UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação

Análise e classificação de comentários da pesquisa "Empresas humanizadas do Brasil" por meio de algoritmos de aprendizado de máquina

Rosival Rodrigues do Nascimento Neto

São Carlos - SP

Análise e classificação de comentários da pesquisa "Empresas humanizadas do Brasil" por meio de algoritmos de aprendizado de máquina

Rosival Rodrigues do Nascimento Neto

Orientador: André Carlos Ponce de Leon Ferreira de Carvalho

Monografia referente ao projeto de conclusão de curso dentro do escopo da disciplina SSC0670 Projeto de Formatura I do Departamento de Sistemas de Computação do Instituto de Ciências Matemáticas e de Computação – ICMC-USP para obtenção do título de Engenheiro de Computação.

Área de Concentração: Inteligência Computacional

USP – São Carlos

2019

“Não tente ser uma pessoa de sucesso. Em vez disso, seja uma pessoa de valor.”

(Albert Einstein)

Dedicatória

Dedico este trabalho à minha família e, especialmente, ao meu primo e irmão Gumercindo Silveira Netto, ou simplesmente Guri, que sempre me apoiou, acreditou no meu potencial e esteve comigo nos momentos fáceis ou difíceis, contribuindo, imensamente, para essa conquista.

Agradecimentos

Agradeço aos meus pais, Gilvanio e Mábia, que me ensinaram, desde muito cedo, através do exemplo, o valor que a educação tem na vida de uma pessoa e que só através dela é possível mudar todo um ecossistema para melhor.

Agradeço à minha avó, Marieta, exemplo de mulher, simplicidade, amor pela família e trabalho duro, que desde sempre me apoiou e comemorou, junto, todas as minhas conquistas.

A minha irmãzinha, Vivi, por ter sido todos esses seus anos a minha inspiração, o meu objetivo e a minha vontade de querer mais e melhor para todos.

À Universidade de São Paulo e principalmente a cidade de São Carlos, ambiente acolhedor e que, sem sombra de dúvidas, mudou a minha forma de pensar e agir enquanto ente social.

Ao meu orientador, André, carinhosamente chamado de Andrezão, que acreditou em mim desde o primeiro contato para realizar este trabalho e que com toda sua humildade vem ajudando diversos alunos a seguirem no mundo do empreendedorismo.

A todos os meus amigos que foram fator chave para o meu desenvolvimento e que foram suportes durante todos esses anos de graduação. Um agradecimento especial ao time Virou Passeio, irmãos que levarei no coração para sempre, à República Espírito de Porco (R.E.P.) que nos anos finais da minha graduação me acolheram como membro da família. E por fim, mas não menos importante, ao time Enactus USP-São Carlos, que me possibilitou desenvolver habilidades que a graduação sozinha não me permitiria.

Resumo

A massiva quantidade de dados produzida nos dias de hoje, por conta do advento da internet e dos sistemas computacionais de alto processamento, já não é mais capaz de ser manipulada manualmente pelos humanos. Por isso, o campo de estudo de aprendizado de máquina vem ganhando cada vez mais espaço e relevância no contexto da interação humano-máquina. Através de métodos e modelos matemáticos surgem diferentes algoritmos que visam obter informações estratégicas a partir desses dados.

Este trabalho propõe comparar e selecionar um dos modelos de análise e classificação de opiniões em comentários textuais qualitativos das *Surveys multistakeholder* da 1ª edição da pesquisa Empresas Humanizadas do Brasil. Para isto, utiliza-se de uma base proprietária de dados (para treinamento e teste) e dos algoritmos de aprendizado de máquina Naive Bayes, com uso da técnica N-gramas, e redes neurais recorrentes LSTM e LSTM bidirecional.

Palavras-chave: Aprendizado de Máquina. Naive Bayes. Redes Neurais Recorrentes.

Sumário

Lista de Tabelas

Lista de Figuras

CAPÍTULO 1: INTRODUÇÃO

1.1. Contextualização e Motivação

É fato notável que a cada dia produz-se uma quantidade maior de dados, carregados de informações, cuja a capacidade humana, limitada, já não é mais capaz de tratar. Por isso que, no mundo moderno, com auxílio da internet e dos sistemas computacionais, gerenciar dados de clientes, produtos e serviços, extraindo a maior quantidade possível de vantagem dessas informações, é uma estratégia chave para qualquer negócio. Assim, da otimização de produtos e serviços até o próprio relacionamento negócio-cliente, torna-se tarefa crucial conseguir extrair opiniões e/ou emoções a partir dessas fontes de dados.

Para Alves et al. (2014), a análise de comentários expressos nessas fontes de dados requer muito esforço quando tratada de forma manual, principalmente devido ao grande volume de dados gerados. Por isso, surgiram novas tecnologias que tornam possível obter novos conhecimentos a partir dessa grande massa de informações. Dentre elas, aprendizado de máquina (ML, do inglês, *Machine Learning*) é um dos campos do conhecimento que mais crescem, no intuito de trazer luz a solução desses problemas, por meio da intersecção da ciência da computação, da matemática aplicada e da estatística. O progresso recente na área de ML foi impulsionado pelo desenvolvimento de novos algoritmos e teoria de aprendizado e pela explosão contínua na disponibilidade de dados on-line e computação de baixo custo (M. I. JORDAN; T. M. MITCHELL, 2015). Sendo que, uma tendência mais recente é a análise de sentimentos ou mineração de opiniões, que busca identificar a opinião por trás de um texto, possibilitando obter *feedbacks* emocionais sobre produtos, serviços, organizações, figuras públicas e outros fontes de informação. Análise de sentimentos também é comumente conhecida por vários outras termos, tais como: extração de opinião, mineração de sentimento, análise de subjetividade, análise afetiva, análise de emoções e mineração de opinião (LIU, 2012).

Ainda segundo Liu (2012), uma opinião é formada por dois elementos principais: um alvo e um sentimento expresso em relação ao alvo. O alvo pode ser definido como uma entidade, ou seja, representa uma pessoa, marca de um determinado produto, ou qualquer

sujeito que esteja relacionado à opinião. Já o sentimento é a opinião ou emoção expressa em relação ao alvo. Por exemplo: “eu amo a empresa onde eu trabalho”. Nesta frase, a parte “a empresa onde eu trabalho” representa o alvo, e a palavra “amo” representa o sentimento expresso em relação a empresa.

Entretanto, em massivas quantidades de dados, inúmeros desafios surgem na análise destes sentimentos ou opiniões, uma vez que podem existir desde erros provenientes dos próprios textos (como os ortográficos ou sintáticos) até erros provenientes dos próprios dados (como a estrutura de dados ou dados que não fazem parte do contexto da aplicação). Além disso, esses desafios se acentuam quando as análises textuais são na língua portuguesa, pois segundo Inoki (1992), a variação dos tempos e formas verbais, regras de concordância e flexões verbais, são desafios comumente encontrados no idioma português.

A análise de opiniões utiliza diversas técnicas e campos da computação moderna de maneira integrada, desde estatística, passando por mineração de dados, até o processamento de linguagem natural (*NLP*, do inglês *Natural Language Processing)*. Abordagens essas que, se baseiam no ML supervisionado, no qual algoritmos são utilizados para induzir modelos preditivos por meio da observação de um conjunto de objetos rotulados (VON LUXBURG E SCHÖLKOPF, 2008).

Neste contexto, análises de opiniões podem ser utilizadas em pesquisas do tipo *Survey* que, segundo Figueiredo (2004), destina-se tanto a obtenção de informações quanto à prevalência, distribuição e inter-relação de variáveis no âmbito de uma população e, nos dias de hoje, coletam massivas informações textuais, qualitativas, para análises, por conta da conectividade que facilita a pulverização, em uma população, desse tipo de pesquisa.

Inspirado nisso, este trabalho explora o uso da análise de opiniões aplicada a pesquisa, do tipo *Survey*, destinada aos *multistakeholders* (diferentes atores de uma organização, como diretores, colaboradores, investidores, sociedade e afins) de diferentes empresas, participantes da 1ª edição da Pesquisa Empresas Humanizadas do Brasil, na qual o autor deste trabalho teve a oportunidade de atuar, em 2018, sob a liderança do pesquisador e doutorando pelo curso de Engenharia de Produção, Pedro Paro (Escola de Engenharia de São Carlos - USP), com o objetivo de ajudar a elevar a humanidade por meio da inspiração

de negócios mais conscientes, humanizados, sustentáveis e inovadores. Durante esse período, o autor desenvolveu uma ferramenta de organização e pré-processamento dos dados qualitativos textuais, das *Surveys*, para serem lidos pelos membros da pesquisa e classificados em: comentários positivos e negativos. Dessa forma, enxergou-se a oportunidade de aproveitar a ferramenta desenvolvida e o conhecimento adquirido no curso de graduação para desenvolver este trabalho.

Este projeto tem como principais contribuições: (i) automatizar o processo de análise das opiniões e diminuir o tempo gasto da equipe para classificação dos comentários qualitativos textuais; (ii) construir a estrutura de base para o desenvolvimento futuro de uma inteligência artificial, agregando valor a pesquisa no contexto de inovação; e (iii) comparar dois algoritmos de ML, muito diferentes, para análise de opiniões em *Surveys multistakeholders.*

1.2. Objetivos

Este trabalho tem como objetivo desenvolver um modelo de análise e classificação de opiniões, a partir de comentários qualitativos textuais das *Surveys multistakeholders* da 1ª edição da Pesquisa Empresas Humanizadas, como também fazer um comparativo entre diferentes algoritmos de ML para tal finalidade. O estado da arte para análise e classificação de opiniões emprega diversas ferramentas e métodos que se baseiam em diferentes estratégias, tais como ML, NLP, mineração de dados, método de Naive Bayes, redes neurais ou mesmo a combinação de tais técnicas.

O modelo proposto neste trabalho será baseado em algoritmos de ML supervisionado, para análise e classificação de opiniões, a partir de um conjunto de treinamento proprietário. Para tal, irá se fazer o uso e comparação das técnicas de Naive Bayes com N-gramas e das redes neurais LSTM (do inglês, *Long Short Term Memory*), visando escolher a arquitetura que retorna os melhores resultados para o conjunto de dados utilizados nesta proposta.

1.3. Organização do Trabalho

No Capítulo 2 são apresentadas as técnicas, métodos e modelos utilizados no desenvolvimento deste projeto, bem como a revisão da terminologia básica utilizada. A seguir, no Capítulo 3, descrito o projeto desenvolvido, todos os seus procedimentos, implementações e resultados obtidos. Finalmente, no Capítulo 4, são apontadas as principais contribuições de projeto, os trabalhos futuros que podem utilizá-lo ou melhorá-lo, assim como sua relação com o curso de graduação.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

Neste capítulo são apresentados os principais conceitos, terminologias abordadas na literatura, além das abordagens e técnicas no que diz respeito a análise e classificação de opiniões textuais com o uso de ML. É apresentada a fundamentação teórica pertinente no que diz respeito aos modelos necessários para o desenvolvimento da aplicação de pré-processamento de dados e NLP. Além disso, discorre-se também sobre as arquiteturas e algoritmos de ML pertinentes à aplicação prática deste trabalho.

2.2. Pré-processamento de dados

Considerando o grande volume de dados disponível em diversas aplicações, com frequência os conjuntos de dados não possuirão uma qualidade boa o suficiente para a extração de conhecimento novo, útil e relevante por algoritmos de ML. As principais causas de baixa qualidade de dados incluem a ocorrência de atributos irrelevantes, valores ausentes ou redundantes (PADILHA, V. A; CARVALHO, A. C. P. L. F, 2017).

Sendo assim, o primeiro passo na elaboração de um modelo de análise e classificação de opiniões é fazer o tratamento da base de dados, ou seja, remover inconsistências, como registros que não estão no formato correto, que não estão classificados ou que não possuam comentários. Além disso, tratar os dados significa também colocá-los em um formato no qual seja mais fácil obter informações.

Uma das técnicas muito utilizadas no pré-processamento de textos (que inclui a análise e classificação de opiniões) é a remoção das chamadas “palavras de parada” (do inglês, *stopwords*), que são palavras removidas antes ou após a aplicação de técnicas de NLP, por serem consideradas irrelevantes no contexto trabalhado. Geralmente é um conjunto de preposições, artigos, alguns advérbios e alguns verbos, como demonstra a Figura 1.

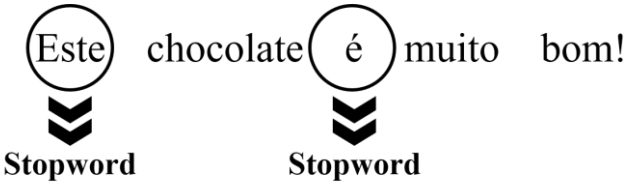


Figura 1 - Exemplo ilustrativo de stopwords. Elaborado pelo autor.

Outra técnica muito comum é o *stemming*, que consiste em reduzir as palavras flexionadas, ou conjugadas, em uma língua para a sua raiz, ou seu radical. O radical de uma palavra é a menor parte da mesma que contém seu significado léxico, sem os afixos ou flexionais, ou seja, é um morfema básico (que mostra o sentido básico da palavra), vide Figura 2.

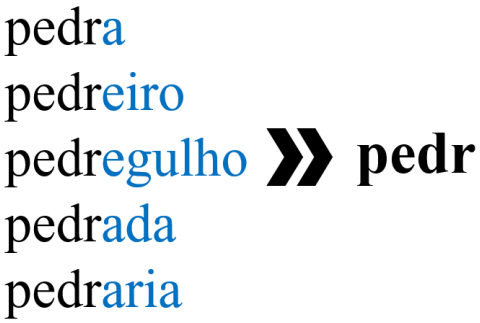


Figura 2 - Exemplo ilustrativo de raiz ou radical de uma palavra. Adaptado Motta A. (2010).

Assim, através das técnicas de pré-processamento obtêm-se um vocabulário menor de palavras no modelo, elimina-se as redundâncias ocasionadas por palavras muito similares e tem-se um ganho de tempo computacional.

2.3. Processamento de Linguagem Natural (NLP)

O NLP, se refere ao conjunto de técnicas computacionais que, combinadas com informações linguísticas, permitem que computadores representem e utilizem conhecimentos expressados em frases de linguagem natural (BARROSO, Y. M, 2016). Ou seja, o objetivo da NLP é ajudar os computadores no entendimento, interpretação e manipulação da informação de texto (como classificação e identificação de idioma) e áudios. Essa técnica pode ser fragmentada em tarefas mais simples como Tokenização e identificação de classes gramaticais, ou em mais complexas como *Semantic Role Labeling* e *Hedge Detection* (CRESTANA, C. E. M, 2010). A seguir são introduzidos dois dos métodos de NLP, o N-Gramas e o Term Frequency - Inverse Document Frequency (TF-IDF).

2.3.1. N-Gramas

A abordagem para construção do dicionário de palavras a partir de um ou mais itens lexicais agrupados é conhecida como N-gramas. Esta abordagem consiste em uma subsequência de “n” elementos em uma sequência maior, os quais são definidos de acordo com a quantidade de elementos que os compõem. Geralmente se trabalha com 3 tipos de N-gramas:

- Unigrama (Figura 3), que é basicamente cada uma das palavras que compõem uma sentença ou frase, muito utilizado em trabalhos de classificação em tópicos e em análises de sentimentos. Muito confundido com os *tokens* em artigos ou publicações onlines.

- Bigrama (Figura 4), que é uma concatenação de cada 2 palavras que compõe o texto, sendo a ordem um fator muito importante. São utilizados em análises de sentimentos para capturar combinações negativas de palavras.

- Trigrama (Figura 5), é uma composição de 3 palavras, pouco utilizado nas atividades de classificação em tópicos ou análises de sentimentos por conta da sua baixa performance em relação aos outros tipos supracitados.

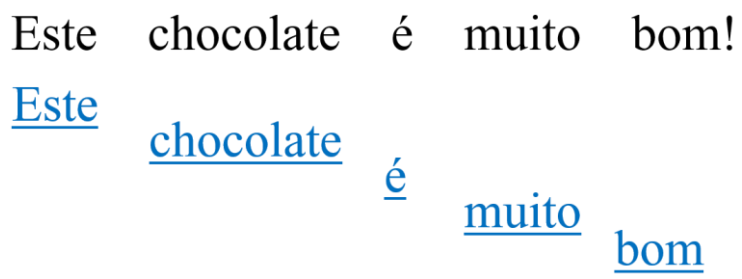


Figura 3 - Exemplo ilustrativo do tipo N-grama: unigrama. O sinal de pontuação não é considerado não tem nenhum valor semântico para a aplicação. Elaborado pelo autor.

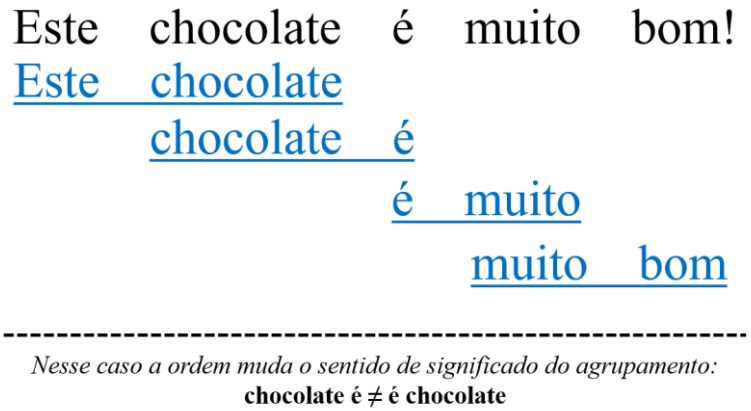


Figura 4 - Exemplo ilustrativo do tipo N-grama: bigrama. O sinal de pontuação não é considerado não tem nenhum valor semântico para a aplicação. Elaborado pelo autor.

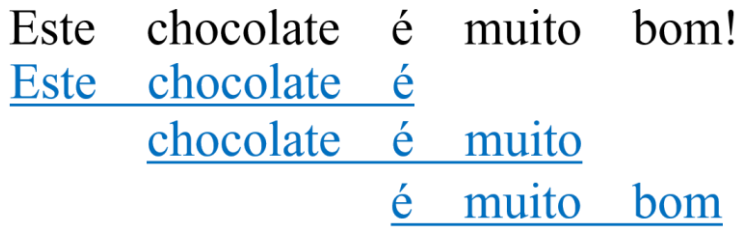


Figura 5 - Exemplo ilustrativo do tipo N-grama: trigrama. O sinal de pontuação não é considerado não tem nenhum valor semântico para a aplicação. Elaborado pelo autor.

+

2.3.2. Term Frequency - Inverse Document Frequency (TF-IDF)

Ao considerarmos uma sentença ou um documento, é possível observar a ordem das letras e das palavras, bem como o número de vezes em que cada palavra se repete no texto. Uma abordagem muito comum para extração de *features* de documentos e sentenças é realizar a contagem de termos (Figura 6), ou seja, contar a quantidade de aparições de um mesmo n-grama, pois as palavras mais comuns podem ditar o contexto de um documento. Entretanto, a contagem de termos dá muita relevância para frases ou documentos que tem mais palavras, ficando com um peso desbalanceado para muitos casos.

1. Este papo estava muito bom
2. Este chocolate era muito bom
3. Este comentário tem chocolate, chocolate, chocolate

	Este	papo	estava	muito	bom	chocolate	era	comentário	tem
1	1	1	1	1	1	0	0	0	0
2	1	0	0	1	1	1	1	0	0
3	1	0	0	0	0	3	0	1	1

Figura 6 - Exemplo ilustrativo da contagem de termos para extração de features de documentos. Elaborado pelo autor.

Uma proposta para mitigar o problema citado é utilizar o *Term Frequency* (TF) ou a frequência dos termos (Figura 7), ou seja, medir a probabilidade de uma determinada palavra aparecer dentro de uma sentença ou documento. O TF é a medida da frequência do termo t_j no documento d_i. A ideia básica, segundo Aranha (2007), é de que os termos que mais aparecem possuem maior relevância/peso do que aqueles que aparecem com menos frequência no documento. Assim sendo, atribui-se a a_{ij} o valor $TF(t_j, d_i)$, como na Equação (1).

$$a_{ij} = TF(t_j, d_i) \tag{1}$$

1. Este papo estava muito bom
2. Este chocolate era muito bom
3. Este comentário tem chocolate, chocolate, chocolate

	Este	papo	estava	muito	bom	chocolate	era	comentário	tem
1	0,2	0,2	0,2	0,2	0,2	0	0	0	0
2	0,2	0	0	0,2	0,2	0,2	0,2	0	0
3	0,16	0	0	0	0	0,5	0	0,16	0,16

Figura 7 - Exemplo ilustrativo do Term Frequency (TF) para extração de features de documentos. Elaborado pelo autor.

Apesar de TF ser uma métrica muito boa, ela possui a limitação no que se refere aos termos que aparecem muitas vezes e que não possuem a informação ou o significado daquele documento (como as *stopwords*) uma vez em que estes possuirão uma frequência relativa muito alta.

Nesse sentido, segundo Aranha (2007), *Inverse Document Frequency* (IDF) é uma medida que varia inversamente ao número de documentos que contém a palavra t_j, c, em um conjunto de documentos N. Logo, essa medida, representada na Equação (2) pode ser utilizada para dar um peso menor ao problema.

$$IDF = log \frac{N}{c} \tag{2}$$

Dessa forma, surge uma métrica mais robusta que é o TF-IDF (Figura 8), oriundo de modelos de análise discriminante estatística baseada em conceitos Bayesianos, que, em linhas gerais, procura achar as palavras que mais discriminam o conjunto do documento analisado. Durante o processo de aplicação do índice no documento, são atribuídos pesos as palavras, baseado em suas frequências, sendo que, o inverso da frequência em documentos, que dá peso as palavras raras, como pode ser visto na Equação (3).

$$a_{ij} = TFIDF(t_j, d_i) = TF(t_j, d_i) \times log\frac{N}{c} \quad (3)$$

1. Este papo estava muito bom
2. Este chocolate era muito bom
3. Este comentário tem chocolate, chocolate, chocolate

	Este	papo	estava	Muito	bom	chocolate	era	comentário	tem
1	0	0,15	0,15	0,06	0,2	0	0	0	0
2	0	0	0	0,06	0,2	0,06	0,15	0	0
3	0	0	0	0	0	0,18	0	0,15	0,15

Figura 8 - Exemplo ilustrativo do TF-IDF para extração de features de documentos. Elaborado pelo autor.

2.4. Aprendizado de máquina supervisionado

Em ML supervisionado, algoritmos são utilizados para induzir modelos preditivos por meio da observação de um conjunto de objetos rotulados (VON LUXBURG E SCHÖLKOPF, 2008), normalmente chamado de conjunto de treinamento. Os rótulos contidos em tal conjunto correspondem a classes ou valores obtidos por alguma função desconhecida. Desse modo, um algoritmo de classificação buscará produzir um classificador capaz de generalizar as informações contidas no conjunto de treinamento, com a finalidade de classificar, posteriormente, objetos cujo rótulo seja desconhecido (PADILHA, V. A; CARVALHO, A. C. P. L. F, 2017).

2.5. Técnicas de análise e classificação de opiniões

A análise e classificação de opiniões, ou análise de sentimentos, é o uso de algoritmos de ML e processamento natural de linguagem para extrair *insights* ou informações valiosas,

automaticamente, das opiniões de pessoas em relação a determinado produto, serviço, organização ou pessoa pública. Sendo que essa análise, classificação e extração de significado pode revelar informações de altíssimo valor estratégico para modelos de negócios e/ou organizações.

Essa análise e classificação em comentários textuais qualitativos pode ser feita de várias formas e utilizar uma série de algoritmos diferentes para o mesmo fim, dentre os quais vale destacar as árvores de decisão, modelos probabilísticos (Naïve Bayes, Redes Bayesianas, Máxima Entropia), lineares (Redes Neurais, e SVM), e modelos baseados em regras (WILSON, WIEBE E HOFFMANN, 2009).

2.5.1. Algoritmo Naive Bayes

Naive Bayes é um método de classificação muito simples, mas poderoso (PADILHA, V. A; CARVALHO, A. C. P. L. F, 2017). Consiste de um classificador probabilístico com base no teorema de Bayes com forte suposição independência entre as características, ou seja, assume que a presença de uma determinada *feature* - características que descrevem um objeto e, também, entradas dos algoritmos de ML - não tem relação com outras, por isso Naive. Esse método utiliza dados de treino para formar um modelo probabilístico baseado na evidência das *features* nos dados.

O algoritmo de Naive Bayes, utiliza a Equação 4 que consiste em encontrar uma probabilidade *a posteriori* (ou condicionada) de A condicional a B, pela probabilidade *a posteriori* de B condicional a A e pelas probabilidades *a priori* de A e B.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{4}$$

Naive Bayes é recomendado como uma das melhores alternativas para problemas de análises de sentimentos e classificação de textos, quando a correlação entre os fatores não é extremamente importante. Além disso, o método tem uma grande aplicação em previsões de

tempo real, por possuir uma velocidade relativamente alta e precisar de poucos dados para realizar classificações.

2.5.2. Redes Neurais Recorrentes (RNR)

As redes neurais recorrentes são redes neurais capazes de processar dados em sequência e têm como propriedade a habilidade de usar informação contextual ao mapear sequências de entrada e saída. Por isso, é amplamente adotada em modelagem de linguagens e em tarefas que envolvem o NLP.

Esse tipo de rede opera em *loops* que fornecem à rede um *feedback* (retroalimentação) constante a respeito do estado das entradas processadas anteriormente, o que permite com que a RNR persista em memória os estados dessas entradas, influenciando o que é obtido na saída.

Logo, a decisão de uma rede recorrente alcançada na etapa de tempo *t-1* afeta a decisão que alcançará um momento mais tarde na etapa de tempo *t*. Assim, as redes recorrentes têm duas fontes de entrada, o presente e o passado recente, que se combinam para determinar como respondem a novos dados, da mesma forma que os humanos fazem na vida.

Em uma RNR a informação da camada oculta é adicionada no período anterior, como pode ser visto na equação 5. Além disso, os parâmetros que fazem a transição da informação entre as camadas ocultas de diferentes períodos são sempre os mesmos. Isso mostra que redes neurais recorrentes compartilham parâmetros através do tempo.

$$h_t = \phi(b_h + xU_x + h_{t-1}W_X) \quad (5)$$

Como ilustrado na Figura 9, as camadas ocultas ***h*** carregam ao longo do tempo as *features* ***x*** combinadas a matriz de peso ***U***. As matrizes de peso ***U, V, W*** são usadas repetidamente pela rede, sendo que todas as camadas ocultas ***h*** compartilham o mesmo peso

W. A influência dos parâmetros computados inicialmente em uma RNR sofre decaimento expressivo ao longo dos ciclos da rede. Isto é conhecido como *vanishing gradient problem.*

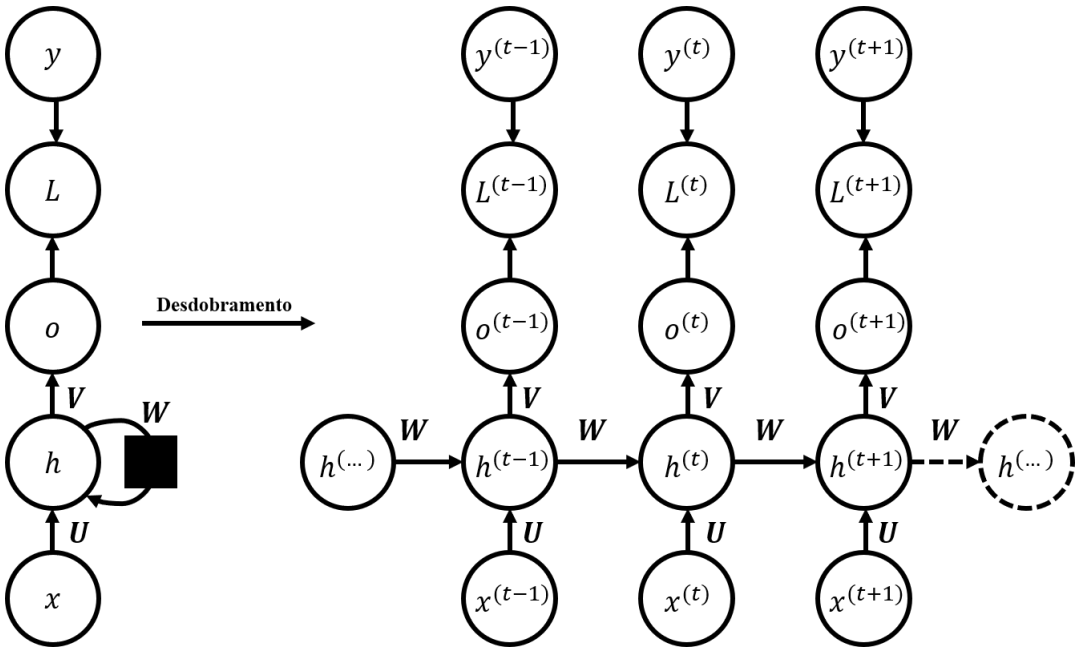


Figura 9 - Representação de uma rede neural recorrente. Adaptado de Luís Fred.

Todos os estados ocultos que ocorrem nos tempos ***t+1*** são sensíveis as novas entradas ***x***, ou seja, as entradas inicialmente computadas perdem a influência ao longo dos ciclos da RNR, acarretando no *vanishing gradient*. Em outras palavras, as camadas ocultas ***h*** são sensíveis às entradas subsequentes, fazendo com que a rede "esqueça" dos parâmetros que aprendeu com as entradas iniciais, como ilustra a Figura 10 na qual a camada ***h*** perde a influência da entrada 1 (marcada na cor preta) ao longo do ciclo da RNR, e no ciclo 7 já não influencia mais a entrada ***x***.

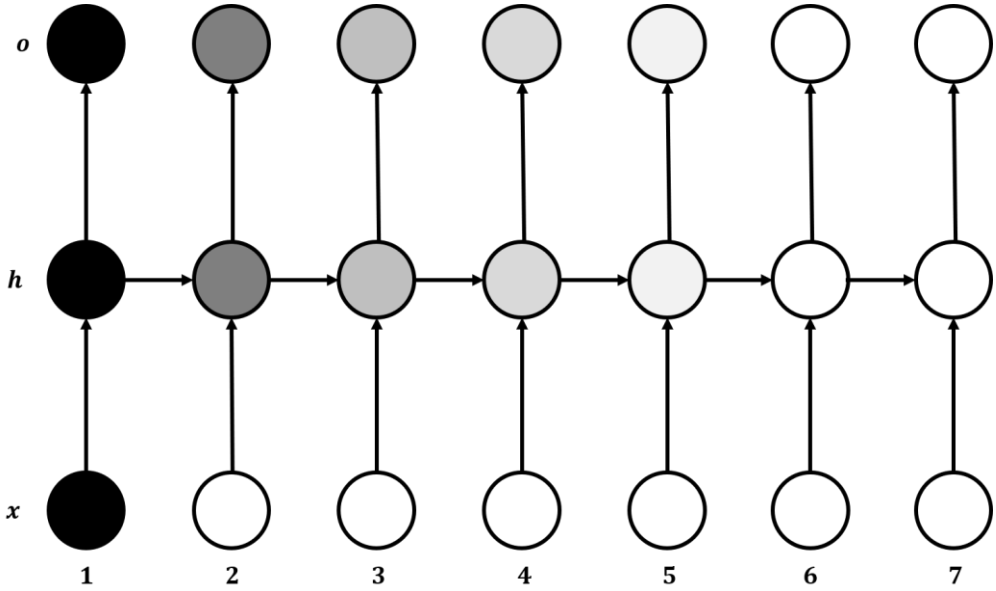


Figura 10 - Vanish Gradient Problem. Adaptado de Luís Fred.

2.5.3. Arquitetura *Long Short-Term Memory* (LSTM)

A arquitetura LSTM foi inicialmente proposta por Hochreiter; Schmidhuber (1997) com o intuito de resolver o problema de persistência da informação de longo prazo nas RNR, o *vanishing gradient*. Para isso, a arquitetura utiliza um mecanismo específico em suas camadas ocultas, denominado células de memória, que calculam os pesos que os conectam de forma a evitar o problema supracitado. Dessa forma, a LSTM tem a capacidade de lembrar das informações que armazenou mesmo depois de várias iterações recorrentes, mas também de esquecer o estado anterior quando a informação não é mais necessária.

Uma célula LSTM é composta por três portões que controlam diferentes comportamentos: portão de entrada (*input gate*), portão de esquecimento (*forget gate*) e portão de saída (*output gate*). Todos os portões têm uma *sigmoid* (σ) como função de linearidade para controlar o fluxo de informações dentro da célula (BISPO T. D, 2018).

Cada célula de uma LSTM (representada na Figura 11) combina os valores do estado anterior, da memória atual e da entrada, executando uma série de operações que define se a

informação computada anteriormente irá seguir inalterada ao longo da rede, ou que parcela dessa informação será eliminada, dando lugar a novas informações dentro da célula.

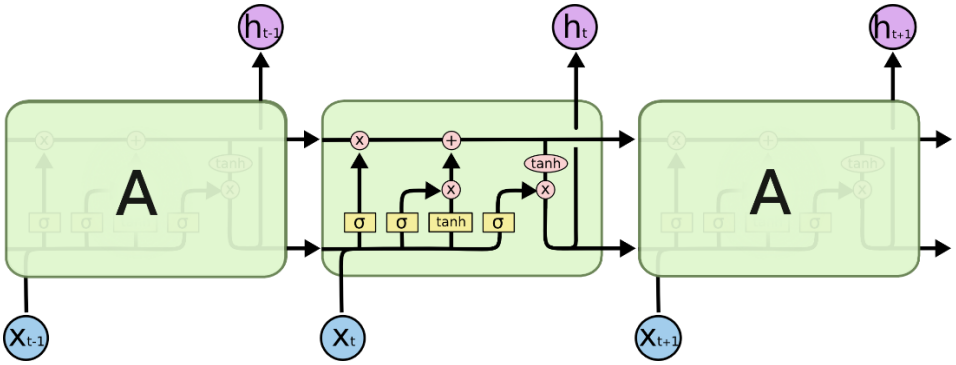


Figura 11 - Estrutura de uma célula LSTM. Retirado do blog de Christopher Olah.

A chave para uma LSTM é o estado da célula, ou a linha horizontal que percorre a parte superior da Figura 12. Esse estado é como uma correia transportadora, que percorre toda a cadeia, com apenas algumas interações lineares menores, tornando mais fácil a fluidez de informações inalteradas.

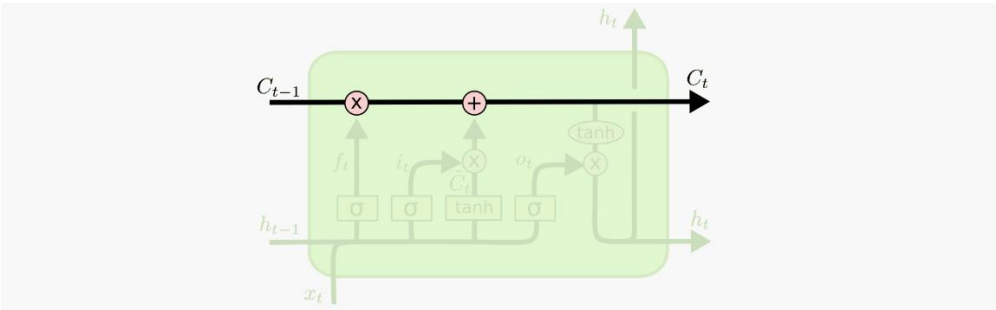
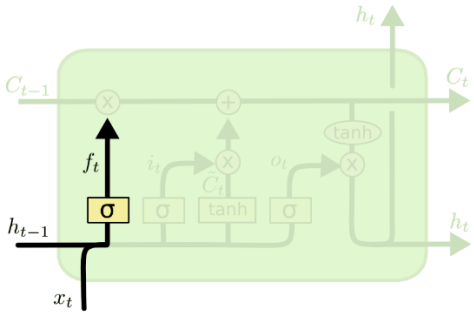


Figura 12 - Estado da célula de uma LSTM. Retirado do blog de Christopher Olah.

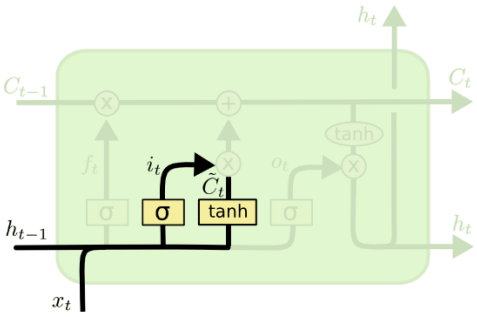
O *forget gate* (Figura 13) controla a entrada da célula LSTM (C_t) através da camada *sigmoide* (σ), que possui a propriedade específica de retornar valores entre 0 e 1. Assim, quanto mais próximo de 1 for o valor da *sigmoide*, mais informações serão mantidas da célula anterior (C_{t-1}) e passadas para frente. W_f e b_f, são, respectivamente, o peso e o valor bias para o portão de entrada.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 13 - Forget Gate de uma célula LSTM. Retirado do blog de Christopher Olah.

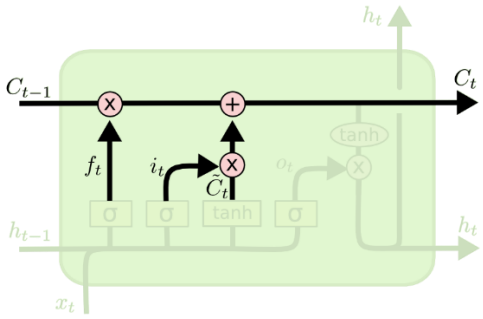
A célula LSTM, em seguida, decide quais informações novas serão armazenadas no estado da célula de memória. Primeiramente, o *input gate* (Figura 14) através da função σ decide quais valores do estado atual da célula serão atualizados. Em seguida, a função $tanh$ calcula um novo valor ($\tilde{C}_t$) para ser multiplicado ponto a ponto com o vetor resultante do passo anterior.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 14 - Input Gate de uma célula LSTM. Retirado do blog de Christopher Olah.

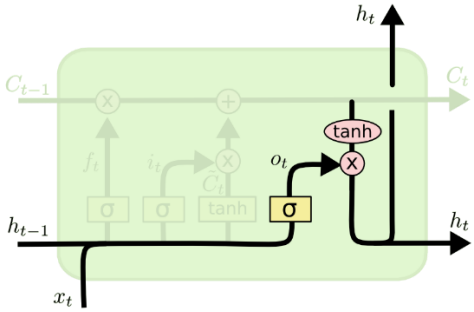
O vetor que contêm os valores a serem adicionados ao estado da célula é combinado com a saída do *input gate*, em seguida somado ao produto entre *forget gate* e a matriz de estados ocultos atual da célula, como pode ser visto na Figura 15.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 15 - Gate de uma célula LSTM. Retirado do blog de Christopher Olah.

Por fim, para decidir o que irá para a saída da célula (h_t) dois passos são necessários: primeiro a *sigmoide* (σ) decide quais partes do estado da célula C_t irão para a saída o_t. Então, o estado da célula é passado como parâmetro para uma função tangente hiperbólica ($tanh$) e então combinada com a saída o valor do σ, como pode ser visto na Figura 16. A função $tanh$ força os valores a ficarem entre -1 e 1, fornecendo uma faixa de valores mais ampla para ser armazenada.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 16 - Output Gate de uma célula LSTM. Retirado do blog de Christopher Olah.

2.5.4. *Long Short-Term Memory* (LSTM) bidirecional

LSTM bidirecional (Figura 17) é uma extensão do LSTM tradicional que pode melhorar o desempenho e a acurácia do modelo em atividades de classificações. Ela pode ser treinada usando todas as informações de entrada, do passado e do futuro, disponíveis de um tempo específico (SCHUSTER; PALIWAL, 1997). Nos problemas onde todos os *steps* de entrada estão disponíveis, o LSTM bidirecional treina dois ao invés de um LSTM na

sequência de entrada. O primeiro como está e o outro como uma cópia invertida da sequência de entrada. Ao utilizar informações do passado e do futuro como entrada é possível minimizar a função objetivo sem a necessidade de atrasos para a inserção de novas informações (SCHUSTER; PALIWAL, 1997). Isso fornece um contexto adicional à rede que resulta em um aprendizado mais rápido e mais completo do problema.

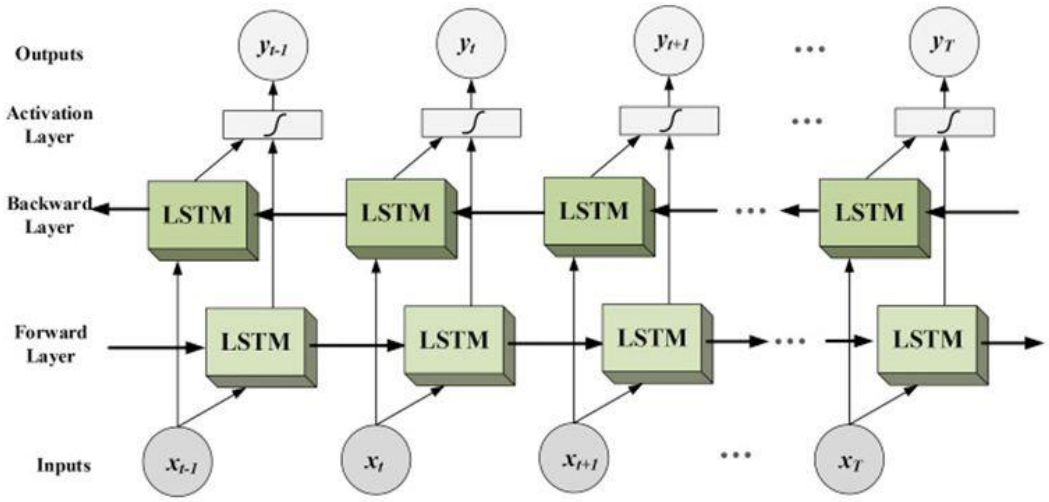


Figura 17 - Estrutura de um LSTM Bidirecional. Retirado do site i2tutorials.

2.6. *Word Embeddings*

No contexto de NLP, convertemos dados textuais em representações vetoriais contendo valores numéricos que refletem várias propriedades linguísticas, tais como relacionamentos semânticos e contextuais. Estes, representam um grande desafio quanto a manutenção da coerência das representações de forma a preservar relações potencialmente importantes para as diversas tarefas.

Os *word embeddings*, são representações vetoriais (a princípio de palavras) capazes de manter a relação entre duas palavras semanticamente relacionadas sem perder a habilidade de codificá-las de maneiras distintas (GOODFELLOW; BENGIO; COURVILLE, 2016). Dessa forma, essas representações são amplamente utilizadas para sanar os desafios supracitados. No espaço vetorial do *word embeddings*, palavras que

aparecem frequentemente em contextos muito parecidos estão mais próximas umas das outras, constituindo uma vizinhança de palavras semelhantes semanticamente. Contudo, ressalta-se que essas representações vetoriais ainda sofrem com o desafio de representar palavras que têm múltiplos significados ou sentidos (LANDEGHEM, 2016).

2.6.1. Modelos de *word embeddings*

Muitos algoritmos foram desenvolvidos para gerar modelos de *word embeddings* e disponibilizá-los para a comunidade. Tais modelos podem ser divididos em duas famílias de métodos (HARTMANN et al., 2017): Os primeiros são aqueles métodos que trabalham com a matriz de co-ocorrência de palavras, como GloVe (PENNINGTON; SOCHER; MANNING, 2014). E os segundos, são aqueles que trabalham com modelos preditivos (baseado na vizinha das palavras), como o Word2Vec (MIKOLOV et al., 2013). Os principais modelos de geração de *word embeddings* são resumidos por Hartmann et al. (2017):

- The Global Vectors (GloVe): algoritmo de aprendizado não supervisionado que computa os vetores através da análise da matriz M de co-ocorrência de palavras construída através das informações contextuais das palavras do corpus.
- Word2vec: possui duas diferentes estratégias de treinamento: (i) *Continuous Bag-ofWords* (CBOW), no qual o modelo tenta prever a palavra do meio suprimida dentro de uma sequência de palavras, e (ii) *Skip-Gram*, o modelo que serve para predizer a vizinhança de uma da palavra.
- Wang2Vec: modificação do Word2vec cujo objetivo é considerar a ordem das sequências, ao contrário da arquitetura original.
- FastText: nesta arquitetura, *word embeddings* são associados N-gramas de caracteres, sendo as palavras codificadas como a combinação dessas representações. Portanto, esse método tenta capturar informações morfológicas para construir os seus *word embeddings*.

Vetores de palavras, gerados através de todos os modelos citados acima, são disponibilizados publicamente, para *download*, pelo Repositório de Word Embeddings do Núcleo Interinstitucional de Linguística Computacional (NILC), inclusive em diferentes dimensões. Esses vetores foram gerados por meio de um *corpus* em português do Brasil e português Europeu.

O modelo GloVe executa significativamente melhor do que as outras linhas de base, geralmente com menor tamanhos de vetor e *corpora* (PENNINGTON J. SOCHER R.; MANNING C. D, 2014). Levando-se em conta essa citação e a necessidade deste trabalho de gerar *word embeddings* usando *corpora* em português, adotar-se-á os vetores do NILC construídos através do modelo GloVe de 50 e 600 dimensões (GloVe50 e GloVe600, respectivamente). O primeiro foi escolhido por ser o menor vetor de palavras do modelo e, para a prática deste trabalho, não se escolheu o maior modelo (Glove 1000) por conta do espaço disponível em disco do dispositivo computacional onde a aplicação foi desenvolvida, selecionando-se, então, o segundo maior modelo: GloVe600.

2.7. Métricas

Uma das coisas mais importantes ao avaliar diferentes algoritmos de ML, é a escolha das métricas que irão avaliar os diferentes modelos, uma vez que existem métricas mais indicadas para cada tipo de problema.

2.7.1. *Logarithmic Loss*

O *Logarithmic Loss* (ou *loss*), é uma métrica de desempenho cuja finalidade é avaliar as predições de probabilidades de uma determinada entrada pertencer a uma determinada classe. Essa métrica possui valores de 0 a 1, que pode ser vista como a porcentagem de confiabilidade de uma classificação. E como se trata de uma medida de *loss*, quanto menor, melhor. Sendo 0, um valor de erro perfeito (BROWNLEE, 2016).

O *Logarithmic Loss* utiliza a Equação 5 quando o número de classificadores é igual a 2, como no caso deste trabalho. Na equação, $\boldsymbol{y}$ é o indicador binário (0 ou 1) para quando uma classe é a classificação correta para uma observação, e $\boldsymbol{p}$ é probabilidade prevista do modelo de que uma observação seja da classe.

$$loss = -(ylog(p) + (1-y)log(1-p)) \quad (5)$$

2.7.2. Acurácia

A acurácia (ou taxa de acerto) é porcentagem de predições feitas corretamente em relação a todas as predições feitas. A acurácia é a métrica mais utilizada na avaliação de algoritmos para problemas de classificação (BROWNLEE, 2016). Na equação 6 pode-se observar como ela é calculada.

$$acurácia = \frac{Nº\ de\ acertos}{Nº\ de\ dados\ classificados} \quad (6)$$

2.7.3. Matriz de confusão

A Matriz de Confusão é uma representação muito útil para a acurácia de um modelo com duas classes. Basicamente, é uma tabela que possui uma linha e uma coluna para cada classe. Cada célula possui o número - ou porcentagem - de predições da classe da linha atual que pertencem à classe da coluna atual (BROWNLEE, 2016).

A partir da Matriz de Confusão, é possível retirar algumas informações sobre cada classe, que são utilizadas para calcular diversas métricas (HAN et. al., 2011), que são:

- Verdadeiro Positivo (TP): dados que foram corretamente classificados pelo classificador.
- Verdadeiro Negativo (TN): dados corretamente classificados como não pertencentes à uma determinada classe.

- Falso Positivo (FP): dados não pertencentes a uma classe, classificados como pertencentes.
- Falso Negativo (FN): dados pertencentes a uma classe, classificados como não pertencentes.

2.4. Considerações Finais

Neste capítulo discorreu-se sobre alguns conceitos e terminologias empregados no campo de estudo do ML e do processamento natural de linguagens. Além disso, abordou-se as técnicas de pré-processamento de dados e os algoritmos e modelos matemáticos necessários para a análise e classificação de opiniões em comentários textuais. Ainda, foi possível discorrer sobre os diferentes modelos de *word* embeddings e sua aplicação no contexto do processamento natural de linguagem. No capítulo seguinte descrever-se-á em detalhes o desenvolvimento deste trabalho.

CAPÍTULO 3: DESENVOLVIMENTO DO TRABALHO

3.1. Considerações Iniciais

Neste capítulo são apresentadas todas as etapas de desenvolvimento do projeto, desde os detalhes de implementação dos métodos utilizados até a discussão dos resultados obtidos em cada etapa do desenvolvimento do trabalho. Além disso, também serão comparados os modelos gerados a partir dos métodos implementados. Ao final, serão discutidas as principais dificuldades e limitações do trabalho desenvolvido.

3.2. Projeto

O objetivo deste trabalho é utilizar o conceito prático de ML supervisionado para a construção de modelos de análise e classificação de opiniões, em comentários textuais qualitativos das *Surveys multistakeholders* da pesquisa Empresas Humanizadas do Brasil. Para isto, utiliza-se de uma base proprietária de dados (algumas das próprias *surveys*, para treinamento e teste) e dos algoritmos de ML Naive Bayes, com uso das técnicas N-gramas, e redes neurais recorrentes LSTM uni e bidirecional. Estes algoritmos foram escolhidos por representarem diferentes contextos de aplicação: i) o algoritmo Naive Bayes é um dos mais simples, muito rápido, retorna um resultado razoável para classificações e é recomendado para aplicações em tempo real; ii) as redes neurais recorrentes LSTM uni e bidirecionais são algoritmos mais trabalhados, com um tempo de execução maior de acordo com a configuração utilizada, portanto não sendo recomendadas para aplicações em tempo real, mas que retornam um resultado muito assertivo. Dessa forma, espera-se, neste trabalho, que os modelos gerados a partir de redes neurais recorrentes LSTM tenham os melhores resultados, por conta dos conceitos já discutidos anteriormente.

Considerando a já existência de uma base de dados proprietária, o primeiro passo é a escolha da linguagem de programação apropriada para o processo de desenvolvimento desta

aplicação. Para isto, a linguagem de programação Python foi escolhida, pois possibilitava uma implementação enxuta e limpa dos métodos necessários e já discutidos anteriormente. A linguagem é de fácil aprendizado, escalável, se integra com diversos tipos de sistemas operacionais, possui uma grande comunidade para apoio nas dificuldades e dúvidas e, principalmente, contém uma grande variedade de bibliotecas em crescimento (quase que exponencial) para aplicações de diversos tipos, como ML e NLP.

Considerando a existência de uma base proprietária de dados, o passo seguinte é fazer o balanceamento dos dados rotulados, buscando não enviesar os algoritmos de classificação. Logo, no conjunto de treino e teste da base de dados equilibra-se a quantidade de comentários textuais rotulados como positivos ou negativos, em 50% para cada (no caso de uma base de dados que contém apenas 2 rótulos).

Em seguida, faz-se o pré-processamento destes dados a partir de técnicas como *stopwords*, *stemming* ou *tokenização*, já discutidas anteriormente. Além disso, caso necessário, pode-se elaborar algumas funções de limpeza, adequadas ao contexto desses dados. Segue a extração dos rótulos, a partir da função de uma das bibliotecas do Python, que serão os classificadores dos nossos dados.

Nos passos que seguem, geram-se as *features*, que irão compor o vocabulário da aplicação, a partir dos métodos de processamento de linguagem de natural, mencionados anteriormente.

Após toda fase de processamento dos dados aplicam-se os algoritmos de análise e classificação de opiniões com diferentes configurações (mantendo-se um backup da base de dados já processada para ser utilizada em cada nova configuração, não importando a ordem de aplicação):

- Aplicação do algoritmo Naive Bayes
    - com *features* unigramas e bigramas
- Aplicação do algoritmo redes neurais recorrentes LSTM
    - sem *word embeddings pré-treinadas*;
    - com *word embeddings pré-treinadas* GloVe50;
    - com *word embeddings pré-treinadas* GloVe600;

- Aplicação do algoritmo redes neurais recorrentes LSTM bidirecional
    - sem *word embeddings pré-treinadas*;
    - com *word embeddings pré-treinadas* GloVe50;
    - com *word embeddings pré-treinadas* GloVe600;

O objetivo de utilizar diferentes algoritmos e configurações é escolher aquela que entrega o modelo com maior acuraria no processo de treinamento e teste de classificação dos comentários. O fluxo da aplicação prática deste trabalho pode ser visto na Figura 18.

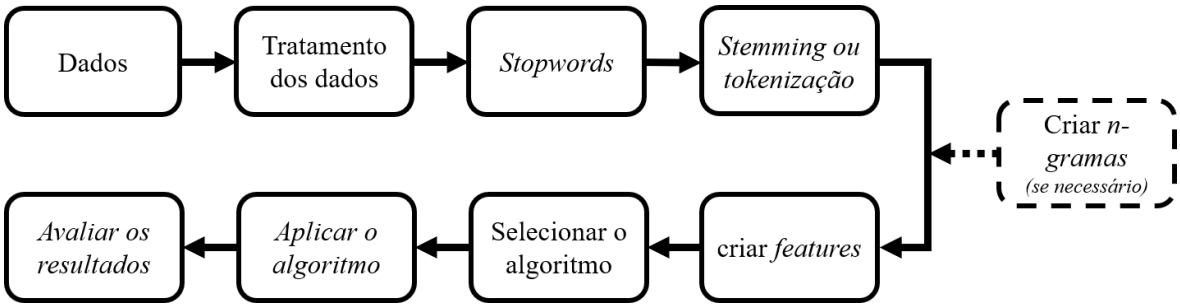


Figura 18 - Fluxo do desenvolvimento deste trabalho. Elaborado pelo autor.

Para fins de uso, esse projeto foi realizado em uma máquina com a seguinte configuração:

- Sistema operacional: Windows 10 Pro 64 bits (10.0, Compilação 18362).
- Processador: Intel® Core™ i7-6700HQ, 2.6 GHz (8 CPUs).
- Memória RAM: 8192 MB.
- GPU: NVIDIA GeForce GTX 960M, 6071MB.
- 120 GB de disco SSD.

3.3. Descrição das Atividades Realizadas

Esta seção tem por objetivo descrever em detalhes as etapas da metodologia descrita na seção anterior.

3.3.1. Escolha da linguagem de programação

A linguagem Python foi, essencialmente, escolhida para o desenvolvimento deste trabalho por conta da grande variedade de bibliotecas, já existentes, para atividades de pré-processamento de dados, processamento de linguagem de natural e implementação de algoritmos de ML. Além disso, a linguagem conta com uma enorme comunidade para a solução de dúvidas e um vasto arsenal de materiais disponíveis na Internet para consulta e uso. Outras características, não menos relevantes, para a escolha da linguagem são: i) linguagem de fácil entendimento (mesmo para os que nunca tiveram contato); ii) integração com diversos sistemas operacionais (este trabalho foi desenvolvido em sistema Windows 10); iii) linguagem de alto nível, onde códigos complexos, densos e de difícil compreensão são dispensados, reduzindo tempo de compilação (Peters, T, 2004).

3.3.2. Balanceamento da base de dados

Este trabalho foi desenvolvido em cima de uma base, de comentários textuais qualitativos, proprietária, ou seja, o seu autor já dispunha dos dados com as classificações pertinentes ao desenvolvimento do projeto. Essa base era um arquivo em formato *.xlsx* (Excel) extraído das *Surveys multistakeholders* respondidas por um grupo de empresas participantes da 1ª edição da pesquisa Empresa Humanizadas do Brasil. O arquivo possuía 6102 comentários, dentre os quais foram escolhidos (a partir da simples aplicação de um filtro no próprio programa Excel), apenas, os que possuíam rótulos ou classificações (positivo ou negativo). Dessa forma, chegou-se à base de dados, de fato, utilizada neste projeto, um arquivo *.xlsx* com 622 comentários já rotulados.

Porém, para o desenvolvimento da aplicação os dados precisavam estar balanceados, para não enviesar os algoritmos de análise e classificação. Logo, dos 622 comentários rotulados, 489 eram positivos e, apenas, 133 negativos. Portanto, foi realizado um corte consciente - garantindo que continuassem a existir comentários de todas as empresas do grupo analisado - de comentários classificados positivamente. Por fim, a base de dados foi reduzida a, apenas, 266 comentários rotulados e balanceados entre positivos e negativos.

3.3.3. Pré-processamento dos dados

Tendo em mãos uma base de dados rotulada e balanceada era necessário fazer o pré-processamento desses dados para os próximos passos do desenvolvimento da aplicação. Para isso, primeiramente utilizou-se a biblioteca *pandas* da linguagem Python que são estruturas de dados poderosas para análise de dados. Através da função *pandas.read_excel()* foi possível ler o arquivo no formato *.xlsx* (Excel) e transferir os dados para serem trabalhados dentro da aplicação. Ademais nesta aplicação foram utilizados mais recursos disponíveis na biblioteca como: i) *apply*, permite a manipulação dos dados, como deixar todas as letras em minúsculo, limpar strings; ii) *get.dummies*, transforma os caracteres dos classificadores, positivo e negativo, em 0 e 1.

Em seguida, para tratar os comentários, efetuando a limpeza de caracteres especiais, pontuações e coisas inúteis, utilizou-se a biblioteca *re* que permite manipular expressões regulares e pode ser visto no Apêndice A.

Por fim, utilizou-se a biblioteca NLTK que tem como objetivo dispor ferramentas para o NLP. Através de suas funções foi possível eliminar as *stopwords* e realizar o *stemming* nos códigos desenvolvidos.

3.3.4. Extração de *features* dos dados

Nesta aplicação de NLP, as *features* do modelo foram basicamente as palavras (também chamadas de *tokens*). Para o teste do algoritmo Naive Bayes, foram utilizados unigramas e bigramas que compõem os comentários textuais, através do método discutido na seção anterior, TF-IDF. Para o teste dos algoritmos LSTM e LSTM bidirecional, foram utilizados apenas os *tokens* por se tratar de um algoritmo com uma maior complexidade.

3.3.5. Aplicação dos algoritmos de análise e classificação de opiniões

No passo que segue a extração de *features* a partir das instâncias dos dados da base, deve-se realizar a aplicação dos algoritmos selecionados com a finalidade de comparação e seleção do modelo com os melhores resultados.

Os algoritmos foram implementados usando a biblioteca *sklearn*, que consiste de um módulo para ML e mineração de dados da linguagem Python. Além disso, os algoritmos LSTM e LSTM bidirecional utilizaram a interface de programação de aplicações (API, do inglês, *Application Programming Interface*) de alto nível para redes neurais, a Keras.

3.3.5.1. Algoritmo Naive Bayes

A implementação do algoritmo se deu, como mencionado, através da biblioteca *sklearn*, mais especificamente através da classe *MultinomialNB()*, um classificador Naive Bayes multinominal, adequado para classificar recursos discretos como contagem de palavras para classificação de textos. Porém, a função se adequa muito bem a contagens fracionárias quando utiliza métricas como TF-IDF, que foi o caso deste trabalho. Além disso, o algoritmo foi testado em duas configurações, uma utilizando unigramas e outra utilizando bigramas.

Segundo Buduma (2015), havia um consenso de se distribuir os dados de treinamento e teste na proporção de 80% e 20%, respectivamente, que fora utilizada neste trabalho. E, como discutido anteriormente, o algoritmo foi utilizado em duas configurações, uma usando unigramas e outra usando bigramas, os resultados da matriz de confusão podem ser observados na Tabela 1 e 2, respectivamente. Os valores da acurácia e do tempo de treinamento do modelo dos algoritmos podem ser vistos na tabela 3.

	Positivo	Negativo
Positivo	51	3
Negativo	4	50

Tabela 1 - Matriz de confusão do algoritmo Naive Bayes utilizando unigramas. Elaborado pelo autor.

	Positivo	Negativo
Positivo	50	4
Negativo	3	51

Tabela 2 - Matriz de confusão do algoritmo Naive Bayes utilizando bigramas. Elaborado pelo autor.

	Acurácia	Tempo de treinamento
Naive Bayes com unigramas	94%	3s
Naive Bayes com bigramas	94%	5s

Tabela 3 - Acurácia e tempo de treinamento dos algoritmos. Elaborado pelo autor.

3.3.5.2. Algoritmo redes neurais recorrentes LSTM e LSTM bidirecional

Além de utilizar a biblioteca *sklearn*, o algoritmo também fez o uso da API Keras que já implementa o LSTM e o LSTM bidirecional, através das funções *LSTM()* e *Bidirectional()*, respectivamente. Para este último, passou-se como parâmetro da função a própria função *LSTM()*.

Para desenvolver o modelo a ser usado como classificador de opiniões, utilizou-se a mesma proporção da aplicação do algoritmo Naive Bayes, uma porcentagem de 20% dos

dados para teste e de 80% para treino do modelo. As etapas úteis a execução desta aplicação: i) carregar os dados pré-processados na memória; ii) quando utilizado, carregar o modelo de *word embedding* (GloVe50 ou GloVe600); iii) realizar o treinamento dos dados; iv) executar a avaliação das classificações no conjunto de testes.

Para as arquiteturas, a etapa de treinamento foi executada em 6 iterações de uma época, logo o treinamento total consiste de 6 épocas, por questões de tempo de processamento e validação através de testes com até 15 épocas de que esse valor (6 épocas) retornava o melhor resultado. Assim, após o treinamento de uma época é feita a classificação dos dados do conjunto de teste e em seguida a avaliação das classificações feitas. Dessa forma, é possível avaliar a evolução do modelo de acordo com o avanço dos treinamentos. Após o fim do treinamento das 6 épocas, cada modelo é salvo para que possa ser reutilizado futuramente.

Após essa etapa, faz-se uma análise de desempenho para permitir avaliar qual arquitetura consegue melhores resultados para o problema de classificação de opiniões em comentários textuais. Para avaliar quão bons foram os resultados obtidos pelas arquiteturas, foram avaliadas as métricas já discutidas acima.

Das métricas avaliadas, para redes neurais, o *loss* é a métrica de maior relevância, uma vez que durante o treinamento, a rede busca sempre o diminuir. Assim, a rede que alcançar o menor valor de *loss*, será considerada a rede com o melhor desempenho. Na aplicação deste trabalho calculamos o *loss* a partir da função de custo já implementada nas bibliotecas em uso *binary_crossentropy*, uma vez em que estamos medindo apenas 2 parâmetros de classificação, representados vetorialmente como 0 e 1.

Porém, o *loss* não tem uma faixa de valores predefinida, sendo, portanto, difícil avaliar o quão bons são os resultados apenas com ele. Para auxiliar nessa análise, utiliza-se a acurácia, para obtenção de uma visão mais ampla do desempenho.

Os resultados das diferentes arquiteturas e configurações citadas na seção 3.2, podem ser observados nas Tabelas 4 e 5.

Arquitetura	epc 1	epc 2	epc 3	epc 4	epc 5	epc 6
LSTM pura	0,6841	0,6491	0,6181	0,3815	0,2876	**0,1986**
LSTM com GloVe50	0,6561	0,6294	0,5948	0,5480	0,5056	**0,4422**
LSTM com GloVe600	0,4340	0,2070	0,1706	0,3258	0,2884	**0,1308**
LSTM bidirecional pura	0,6892	0,6735	0,5667	0,3984	0,2693	**0,1183**
LSTM bidirecional com GloVe50	0,5094	0,6038	0,6651	0,7264	0,7170	**0,7170**
LSTM bidirecional com GloVe600	0,4682	0,3505	0,4239	0,4053	0,2607	**0,1997**

Tabela 4 - Valores de loss por época (epc) e por arquitetura. Elaborado pelo autor.

Arquitetura	epc 1	epc 2	epc 3	epc 4	epc 5	epc 6
LSTM pura	68,52	75,93	68,52	93,33	87,04	**91,44**
LSTM com GloVe50	66,67	70,39	70,37	75,93	79,63	**79,63**
LSTM com GloVe600	85,19	87,04	92,59	87,04	90,74	**92,59**
LSTM bidirecional pura	62,96	53,70	75,93	92,59	90,74	**94,44**
LSTM bidirecional com GloVe50	57,41	61,11	72,22	74,07	75,93	**83,33**
LSTM bidirecional com GloVe600	81,48	85,19	88,89	88,89	90,74	**92,59**

Tabela 5 - Valores de acurácia em porcentagem por época e por arquitetura. Elaborado pelo autor.

Os tempos de execução dos treinamentos dos dados dos algoritmos podem ser vistos na Tabela 6.

Arquitetura	Tempo de execução do treinamento
LSTM pura	15 s
LSTM com GloVe50	40 s
LSTM com GloVe600	123 s
LSTM bidirecional pura	491 s
LSTM bidirecional com GloVe50	26 s
LSTM bidirecional com GloVe600	856 s

Tabela 6 - Tempos de execução dos treinamentos dos dados dos algoritmos utilizados. Elaborado pelo autor.

3.4. Resultados Obtidos

Utilizando o algoritmo Naive Bayes os resultados obtidos, nos casos de teste e treinamento, foram acima do esperado e com um tempo de execução baixa. Esse algoritmo é reconhecido, inclusive, por este fato, que o leva a ser adequado para aplicações de tempo real. Atentou-se, também, o fato de que o resultado não se alterou em termos de acurácia de acordo com a escolha da configuração n-gramas.

Na aplicação dos algoritmos de redes neurais recorrentes testadas, percebe-se que a melhor configuração, ou seja, aquela que possui o menor valor de *loss* e a maior acurácia, é a LSTM bidirecional pura. Percebe-se, também, o fato de que com o uso dos *word embeddings* as redes que retornaram os melhores resultados foram as que utilizaram a dimensão 600, ou seja, as que possuíam um maior número de vetor de palavras.

Para fins de novos testes dos modelos implementados, treinados e testados com a mesma base de dados, montou-se uma nova base contendo novos comentários de outras empresas participantes da pesquisa, os quais também já haviam sido classificados, porém não pertenceram à base inicial deste trabalho. A nova base continha 52 comentários, dos

quais 33 eram positivos e 19 eram negativos, ou seja, não haviam balanceamento dos dados porque não era preciso, uma vez que essa base foi utilizada apenas para validação dos modelos gerados.

Dessa forma, os comentários textuais de opiniões da nova base foram dados como entrada para os modelos gerados e sua saída coletada em um arquivo em formato *.xlsx* (Excel) para fins de comparação com a classificação previamente realizada. Os resultados dos modelos em relação à classificação desses dados podem ser observados na Tabela 7.

Modelo aplicado	Acertos
Naive Bayes com unigramas	67
Naive Bayes com bigramas	75
LSTM pura	88
LSTM com GloVe50	73
LSTM com GloVe600	87
LSTM bidirecional pura	92
LSTM bidirecional com GloVe50	81
LSTM bidirecional com GloVe600	87

Tabela 7 - Acertos em porcentagem na nova base de dados de acordo com o modelo gerado. Elaborado pelo autor.

Nota-se que os resultados does testes através da nova base seguem razoavelmente os resultados discutidos anteriormente. As porcentagens de acerto das redes neurais recorrentes seguiram aproximadamente os valores de acurácia dos dados que foram testados da base inicial no modelo treinado. Entretanto, as porcentagens de acerto para o algoritmo Naive Bayes não seguiram os valores de acurácia vistos na aplicação com a base inicial, o que pode

ser explicado por conta da baixa quantidade de dados rotulados nos treinos e testes para gerar o modelo. Nas dissertações e artigos lidos na internet as bases que geram os modelos de classificação, geralmente, possuem milhares ou milhões de dados classificados, mas neste trabalho utilizou-se uma base de, apenas, algumas centenas de dados.

É interessante notar que, assim como esperado as redes neurais recorrentes tiveram um desempenho superior ao algoritmo de Naive Bayes, entretanto esperava-se que as LSTMs que utilizaram os *word embeddings* tivessem uma taxa de acerto maior, o que não foi constatado nos resultados de testes com a nova base de dados.

3.5. Dificuldades e Limitações

A maior dificuldade encontrada para o desenvolvimento deste trabalho foi a baixa quantidade de dados rotulados para treinamento e teste dos modelos gerados por algoritmos de ML. Como discutido anteriormente, normalmente utilizam-se bases de milhares ou milhões de dados para treinamento dos modelos. Dessa forma, possivelmente essa seja também a maior limitação das análises e comparações entre os diferentes métodos utilizados.

O autor não chegou a explorar a utilização de outros métodos para classificação de opiniões, como os classificadores KNN e suas variantes, o SVM, as árvores de decisão, dentre outros tantos que existem na literatura. Foi feita a opção de analisar e testar métodos muito distantes em relação a complexidade de implementação, tempo de execução e ao contexto de aplicação, por conta do tempo do autor disponível para pesquisa e implementação destes métodos.

Dessa forma, a principal lição que o autor deste trabalho levará de sua execução é que a pesquisa científica deve explorar todos os possíveis caminhos que levam a solução desejada. Além disso, a ciência enquanto arte precisa de tempo disponível para ser apreciada, analisada, incorporada e, só então, colocada em prática.

3.6. Considerações Finais

Neste capítulo foi apresentado todo o desenvolvimento do trabalho, os pensamentos que levaram a cada processo e os resultados obtidos. Foram analisados os desempenhos de dois métodos de ML para classificação de opiniões, além da comparação de seus modelos gerados por meio de diferentes configurações. Suas dificuldades e limitações foram expostas e consideradas para o contexto deste trabalho, destacando-se a importância de uma quantidade, relativamente, alta de dados para gerar modelos de classificação mais assertivos. No próximo capítulo será feita a conclusão do trabalho, e algumas considerações de importância para o curso de graduação do autor.

CAPÍTULO 4: CONCLUSÃO

4.1. Contribuições

Este trabalho teve como objetivo utilizar o ML supervisionado para a construção de modelos de análise e classificação de opiniões, em comentários textuais qualitativos. Para isto, utiliza-se de uma base proprietária de dados (para treinamento e teste) e dos algoritmos de ML Naive Bayes, com uso das técnicas N-gramas, e redes neurais recorrentes LSTM e LSTM bidirecional, tendo como finalidade a comparação e seleção do modelo com melhores resultados, para uso real nas próximas edições da pesquisa Empresas Humanizadas do Brasil. Assim sendo, elenca-se as seguintes contribuições científicas e práticas deste projeto:

1. Comparação entre diferentes e distantes métodos de classificação de textos utilizando-se ML.
2. Validar o uso de redes neurais recorrentes para classificação de opiniões textuais num cenário de aplicação real.
3. Implementação de algoritmos de ML no escopo das próximas edições da pesquisa Empresas Humanizadas do Brasil

4.2. Relacionamento entre o Curso e o Projeto

O autor deste trabalho não teve a oportunidade de cursar, durante a graduação, nenhuma disciplina sobre ML e, também, não teve contato com a linguagem de programação Python através do curso. Algo similar, mas ainda muito distante, foi a disciplina de Inteligência Artificial, que porventura teve uma abordagem muito prática. Dessa forma, o autor considera as disciplinas de Introdução a Ciência da Computação como cruciais para o desenvolvimento deste presente trabalho, pois a partir do conhecimento da lógica de programação e da linguagem C, o aprendizado de Python e o entendimento dos algoritmos de ML foram muito fluidos.

O autor deste trabalho teve a oportunidade de atuar, em 2018, na 1ª edição da pesquisa Empresas Humanizadas do Brasil, com o objetivo ajudar a elevar a humanidade por meio da inspiração de negócios mais conscientes, humanizados, sustentáveis e inovadores. Durante esse período, o autor desenvolveu uma ferramenta de organização e pré-

processamento dos dados qualitativos textuais, das *Surveys* da pesquisa, o que foi um fator essencial para desenvolver este trabalho.

4.3. Considerações sobre o Curso de Graduação

O curso de Engenharia de Computação da USP de São Carlos, quando comparado a cursos da mesma área da mesma instituição, como o curso de Engenharia Elétrica e o de Ciência da Computação, possui uma carga horária de aulas semestral absurdamente grande, o que faz com que trabalhos fora de sala de aula não sejam tão priorizados quanto deveriam. É notável que para o aprendizado de conceitos complexos de computação, assim como desenvolvimento de habilidades técnicas de criação de códigos eficientes requer prática. Nesse sentido, a carga horária de aulas extremamente carregada prejudica o trabalho prático dos estudantes, assim como é deixado de lado cada vez mais as atividades extra-curriculares, importantes para o desenvolvimento pessoal e profissional dos estudantes.

Sendo o curso altamente relacionado às tecnologias emergentes da época, é preciso que o mesmo se adapte rapidamente a elas, de forma a não ficar atrasado tecnologicamente em relação ao mundo como um todo.

4.4. Trabalhos Futuros

Este projeto foi realizado tendo em mente que, caso bem-sucedido, como foi o caso, seria o princípio de uma incorporação de ML nas próximas edições da pesquisa Empresas Humanizadas do Brasil.

Assim, como trabalho futuro, propõe-se a implementação da técnica que se mostrou mais apropriada para reconhecimento de opiniões da pesquisa supracitada nas *Surveys multistakeholders*, visando reduzir o tempo gasto da equipe para leitura e classificação dessas opiniões. Além disso, através da construção de uma base de dados muito maior, para

treinamento a partir do algoritmo de ML, a incorporação *online* e em *on demand* deste trabalho, para que as empresas possam ter seus relatórios gerados automaticamente.

REFERÊNCIAS

ALVES, F. A. L.; BAPTISTA, C. D. S.; FIRMINO, A. A.; OLIVEIRA, M. G. D.; PAIVA, A. C. D. A. **Comparison of svm versus naïve-bayes techniques for sentiment analysis in tweets: a case study with the 2013 FIFA confederations cup**. In: Brazilian Symposium On Multimedia And The Web, João Pessoa, 2014.

ARANHA, C. N. **Uma abordagem de pré-processamento automático para mineração de textos em português: sob o enfoque da inteligência computacional**. Pontifica Universidade Católica do Rio de Janeiro, 2007. Disponível em: <http://eds.a.ebscohost.com/eds/detail/detail?vid=1&sid=1963d31b-9887-4b50-ad3f5d3bc06dc934%40sessionmgr4008&bdata=Jmxhbmc9cHQtYnImc2l0ZT1lZHMtbGl2ZSZzY29wZT1zaXRl#AN=edsndl.oai.union.ndltd.org.IBICT.oai.agregador.ibict.br.BDTD.oai.bdtd.ibict.br.PUC.RIO.o>. Acesso em: 23 out. 2019, 23:37:52.

BARROSO, Y. M. **Structured learning with incremental feature induction and selection for portuguese dependency parsing**. Dissertação (mestrado), Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2016.

BISPO T. D. **Arquitetura LSTM para classificação de discursos de ódio cross-lingual Inglês-PtBR**. Tese (mestrado) - Universidade Federal De Sergipe, 2018.

BROWNLEE, J. **Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models and Work Projects End-to-end.** [s.n.], 2016.

BUDUMA, N. **Fundamentals of Deep Learning**, 2017.

CHAKRABORTY, G.; KRISHNA, M. **Analysis of unstructured data: applications of text analytics and sentiment mining**. In: SAS GLOBAL FORUM, 2014, Washington. Proceedings... Cary: SAS Institute Inc, 2014. p. 1288-1302.

CRESTANA, C. E. M. **A token classification approach to dependency parsing**. Dissertação (mestrado), Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2010.

FIGUEIREDO, N. M. A. **Método e Metodologia na Pesquisa Científica**. (Org.) s.l., Difusão Editora, 2004. p. 247.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning.** [S.l.]: MIT Press, 2016.

HAN, J.; PEI, J.; KAMBER, M. **Data mining: concepts and techniques**. [S.l.]: Elsevier, 2011.

HARTMANN, N. et al. **Portuguese word embeddings: Evaluating on word analogies and natural language tasks**. arXiv preprint arXiv:1708.06025, 2017.

HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. Neural Computation, MIT Press, v. 9, n. 8, 1997. p. 1735-1780.

INOKI, S. R. **Uma Gramática de um Fragmento do Português Baseado na Lógica Ilocutória**. Dissertação (Mestrado) – Curso de Sistemas e Computação, Instituto Militar de Engenharia, Rio de Janeiro, 1992.

JORDAN, M. I.; MITCHELL, T. M. **Machine learning: Trends, perspectives, and prospects**. *Science* 349, 255, 2015.

LANDEGHEM, J. V. **A survey of word embedding literature**. 2016.

LIU, B. **Sentiment analysis and opinion mining**. Synthesis Lectures On Human Language Technologies, v. 5, n. 1, 2012. p. 1-167.

MIKOLOV, T. et al. **Efficient estimation of word representations in vector space**. arXiv preprint arXiv:1301.3781, 2013.

OLAH, Christopher. **Understanding LSTM Networks**. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 23 out. 2019, 22:42:30.

PADILHA, V. A.; CARVALHO, A. C. P. L. F. **Mineração de Dados em Python**, 2017.

PENNINGTON, J.; SOCHER, R.; MANNING, C. **Glove: Global vectors for word representation**. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). [S.l.: s.n.], 2014. Disponível em: <https://nlp.stanford.edu/pubs/glove.pdf> Acesso em: 22 out. 2019, 19:28:23.

PETERS, T. **Zen of Python**, 2004. Disponível em: <https://www.python.org/dev/peps/pep-0020/>. Acesso em 24 out. 2019.

SCHUSTER, M.; PALIWAL, K. K. **Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing**, IEEE, v. 45, n. 11, p. 2673–2681, 1997.

VON LUXBURG, U.; SCHÖLKOPF, B. **Statistical learning theory: models, concepts, and results**, 2008.

WILSON, T.; WIEBE, J.; HOFFMANN, P**. Recognizing contextual polarity: an exploration of features for phrase-level sentiment analysis**. Computational linguistics, v. 35, n. 3, 2009. p. 399-433.

ZHANG, Y.; JIN, R.; ZHOU, Z. **Understanding bag-ofwords model: a statistical framework. International journal of machine learning and cybernetics.** V.1, n. 1, 2010. p. 43-52.

APÊNDICE A – Código Fonte 1

```
def clean_str(string):
    string = re.sub(r"[^A-Za-z0-9(),!?'`$%@#.&+/\|-]", " ", string)
    string = re.sub(r",", " , ", string)
    string = re.sub(r"!", " ! ", string)
    string = re.sub(r"\(", " \(", string)
    string = re.sub(r"\)", " \) ", string)
    string = re.sub(r"\?", " \? ", string)
    string = re.sub(r"\s{2,}", " ", string)

    cleanr = re.compile('<.*?>')

    string = re.sub(r'\d+', '', string)
    string = re.sub(cleanr, '', string)
    string = re.sub("'", '', string)
    string = re.sub(r'\W+', ' ', string)
    string = string.replace('_', '')


    return string.strip()
```