o próprio classificador pudesse alimentar a sua base de dados com novas frases, classificadas por ele mesmo, para melhorar a precisão de acerto.

REFERÊNCIAS

GRANATYR, Jones. Mineração de Emoção em Textos com Python e NLTK. **Udemy**, Disponível em: <https://www.udemy.com/mineracao-de-emocao-em-textos-com-python-e-nltk/learn/v4/overview>. Acesso em 16 jun. 2017

LAMPROPOULOS, Aristomenis S.; TSIHRINTZIS, George A. **Machine Learning Paradigms: Applications in Recommender Systems**. Springer, 2015.

LIDDY, E. ***Natural Language Processing***. New York: Marcel Decker, Inc., 2001.

PRADO, Jean. Por dentro da Stilingue, ferramenta que usa inteligência artificial para resumir a internet. **Tecnoblog**, Disponivel em: <https://tecnoblog.net/189060/stilingue-resumo-internet-ia/>. Acesso em 02 nov. 2017

RAY, Sunil. 6 passos fáceis para aprender o algoritmo naive bayes com o código em python. **VOOO – INSIGHTS**, Disponível em: <https://www.vooo.pro/insights/6-passos-faceis-para-aprender-o-algoritmo-naive-bayes-com-o-codigo-em-python/>. Acesso em: 22 set. 2017

RUSSEL, Stuart; NORVIG, Peter. ***Inteligência Artificial***. Rio de Janeiro: Campus Elsevier, 2010.

Site Oficial da Organização Pan-Americana da Saúde, Disponivel em: <http://www.paho.org/bra/index.php?option=com_content&view=article&id=5354:aumenta-o-numero-de-pessoas-com-depressao-no-mundo&Itemid=839>. Acesso em 19 out. 2017

Site Oficial do NLTK, Disponível em: <http://www.nltk.org/>. Acesso em 17 ago. 2017.