



ANÁLISE DE SENTIMENTO EM REDES SOCIAIS POR CLASSIFICADORES MULTIMODAIS DE APRENDIZADO DE MÁQUINA

Breno Vieira Arosa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Luiz Pereira Calôba
Natanael Nunes de Moura
Junior

Rio de Janeiro
Setembro de 2019

ANÁLISE DE SENTIMENTO EM REDES SOCIAIS POR CLASSIFICADORES
MULTIMODAIS DE APRENDIZADO DE MÁQUINA

Breno Vieira Arosa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE
ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE
JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA
ELÉTRICA.

Examinada por:

Prof. Luiz Pereira Calôba, Dr.Ing.

Natanael Nunes de Moura Junior, D.Sc.

Prof. Nome do Terceiro Examinador Sobrenome, D.Sc.

Prof. Nome do Quarto Examinador Sobrenome, Ph.D.

Prof. Nome do Quinto Examinador Sobrenome, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2019

Vieira Arosa, Breno

Análise de Sentimento em Redes Sociais por Classificadores Multimodais de Aprendizado de Máquina/Breno Vieira Arosa. – Rio de Janeiro: UFRJ/COPPE, 2019.

XI, 62 p.: il.; 29, 7cm.

Orientadores: Luiz Pereira Calôba

Natanael Nunes de Moura Junior

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 53 – 62.

1. Análise de sentimento. 2. Processamento de linguagem natural. 3. Redes complexas. 4. Aprendizado de máquina. I. Pereira Calôba, Luiz *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

*A alguém cujo valor é digno
desta dedicatória.*

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ANÁLISE DE SENTIMENTO EM REDES SOCIAIS POR CLASSIFICADORES MULTIMODAIS DE APRENDIZADO DE MÁQUINA

Breno Vieira Arosa

Setembro/2019

Orientadores: Luiz Pereira Calôba
Natanael Nunes de Moura Junior

Programa: Engenharia Elétrica

Nos últimos anos, as redes sociais se tornaram um dos principais meios de comunicação, e com isso, houve um aumento da influência que exercem sobre os usuários. Por esse motivo, as mensagens que trafegam por elas passam a ter importância para as mais diversas finalidades como, por exemplo, a avaliação de produtos e eventos. Dessa forma, esse tipo de texto se faz passível de diversas análises, sendo a mineração de opinião é uma das operações com mais aplicações diretas. Nesse sentido, ferramentas de processamento de linguagem natural e de redes complexas são capazes de auxiliar a geração destas análises. Entretanto, o desempenho dessas técnicas, em geral, depende da existência e do volume de bases de treinamento anotadas manualmente, dificultando assim a sua utilização. O presente trabalho aplica métodos de geração de bases de treinamento automatizadas para contornar tal obstáculo e gerar classificadores de análise de sentimento. São avaliados diferentes modelos de classificação textual, tanto lineares, como Naïve Bayes e SVM, quanto não lineares, como por *Deep Learning*. Também são analisadas técnicas de redes complexas para caracterização dos usuários das redes, abordando, assim, diferentes aspectos das informações fornecidas por estas mídias.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

SENTIMENT ANALYSIS ON SOCIAL NETWORKS BY MULTIMODAL
MACHINE LEARNING CLASSIFIERS

Breno Vieira Arosa

September/2019

Advisors: Luiz Pereira Calôba

Natanael Nunes de Moura Junior

Department: Electrical Engineering

In this work, we present ...

Sumário

Lista de Figuras	ix
Lista de Tabelas	xi
1 Introdução	1
1.1 Motivação	4
1.2 Objetivo	4
1.3 Organização do Texto	5
2 Análise de Sentimento de Redes Sociais	7
3 Processamento de Linguagem Natural	13
3.1 Pré-Processamentos	13
3.2 Representações	14
3.2.1 Codificação One-Hot	15
3.2.2 Bag-of-Words	15
3.2.3 Word2Vec	17
3.2.4 Representações por Redes Neurais Recorrentes	17
3.2.5 BERT	20
3.3 Classificadores	23
3.3.1 Algoritmos Baseados em Dicionários	23
3.3.2 Modelos de Aprendizado de Máquina Lineares	24
3.3.3 Modelos de Aprendizado de Máquina Não-Lineares	27
4 Modelos de Redes Complexas	30
4.1 Modelos Baseados em Fatoração Matricial	31
4.2 Modelos Baseados em Passeios Aleatórios	33
4.3 Redes Convolucionais de Grafos	35
5 Método	37
5.1 Bases de Dados	37
5.2 Figuras de Mérito	38

5.3	Desenvolvimento	38
5.3.1	Classificadores de Processamento de Linguagem Natural . . .	39
5.3.2	Classificadores Multimodais	40
6	Resultados e Discussões	43
6.1	Classificadores textuais não Neurais	44
6.2	Classificadores textuais por Redes Neurais	46
6.3	Classificadores Multimodais	47
7	Conclusões	52
	Referências Bibliográficas	53

Lista de Figuras

3.1	Etapas de classificadores de Processamento de Linguagem Natural.	13
3.2	Vetores <i>One-Hot</i> de palavras de um dicionário. A dimensionalidade do vetor é igual ao tamanho do dicionário.	15
3.3	Processo de representação por <i>Bag-of-Words</i> da frase “Preparando minha mochila para viagem.”. A palavra “para” é removida durante o pré-processamento por ser uma <i>stopword</i>	16
3.4	Esquemas de treinamento do Word2Vec. Exemplo composto por janela de contexto de 3 palavras. A matriz X_t representa a palavra da janela na posição t e \hat{Y}_t a predição do modelo para a mesma. W , por sua vez, representa o conjunto de pesos treináveis. A representação resultante de uma palavra é dada pelo seu referente vetor na matriz de pesos de entrada W_x	18
3.5	Arquitetura Encoder-Decoder, constituída por duas camadas de redes recorrentes: a de codificação e a de decodificação, ligadas por um conjunto de pesos C , que captura todo o contexto provindo da camada de codificação.	19
3.6	Diagrama da camada ℓ do Transformer com K cabeças de <i>self-attention</i> para uma palavra de índice i , dada a sentença S . As matrizes $Q^{k,\ell}$, $K^{k,\ell}$ e $V^{k,\ell}$ representam respectivamente os pesos de atenção de busca, resposta e valor.	22
3.7	SVM classificando duas classes linearmente separáveis.	26
5.1	Arquitetura composto pela combinação de classificadores de informação textual e de usuário concatenados. Adicionou-se também uma camada de neurônios que será treinado para combinar as representações.	42
6.1	Gráfico da <i>CCDF</i> (<i>complementary cumulative distribution function</i>) dos graus de entrada e saída da rede de usuários. O eixo horizontal representa o grau k enquanto o eixo vertical demonstra a probabilidade de um vertice ter grau pelo menos k	44

6.2	Distância entre pares de nós da rede de usuários.	45
6.3	Resposta do modelo de Naive Bayes a variação do parâmetro de suavização. A linha cheia representa a média dos valores das 10 partições enquanto a área em azul, limitada pela linha rachurada, representa um desvio padrão. A linha vertical vermelha ressalta o parâmetro de maior média.	46
6.4	Curva ROC do modelo de Naive Bayes aplicado aos dados de teste.	47
6.5	Matrix confusão do modelo de Naive Bayes.	48
6.6	Resposta do modelo de SVM a variação do parâmetro de regularização. A linha cheia representa a média dos valores das 10 partições enquanto a área em azul, limitada pela linha rachurada, representa um desvio padrão. A linha vertical vermelha ressalta o parâmetro de maior média.	49
6.7	Curva ROC do modelo de SVM aplicado aos dados de teste.	50
6.8	Matrix confusão do modelo de SVM.	50
6.9	Curvas de treinamento de classificadores por redes neurais convolucionais.	51
6.10	Curvas de treinamento de classificadores por redes neurais LSTM.	51

Lista de Tabelas

2.1	Dificuldades encontradas na classificação de sentimento de <i>tweets</i> . . .	9
2.2	Emoticons selecionados por GO <i>et al.</i> [1] para supervisão distante de <i>tweets</i>	11
2.3	Exemplos de <i>tweets</i> anotados por supervisão distante com emoticons selecionados por GO <i>et al.</i> [1].	11
6.1	Emoticons selecionados para aplicação de supervisão distante.	43

Capítulo 1

Introdução

Nas últimas duas décadas, as redes sociais se tornaram um dos principais meios de comunicação. Esse crescimento se justifica, em parte, pela massificação do acesso a internet, proporcionada em grande medida por dispositivos móveis como *smartphones* e *tablets*. Também alavancado pelos avanços computacionais e pelo desenvolvimento acelerado de novas técnicas e algoritmos, o aprendizado de máquina, em especial o processamento de linguagem natural, tem essas redes como importante objeto de estudo.

Desde a chamada Revolução Digital, observamos um progressivo barateamento e facilitação do uso de dispositivos eletrônicos. À medida que essas tecnologias passaram a ser acessíveis, não apenas para as corporações, mas também para os indivíduos, houve um crescente processo de digitalização de diversos aspectos de nossas vidas. Com a comunicação não foi diferente. O email, por exemplo, substituiu desde os anos 70 operações que até então eram apenas possíveis de forma analógica, como pelo uso de cartas. Nesse contexto, as redes sociais, ou mídias sociais, abordam aspectos diferentes da comunicação, mais dinâmica e informalmente.

Apesar de já existirem desde os anos 90, é com a virada do milênio que as primeiras grandes mídias sociais online aparecem, como *LinkedIn*, *MySpace* e *Orkut*. Desde então há um aumento anual da quantidade de seus usuários. Atualmente, estima-se que 3,5 bilhões de pessoas, isto é, 45% da população mundial utilize pelo menos uma rede social. Este número torna-se ainda mais interessante quando considerado que 4,4 bilhões de pessoas têm acesso à internet. Portanto, quase 80% dos internautas estão em alguma das mídias sociais. No Brasil, esses números se acentuam ainda mais: 70% da população tem acesso à internet e 66% utiliza as redes sociais [2].

Além da alta penetração na sociedade, devido à disponibilidade proporcionada pelos dispositivos móveis, os usuários consomem boa parte de seu tempo nessas redes. No mundo, gasta-se em média 2 horas e 16 minutos por dia. Novamente, esse número é ainda superior no Brasil, onde a média é de 3 horas e 34 minutos, caracterizando-se como segundo país no mundo a usar por mais tempo as redes,

atrás somente das Filipinas.

Essa forte presença fez com que as mídias sociais não impactassem apenas as comunicações. Hoje em dia, esses meios também são comumente utilizados para buscas de relacionamentos, compartilhamento de notícias, divulgação de serviços, atendimento ao público, entre outros. As informações que trafegam nas redes exercem grande influência na formação de opinião das pessoas, seja em relação a um produto, a um evento ou até mesmo a temas políticos, como pôde-se observar em eleições pelo mundo nos últimos anos.

Portanto, a análise de tais informações, presentes nas redes, é importante para as mais diversas aplicações. Contudo, essa grande quantidade de usuários também se reflete no número de dados providos das mídias sociais. Dentre as estatísticas de uso do ano de 2018 fornecidas pelas próprias redes sociais, tem-se que, diariamente, 300 milhões de fotos são publicadas no *Facebook*, 5 bilhões de vídeos são vistos no *YouTube*, 43 bilhões de mensagens são enviadas no *WhatsApp* e 100 milhões de usuários interagem pelo *Twitter*.

O massivo volume de dados inviabiliza que essas análises sejam feitas manualmente, tornando-se necessário o desenvolvimento de ferramentas capazes de automatizar esse processo. Entram aí as técnicas desenvolvidas pelo campo do Processamento de Linguagem Natural, do inglês *Natural Language Processing* (NLP). Foi a partir dos anos 50 que esse termo passou a aparecer como um ramo da Inteligência Artificial. Devido à sua complexidade, a linguagem natural acabou inclusive se tornando um critério de inteligência, como proposto no teste de Turing [3].

O NLP é uma ampla área de pesquisa, abrangendo diferentes estágios da língua, desde os níveis de abstração mais baixos, como o estudo da fonologia e da sintaxe, até os níveis de abstração mais altos, que lidam com a semântica de determinado conteúdo. Nesse ramo, busca-se desenvolver métodos capazes de auxiliar e/ou automatizar tarefas como o reconhecimento de fala, a análise sintática de frases, a extração de entidades, a segmentação por tópicos, entre outros.

Esse conjunto de ferramentas é essencial para tratar o volume de textos gerado pelas redes sociais. Com o avanço de técnicas de *Deep Learning*, nos últimos anos pôde-se observar uma melhora significativa de desempenho, que permitiu a realização mesmo de tarefas de grande dificuldade, como a automatização de traduções e de aplicações de atendimento ao cliente por conversação.

Entretanto, apesar do grande potencial dessas técnicas, as mídias sociais apresentam características que diferenciam seu conteúdo dos gêneros textuais que tradicionalmente se analisam em NLP, como artigos científicos e textos jornalísticos, dificultando o processamento dessa informação. Em geral, as redes sociais se apresentam como um meio de conversação ágil, logo, as mensagens que circulam por elas costumam ser extremamente curtas, com amplo uso de abreviações. Por seu caráter

informal, observa-se uma alta taxa de erros gramaticais e uma grande utilização de *emoticons* ¹. Somado a isso, o fato de serem meios de comunicação globais também ressalta a presença de estrangeirismos ². Ademais, a dinamicidade das redes sociais faz com que a evolução do sentido das palavras seja acelerada. Esses elementos trazem a necessidade de adaptação ou desenvolvimento de novas técnicas para se reproduzir o sucesso obtido pelas técnicas de processamento de texto em documentos com escrita mais formal e estruturada.

No entanto, o principal fator que distingue as informações de redes de outros meios é a forte interligação entre diferentes tipos de mídia, como textos, imagens, áudios, fotos e vídeos. Além de metadados importantes, como localização, data e horário, uma propriedade importante das mídias sociais são os atributos referentes às redes de usuário. Exemplos desses dados são o número de amizades de um usuário da rede e o número de re-compartilhamentos de uma mensagem. Logo, apesar da capacidade das ferramentas de NLP, existe um conjunto de informações que essas técnicas desconsideram, abrindo espaço para que as abordagens sejam multimodais, ou seja, que explorem diversas destas propriedades.

De certa maneira, as interações entre usuários são o cerne das mídias sociais. Logo, técnicas que se dispõem a analisar esse tipo de informação também são de grande relevância. Nesse quesito, o campo das Redes Complexas, também chamado de Ciência de Redes, é responsável por estudar os algoritmos e comportamentos observados em grafos que representam sistemas reais, como no caso das redes sociais. Assim como o aprendizado de máquina, essa esfera do conhecimento também demonstra um grande crescimento nos últimos anos, fornecendo um novo leque de tecnologias, de forma a descobrir-se aplicabilidades até então inexploradas. Dentre suas aplicações, que possuem importância para o estudos das mídias sociais, pode-se ressaltar, por exemplo, a detecção de comunidades, a identificação de principais influenciadores e a modelagem de propagação de informação.

Finalmente, esses métodos são poderosas ferramentas de análise das redes sociais, principalmente quando aplicados em complemento à informação textual. A complementariedade entre os métodos em questão permitem, por exemplo, distinguir conotações de uma mesma mensagem quando escritas por usuários de comunidades de ideias opostas. Por conta de casos complexos como esse, nos quais a análise de um único elemento da mensagem não é suficiente para sua classificação, é que se fazem necessários estudos que contemplem a multimodalidade das redes sociais.

¹Sequência de caracteres ou pequena imagem que transmite um estado emotivo

²Uso de palavras ou expressões estrangeiras

1.1 Motivação

Dados são considerados um dos bens mais valiosos da atualidade, de forma mesmo a serem chamados de “o novo petróleo”. Isso porque, assim como o óleo, os dados são preciosos e precisam ser refinados para terem utilidade. Um dos aspectos dessa transformação pode ser observado na mudança cultural de organizações e empresas que passam a tomar decisões baseadas em dados e métricas coletados.

A busca por informação de qualidade sobre um serviço ou produto sempre foi importante para consumidores. Quando não havia as tecnologias utilizadas atualmente, essas pesquisas eram feitas majoritariamente ou boca-a-boca ou a partir de revistas especializadas. Com a criação da internet e das redes sociais, estas passaram também a exercer essa função, pelo benefício de se encontrar opiniões de forma espontânea e em grande quantidade. As empresas, por sua vez, têm a oportunidade de utilizar as opiniões que trafegam nas redes para identificar falhas em suas mercadorias, melhorar sua segmentação, planejar novos produtos, entre outras atividades. Com o fácil acesso à coleta desses dados, as ferramentas capazes de extrair o sentimento dessas mensagens tornam-se fundamentais para viabilizar esse procedimento na escala em que elas ocorrem.

Mesmo das dificuldades inerentes à classificação de mensagens de redes sociais, técnicas de aprendizado de máquina, sobretudo *Deep Learning*, e de Redes Complexas apresentam êxito em várias tarefas realizadas sobre elas. Entretanto, o sucesso desses modelos depende, em geral, da quantidade de dados anotados disponíveis para treinamento. Como o processo de anotação é manual, esse passa a ser o gargalo da construção de classificadores de sentimento.

Tal empecilho se torna ainda mais notável quando considera-se aplicações que requerem a análise de diferentes línguas ou que tenham foco em um tema específico, necessitando criação de bases de dados próprias para cada caso de uso. Esses fatos motivam a elaboração de métodos que sejam independentes de bases de treinamento.

1.2 Objetivo

Este projeto visa a desenvolver um método capaz de formar classificadores de análise de sentimento sem a necessidade anotação de bases de dados de treinamento. Essas análises serão feitas sobre dados de mídia sociais e tanto atributos textuais, quanto os das redes de usuários serão explorados. A principal meta deste trabalho é viabilizar o emprego de modelos complexos, que se beneficiam de um grande volume de dados, sem o custo proveniente da anotação de bases.

Para a análise das mensagens será avaliado o impacto da utilização de classificadores de *Deep Learning* em comparação a métodos lineares tradicionalmente

aplicados em NLP. Diferentes estratégias de representação de palavras e arquiteturas de classificadores de redes de aprendizado profundo serão testados de acordo com o estado da arte em processamento de texto. O processo será feito de maneira semi-supervisionada com supervisão distante, para anotação automática dos dados de treinamento.

Técnicas de Redes Complexas serão aplicadas para a caracterização dos autores das mensagens. Serão comparados modelos de representação de nós em grafos, também baseados em aprendizado de máquina e com treinamento não supervisionado, cujo custo computacional permita operar em volumes de dados compatíveis com os de redes sociais. Nesse caso, além de avaliar os modelos entre si, será analisada a informação adicional provinda dos usuários quando aplicada em conjunto com o classificador textual.

Por fim, há um amplo conjunto de estudo aplicando processamento de linguagem natural em redes sociais. Apesar de menor, a ciência de redes também têm um grande repertório de pesquisa sobre esse meio de comunicação. Assim, este trabalho visa a preencher a lacuna de sistemas que não necessitam de investimento em anotação de bases e que abrangem a multimodalidade da informação característica desse tipo de dado.

1.3 Organização do Texto

Esse documento é organizado da seguinte maneira:

- O Capítulo 2 apresenta o problema da análise de sentimento aplicada em mídias sociais e os seus desafios. Esse Capítulo contém uma breve revisão bibliográfica de classificadores de análise de sentimento e de técnicas de redes aplicadas a essa tarefa.
- No Capítulo 3, as técnicas de processamento de linguagem natural são apresentadas. São caracterizadas as diferentes formas de representação numérica das mensagens e sobre quais premissas se baseiam. Também são descritos os classificadores, tanto os métodos lineares tradicionalmente aplicados a textos, quanto os de *Deep Learning*.
- As ferramentas de ciência de redes são apresentadas no Capítulo 4. Nele, são mostradas as diferentes técnicas de representação de vértices de uma rede, que serão aplicadas na modelagem de usuários de mídias sociais.
- O Capítulo 5 descreve o método proposto para desenvolvimento dos classificadores. São apresentadas as etapas de formação de bases de dados, de anotação automática das mesmas e de sua classificação.

- Os resultados obtidos pelos experimentos propostos são mostrados no Capítulo 6.
- Por fim, o Capítulo 7 avalia tais resultados, apresenta as conclusões e enumera possíveis desdobramentos do trabalho realizado.

Capítulo 2

Análise de Sentimento de Redes Sociais

A Análise de Sentimento é o campo de estudos que analisa opinião, sentimento, atitude e emoções de pessoas em relação a entidades. Essas entidades podem ser outras pessoas, eventos, produtos, tópicos etc. Na literatura, também se encontram os seguintes nomes relacionados a esse ramo: *mineração de opinião*, *extração de opinião*, *mineração de sentimento*, *análise de subjetividade*, *análise de emoção* e *extração de críticas* [4]. Por ser aplicado majoritariamente a textos escritos, este campo é filiado ao processamento de linguagem natural, sendo uma de suas ramificações mais ativas.

Segundo CAMBRIA [5], a Análise de Sentimento é dividida entre duas principais tarefas, a extração de polaridade e a detecção de emoções. Enquanto a primeira se concentra em discernir conteúdo positivo de negativo, a segunda é responsável por classificar o documento em emoções como felicidade, medo, raiva e tristeza.

A autora LIU [4], por sua vez, ressalta que há diferentes níveis de granularidade de execução desta tarefa. A escolha do nível a ser utilizado depende da finalidade pela qual se aplicará a classificação e será uma das principais características para definir o tipo de técnica empregada. A Análise de Sentimento pode ser realizada a nível de documento, no qual um documento, como uma avaliação de produto, é analisado como um todo. Nesses casos, assume-se implicitamente que um documento expressa uma opinião sobre uma única entidade, como o produto em questão no exemplo citado. Também fica implícito que o documento expressa um sentimento único sobre a entidade. Por serem textos mais extensos, logo com mais informação, a assertividade de modelos nesses casos geralmente é mais alta. Portanto, mesmo as técnicas mais simples como as baseadas em dicionário, como serão apresentadas na seção 3.3.1, podem obter resultados suficientemente bons. TABOADA *et al.* [6] exemplifica a extração de opinião de documentos a partir de técnicas de dicionário aplicadas em diferentes bases de dados de avaliações de produtos. Por sua vez, DAS e CHEN [7] realiza predições de valores de ações a partir de análise de sentimento

das mensagens de um fórum online de investidores.

Visto que as limitações decorrentes de se classificar documentos por inteiro reduzem o escopo de aplicações possíveis, pode-se recorrer ao nível seguinte de granularidade: a classificação de sentimento de sentenças. A principal diferença entre essa abordagem e a anterior é a quantidade de informação disponível, dado que uma sentença é composta, geralmente, por poucas palavras. Por outro lado, a premissa do sentimento único no conteúdo de uma frase é mais coerente com a realidade quando comparada com a classificação do documento como um todo, sendo uma aproximação suficientemente boa para uma nova gama de casos de uso. Entretanto, LIU [4] ressalta que para o caso de sentenças é importante que a classificação de polaridade leve em consideração o sentimento neutro. Isso se torna relevante pois, até mesmo dentro de documentos opinativos, como avaliações de produto, há sentenças puramente objetivas, que não expressam polaridade sobre uma entidade. Já RILOFF *et al.* [8] apresentam o primeiro trabalho especificamente voltado para classificação de subjetividade de documentos. Devido à quantidade limitada de informação presentes em uma sentença, classificadores baseados tanto em dicionários, quanto em técnicas de aprendizado de máquina por modelos lineares apresentam indicadores piores na execução de tal tarefa. É nesse cenário que nos últimos anos os modelos não lineares começaram a sobressair, como apresentam SOCHER *et al.* [9] e SOCHER *et al.* [10], que aplicam diferentes técnicas baseadas em *Deep Learning* para classificação de sentenças.

Por fim, apresenta-se a análise de sentimento de características. Uma entidade pode ser composta de diversos atributos. Ao falar sobre um filme, pode-se avaliar aspectos dele como o roteiro, os atores, os personagens etc. Uma crítica a esse determinado filme é composta de sentimentos distintos para cada um desses atributos e o objetivo da análise de sentimento de características é identificar a polaridade de uma mensagem em relação a cada um dos atributos presentes. Para realizar essa análise, são necessários elementos novos, como o reconhecimento de entidade e dos aspectos dos quais ela é composta. Também podem ser relevantes a identificação do autor e do momento do documento analisado. NASUKAWA e YI [11] e SNYDER e BARZILAY [12] são exemplos de trabalhos que aplicam mineração de opinião focada em aspectos.

A Análise de Sentimento aplicada a redes sociais, em especial ao Twitter, se assemelha à classificação de sentimento de sentenças. Entretanto, o perfil das mensagens que circulam nas mídias sociais apresenta peculiaridades quando comparado a meios convencionais de comunicação escrita. Por se tratar de um ambiente informal e de comunicação rápida, é comum encontrar erros gramaticais e abreviações. Similarmente, os *emoticons* também tem amplo uso por serem métodos práticos de expressar sentimentos em poucos caracteres. Além disso, por se tratarem de redes

globais, é frequente o emprego de palavras ou expressões de outras línguas em uma mesma mensagem. Tais fatores são obstáculos aos classificadores de linguagem natural, dificultando a tarefa de extração de polaridade. A Tabela 2.1 mostra exemplos de mensagens extraídas do Twitter que demonstram esses problemas.

Fator	<i>Tweet</i>
Ironia	Recomendo chegar para dar aula e descobrir que mudaram seu horário sem avisar.
Ambiguidade	Estou igualmente fascinada e enojada.
Multiplicidade de idiomas	Macarrão de arroz is the new miojo.

Tabela 2.1: Dificuldades encontradas na classificação de sentimento de *tweets*.

O Twitter é uma rede social baseada em interações por mensagens curtas. Suas principais características são a brevidade e instantaneidade das mensagens, também chamadas *tweets*. Elas são limitadas atualmente em 280 caracteres, mas ficaram famosas pelo seu limite anterior de 140 caracteres. Criada em 2006, a rede conta com 139 milhões de usuários ativos diários ¹ e está entre as com maior número de usuários do mundo.

O Twitter foi responsável pela criação e popularização das *hashtags*, mecanismo que funciona como marcação de palavra-chave ou tópico. Seu funcionamento se dá pela utilização do símbolo da tralha (#) e pela ausência de espaço e pontuação nas construções que são formadas por mais que uma palavra. As *hashtags* funcionam como um agregador de *tweets* e o site ainda apresenta uma lista das mais populares no momento. O sucesso dessa funcionalidade fez com que a mesma fosse posteriormente aderida por outras mídias sociais como o Facebook e o Instagram, se tornando um atributo marcante das redes sociais.

Além das *hashtags*, as mensagens no Twitter também possibilitam a inserção de mídias como imagens, vídeos e áudios. Outro atributo comum em mídias sociais são as redes formadas pela conexão de usuários. Essas redes podem ser formadas por relações de amizade, seguidores e outras interações entre essas pessoas ou entidades. O re-compartilhamento de mensagem, que se dá quando um usuário divulga em seu perfil uma mensagem criada por outra pessoa, é outro componente capaz de gerar grafos de usuários. A título de informação, no Twitter o re-compartilhamento também é chamado de *retweet*.

Como citado anteriormente, textos que circulam nas mídias sociais contêm desafios adicionais em comparação a documentos tradicionalmente estudados pela área de NLP, como artigos jornalísticos e avaliações de produto. Apesar disso, técnicas de classificação de sentimento puramente textuais obtiveram uma crescente melhora

¹dados do relatório trimestral para investidores [13]

de desempenho, principalmente com a aplicação de técnicas de *Deep Learning*.

Um dos primeiros e mais influentes trabalhos sobre aplicação de aprendizado de máquina para análise de sentimento foi feito por PANG *et al.* [14], que comparou técnicas de classificação feitas a partir de dicionário com modelos de *Naive Bayes*, Máxima Entropia e Máquinas de Vetor Suporte (SVM). Em seu estudo, PANG *et al.* [14] também compara a eficácia de diferentes métodos de representação do texto para o uso de modelos de aprendizado de máquina.

Posteriormente, inspirado nos métodos de PANG *et al.* [14], GO *et al.* [1] utilizou as mesmas técnicas para classificação de sentimento em Twitter. A grande diferença entre os trabalhos de PANG *et al.* [14] e GO *et al.* [1] é que, enquanto no primeiro caso utiliza dados de críticas de filmes, cujo coleta já fornece uma anotação, dado que as críticas são acompanhadas de um sistema de avaliação (entre 1 e 5 estrelas), o posterior não contou com disponibilidade parecida e, por isso, aplicou um método de anotação automática.

A maneira convencional de abordar bases de dados não anotadas é realizar uma classificação manual dos dados. Por depender de iteração humana, essa se torna a etapa mais custosa do processo de criação de classificadores. Além disso, a alta dinamicidade dos temas e do vocabulário presente nas redes sociais faz com que se reduza o prazo em que a base de dados anotada é relevante. Considerando também que um maior volume das bases de dados, em geral, permitem que os modelos treinados obtenham melhores resultados, a soma desses fatores torna ineficiente o processo de anotação manual.

O processo utilizado por GO *et al.* [1] foi criado por READ [15] e denominado supervisão distante. Este método consiste em selecionar alguma característica que tenha alta correlação com a predição desejada e utilizá-la para anotar a base de dados. No caso dos trabalhos citados, foram escolhidos conjuntos de *emoticons* positivos e negativos que serviram para a anotação ruidosa. Desta forma, a restrição para construção de bases de dados passou a ser a coleta de *tweets* e de recursos computacionais. Em GO *et al.* [1], a base de treinamento, que foi anotada por supervisão distante, conteve 800 mil *tweets*. Para fins de validação do modelo, ainda foi necessário elaborar uma base de dados anotada manualmente, sendo esta de apenas 359 *tweets*.

Entre as desvantagens dessa prática está o fato de o conjunto de *emoticons* selecionados para anotação ruidosa precisar ser removido dos dados de treinamento do modelo para não introduzir viés na classificação. A qualidade da supervisão distante também depende da seleção dos *emoticons*. Ademais, não se pode descartar que é possível que uma classe contenha subclasses que não sejam correlacionadas com a característica escolhida para executar a supervisão distante. Por exemplo, dentro do conjunto de *tweets* positivos, a subclasse de *tweets* irônicos positivos pode não conter

Emoticons Positivo	Emoticons Negativos
:)	: (
:-)	:- (
:)	: (
:D	
=)	

Tabela 2.2: Emoticons selecionados por GO *et al.* [1] para supervisão distante de *tweets*.

<i>Tweet</i>	Sentimento
Adoro aquelas amizades que alinham em tudo mesmo em cima da hora :)	Positivo
Fico mais triste ainda porque tenho android :(Negativo
To com um mal estar tão grande no corpo desde ontem :- (zero forças	Negativo
Muito feliz com minha nova tattoo :D	Positivo

Tabela 2.3: Exemplos de *tweets* anotados por supervisão distante com emoticons selecionados por GO *et al.* [1].

emoticons e assim não estar presente no conjunto de treinamento. Mesmo com essas limitações, a anotação automática foi fundamental para alavancar a performance dos classificadores de sentimento de redes sociais.

A forma de se transformar o corpus textual em números foi objeto de muitos estudos. Como alternativa ao *bag-of-words*, WANG e MANNING [16] apresentaram um método de utilizar pesos obtidos a partir do treinamento de um modelo de *Naive Bayes* como entrada de um classificador de SVM. Por sua vez, PALTOGLOU e THELWALL [17] estudaram variações de técnicas de Recuperação de Informação com a mesma finalidade. Outro eixo importante foram as representações densas, ou de baixa dimensionalidade. Os principais trabalhos relacionados a essas representações foram feitos por MIKOLOV *et al.* [18], conhecido como *Word2Vec*, PENNINGTON *et al.* [19], denominado *GloVe* e mais recentemente *ELMo*, produzido por PETERS *et al.* [20], e *BERT*, de DEVLIN *et al.* [21].

A disposição de grandes bases de dados e as representações densas foram a condição *sine qua non* para viabilizar a aplicação de modelos de *Deep Learning* em análise de sentimento. O Aprendizado Profundo foi responsável por romper barreiras de performance de aprendizado de máquina em diversas áreas [22], como classificação de imagem [23], reconhecimento de fala [24] e detecção de doenças [25]. O processamento de linguagem natural foi um dos campos mais impactados por esse conjunto de técnicas, seja na realização de traduções [26], na correção de erros gramaticais [27], no reconhecimento de entidades [28], na criação de resumos [29],

ou em outras tarefas. Dentre as aplicações de *Deep Learning* em análise de sentimento, temos KIM [30], com a utilização de redes neurais convolucionais para classificação de sentenças, ZHOU *et al.* [31], que, por sua vez, demonstraram o uso de redes LSTM baseadas em redes recorrentes e SOCHER *et al.* [10], que aplicam um modelo recursivo.

Entretanto, a mistura entre diferentes modalidades de comunicação (texto, imagem, vídeo, etc.) gera uma dificuldade adicional para análise das mensagens. O texto presente em um *tweet* que possui uma imagem pode fazer interlocução com a mesma. Uma mensagem enviada por um usuário pode ter sentido oposto a mesma mensagem quando comunicada por alguém que pertença a um grupo opositor. Nesses casos, a análise do conteúdo textual por si só é incapaz de captar a essência da mensagem. Dá-se, assim, a necessidade de abordagens multimodais para qualquer tipo de análise de redes sociais.

Como a interação entre usuários é um dos principais componentes das mídias, esse se torna um objetivo natural para se estudar em complemento ao texto. Para explorar esse componente, é possível aplicar ferramentas desenvolvidas pela área de pesquisa de Redes Complexas, também chamada de Ciência de Redes. Assim como o Aprendizado de Máquina, o ramo de estudo de Redes Complexas apresenta um grande crescimento nos últimos anos, dado à variedade de sistemas em que suas análises e modelos são aplicados com efetividade [32]. Entre os exemplos de sucesso estão a identificação de doenças [33], a predição de propagação de epidemias [34], o estudo de robustez de redes de roteadores [35] e a identificação de operadores de lavagem de dinheiro [36].

No caso da aplicação dessas técnicas em redes sociais, podemos citar RATKIEWICZ *et al.* [37]. Eles utilizaram características da rede formada por usuários do Twitter que interagiram com *tweets* sobre as campanhas eleitorais de 2010 dos Estados Unidos, visando a detectar a propagação orquestrada de conteúdo. Já VAROL *et al.* [38] utilizaram propriedades como densidade da rede e coeficiente de clusteração em conjunto com outros atributos não relacionados aos grafos, buscando a detecção de contas de usuário robôs. BACKSTROM e LESKOVEC [39] aplicaram passeios aleatórios com pesos definidos por atributos da rede de amizades para sugerir conexão entre dois usuários de uma rede social. QIU *et al.* [40], por sua vez, desenvolveram uma técnica de representação de nós baseada em redes neurais convolucionais para prever a influência de usuários em redes sociais. Os exemplos citados reforçam que ferramentas de Ciência de Redes são capazes de acessar informações codificadas pela rede de usuários de mídias sociais e que essa informação é útil para diversas aplicações.

Capítulo 3

Processamento de Linguagem Natural

Neste capítulo serão apresentadas as técnicas de Processamento de Linguagem Natural que compõem um classificador textual, como a de análise de sentimento. Esse processo é comumente formado por 3 fases, como demonstrado no diagrama 3.1. As seções a seguir descrevem cada uma destas delas.

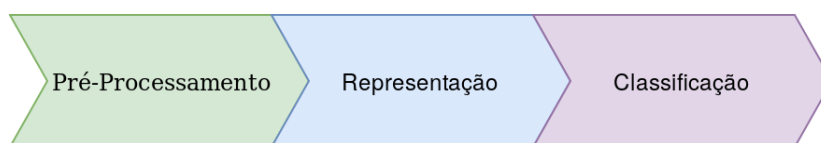


Figura 3.1: Etapas de classificadores de Processamento de Linguagem Natural.

3.1 Pré-Processamentos

A primeira etapa aplicada para a elaboração de modelos de NLP é o pré-processamento. Essa fase consiste na limpeza e preparação dos dados, visando a melhorar a performance do classificador, seja retirando ruídos dos textos, reduzindo o tamanho do vocabulário ou formatando o documento de maneira a facilitar a modelagem.

O volume do vocabulário adotado costuma ser limitado tanto pelos recursos computacionais, quanto pelo requisito mínimo de estatística das palavras na base de dados. Portanto, técnicas de pré-processamento que reduzam o tamanho total do vocabulário têm um importante papel na garantia de um bom funcionamento dos classificadores.

Como a maior parte dos modelos de NLP trabalham a nível de palavra, é necessário separar o documento em frases, com algoritmos como o Punkt [41], e, pos-

teriormente, em palavras. Esse processo, chamado *tokenização*, precisa ser robustos a abreviações, números e outras características específicas do idioma ao qual será aplicado, como a contração de palavras em português. Se tratando de redes sociais, também é relevante tratar os links, as *hashtags* e as menções a usuários.

Algoritmos de correção ortográfica [42][43] podem ser eficientes para aprimorar a qualidade dos textos, principalmente se tratando de mensagens providas de meios de comunicação como as redes sociais.

Técnicas de stemização consistem na extração do radical das palavras, como o obtido pelo algoritmo de Porter [44]. Um exemplo é dado com a palavra “montanha”, que possui radical “mont”, o mesmo obtido pela palavra “monte”. Por outro lado, o processo de lematização tem finalidade parecida, porém transforma a palavra em sua forma base, isto é, forma como ela aparece no dicionário, podendo então diferenciar palavras com o mesmo radical, como “banco” e “bancários”. Ambas as técnicas visam a tornar as etapas posteriores menos sensíveis a flexões gramaticais, além de colaborar na redução do vocabulário.

Uma das principais etapas do pré-processamento é a remoção das *stopwords*, conjunto de palavras que contêm informação pouco discriminante para uma dada aplicação [45]. Geralmente, elas são compostas pelas palavras mais comuns da língua, principalmente artigos e preposições. O objetivo da remoção das *stopwords* é diminuir ruídos dos dados textuais, assim simplificando a etapa de modelagem. SAIF *et al.* [46] fizeram um estudo comparando diversos métodos de seleção de *stopwords* e o impacto das mesmas na classificação de sentimento de *tweets*. A identificação de classe gramatical, em inglês *part-of-speech*, além de ser tipicamente utilizada como entrada de modelos de NLP, também pode ser útil para selecionar *stopwords*.

3.2 Representações

Uma vez que os tratamentos iniciais dos textos são feitos, chega-se à etapa de preparação desses dados para serem processados pelo modelo. Para isso, os documentos são transformados de sequências de palavras em vetores ou matrizes. Há diversas técnicas desenvolvidas com essa finalidade, que podem ser divididas em representações esparsas e densas. As representações esparsas são as mais simples de serem aplicadas dado que, em geral, não dependem do treinamento de nenhum modelo. Entretanto, elas resultam em vetores ou matrizes de dimensão na ordem de, pelo menos, o tamanho do vocabulário escolhido. Como os vocabulários costumam ser muito extensos (centenas de milhares de palavras em alguns casos), o tamanho e a esparsidade da representação obtida podem dificultar o treinamento dos classificadores. Para contornar essa dificuldade foram criados algoritmos de representações densas. Por transformarem os documentos em vetores ou matrizes de baixa di-

mensão, estes algoritmos foram os responsáveis pela viabilidade de utilização de técnicas de *Deep Learning* aplicadas ao processamento de linguagem natural. Nessa seção, descreveremos algumas das principais técnicas de representação de texto.

3.2.1 Codificação One-Hot

A codificação *One-Hot* representa cada palavra de maneira maximamente esparsa. Para tal, é definido um espaço vetorial em que cada palavra do vocabulário utilizado é equivalente a uma dimensão do espaço. Portanto, um documento pode ser transcrito dessa forma em uma sequência de vetores, ou matriz, em que cada palavra é um vetor com valor unitário na dimensão da própria palavra e zero nas outras, como mostra a Figura 3.2.



Figura 3.2: Vetores *One-Hot* de palavras de um dicionário. A dimensionalidade do vetor é igual ao tamanho do dicionário.

Frequentemente encontramos nas línguas palavras compostas ou expressões. Essas informações são perdidas na codificação *One-Hot*. Uma forma de se atenuar esse problema são com os chamados *n-gramas*. A ideia do *n-grama* é formar tokens de 2, 3, ou n palavras, resultando em uma dimensionalidade equivalente à quantidade de *n-gramas* do corpus formador do dicionário. Entretanto, o aumento no número de palavras por token também gera um aumento significativo do número de dimensões, dificultando o treinamento do classificador.

3.2.2 Bag-of-Words

A codificação *Bag-of-Words* é uma alternativa à representação de mensagens como matrizes compostas de vetores *One-Hot* de suas palavras. Esta é feita pela soma desses mesmos vetores [47]. Portanto, a representação final é dada por um único vetor, de tamanho correspondente ao do vocabulário utilizado. Esta técnica também tem a

vantagem de transformar documentos de tamanho variados em vetores de dimensões fixas, fator que precisa ser contornado em codificações baseadas em palavras.



Figura 3.3: Processo de representação por *Bag-of-Words* da frase “Preparando minha mochila para viagem.”. A palavra “para” é removida durante o pré-processamento por ser uma *stopword*.

Entretanto, a distribuição de palavras em um corpus, em geral, segue a lei de Zipf [48], ou seja, sua frequência segue uma distribuição em lei de potência. Sendo assim, mesmo retirando *stopwords*, as palavras mais comuns do vocabulário ainda dominarão os documentos e esse comportamento pode ser prejudicial para o treinamento dos modelos que serão menos expostos a palavras incomuns.

Para atenuar tal problema, pode-se aplicar o *term frequency-inverse document frequency* (TF-IDF) [49]. Nesse caso, a representação segue a mesma estrutura proposta pela codificação *bag-of-words*. Entretanto, cada palavra é ponderada por um multiplicador inversamente proporcional a sua frequência nos documentos.

Bag-of-words e TF-IDF são métodos de representação muito presentes na literatura pela sua simplicidade de implementação e pelos benefícios anteriormente descritos, em especial em conjunto com modelos como a SVM (Seção 3.3.2), que apresenta menos dificuldades de ser treinada em dados esparsos. No entanto, um componente fundamental da linguagem é perdido durante processo: o contexto em que cada palavra se insere. Pela representação agrupar todos os tokens em um único vetor, a ordem das palavras no documento é perdida, o que em muitos casos pode corresponder à inviabilidade de uma classificação precisa do mesmo.

As técnicas de representações densas que serão apresentadas a seguir, além de resolverem o obstáculo da dimensionalidade dos dados, também viabilizam a classificação por modelos que levem em conta o contexto de cada palavra.

3.2.3 Word2Vec

Word2Vec [18] foi uma das primeiras técnicas de representação densa de palavras amplamente adota pela indústria e pela academia. Representações densas são aquelas em que cada palavra é transformada em um vetor de números reais de baixa dimensionalidade, tipicamente dezenas ou poucas centenas de dimensões. MIKOLOV *et al.* [18] mostraram que essa representação é capaz de capturar parte dos sentidos semânticos e sintáticos das palavras, aproximando as que são sinônimas ou exerçam a mesma função gramatical, atenuando assim o problema da disparidade de frequência das palavras.

Esta técnica é um modelo constituído de uma rede neural de uma camada escondida. Para o treinamento do mesmo, são feitas janelas de um número arbitrário de palavras que servirão como entrada do modelo. Dessas janelas, há duas variações: o *continuous bag-of-words*, em que o modelo é treinado para prever a palavra central da janela de entrada a partir das outras palavras da janela, e o *skipgram*, que a partir da palavra central é treinado para prever o seu contexto. A Figura 3.4 ilustra a diferença entre esses métodos. As entradas e saídas do modelo são representadas pela codificação *one-hot* dos termos. A quantidade de neurônios escolhida para a camada escondida da rede resultará na dimensionalidade da representação das palavras. Estas, por sua vez, serão os pesos da camada de entrada do modelo treinado.

Apesar de o *Word2Vec* ser um modelo que precisa ser treinando, esse treinamento é não-supervisionado, dado que tanto as entradas quanto as saídas do modelo são obtidas diretamente dos documentos, sem anotações de classes. Desta forma, foi possível aplicar o algoritmo a grandes bases de dados mineiradas da internet. Esse tamanho volume de dados de treinamento foi essencial para que bons resultados demonstrados por MIKOLOV *et al.* [18] fossem alcançados.

Por ser a primeira representação densa de sucesso, o *Word2Vec* foi essencial para a aplicação de classificadores também baseados em redes neurais, como os de *Deep Learning*, que obtiveram grande êxito em diversas tarefas de processamento de linguagem natural. Outras técnicas de representação densas semelhantes e também amplamente adotadas na indústria foram desenvolvidas neste mesmo período de tempo, como o GloVe [19] e o FastText [50].

3.2.4 Representações por Redes Neurais Recorrentes

O sucesso do *Word2Vec*, baseado em redes neurais *feed forward*, inspirou a experimentação de outros tipos de redes neurais. Dentre elas, redes neurais recorrentes e suas variações obtiveram bons resultados na representação de palavras. Nesta seção, serão descritas as variações deste algoritmos.

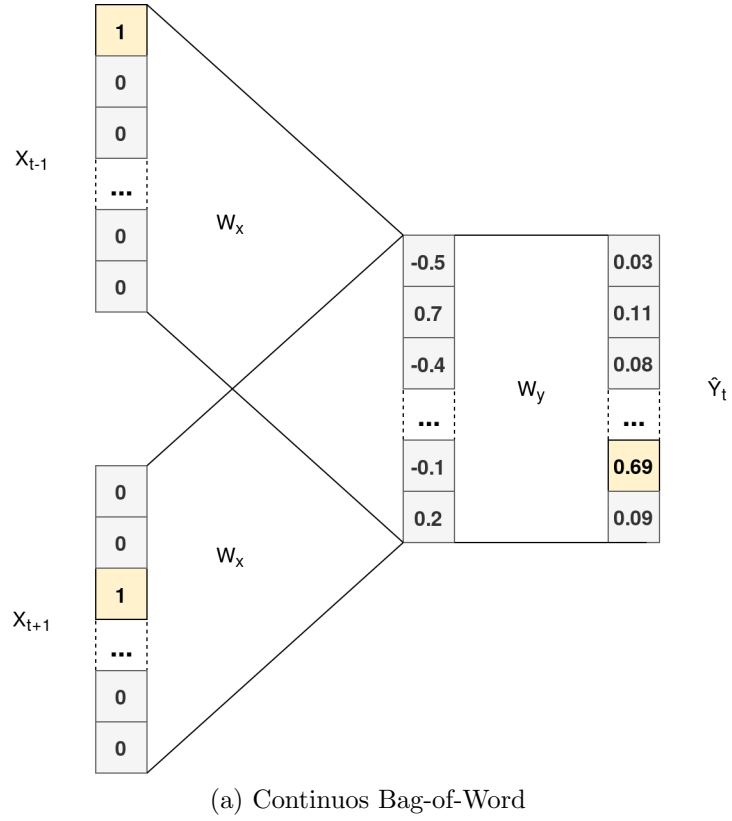


Figura 3.4: Esquemas de treinamento do Word2Vec. Exemplo composto por janela de contexto de 3 palavras. A matriz X_t representa a palavra da janela na posição t e \hat{Y}_t a predição do modelo para a mesma. W , por sua vez, representa o conjunto de pesos treináveis. A representação resultante de uma palavra é dada pelo seu referente vetor na matriz de pesos de entrada W_x .

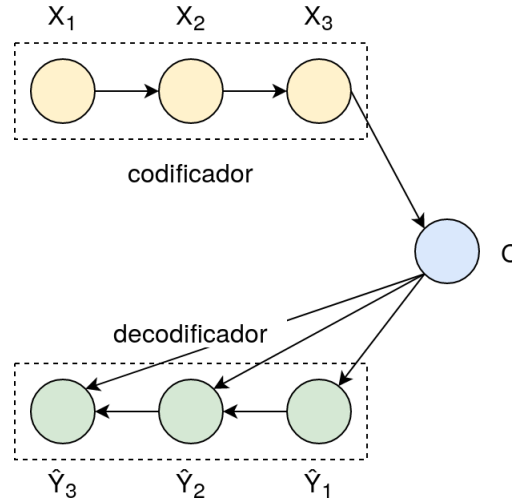


Figura 3.5: Arquitetura Encoder-Decoder, constituída por duas camadas de redes recorrentes: a de codificação e a de decodificação, ligadas por um conjunto de pesos C , que captura todo o contexto provindo da camada de codificação.

A estrutura chamada *Encoder-Decoder*, desenvolvida por CHO *et al.* [51], se tornou a base das principais representações por redes recorrentes. Ela consiste em uma rede neural recorrente dividida em duas etapas: a codificação, na qual uma sequência de tamanho variável é representada em um vetor de contexto; e a decodificação, em que esse mesmo vetor é utilizado para obter uma outra sequência de tamanho variável. O *Encoder-Decoder* é um modelo de aprendizado de sequências de tamanho variável a partir de outra sequência de tamanho variável como entrada, não necessariamente com os mesmos tamanhos. Isso é reflexo do problema inicial que o mesmo foi desenvolvido para atacar: a tradução de textos.

Toda essa estrutura é treinada conjuntamente e pode ser utilizada para gerar sequências de saída a partir de entradas, ou para avaliar um par de entrada e saída a partir da probabilidade $p_{\theta}(\mathbf{y} \mid \mathbf{x})$. Esta última sendo aprendida pelo treinamento, no qual \mathbf{x} e \mathbf{y} representam respectivamente as sequências de entrada e saída e θ o conjunto de pesos do modelo treinado.

A etapa de codificação desse modelo pode ser vista como um resumo da sequência de entrada em um vetor de tamanho fixo. Assim sendo, o mesmo pode ser utilizado para representações de documentos ou frases, ou até mesmo de palavras, caso aplicada uma sequência de tamanho unitário. CHO *et al.* [51] demonstram brevemente em seu trabalho que a codificação é capaz de capturar significados semânticos e sintáticos das palavras, assim como o Word2Vec.

ELMo

ELMo [20], sigla para *Embeddings from Language Models*, em português Representações para Modelos Linguísticos, é um modelo capaz de obter representação de palavras, e se diferencia de modelos de representação de palavras como o Word2Vec por levar em consideração o contexto da palavra para obter sua representação, tanto durante o treinamento, quanto durante a inferência. Isto é, uma palavra que possua múltiplos sentidos terá representações diferentes dependendo da frase em que está inserida. A ideia de utilizar o contexto da palavra para selecionar sua representação durante a inferência foi inspirado pelos trabalhos de PETERS *et al.* [52] e MCCANN *et al.* [53]. O modelo é composto de uma rede LSTM [54] bi-direcional e com múltiplas camadas, treinada de maneira não supervisionada a partir do objetivo de prever a palavra seguinte.

Estudos anteriores mostram que em modelos de redes recorrentes multicamadas, a melhor camada escolhida para representação das palavras depende da finalidade em que se deseja aplicá-la [55] [56] [57] [58] e que, em geral, camadas inferiores codificam melhor a informação sintática e camadas superiores a informação semântica. A proposta de ELMo é, para cada tarefa, treinar uma combinação linear das representações obtidas por cada camada da rede.

PETERS *et al.* [20] ressaltam que ELMo foi capaz de obter resultados significativamente melhores que as técnicas do estado da arte até então, considerando diversas tarefas do processamento de linguagem natural. Além de capturar informação sintática e semântica, ELMo se mostrou capaz de desambiguar o contexto de palavras, problema até então não resolvido pelos principais algoritmos de representação.

3.2.5 BERT

Outra técnica de representação bastante utilizada atualmente para representação de palavras é a BERT [21], *Bidirectional Transformers*. No entanto, antes de explicá-la, será apresentada abaixo a arquitetura Transformers, em que a mesma se baseia.

Transformers

Um dos principais problemas de redes neurais recorrentes é o “esquecimento”. O esquecimento é observado quando se perde influência das primeiras palavras de uma longa sequência ao se tentar prever a palavra no final da mesma. Essa situação decorrente das sucessivas funções de ativação dos neurônios entre as palavras do começo e do final do documento. Mesmo com o algoritmo LSTM [54], *Long-Short Term Memory*, que visa a atenuar esse efeito criando um peso adicional, chamado de portão (que controla o quanto os pesos de contexto são atualizados entre cada

iteração de palavras de uma sequência), o esquecimento continua sendo um fator limitante.

O mecanismo de atenção desenvolvido por BAHDANAU *et al.* [59] também tem como objetivo diminuir o efeito do esquecimento. Este consiste em um conjunto de pesos adicionais que, multiplicados pelos pesos de estados das iterações anteriores, formam os pesos de estado da iteração corrente, como também descrito em mais detalhes por LUONG *et al.* [60]. A atenção se tornou uma das principais adições feitas a redes recorrentes aplicadas a processamento de linguagem natural.

Inspirado nesse mecanismo, VASWANI *et al.* [26] propuseram a arquitetura Transformer para tradução de textos. Essa arquitetura consiste em um sistema de *Encoder-Decoder*, desta vez composto por redes neurais *feed forward* multi camadas. Nelas a dependência temporal entre as palavras de um documento é atacada apenas pelo sistema de *self-attention*, ou de atenção própria.

Enquanto o mecanismo de atenção se dá por um único conjunto de pesos, a *self-attention* é dividida em 3: os pesos de busca, de resposta e de valor. Cada palavra do vocabulário terá um de cada vetor citado acima associado a ela. Para cada palavra do documento, será feito um produto interno do seu vetor de busca e os vetores de resposta das outras palavras da sentença, resultando em um valor único de atenção entre cada par possível de palavras de uma sentença. Após calcular esse valor, será realizada uma operação de Softmax para que a norma do vetor de atenção seja unitária. Posteriormente, se multiplica o vetor resultante pelo vetor de valor da palavra de resposta. Essa operação tem a finalidade de diminuir a importância de palavras sem potencial discriminatório, como as *stopwords*. Assim são calculadas as entradas da camada *feed forward* do algoritmo.

A etapa de codificação do Transformer é composta por uma cascata de unidades formadas por uma camada de *self-attention* seguida de uma camada de rede *feed forward*. A Figura 3.6 ilustra uma camada da arquitetura descrita. Diferentemente das redes recorrentes, o Transformer requer um tamanho fixo de entrada, sendo um parâmetro a ser definido dependendo da base de treinamento escolhida. Apesar de as redes *feed forward* não possuírem um conceito de sequência, como as redes recorrentes, essa arquitetura provou ser eficiente em tarefas de processamento de linguagem natural, ao passo que possui treinamento significativamente menos custoso que as redes recorrentes [26].

Arquitetura BERT

O sucesso do Transformer em traduções inspirou sua aplicação em outras tarefas. RADFORD *et al.* [62] propuseram utilizar apenas a camada de decodificação do Transformer, junto com uma adaptação da sequência de entrada, para adequar a tarefa a ser realizada. O modelo é pré-treinado com uma quantidade massiva de



Figura 3.6: Diagrama da camada ℓ do Transformer com K cabeças de *self-attention* para uma palavra de índice i , dada a sentença S . As matrizes $Q^{k,\ell}$, $K^{k,\ell}$ e $V^{k,\ell}$ representam respectivamente os pesos de atenção de busca, resposta e valor.

Figura retirada de [61].

dados na predição da palavra seguinte e posteriormente é feito um ajuste fino com uma base de dados da tarefa desejada. Essa nova arquitetura foi capaz de propagar os ganhos de performance do Transformer para toda a gama de tarefas de NLP.

Entretanto, tanto o Transformer original quanto o de RADFORD *et al.* [62] são modelos unidirecionais, prevendo a palavra, ou sentença à direita, a partir do contexto à esquerda. DEVLIN *et al.* [21] defendem que, por serem unidirecionais, os modelos possuem capacidade limitada, principalmente em tarefas que utilizam a sentença inteira, sendo a análise de sentimento um exemplo. Proporam, então, um modelo bidirecional praticamente idêntico ao Transformer de RADFORD *et al.* [62] sendo sua única modificação ter o mecanismo de atenção considerando bidirecionalmente o contexto.

Além de representar palavras, o BERT pode ser usado diretamente como classificador, assim como o ELMo. DEVLIN *et al.* [21] mostram as diferenças entre as camadas do modelo quando utilizado como representação. Assim como nas arquiteturas apresentadas anteriormente, as diferentes camadas possuem aprendizado de características distintas da língua.

3.3 Classificadores

Nessa seção serão abordadas as diferentes estratégias para classificação de sentimento. As etapas descritas anteriormente lidam com a preparação dos documentos para a realização da classificação. Mesmo com complexidade dos métodos apresentados, os classificadores podem ser tão simples quanto contadores de palavras positivas e negativas. Podemos dividi-los em algoritmos baseados em dicionário e algoritmos de aprendizado de máquina [6], estando suas respectivas características descritas abaixo.

3.3.1 Algoritmos Baseados em Dicionários

Uma das técnicas mais simples para a classificação de sentimento é feita a partir da elaboração de um dicionário composto por palavras que tenham conotação positiva ou negativa. A formação de um dicionário de sentimento pode ser feita de forma manual [63] [64] ou de forma automática. Dentre as maneiras automáticas, temos a aplicada por HU e LIU [65], que é feita a partir de um conjunto inicial de palavras anotadas e de um dicionário de sinônimos e antônimos, tal qual o WordNet [66]. Um exemplo é selecionar as palavras semente “bom” e “ruim” e montar uma base de palavras recursivamente a partir dos seus sinônimos e antônimos. Formas similares de identificação de palavras com sentimento baseada em dicionários e palavras semente foram desenvolvidas por BLAIR-GOLDENSOHN *et al.* [67], RAO

e RAVICHANDRAN [68], HASSAN e RADEV [69], entre outros.

Existindo um dicionário, a classificação de polaridade de um documento é dada a partir de alguma função dos sentimentos das palavras que o compõem. É comum a inclusão de algumas características sintáticas como a identificação de adjetivos intensificadores e de identificação de negação, para ponderar a pontuação de sentimento de uma palavra ou expressão [6]. A função de classificação pode ser tão simples quanto uma média do sentimento das palavras ou, por exemplo, um cálculo de proximidade entre o conjunto de palavras do documento e as palavras “excelente” e “ruim”, representando as classes positivas e negativas, como propõe TURNEY [70].

3.3.2 Modelos de Aprendizado de Máquina Lineares

Naturalmente, as primeiras aplicações de aprendizado de máquina em análise de sentimento fizeram uso de modelos simples, como os lineares. PANG *et al.* [14] fizeram um dos primeiros estudos comparativos entre esses métodos. A seguir, serão descritos os dois principais modelos lineares utilizados em processamento de linguagem natural: Naïve Bayes e Máquina de Vetor de Suporte.

Naïve Bayes

O classificador de Naïve Bayes se baseia no teorema de Bayes (Equação 3.1) e na premissa de independência entre as dimensões da representação dos dados. Quando representamos um documento textual com, por exemplo, *bag-of-words*, a premissa de independência significa assumir que existe independência entre as palavras do documento. Apesar de não existir independência entre palavras na linguagem, tal modelo foi amplamente aplicado a tarefas de processamento de texto, devido a sua performance e simplicidade.

$$P(A | B) = \frac{P(A)P(B | A)}{P(B)} \quad (3.1)$$

SCHÜTZE *et al.* [71] descreveram a formulação matemática desse modelo, como será demonstrado. Será denominado \mathbf{X} o corpus de treinamento e \mathbf{x} um documento desse corpus. As classes a que os documentos pertencem, como positiva e negativa, são chamadas de c_k , na qual \mathbf{c} é o conjunto de classes. Portanto, substituindo as variáveis da equação 3.1 tem-se que a probabilidade de um documento pertencer a uma determinada classe é dada por:

$$p(c_k | \mathbf{x}) = \frac{p(c_k) p(\mathbf{x} | c_k)}{p(\mathbf{x})} \quad (3.2)$$

Como se assume independência entre as variáveis, tem-se que:

$$p(x_i \mid x_{i+1}, \dots, x_n, c_k) = p(x_i \mid c_k) \quad (3.3)$$

Logo, pode-se substituir $p(\mathbf{x} \mid c_k)$ pelo produtório de $p(x_i \mid c_k)$:

$$p(c_k \mid \mathbf{x}) = \frac{p(c_k) \prod_{i=1}^n p(x_i \mid c_k)}{p(\mathbf{x})} \quad (3.4)$$

A probabilidade $p(\mathbf{x})$ será constante, logo não influenciará o treinamento. Assim tem-se que a relação entre a probabilidade de um documento pertencer a classe é dada por:

$$p(c_k \mid \mathbf{x}) \propto p(c_k) \prod_{i=1}^n p(x_i \mid c_k) \quad (3.5)$$

Dessa forma, a tarefa da classificação, que é encontrar a classe de maior probabilidade, é representada por:

$$\hat{y} = \max p(c_k \mid \mathbf{x}) = \max_{k \in \{1, \dots, K\}} p(c_k) \prod_{i=1}^n p(x_i \mid c_k) \quad (3.6)$$

A dependência entre a predição e o conjunto de treino, então, é dada por $p(c_k)$ e $p(\mathbf{x} \mid c_k)$. Esses valores são obtidos por estimadores de máxima verossimilhança. O parâmetro $p(c_k)$ é dado pela proporção de vezes em que a classe aparece no conjunto de treino, tendo o conjunto de treino o tamanho m de documentos:

$$\hat{p}(c_k) = \frac{\sum_{i=1}^m [y_i = c_k]}{m} \quad (3.7)$$

Por sua vez, $p(x_r \mid c_k)$ é estimado pela proporção entre uma determinada característica x_r e o total de características do subconjunto de dados $\mathbf{X}_{\mathbf{c}_k}$ pertencentes a classe c_k :

$$\hat{p}(x_r \mid c_k) = \frac{\sum_{j=i}^{m'} \sum_{i=1}^n [x_{ji} = x_r]}{|\mathbf{X}_{\mathbf{c}_k}|} \quad (3.8)$$

O Naïve Bayes pode ser aplicado tanto a vetores inteiros da representação *bag-of-words* de frequência de palavras em um documento, quanto pelo seu equivalente binário que assinala ou não a presença de uma determinada palavra. Ao considerar a frequência de palavras, aplicamos o modelo multinomial, enquanto ao utilizar a representação de forma binária temos o modelo de Bernoulli. O modelo de Bernoulli é mais eficiente para documentos e vocabulários pequenos, ao passo que o multimodal se sobressai no caso oposto [71].

Apesar de a premissa de independência não ser verdadeira para a linguagem e não ser um bom estimador de probabilidade das classes [71], Naïve Bayes se mostra eficiente na classificação, em especial quando existe um conjunto de características

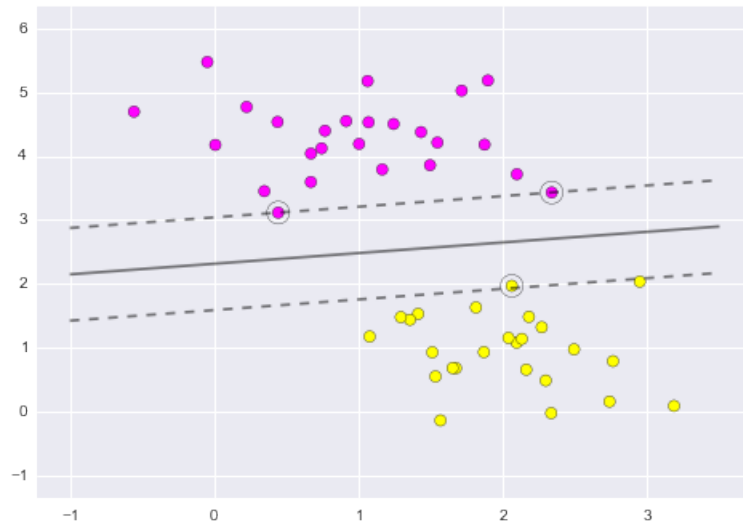


Figura 3.7: SVM classificando duas classes linearmente separáveis.
Figura retirada de [72].

de semelhante importância. Sua principal vantagem se dá pelo baixo custo computacional de treinamento, visto que seus parâmetros são estimados apenas por contagens na base de dados.

Máquina de Vetor de Suporte

A Máquina de Vetor de Suporte, também chamada de SVM, é um algoritmo que busca encontrar um vetor que melhor separe duas classes. A diferença entre esse algoritmo e outros classificadores, como a regressão linear, é que o SVM tem como objetivo obter o vetor que maximize a distância entre as margens das classes. A regressão linear, por sua vez, geralmente minimiza uma função custo baseada na distância entre os dados e o centro de massa das classes.

Entretanto, por se basear na margem das classes, o SVM não é capaz de separar classes que tenham sobreposição. Como maneira de contornar esse problema, foi criada uma variável de relaxamento de que controla o quanto de sobreposição é permitido entre as classes, como explicam CORTES e VAPNIK [73]. Ainda assim, até então a Máquina de Vetor de Suporte fica limitada a problemas lineares. Apenas com o desenvolvimento do chamado *kernel trick* é que o SVM pôde ser aplicado à problemas não-lineares, ganhando assim alta adoção. O *kernel trick* consiste em realizar uma transformação na representação dos dados de forma que os mesmos sejam linearmente separáveis após a transformação. Mapeamentos radiais e polinomiais são dois dos exemplos de *kernel* amplamente utilizados.

Uma das principais propriedades do SVM é que seu treinamento independe da dimensionalidade das características dos dados, visto que seu aprendizado é feito com

base nas margens entre as classes. Assim, uma vez que os dados sejam separados por uma margem, mesmo que tenham alta dimensionalidade, o vetor de suporte de classificação pode ser encontrado. A aplicação de SVM para classificadores de texto foi proposta por JOACHIMS [74], visto que dados textuais representados por *bag-of-words* possuem 3 propriedades que se adequam à classificação por SVM: *a)* possuem alta dimensionalidade *b)* são esparsos *c)* tem poucas dimensões irrelevantes. Apesar de JOACHIMS [74] demonstrar a efetividade do SVM para classificação de texto utilizando *kernels* não lineares, bons resultados são obtidos mesmo sem a aplicação de *kernel*, como mostram PANG *et al.* [14].

3.3.3 Modelos de Aprendizado de Máquina Não-Lineares

Classificadores lineares a partir de representação por *bag-of-words* ou TF-IDF dos textos foram o estado da arte em tarefas de processamento natural durante anos. Entretanto, como visto anteriormente, ao aplicar técnicas como *bag-of-words*, abre-se mão do contexto em que cada palavra aparece no documento. Tal fator limita a performance da classificação. Sem conseguir capturar o contexto, essas técnicas se distanciam de como interpreta-se a língua. Para diminuir essa lacuna, possibilitados pelo desenvolvimento de técnicas de representações densas, passou-se utilizar classificadores não-lineares que visam a adicionar alguma forma de contexto ao processo.

Deep Learning

Dentre os modelos não-lineares, os que mais se destacam em processamento de linguagem natural são os de *Deep Learning*, baseados em redes neurais. Uma das principais características das redes neurais é serem algoritmos capazes de aproximar qualquer função [75]. Com o desenvolvimento do método de treinamento por auto diferenciação, chamado de *backpropagation*, nos anos 80 [76] as redes neurais foram amplamente adotadas. Entretanto, dois principais fatores inviabilizaram o treinamento de redes neurais de larga escala por muitos anos: o custo computacional do *backpropagation* e o efeito do *vanishing gradient* [77]. O *vanishing gradient* é decorrente do efeito multiplicativo do gradiente desde a camada de neurônios da saída até a camada de entrada. O mesmo resulta em um treinamento menor da camada à medida em que ela está mais distante da saída, estabelecendo assim uma dificuldade exponencial no treinamento com relação ao número de camadas da rede.

Com o desenvolvimento tecnológico e a elaboração de uma técnica de treinamento camada-a-camada, foi possível se sobrepor a esta barreira. Nesse método, proposto por HINTON *et al.* [78], cada camada de neurônio é treinada individualmente e os valores obtidos por esse treinamento são utilizados como inicialização do treinamento da rede completa. Após esse trabalho, começaram a testar redes neurais com um

número cada vez maior de camadas. Denominou-se *Deep Learning* o conjunto de modelos de redes neurais profundas.

O salto de performance obtido pela aplicação de redes neurais profundas [22] em diversas áreas de conhecimento fez deste conjunto de técnicas um objeto de estudo de muito interesse por pesquisadores e para indústria. Parte de seu sucesso se atribui ao fato de que cada adicional na rede neural permite que a mesma obtenha uma representação mais complexa dos dados. Considerando o processamento de linguagem, diferentes tipos de redes neurais foram testados para tentar capturar a complexidade dessa informação, com diferentes abordagens para adicionar o contexto em que cada palavra se insere no processo de modelagem. As seções abaixo descrevem os principais tipos de redes neurais aplicados a NLP.

Redes Neurais Convolucionais

As redes convolucionais foram desenvolvidas prioritariamente para resolver problemas como a classificação de imagens e o reconhecimento de fala. Um fator comum que essas tarefas compartilham é a possível translação da informação. Na classificação de imagens, por exemplo, um determinado objeto pode estar em diferentes posições ou ocupar tamanhos distintos em diferentes imagens. Apesar de ser possível aplicar redes neurais MLP ao problema, esta precisaria aprender os mesmos padrões de neurônios em diferentes posições [79]. Além disso, nesse tipo de problema existe uma característica da localidade da informação. Por exemplo, para reconhecimento de fala de uma palavra no final de uma frase, a palavra imediatamente anterior é, em geral, mais importante que a primeira palavra da mesma. Ao se utilizar redes neurais MLP completamente conectadas entre camadas, despreza-se essa propriedade, incorrendo em um maior custo computacional sem que necessariamente se reflita em performance ou, pior, aumentando a chance de o treinamento resultar em *overfitting*.

A solução para a rede convolucional atacar o problema da correlação espacial da informação é utilizar filtros espaciais. Cada uma de suas camadas é formada por um conjunto de filtros que são compostos por neurônios. A iteração de treinamento da rede consiste no deslocamento de cada filtro por toda a dimensão da camada anterior. Desta maneira, os padrões locais são capturados por um mesmo filtro independente de onde os padrões apareçam nos dados de entrada. O deslocamento dos filtros durante o treinamento funciona como um compartilhamento de pesos, reduzindo a quantidade de parâmetros livres da rede e, logo, sua probabilidade de *overfitting*.

A interpretação de redes convolucionais aplicadas a processamento de texto não é muito diferente de sua utilização com imagens. Enquanto os filtros convolucionais exploram os padrões locais com filtros 2D de poucos pixels em imagens, no caso tex-

tual os filtros são unidimensionais e percorrem a sequência de vetores de palavras com uma janela, capturando assim parte do contexto em que cada palavra se insere. Esse modelo já havia sido aplicado a documentos representados com *bag-of-words*, como demonstrado por KALCHBRENNER *et al.* [80] e YIH *et al.* [81]. Entretanto, apenas quando utilizado em conjunto com a representação Word2Vec, como apresentou KIM [30], esse algoritmo ganhou tração nas tarefas de classificação de texto. A disponibilidade de modelos Word2Vec pré-treinadas, somada à representação densa que resulta em menor quantidade de parâmetros e menor probabilidade de *overfitting*, torna essa combinação acessível para treinamento mesmo em casos de grandes bases de dados, dessa forma resultando em performance superior, como mostra KIM [30].

Redes Neurais Recorrentes

Outra forma de encapsular o contexto presente na linguagem é utilizando redes neurais recorrentes. Elas são, em geral, treinadas com o algoritmo *Backpropagation Through Time* [82] (BPTT). Nesse algoritmo a rede recorrente se comporta como uma rede neural *feedforward* na qual cada camada representa um passo temporal nos dados de entrada. No caso dos dados serem textuais cada passo pode ser uma palavra. Portanto, ao aplicar esse algoritmo sobre o texto, o contexto sequencial do documento será capturado pela própria arquitetura da rede.

As variações de redes neurais recorrentes, como apresentado em 3.2.4, também realizam classificação em grande parte. Um exemplo desta aplicação é mostrado por TAI *et al.* [83] classificando sentimento por uma rede LSTM. Posteriormente, técnicas de classificação que misturam diferentes tipos de redes foram testadas com êxito, como mostram ZHOU *et al.* [31], sendo a arquitetura constituída de uma camada de rede recorrente seguida por uma camada de rede convolucional.

Mesmo com os bons resultados obtidos nesses trabalhos, o treinamento por BPTT é computacionalmente mais custoso que o *backpropagation* de redes *feedforward* ou convolucionais, dado que o BPTT apresenta menos oportunidade de paralelismo do treinamento. Assim, esse algoritmo acaba sendo menos utilizado do que as redes convolucionais, ou até mesmo do que os Transformers, para a classificação de documentos.

Capítulo 4

Modelos de Redes Complexas

Grafos são estruturas de dados que codificam relações e que estão presentes em uma grande variedade de cenários. São compostos por nós que representam elementos, quaisquer que sejam, e arestas que são as relações entre os elementos. Esses elementos podem ser de um único tipo, formando grafos homogêneos, como por exemplo uma rede de amigos em que todos os elementos são pessoas. Os grafos heterogêneos são os que possuem vértices de mais de um tipo, neste caso, um exemplo é grafo de cinema no qual tem como vértices tanto filmes quanto atores. Em relação às arestas, quando estas representam uma ligação de sentido único são formados grafos direcionais. Um grafo de empréstimos financeiros entre um grupo de pessoas será um grafo direcional, enquanto a rede de pessoas que se abraçaram será não-direcionada. As arestas também podem ter pesos distintos. Também podemos categorizar o grafo como estático caso ele não sofra alterações, ou dinâmico, como os grafos que evoluem com o passar do tempo.

Tratando-se de redes sociais, grafos são especialmente importantes, dado que a relação entre os usuários é a principal componente desses serviços. Portanto, ao estudar as redes sociais é essencial entender essas relações. Há diversas maneiras de representar essas redes, sendo a mais comum formando um grafo homogêneo em que cada usuário é um vértice e as arestas codificando as interações entre os mesmos.

Há quatro principais categorias de modelos de redes: categorização de nós, predição de arestas, categorização do grafo e detecção de comunidade. A categorização de nós é a tarefa que classifica cada nó em uma ou mais classes. Um exemplo comum na literatura é a classificação de artigos acadêmicos em áreas de conhecimento [84]. A predição de arestas consiste em dado um par de vértices determinar a probabilidade de existir uma aresta entre eles. Uma aplicação dessa tarefa no contexto de redes sociais é a recomendação de conexão entre usuários. A detecção de comunidade consiste no agrupamento de nós de uma rede, identificando os conjuntos de nós que tem maior conexão entre si. Em algumas circunstâncias o que se tem interesse é entender alguma propriedade formada pelo grafo inteiro. Nesses

casos a categorização de grafos classifica a rede como um todo em uma ou mais classes. Este método pode ser usado, por exemplo, na classificação de função de proteínas a partir de sua estrutura molecular [85]. Além dessas quatro categorias, a literatura de redes complexas apresenta outras variedades de modelos como o de propagação de epidemias em redes, modelagem de evolução de grafos dinâmicos, entre outros.

Devido ao foco desse trabalho na representação de usuários de redes sociais, serão estudadas nesse capítulo técnicas de representações de vértices. Assim como no caso de documentos textuais, esses modelos geralmente visam a codificar a informação de um nó em um vetor de baixa dimensionalidade, capturando tanto seus atributos individuais, quanto os decorrentes da estrutura de conexões do mesmo. Desta forma, essa informação poderá ser combinada com a representação das mensagens e assim treinar um classificador que considere ambas as partes para realização da tarefa desejada.

4.1 Modelos Baseados em Fatoração Matricial

Uma das forma mais comuns de descrever um grafo é por suas matrizes, sendo a principal delas a matriz de adjacência. A matriz de adjacência A é uma matriz quadrada de dimensionalidade $n \times n$, sendo n o número de vértices, na qual o elemento a_{ij} tem valor 1 caso o vértice v_i seja conectado ao v_j e 0 caso contrario. Há outras matrizes extraídas dos grafos, como a matriz Laplaciana e a matriz de centralidade de Katz.

Essas matrizes podem ser exploradas para extração de informação das redes. Os modelos de representação de vértices por fatoração matricial foram um dos primeiros métodos aplicados. Nos casos em que a matriz é positiva e semi-definida, como a matriz Laplaciana, é possível aplicar a decomposição em autovalores para obter uma representação. Nos outros casos, em geral, são utilizados métodos iterativos visando a minimizar uma função custo relacionada à qualidade da representação obtida [86]. Serão explicados em seguida um conjunto de técnicas baseados na fatoração de matrizes.

Locally Linear Embedding (LLE)

Um dos modelos de representação mais simples é o desenvolvido por ROWEIS e SAUL [87]. Esse algoritmo considera a premissa de que um nó é uma combinação linear de seus vizinhos. Portanto, tendo a matriz de adjacência W_{ij} e o vetor de representação de um nó como Y_i , definimos o mesmo como:

$$Y_i \approx \sum_j W_{ij} Y_j \quad (4.1)$$

Assim, função custo a ser minimizada é a que minimiza o erro de representação do somatório de todos os nós, de forma que a função custo $\phi(Y)$ é dada por:

$$\phi(Y) = \sum_i |Y_i - \sum_j W_{ij} Y_j|^2 \quad (4.2)$$

Sendo necessário respeitar a restrição $\frac{1}{N} Y^T Y = 1$ para remover as soluções degeneradas. Essa técnica obtém uma representação de nós que preserva as distância de primeira ordem da rede, ou seja, vizinhos diretos no grafo original vão obter representações próximas.

Automapas Laplacianos

Esta técnica, desenvolvida por BELKIN e NIYOGI [88], também tem como objetivo obter representações próximas entre vizinhos. No entanto, desta vez é inspirada na equação de fluxo de calor, na qual a transmissão de energia decai quadraticamente com a distância entre os corpos. Para isso, é utilizada a matriz laplaciana do grafo. Sua função custo é dada pela equação 4.3, na qual L é a laplaciana do grafo.

$$\begin{aligned} \phi(Y) &= \frac{1}{2} \sum_{ij} |Y_i - Y_j|^2 W_{ij} \\ &= \text{tr}(Y^T L Y) \end{aligned} \quad (4.3)$$

HOPE

Os algoritmos apresentados anteriormente são efetivos para manter a distância entre vizinhos de um grafo não-direcionado. Entretanto, grafos direcionados reais têm, em grande parte das vezes, sua distribuição de grau obedecendo a uma lei de potência, como é o caso dos grafos formados em redes sociais. Não apresentam, assim, propriedades simétricas entre pares de vértices. Para obter representações que respeitem essa assimetria OU *et al.* [89], desenvolveram o algoritmo *High-Order Proximity preserved Embedding* (HOPE). A chave deste algoritmo é utilizar métricas de proximidade de ordem superior em sua função custo, preservando assim as propriedades da assimetria. Como qualquer medida de proximidade pode ser aplicada nesse algoritmo, OU *et al.* [89] testaram medidas como a proximidade de Katz e Pagerank personalizado, mostrando que em vários casos essas medidas possuem aproximações que diminuem o custo computacional do algoritmo.

HOPE representa cada vértice em 2 vetores, um de entrada e um de saída, que denominaremos respectivamente Y^s e Y^t . A função custo é dada pela equação 4.4,

na qual S representa a matriz de proximidade entre os nós, dada por qualquer uma das medidas escolhidas. O termino da otimização resulta no par de vetores, de entrada e saída, que representam cada um dos nós da rede.

$$\phi = \|S - Y^s Y^{t^T}\|_F^2 \quad (4.4)$$

4.2 Modelos Baseados em Passeios Aleatórios

Outras estratégias amplamente adotada são as baseadas em Passeios Aleatórios. O Passeio Aleatório é um processo estocástico que consiste em selecionar um vértice inicial no grafo e completar uma sequência de passos aleatórios pelos seus vizinhos. Comos o resultado do Passeio Aleatório é uma sequência de vértices, podemos utiliza-lo como uma função de proximidade entre os nós. Com essa finalidade costuma-se realizar múltiplos Passeios Aleatórios de tamanho fixo partindo de cada vértice da rede, formando um conjunto de realizações.

Por ser um método iterativo e não necessitar o cálculo de nenhuma matriz completa do grafo, esse método apresenta vantagem em custo operacional para redes grandes, dado que essas matrizes tem custo quadrático em memória. Além disso, a iteratividade do método também permite que novos elementos sejam adicionados ao grafo sem requerer um re-treinamento completo do modelo, característica essencial para seleção de algoritmos em diversas aplicações. A seguir serão apresentados os principais modelos deste grupo.

Deepwalk

Deepwalk [90] se inspira em modelos de representações de palavras, utilizando como entrada sequências de iterações de Passeios Aleatórios truncadas em poucos passos. Os autores observaram que, assim como a distribuição de ocorrência de palavras de uma língua segue uma lei de potência, o mesmo comportamento é observado em boa parte das redes formadas no mundo real, como a de conexão entre pessoas. Desta forma, eles adaptaram algoritmos previamente aplicados em predição de palavras para obter as representações dos vértices.

O modelo proposto é muito similar à variação *skipgram* do Word2Vec 3.2.3: os passeios aleatórios truncados formam sequências de tamanho $2k + 1$, e o treinamento do algoritmo é feito de maneira a maximizar a probabilidade de predição do contexto a partir do vértice na posição central do passeio. A fórmula 4.5 formaliza o problema citado. Dado que v_i é o vértice central de um certo passeio e Φ é o mapeamento de um vetor em sua representação, que queremos encontrar, então:

$$\max P(\{v_{i-k}, \dots, v_{i-1}, v_{i+1}, \dots, v_{i+k}\} | \Phi(v_i)) \quad (4.5)$$

O processo é feito de forma iterativa por conta de cada realização do Passeio Aleatório compor uma entrada do algoritmo. A otimização é dada por gradiente descendente da função custo ϕ que maximiza a probabilidade 4.5, que é apresentada na equação 4.6. Outras adaptações propostas posteriormente ao Word2Vec para diminuir o custo computacional de treinamento também podem ser adotadas no Deepwalk, como *softmax* hierárquico e *negative sampling*.

$$\phi = -\log P(\{v_{i-k}, \dots, v_{i-1}, v_{i+1}, \dots, v_{i+k}\} | \Phi(v_i)) \quad (4.6)$$

node2vec

O node2vec [91] também é um algoritmo inspirado no Word2Vec. Seu processo de otimização é idêntico ao apresentado pelo Deepwalk, também aplicado sobre sequências de passeios aleatórios de tamanho fixo. A diferença entre ambos está na estratégia de passeio aleatório, que além de capturar a vizinhança de um dado vértice, visa também a capturar a função do mesmo na estrutura ao seu redor. Enquanto a vizinhança de um nó compõe uma parte de sua informação, o posicionamento do mesmo em sua comunidade também é relevante. Dentro de uma rede, um nó central de uma comunidade pode apresentar características mais semelhantes ao ponto central de outra comunidade do que a algum vértice vizinho a ele que tenha poucas conexões.

A proposta desenvolvida por GROVER e LESKOVEC [91] consiste em um Passeio Aleatório enviesado que parametriza quão relevante é cada um desses dois comportamentos. Ao se escolher parâmetros que priorizem a vizinhança do vértice, a amostragem do Passeio Aleatório enviesado passa a se assemelhar a uma busca em profundidade. Por outro lado, focando-se na estrutura formada pelas conexões do nó, obtemos uma amostragem semelhante à busca em largura.

No Passeio Aleatório não enviesado, a probabilidade de transição π_{uv} do nó u para o nó v é dada por $\pi_{uv} = \frac{w_{uv}}{\sum w_u}$, no qual w_{uv} é o peso da aresta entre u e v e $\sum w_u$ é o somatório dos pesos das arestas de u , neste caso operando como regularizador da probabilidade. Para atingir seu objetivo, node2vec multiplica a probabilidade de transição original do Passeio Aleatório por um fator $\alpha(s, v)$ parametrizado pelos parâmetros p e q , como mostra a equação 4.7. Nela, s representa o vértice em que o Passeio Aleatório amostrou no passo anterior ao passo de posição u , enquanto d_{sv} representa a distância mínima entre o par de vértices s e v . Esses parâmetros controlam quão rápido o Passeio Aleatório explora a rede.

$$\alpha(s, v) = \begin{cases} \frac{1}{p}, & \text{se } d_{sv} = 0 \\ 1, & \text{se } d_{sv} = 1 \\ \frac{1}{q}, & \text{se } d_{sv} = 2 \end{cases} \quad (4.7)$$

O parâmetro p é chamado de parâmetro de retorno. A escolha de valores altos para esse fator faz com que o Passeio Aleatório evite re-amostrar vértices recém amostrados, enquanto valores baixos forçam o algoritmo a manter a amostragem localizada em uma distância pequena de seu ponto de partida. Por sua vez, o parâmetro q regula a probabilidade de se amostrar nós distantes do vértice de partida. Valores baixos de q resultam em uma amostragem semelhante à busca em profundidade enquanto valores altos levam a um padrão de busca em largura. Resumidamente, a probabilidade de transição do passeio aleatório proposto por node2vec é dada pela equação 4.8.

$$\pi(s, u, v) = \frac{w_{uv}\alpha(s, v)}{\sum_x w_{ux}\alpha(s, x)} \quad (4.8)$$

Os autores mostram que a adoção dessa estratégia de amostragem resultou em ganhos significativos de performance do algoritmo para tarefas de classificação de vértices. Apesar da maior complexidade, o algoritmo mantém o aumento linear do custo computacional com relação ao número de nós da rede, possibilitando sua aplicação em redes de grande volume.

4.3 Redes Convolucionais de Grafos

O sucesso de *Deep Learning* em outras áreas de estudo, como em processamento de linguagem natural, também impactou o estudo de redes complexas. Ao longo da última década observamos adaptações desses modelos para serem usados em grafos, como por exemplo os *autoencodes* [92][93] e redes recorrentes [94][95]. Dentre eles as variações de redes convolucionais aplicadas a grafos se tornaram o principais algoritmos deste grupo, superando em performance os algoritmos mais utilizados a época em varias tarefas. Assim como uma rede convolucional convencional, o treinamento dessas redes é supervisionado, feito por *back-propagation* de uma função custo específica da tarefa.

Para tal, se faz necessária uma adaptação do operador de convolução para aplicação em grafos. Estas adaptações são divididos em dois tipos [96]: convoluções espectrais [97][98] e convoluções espaciais [99]. A convolução espacial desenvolvida por KIPF e WELLING [99], chamada rede convolucional de grafos (GCN) se tornou uma das mais populares. A equação 4.9 define a camada de convolução usada nesse algoritmo.

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l W^l) \quad (4.9)$$

A matriz $\tilde{A} = A + I$ é a matriz de adjacência somada à matriz identidade, correspondente a um grafo com um laço ¹ adicionado a cada vértice. A matriz diagonal \tilde{D} é composta pela diagonal $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, sendo que H^l representa a saída da camada l e W^l representa os pesos treináveis da mesma. Por fim, temos a função de ativação escolhida σ . Os autores mostram em seu trabalho a relação desta camada com a aproximação de primeira ordem da convolução espectral do grafo.

Apesar do GCN não ser especificamente voltado para obtenção de representações, os filtros convolucionais obtidos após o treinamento podem ser utilizados como uma. Uma dificuldade deste modelo é que, assim como os modelos de fatoração matricial, só é possível aplicar GCN em grafos fixos, pois ao precisar adicionar novos vértices, precisa-se retreinar o modelo. Dentre os algoritmos que se propõem a atacar esse problema, destacamos o GraphSAGE [100], que obtém a representação de um vértice a partir de uma função de agregação de seus vizinhos, conseguindo por sua vez incorporar novos nós mesmo que estes não tenham participado do treinamento.

Por se tratar de um método treinado supervisionado este vai contra o objetivo deste trabalho de reduzir a necessidade de anotação de dados. Porém, VELIČKOVIĆ *et al.* [101] desenvolveram um método, chamado Graph Infomax, que visa permitir o treinamento não supervisionado mesmo para modelos originalmente supervisionados. Este tem como objetivo maximizar a informação mútua entre representações globais e locais do grafo, codificando assim informação referente tanto a rede inteira quanto a vizinhança de cada vértice.

¹aresta conectando um vértice a ele mesmo

Capítulo 5

Método

Este capítulo descreve a elaboração de classificadores de sentimento de tweets usando técnicas de processamento de linguagem natural e ciência de redes. Serão detalhados as etapas de coleta de base de dados, as figuras de mérito utilizadas nas avaliações dos modelos e o processo de treinamento dos algoritmos.

5.1 Bases de Dados

Serão utilizadas duas bases de dados nesse trabalho, uma de treinamento e uma de teste. As bases serão formadas por tweets captados utilizando a API ¹ do Twitter, a qual fornece amostras de mensagens públicas, com a possibilidade de filtrar pelas que contenham palavras-chave de interesse.

As mensagens de texto receberão alguns pré-processamentos antes de formarem as bases de dados. O primeiro passo será aplicar a tokenização, quebrando a mensagem em sequência de palavras e removendo as pontuações. Posteriormente as palavras serão postas em forma minúscula. Por fim, menções a usuários e hiperlinks serão substituídos por tokens especiais.

Tendo objetivo de diminuir o custo de desenvolvimento do classificador textual será adotado o método de anotação distante descrito por GO *et al.* [1] para formação da base de dados de treinamento, usando emoticons como característica de anotação. Serão selecionados manualmente emoticons positivos e negativos dentre os mais frequentes nos textos captados. As mensagens que possuírem um ou mais dos emoticons selecionados serão categorizadas com a classe correspondente na base de treinamento. Para remover possíveis ambiguidades, serão removidos da base os textos que possuírem mais do que uma categoria. Além disso, os emoticons utilizados na supervisão distante serão removidos dos textos de treinamento para evitar a introdução de vies no classificador.

¹<https://developer.twitter.com/en/docs/twitter-api>

A base de testes será composta de tweets anotados manualmente. Esta será utilizada para avaliar os modelos treinados com os métodos semi-supervisionados. Para esta finalidade, serão reservadas mensagens de um conjunto de usuários selecionados aleatoriamente para compor essa base. Esta separação é importante para evitar vies pois apesar de os métodos de representação de usuários serem não supervisionados, como será descrito na Seção 5.3, essa representação será utilizada no treinamento do classificador. As mensagens destes usuários serão disponibilizadas para anotação utilizando a ferramenta Doccano ². Por fim, as mensagens anotadas comporão a base de dados de teste.

5.2 Figuras de Mérito

A avaliação dos algoritmos aplicados nestas bases será feita utilizando as figuras de mérito descritas nessa seção.

Levando em consideração o desbalanceamento de classes das bases de dados não é adequado a aplicação da acurácia pois esta favoreceria a classe majoritariamente presente nas bases. Como alternativa será adotada a área sob a curva ROC (*receiver operating characteristic*). A curva ROC apresenta a relação entre probabilidade de detecção e falso alarme para diferentes patamares de decisão de classificadores binários. A área sob a curva ROC é uma medida que agrupa esse conjunto de taxas em um número único, que representa o modelo em todos possíveis pontos de operação.

Como o objetivo abordado não apresenta um ponto de operação pré-definido, esse também pode ser definido pela otimização de uma figura de mérito. Desta forma, A seleção do limiar do classificador treinado será feita a partir do índice SP [102]. Este índice consiste na média geométrica entre os pontos de média geométrica e média aritmética entre a probabilidade de detecção da classe positiva P_c e a probabilidade de não obtenção de falso alarme P_{nc} , como mostra a Equação 5.1.

$$SP = \sqrt{\sqrt{P_c P_{nc}} \left(\frac{P_c + P_{nc}}{2} \right)} \quad (5.1)$$

5.3 Desenvolvimento

O objetivo deste trabalho é desenvolver classificadores de análise de sentimento de tweets sem a necessidade de anotações de dados e que explorem a multi-modalidade do problema. Desta forma, serão dívidos os experimentos em duas etapas. A primeira considerando apenas o texto das mensagens, serão avaliadas as técnicas descri-

²<https://github.com/doccano/doccano>

tas no Capítulo 3. Posteriormente, serão utilizadas as representações das mensagens e as representações da rede de usuários conjuntamente. Desta forma analisaremos se a adição de informação do usuário infere em alguma vantagem de performance no classificador.

5.3.1 Classificadores de Processamento de Linguagem Natural

Seguindo os classificadores apresentados no Capítulo 3, o primeiro algoritmo aplicado será o Naive Bayes. As mensagens serão representadas por Bag-of-Words e, por isso, será utilizado Naive Bayes com distribuição multinomial. Será utilizada a suavização de Laplace como regularizador do modelo. O valor do parâmetro de suavização será selecionado de maneira a maximizar a área sob a curva ROC (AUC). A validação cruzada por K-partições servirá para maximizar esse valor, sendo selecionadas 10 partições.

Posteriormente será treinado o classificador de SVM, também com representação de Bag-of-Words. Dado o grande volume de dados e a complexidade computacional será utilizado a variação de SVM por método de gradiente como proposto por SUYKENS e VANDEWALLE [103]. Neste caso, será utilizada regularização L_2 , assim como no classificador de Naive Bayes, este parâmetro será selecionado por validação cruzada de K-partições de 10 partições, de maneira a maximizar a AUC. Tanto o modelo de Naive Bayes quanto o SVM foram treinados utilizando as implementações disponibilizadas pela biblioteca Scikit-Learn [104].

Para os modelos CNN e LSTM usaremos como representação o Word2Vec. O treinamento do Word2Vec será feito a partir das mensagens amostradas do Twitter, as mesmas mensagens utilizadas para formar a base de dados de treinamento por supervisão distante, seguindo os mesmos pré-processamentos. O modelo será treinado com o método *continous bag-of-words* com amostragem negativa de 5 palavras, terá tamanho de janela de contexto 5 e dimensionalidade 100 do vetor de representação. O treinamento será feito utilizando a biblioteca Gensim [105].

O treinamento dos classificadores de *Deep Learning* por CNN e LSTM serão treinados de formas similares. Ambos terão como entrada os textos representados por Word2Vec. Como as redes convolucionais dependem de tamanhos fixos de entrada, as mensagens serão limitados a um tamanho máximo de *tokens* definido pelo conjunto de dados de treino. As mensagens de tamanho menor que o tamanho máximo definido serão preenchidas com vetores nulos.

Nesses classificadores, além das camadas convolucionais e LSTM, respectivamente, uma camada será adicionada com um único neurônio para realização da classificação. Por serem modelos de alto custo computacional, não será realizada busca

em *grid*. Os hiperparâmetros selecionados foram baseados nos trabalhos apresentados no Capítulo 3 que exploram o uso dessas técnicas em processamento de texto. A rede convolucional será composta por três camadas convolucionais em paralelo, cada uma com 100 filtros convolucionais com tamanhos respectivamente de 3, 4 e 5 tokens. O classificador de LSTM, por sua vez, conterá duas camadas bi-direcionais em série de 100 neurônios cada. Ambos os modelos utilizaram *dropout* durante o treinamento com probabilidade de 50% de desligamento do neurônio e regularização L2 de fator 10^{-3} . Serão realizados 10 treinamentos de cada modelo, dado que o treinamento de redes neurais é propenso a estagnar em mínimos locais. Como estes métodos tem alto custo computacional, não será utilizada a validação de K-partições. Será selecionado aleatoriamente um conjunto de validação composto por 20% dos dados de treinamento, sendo este usado para seleção dos hiperparâmetros. Para reduzir o efeito de *overfitting* será aplicado o *early stopping*.

Por fim, será avaliada a representação de palavras pela técnica ELMo. Como este modelo contém um número elevado de parâmetros, será utilizado um conjunto de pesos pré-treinados³, os quais foram treinados em textos em português retirados da Wikipedia⁴. O modelo gera vetores de representações de tamanho 1024 e foi treinado com duas camadas. Será utilizada apenas a camada superior do modelo, dado que para tarefa de análise de sentimento se faz mais relevante a informação semântica das mensagens. Em conjunto com essa representação, será utilizado o modelo de CNN para classificação do texto. Este seguirá o mesmo método do anterior, limitando as mensagens em tamanho, preenchendo com vetores nulos e otimizando hiperparâmetros por busca em *grid*.

Em virtude do alto custo computacional do modelo BERT apresentado no Capítulo 3 não será possível o treinamento do mesmo para comparação, apesar dos bons resultados obtidos em seu artigo de referência.

Dessa forma se concluirá a comparação de desempenho dos algoritmos clássicos de processamento de linguagem natural com as técnicas neurais desenvolvidas no decorrer da última década. Neste estágio foram analisadas apenas os dados textuais das mensagens, a seguir será apresentado o método que unificará a informação provinda dos textos com as informações obtidas do usuário.

5.3.2 Classificadores Multimodais

A primeira etapa para formação dos classificadores multimodais é o treinamento não-supervisionado das representações de usuário. Serão empregadas 3 das técnicas apresentadas no Capítulo 4, uma de cada um dos tipos de modelos. O grafo formado será homogêneo em que os usuários são codificados como vértices e os *retweets*,

³<https://allennlp.org/elmo>

⁴<https://www.wikipedia.org>

operação de compartilhamento de uma mensagem de outro usuário, formam uma aresta direcional entre ambos.

A rede obtida é formada por aproximadamente 5,5 milhões de nós e 28 milhões de arestas. Como elucidado no Capítulo 4, a maioria dos algoritmos de representação de vértices tem grandes requisitos de memória. Portanto, algumas restrições foram necessárias para viabilizar o treinamento dos modelos. Cada modelo tem sua limitação própria, mas será seguida a mesma estratégia para atacar esse problema. O grafo original será reduzido para o sub-grafo composto pelos nós da maior componente fortemente conexa, ou seja, o maior sub-grafo em que todos os nós apresentam caminhos entre si. Este procedimento reduz o grafo a um total de x vértices e y arestas. Ainda assim, nos casos em que esta rede ainda seja grande demais para viabilizar o treinamento, serão selecionados aleatoriamente vértices, garantindo apenas que os vértices presentes na base de teste estejam incluídos na rede a ser treinada. Para facilitar a comparação, todas as técnicas usaram o mesmo tamanho de *embedding*.

O primeiro método para representação dos vértices avaliado será o modelo de fatoração matricial *Locally Linear Embedding*. Este utiliza a matriz de adjacência para criar representações que aproximem nós vizinhos.

Em seguida, será aplicado o node2vec como representante dos algoritmos baseados em passeios aleatórios. Este será treinado com parâmetros fixos. Os valores de p e q serão 1. Os passeios aleatórios terão 80 passos e serão realizados 10 inicializações aleatórias por cada nó. O treinamento usará janelas de contexto de tamanho 10 e serão treinados por 3 épocas.

Por fim, será avaliado o método de redes convolucionais aplicadas a grafos. Como o modelo GCN é um modelo supervisionado, será aplicado a técnica Graph Infomax para treinamento não-supervisionado. Outra adaptação necessária é que enquanto os outros métodos dependem apenas da estrutura da rede, o GCN usa também características dos vértices para treinamento da representação. Se tratando de redes sociais há várias possibilidades de característica para ser exploradas como número de contatos, descrição do perfil, foto, entre outras. Entretanto, essas abordagens não serão exploradas nesse trabalho, usaremos como característica o grau do nó na rede.

Desta forma, haveremos treinado os classificadores textuais e as representações dos usuários, restando portanto a testar se a combinação dos mesmos dos mesmos infere em algum aumento de performance. Serão retirados os neurônios da última camada, a camada de classificação, dos modelos de CNN e LSTM já treinados. Serão concatenadas as camadas restantes dos modelos textuais com as representações de usuários e adicionada uma nova camada de classificação. A figura 5.1 demonstra a arquitetura final resultante da composição dos dois classificadores. O modelo será treinado novamente nos dados anotados por supervisão distante, com os pesos

inicializados pelos modelos pré-treinados. Concluindo-se, portanto, o conjunto de experimentos realizados nesse trabalho.

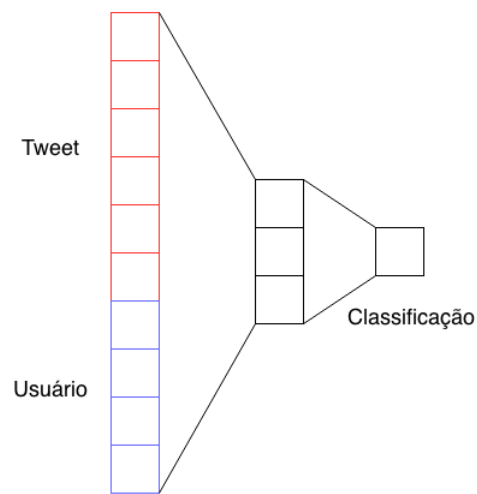


Figura 5.1: Arquitetura composto pela combinação de classificadores de informação textual e de usuário concatenados. Adicionou-se também uma camada de neurônios que será treinado para combinar as representações.

Capítulo 6

Resultados e Discussões

A etapa de coleta de dados captou aproximadamente 9,2 milhões de tweets únicos entre 2018 e 2019. Parte desses tweets foi coletada da amostragem do conjunto total de mensagens publicadas e outra parte da amostrada por dentro os tópicos: *Libertadores*, *ENEM*, *Amazônia* e *Rock in Rio*. Nessa base há aproximadamente 45 milhões de retweets, fornecendo a conexão entre pessoas que será utilizada para montar a rede de usuários.

Para anotação automática foram identificados os emoticons mais frequentes da base de dados e classificados manualmente entre positivos e negativos, desconsiderando os neutros. A Tabela 6.1 mostra as classes dos emoticons selecionados. Com esse conjunto de emoticons 580 mil tweets foram anotados por supervisão distante, dentre os quais 130 mil marcados como negativos e 450 mil marcados como positivos.





































Classe	Emoticons
Positiva	                  
Negativa	                

Tabela 6.1: Emoticons selecionados para aplicação de supervisão distante.

A rede de usuários foi formada por retweets entre usuários. Os 45 milhões de retweets formam uma rede de 28 milhões de arestas únicas conectando 5,5 milhões de vértices, representando uma densidade de $9,4e-7$. O coeficiente de clusterização global é de $9,6e-4$, portanto, havendo um vizinho em comum o nó tem 1000 vezes mais chance de estar conectado a outro nó do que quando ambos não compartilham conexões. Aproximadamente 4,5% dos nós pertencem a componente fortemente conexa, ou seja, sub-grafo direcional em que existe um caminho entre todos os vértices. Enquanto 97% dos nós pertencem a componente fracamente conexa, que leva em consideração o sub-grafo não direcional. Observa-se que a rede inteira é formada praticamente por uma única componente gigante enquanto os outros 3% dos usuários estão distribuídos em um grande volume de pequenas componentes.

A Figura 6.1 mostra as distribuições de grau dos vértices. Analisa-se que tanto a distribuição de grau de entrada quanto a de saída tem caudas longas apesar da curva da distribuição de grau de entrada ser significativamente mais extensa. Ressalta-se também que mais que 80% dos nós tem grau de entrada 0, ou seja, são usuários que não receberam nenhum retweet. A Figura 6.2 mostra a distância entre os nós da rede, que tem média 8,8. Conclui-se portanto que dentre os dados captados formou-se uma rede com número significativo de usuários e possui características comumente observadas em redes como a formação de *hubs*, pequena distância média e grande componente conexa.

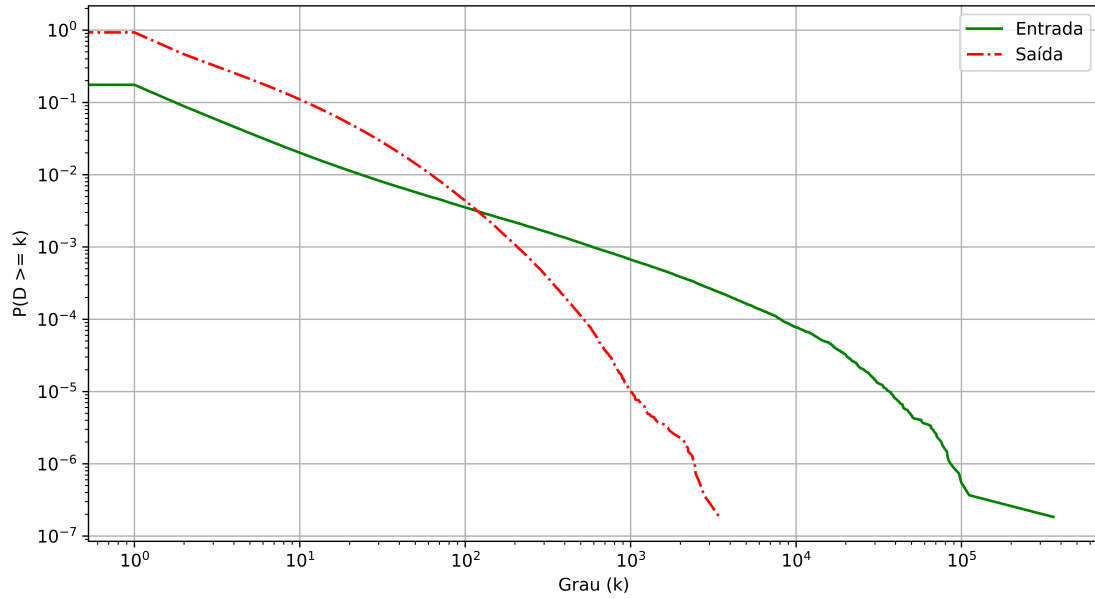


Figura 6.1: Gráfico da *CCDF* (*complementary cumulative distribution function*) dos graus de entrada e saída da rede de usuários. O eixo horizontal representa o grau k enquanto o eixo vertical demonstra a probabilidade de um vertice ter grau pelo menos k .

Uma vez finalizada a etapa de coleta dos dados, o processo de anotação e a formação da rede de usuários, se prosseguiu para a etapa do classificador de sentimento textuais. Esses classificadores serão usados como base de comparação para avaliar a eficiência dos classificadores multi-modais.

6.1 Classificadores textuais não Neurais

Os primeiros classificadores textuais treinados foram os por Naive Bayes e SVM. Nesse caso, ambos usaram a representação Bag-of-Words. O vocabulário formado a partir da base coletada é formado por um 127 mil palavras. O treinamento do algoritmo de Naive Bayes multinomial foi feito utilizando validação cruzada por

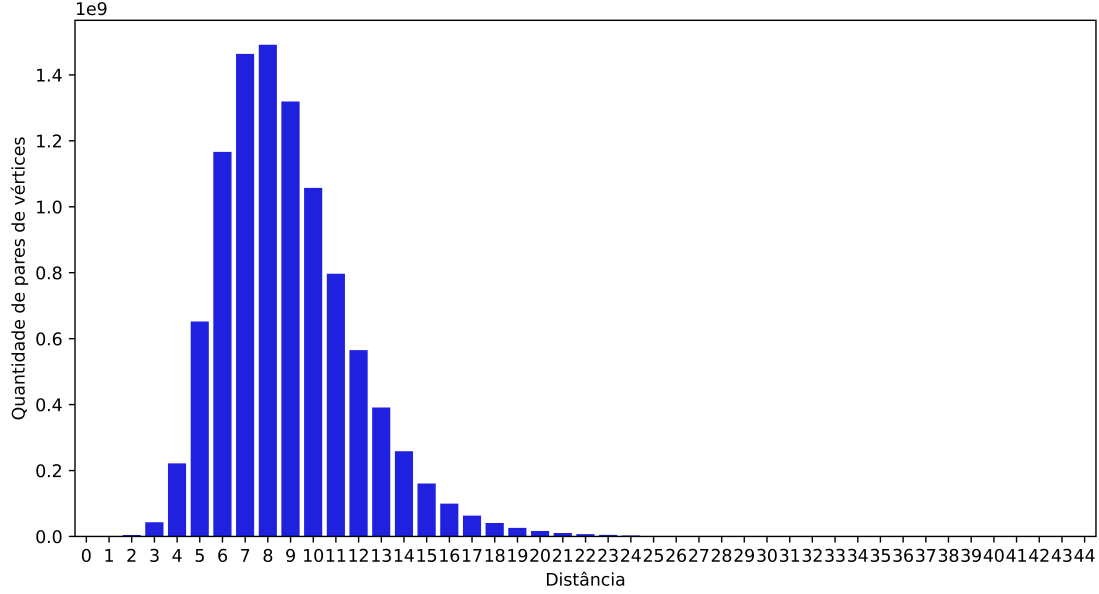


Figura 6.2: Distância entre pares de nós da rede de usuários.

K-partições, no total de 10 partições, sob os dados de treinamento anotados com supervisão distante. O parâmetro de suavização foi escolhido por *grid search*, de maneira a maximizar a área sob a curva ROC. A Figura 6.3 mostra a resposta do modelo para os diferentes fatores de suavização testados. O hiperparâmetro selecionado foi o de valor 0,244, que obteve AUC de $0,862 \pm 0,002$. Uma vez selecionado o melhor modelo na base de treino, o mesmo foi avaliado com os dados da base de teste, que foram anotados manualmente. A Figura 6.4 apresenta a curva ROC dos dados de treino. Observa-se que o mesmo modelo ao ser aplicado aos dados de teste tem resultado de 0,801 de área sob curva ROC. Vê-se que há uma diferença de performance entre as diferentes bases, que sobressai a dispersão observada nas partições do treinamento. Concluindo a análise do modelo de Naive Bayes, utilizou-se os dados de treino para encontrar o limiar de classificação que maximiza o índice SP. O limiar encontrado teve o valor de 0,88 e a Figura 6.5 apresenta a matriz confusão correspondente a esse valor.

Em seguida, testou-se o classificador por SVM, modelo que obteve melhor resultados nos testes feitos por GO *et al.* [1]. O vocabulário utilizado para codificação one-hot foi o mesmo aplicado ao Naive Bayes. Também se aplicou validação cruzada com 10 partições para determinar o melhor parâmetro de regularização L_2 . A Figura 6.6 apresenta os valores de AUC obtidos nos dados de treino para o intervalo de parâmetro testado. Observamos que o valor máximo obtido de AUC nos dados de treino foi $0,904 \pm 0,001$, para o fator de regularização $7e^{-05}$. Porém, ao aplicar o modelo aos dados de teste, a performance do modelo cai consideravelmente, como vemos na Figura 6.7, com 0,647 de área da curva ROC. O threshold de maior

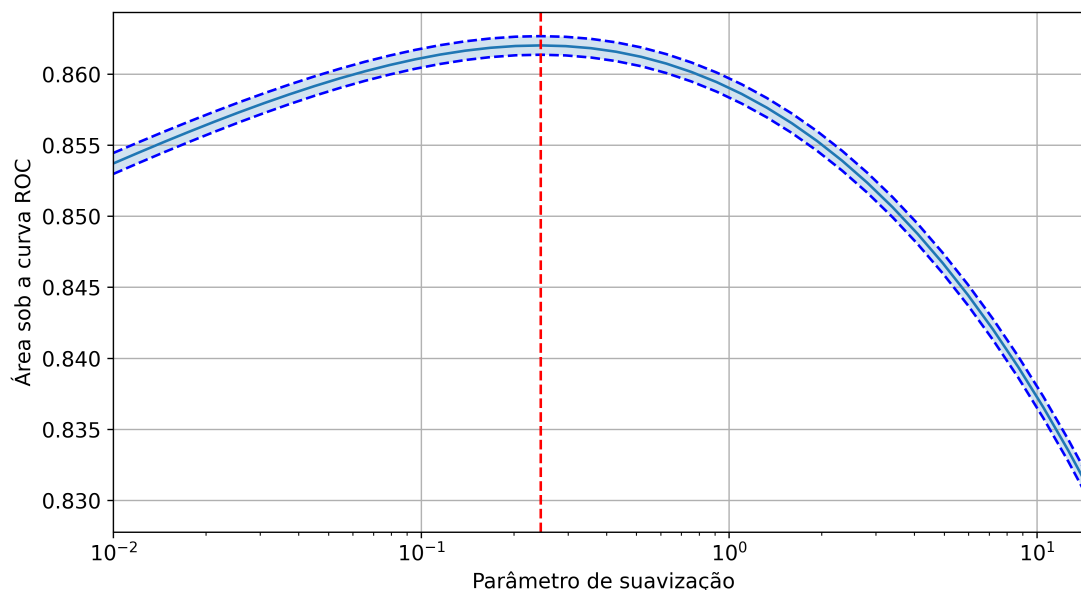


Figura 6.3: Resposta do modelo de Naive Bayes a variação do parâmetro de suavização. A linha cheia representa a média dos valores das 10 partições enquanto a área em azul, limitada pela linha rachurada, representa um desvio padrão. A linha vertical vermelha ressalta o parâmetro de maior média.

índice SP foi 0,776, obtendo a matriz confusão apresentada na Figura 6.8. A matriz confusão mostra que o modelo consegue distinguir elementos da classe positiva, entretanto, ao ser aplicado em frases negativas o mesmo não conseguiu performance melhor que escolha aleatória. Observa-se também que apesar do modelo SVM superar as métricas do modelo de Naive Bayes nos dados de treino, SVM generaliza significativamente menos quando aplicado ao banco de dados de teste. Apresentando assim uma grande discrepância entre a performance da base anotada por supervisão distante e a classificada manualmente.

6.2 Classificadores textuais por Redes Neurais

Realizou-se o treino do *Word2Vec* para representação do texto com os parâmetros apresentados na Seção 5.3.1. O treinamento foi executado até a convergência da função custo.

Com o *Word2Vec* treinado, foi realizado o treino dos classificadores de redes neurais convolucionais. Algumas das curvas de treinamento são demonstradas na Figura 6.9. Observamos que, normalmente, após a terceira época a função custo de treinamento apresenta uma estabilidade. Devido ao *early stopping* cada treinamento acaba em épocas diferentes. Como aplicamos *dropout* e regularização durante o treinamento temos que a entropia cruzada de treinamento é mais alta do que a

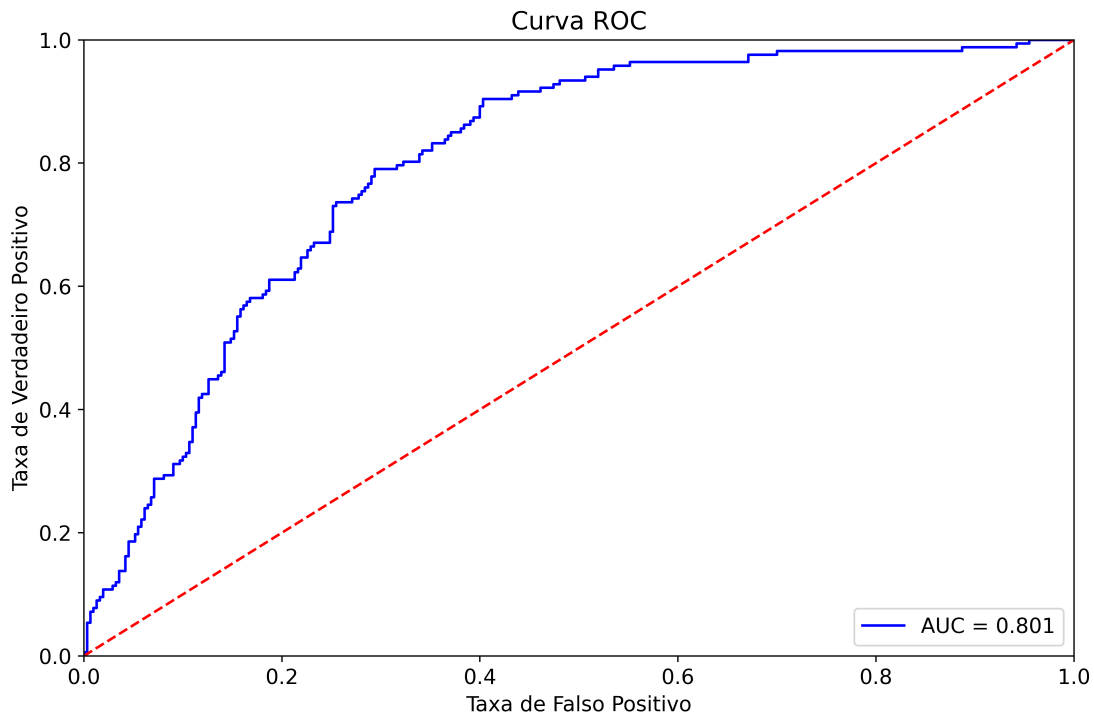


Figura 6.4: Curva ROC do modelo de Naive Bayes aplicado aos dados de teste.

de validação. Ressaltamos que os treinamentos convergiram para valores parecidos, não oscilando entre diferentes mínimos locais. A Figura 6.10 apresenta uma amostra dos 10 treinamentos realizados da rede LSTM. Vemos que no caso da LSTM, foram precisos 30 a 50 épocas de treino para conversão do modelo. Observamos também que assim como as redes convolucionais, não houve grande variação entre os mínimos encontrados durante o treinamento das redes.

6.3 Classificadores Multimodais

Seguimos para a incorporação da informação dos grafos de usuários nos modelos. Como descrito na seção 5.3.2, o grafo foi reduzido a sua componente principal para reduzir os requisitos de memória necessários para o treinamento dos modelos. O aprendizado foi realizado em duas etapas: aquisição de *embeddings* a partir de modelagem não-supervisionada do grafo; posteriormente foi realizado o treinamento em conjunto com os modelos textuais treinados na anteriormente.

As redes conjuntas compostas pelas camadas concatenadas da representação textual do *tweet* com a representação do grafo do usuário, seguida de 2 camadas classificatórias de tamanhos repectivamente de 100 e 32 neurônios completamente conectados. Os pesos das representações do usuário e textuais continuaram a sofrer *fine-tunning* durante a otimização do modelo.



Figura 6.5: Matrix confusão do modelo de Naive Bayes.

Começando pelo treinamento da representação por *Locally Linear Embedding*. Pelo algoritmo depender da matriz de adjacência, que tem custo quadrático em memória, foi necessário limitar o número de usuários da rede. Foram selecionados aleatoriamente 10 mil usuários dentro da sub-grafo de usuários da maior componente conectada para o treinamento do algoritmo. Foram feitas 10 realizações do treinamento, cada uma com seu próprio conjunto de nós selecionados. O modelo foi treinado para extrair uma representação de dimensão 128. Para melhor comparar os diferentes métodos esse número será mantido pelos outros métodos. Os usuários que não foram selecionados no sorteio do treinamento tiveram sua representação no vetor zero.

Posteriormente foram treinados o algoritmo *Node2Vec*. Mantiveram-se as 10 realizações do treinamento e o tamanho do vetor de representação. Como este algoritmo não depende da matriz de adjacência, foram realizados os treinamentos considerando todos os usuários presentes na maior componente. Neste algoritmo consideramos o grafo como não-direcionado e removemos as arestas de um nó a si mesmo. Os parâmetros de treinamento do algoritmo foram baseados no proposto por GROVER e LESKOVEC [91]. Os valores de parâmetro p e q de amostragem foram definidos em 1. Os passeios aleatórios para definição dos contextos tiveram 80 passos de interação e foram feitas 10 realizações de passeio por vértice. Os pesos das arestas não foram considerados durante a realização dos passeios aleatórios.

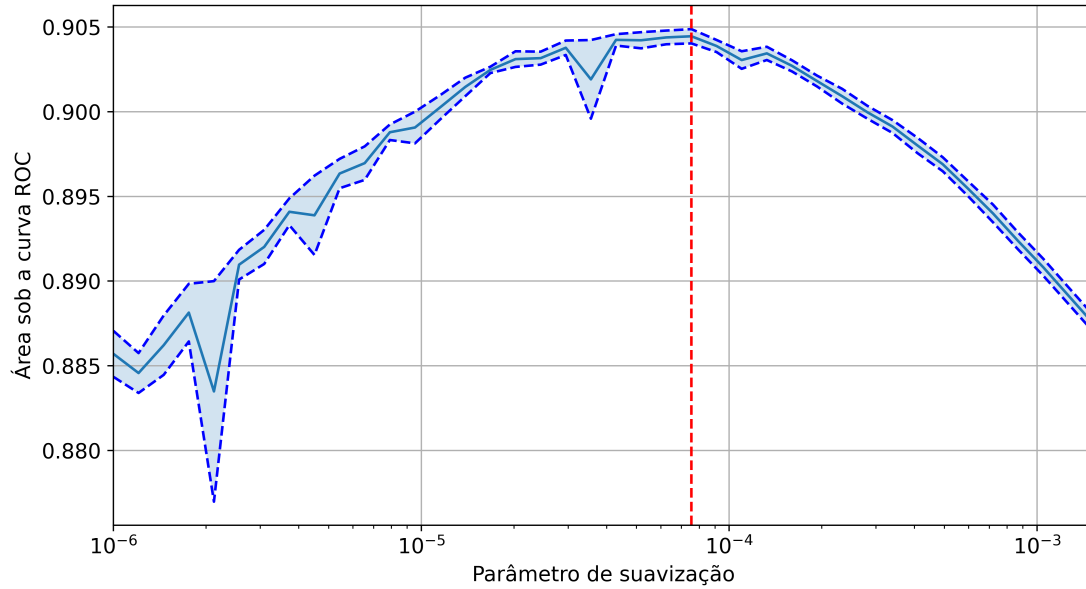


Figura 6.6: Resposta do modelo de SVM a variação do parâmetro de regularização. A linha cheia representa a média dos valores das 10 partições enquanto a área em azul, limitada pela linha rachurada, representa um desvio padrão. A linha vertical vermelha ressalta o parâmetro de maior média.

Utilizou-se janelas de contexto de tamanho 10 e treinou-se a otimização dos pesos por 3 épocas.

Por fim, treinou-se a rede convolucional de grafos (GCN). Assim como no caso do *Node2Vec* também se utilizou todo o subgrafo presente na principal componente. Assim como nos modelos anteriores continuou-se com as 10 inicializações e o tamanho da representação dos nós em 128 dimensões. Pelo GCN depender de atributos do vértices para o treinamento da representação, utilizou-se um atributo obtido pelo próprio grafo, o log do grau do nó. Essa escolha se deu pois não necessita de extrações adicionais de dados. Outros atributos dos usuários poderiam ter sido incorporados caso houvesse a disponibilidade desses dados como: número de seguidores, localização, frequência de uso, etc. O modelo foi treinado até a finalização por *early stopping*.

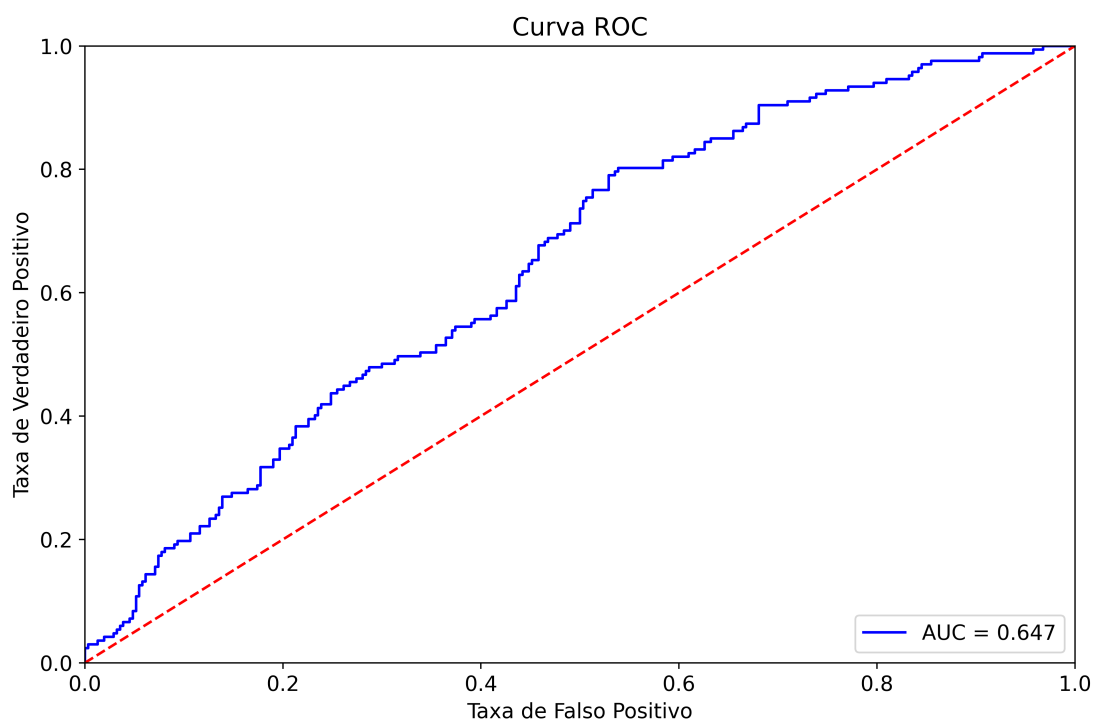


Figura 6.7: Curva ROC do modelo de SVM aplicado aos dados de teste.

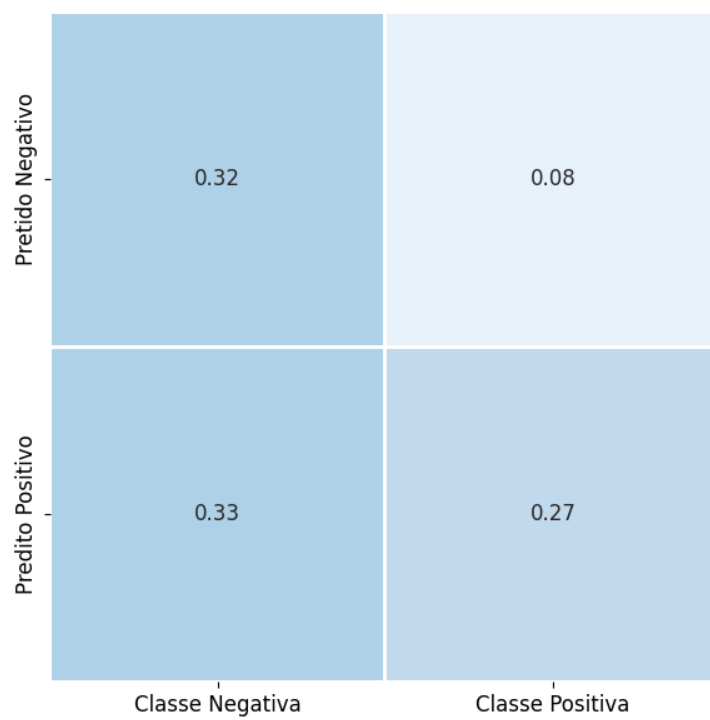


Figura 6.8: Matrix confusão do modelo de SVM.

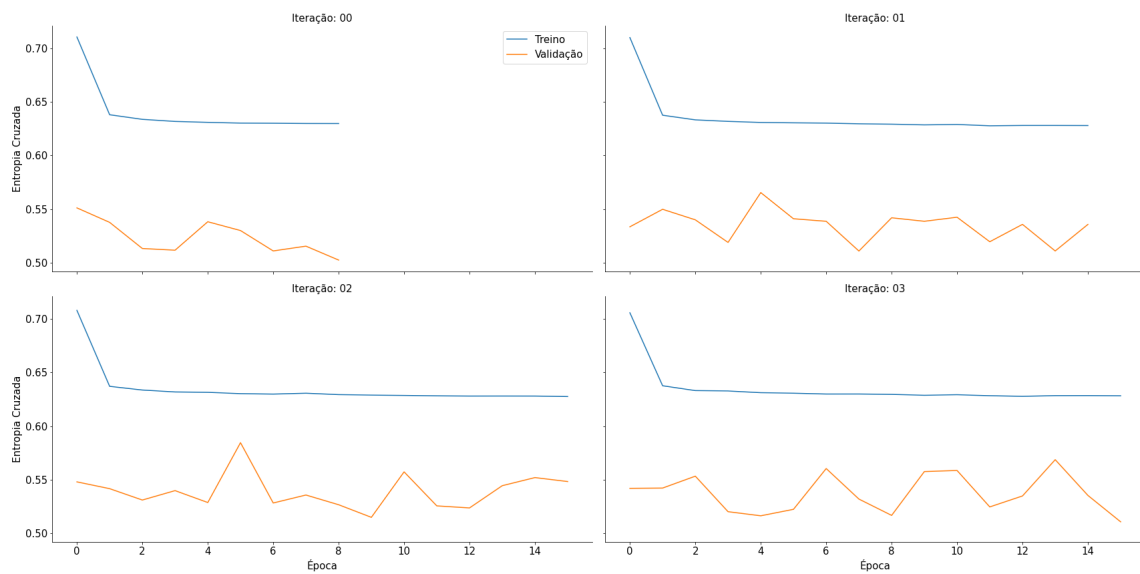


Figura 6.9: Curvas de treinamento de classificadores por redes neurais convolucionais.

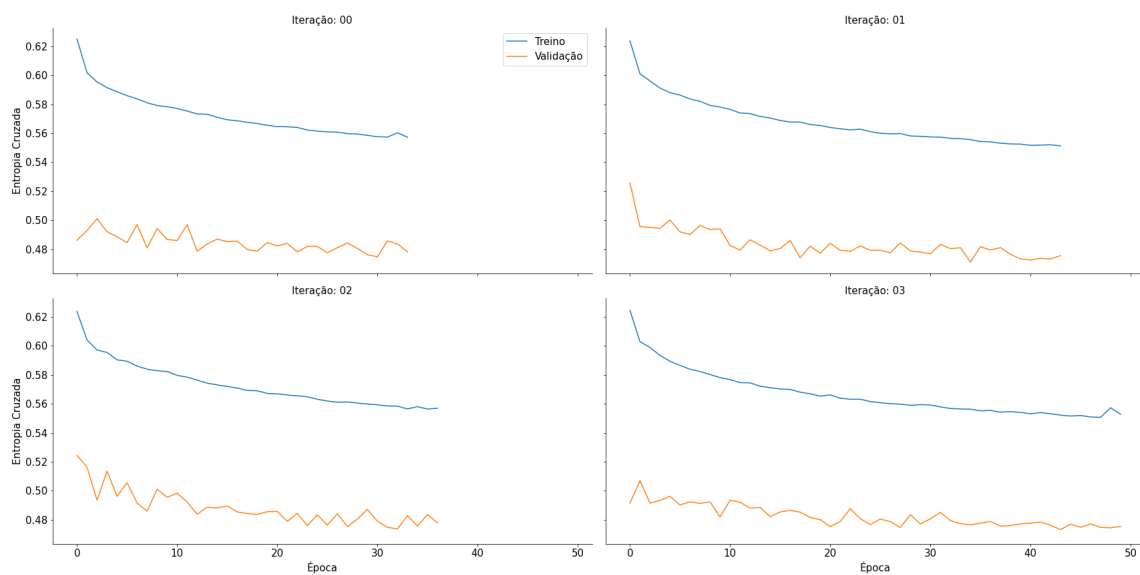


Figura 6.10: Curvas de treinamento de classificadores por redes neurais LSTM.

Capítulo 7

Conclusões

Referências Bibliográficas

- [1] GO, A., BHAYANI, R., HUANG, L. “Twitter sentiment classification using distant supervision”, *CS224N Project Report, Stanford*, v. 1, n. 12, pp. 2009, 2009.
- [2] SOCIAL, W. A. “Digital in 2019: Global Overview”. <https://wearesocial.com/global-digital-report-2019>. acessado em 3 de Junho de 2019.
- [3] TURING, A. M. “Computing machinery and intelligence”, *Mind*, v. 59, n. 236, pp. 433–460, 1950.
- [4] LIU, B. *Opinions, Sentiment, and Emotion in Text*. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge University Press, 2015. ISBN: 9781107017894. Disponível em: <<https://books.google.com.br/books?id=6IdsCQAAQBAJ>>.
- [5] CAMBRIA, E. “Affective computing and sentiment analysis”, *IEEE Intelligent Systems*, v. 31, n. 2, pp. 102–107, 2016.
- [6] TABOADA, M., BROOKE, J., TOFILOSKI, M., et al. “Lexicon-based methods for sentiment analysis”, *Computational linguistics*, v. 37, n. 2, pp. 267–307, 2011.
- [7] DAS, S. R., CHEN, M. Y. “Yahoo! for Amazon: Sentiment extraction from small talk on the web”, *Management science*, v. 53, n. 9, pp. 1375–1388, 2007.
- [8] RILOFF, E., WIEBE, J., PHILLIPS, W. “Exploiting subjectivity classification to improve information extraction”. In: *AAAI*, pp. 1106–1111, 2005.
- [9] SOCHER, R., PENNINGTON, J., HUANG, E. H., et al. “Semi-supervised recursive autoencoders for predicting sentiment distributions”. In: *Proceedings of the conference on empirical methods in natural language processing*, pp. 151–161. Association for Computational Linguistics, 2011.

- [10] SOCHER, R., PERELYGIN, A., WU, J., et al. “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.
- [11] NASUKAWA, T., YI, J. “Sentiment analysis: Capturing favorability using natural language processing”. In: *Proceedings of the 2nd international conference on Knowledge capture*, pp. 70–77. ACM, 2003.
- [12] SNYDER, B., BARZILAY, R. “Multiple aspect ranking using the good grief algorithm”. In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 300–307, 2007.
- [13] TWITTER. “Q2-2019 Letter to Shareholders”. https://s22.q4cdn.com/826641620/files/doc_financials/2019/q2/Q2-2019-Shareholder-Letter.pdf. acessado em 07 de Novembro de 2019.
- [14] PANG, B., LEE, L., VAITHYANATHAN, S. “Thumbs up?: sentiment classification using machine learning techniques”. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86. Association for Computational Linguistics, 2002.
- [15] READ, J. “Using emoticons to reduce dependency in machine learning techniques for sentiment classification”. In: *Proceedings of the ACL student research workshop*, pp. 43–48. Association for Computational Linguistics, 2005.
- [16] WANG, S., MANNING, C. D. “Baselines and bigrams: Simple, good sentiment and topic classification”. In: *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pp. 90–94. Association for Computational Linguistics, 2012.
- [17] PALTOGLOU, G., THELWALL, M. “A study of information retrieval weighting schemes for sentiment analysis”. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 1386–1395. Association for Computational Linguistics, 2010.
- [18] MIKOLOV, T., SUTSKEVER, I., CHEN, K., et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*, pp. 3111–3119, 2013.

- [19] 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)*, pp. 1532–1543, 2014.
- [20] PETERS, M. E., NEUMANN, M., IYYER, M., et al. “Deep contextualized word representations”. In: *Proc. of NAACL*, 2018.
- [21] DEVLIN, J., CHANG, M.-W., LEE, K., et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”, *arXiv preprint arXiv:1810.04805*, 2018.
- [22] LECUN, Y., BENGIO, Y., HINTON, G. “Deep learning”, *nature*, v. 521, n. 7553, pp. 436–444, 2015.
- [23] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [24] HINTON, G., DENG, L., YU, D., et al. “Deep neural networks for acoustic modeling in speech recognition”, *IEEE Signal processing magazine*, v. 29, 2012.
- [25] ESTEVA, A., KUPREL, B., NOVOA, R. A., et al. “Dermatologist-level classification of skin cancer with deep neural networks”, *Nature*, v. 542, n. 7639, pp. 115, 2017.
- [26] VASWANI, A., SHAZEER, N., PARMAR, N., et al. “Attention is all you need”. In: *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- [27] GE, T., WEI, F., ZHOU, M. “Reaching human-level performance in automatic grammatical error correction: An empirical study”, *arXiv preprint arXiv:1807.01270*, 2018.
- [28] AKBIK, A., BLYTHE, D., VOLLGRAF, R. “Contextual string embeddings for sequence labeling”. In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, 2018.
- [29] WU, Y., HU, B. “Learning to extract coherent summary via deep reinforcement learning”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [30] KIM, Y. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A*

- meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1746–1751, 2014. Disponível em: <<http://aclweb.org/anthology/D/D14/D14-1181.pdf>>.
- [31] ZHOU, P., QI, Z., ZHENG, S., et al. “Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling”, *arXiv preprint arXiv:1611.06639*, 2016.
 - [32] ALBERT, R., BARABÁSI, A.-L. “Statistical mechanics of complex networks”, *Rev. Mod. Phys.*, v. 74, pp. 47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47. Disponível em: <<https://link.aps.org/doi/10.1103/RevModPhys.74.47>>.
 - [33] BARABÁSI, A.-L., GULBAHCE, N., LOSCALZO, J. “Network medicine: a network-based approach to human disease”, *Nature reviews genetics*, v. 12, n. 1, pp. 56, 2011.
 - [34] HUFNAGEL, L., BROCKMANN, D., GEISEL, T. “Forecast and control of epidemics in a globalized world”, *Proceedings of the National Academy of Sciences*, v. 101, n. 42, pp. 15124–15129, 2004.
 - [35] ALBERT, R. “Attack and error tolerance in complex networks”, *Nature*, v. 406, pp. 387–482, 2000.
 - [36] COLLADON, A. F., REMONDI, E. “Using social network analysis to prevent money laundering”, *Expert Systems with Applications*, v. 67, pp. 49–58, 2017.
 - [37] RATKIEWICZ, J., CONOVER, M. D., MEISS, M., et al. “Detecting and tracking political abuse in social media”. In: *Fifth international AAAI conference on weblogs and social media*, 2011.
 - [38] VAROL, O., DAVIS, C. A., MENCZER, F., et al. “Feature engineering for social bot detection”, *Feature engineering for machine learning and data analytics*, p. 311, 2018.
 - [39] BACKSTROM, L., LESKOVEC, J. “Supervised random walks: predicting and recommending links in social networks”. In: *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 635–644. ACM, 2011.
 - [40] QIU, J., TANG, J., MA, H., et al. “Deepinf: Social influence prediction with deep learning”. In: *Proceedings of the 24th ACM SIGKDD Internatio-*

nal Conference on Knowledge Discovery & Data Mining, pp. 2110–2119. ACM, 2018.

- [41] KISS, T., STRUNK, J. “Unsupervised multilingual sentence boundary detection”, *Computational Linguistics*, v. 32, n. 4, pp. 485–525, 2006.
- [42] DAMERAU, F. J. “A technique for computer detection and correction of spelling errors”, *Communications of the ACM*, v. 7, n. 3, pp. 171–176, 1964.
- [43] NAVARRO, G. “A guided tour to approximate string matching”, *ACM computing surveys (CSUR)*, v. 33, n. 1, pp. 31–88, 2001.
- [44] PORTER, M. F. “An algorithm for suffix stripping”, *program*, v. 14, n. 3, pp. 130–137, 1980.
- [45] LO, R. T.-W., HE, B., OUNIS, I. “Automatically building a stopwords list for an information retrieval system”. In: *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, v. 5, pp. 17–24, 2005.
- [46] SAIF, H., FERNÁNDEZ, M., HE, Y., et al. “On stopwords, filtering and data sparsity for sentiment analysis of twitter”, 2014.
- [47] MANNING, C., RAGHAVAN, P., SCHÜTZE, H. “Introduction to information retrieval”, *Natural Language Engineering*, v. 16, n. 1, pp. 100–103, 2010.
- [48] POWERS, D. M. “Applications and explanations of Zipf’s law”. In: *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, pp. 151–160. Association for Computational Linguistics, 1998.
- [49] SALTON, G., BUCKLEY, C. “Term-weighting approaches in automatic text retrieval”, *Information processing & management*, v. 24, n. 5, pp. 513–523, 1988.
- [50] BOJANOWSKI, P., GRAVE, E., JOULIN, A., et al. “Enriching word vectors with subword information”, *Transactions of the Association for Computational Linguistics*, v. 5, pp. 135–146, 2017.
- [51] CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”, *arXiv preprint arXiv:1406.1078*, 2014.

- [52] PETERS, M. E., AMMAR, W., BHAGAVATULA, C., et al. “Semi-supervised sequence tagging with bidirectional language models”, *arXiv preprint arXiv:1705.00108*, 2017.
- [53] MCCANN, B., BRADBURY, J., XIONG, C., et al. “Learned in translation: Contextualized word vectors”. In: *Advances in Neural Information Processing Systems*, pp. 6294–6305, 2017.
- [54] HOCHREITER, S., SCHMIDHUBER, J. “Long short-term memory”, *Neural computation*, v. 9, n. 8, pp. 1735–1780, 1997.
- [55] HASHIMOTO, K., XIONG, C., TSURUOKA, Y., et al. “A joint many-task model: Growing a neural network for multiple nlp tasks”, *arXiv preprint arXiv:1611.01587*, 2016.
- [56] SØGAARD, A., GOLDBERG, Y. “Deep multi-task learning with low level tasks supervised at lower layers”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 231–235, 2016.
- [57] BELINKOV, Y., DURRANI, N., DALVI, F., et al. “What do neural machine translation models learn about morphology?” *arXiv preprint arXiv:1704.03471*, 2017.
- [58] MELAMUD, O., GOLDBERGER, J., DAGAN, I. “context2vec: Learning generic context embedding with bidirectional lstm”. In: *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pp. 51–61, 2016.
- [59] BAHDANAU, D., CHO, K., BENGIO, Y. “Neural machine translation by jointly learning to align and translate”, *arXiv preprint arXiv:1409.0473*, 2014.
- [60] LUONG, M.-T., PHAM, H., MANNING, C. D. “Effective approaches to attention-based neural machine translation”, *arXiv preprint arXiv:1508.04025*, 2015.
- [61] JOSHI, C. “Transformers are Graph Neural Networks”. <https://graphdeeplearning.github.io/post/transformers-are-gnns/>. acessado em 1 de Março 2020.
- [62] RADFORD, A., NARASIMHAN, K., SALIMANS, T., et al. “Improving language understanding by generative pre-training”, Disponível em: <<https://s3-us-west-2.amazonaws.com/>

openai-assets/research-covers/language-unsupervised/
language_understanding_paper.pdf>.

- [63] STONE, P. J., DUNPHY, D. C., SMITH, M. S. “The general inquirer: A computer approach to content analysis.” 1966.
- [64] TONG, R. M. “An operational system for detecting and tracking opinions in on-line discussion”. In: *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*, v. 1, 2001.
- [65] HU, M., LIU, B. “Mining and summarizing customer reviews”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177. ACM, 2004.
- [66] MILLER, G. A., BECKWITH, R., FELLBAUM, C., et al. “Introduction to WordNet: An on-line lexical database”, *International journal of lexicography*, v. 3, n. 4, pp. 235–244, 1990.
- [67] BLAIR-GOLDENSOHN, S., HANNAN, K., MCDONALD, R., et al. “Building a sentiment summarizer for local service reviews”. In: *Proceedings of WWW-2008 workshop on NLP in the Information Explosion Era*, 2008.
- [68] RAO, D., RAVICHANDRAN, D. “Semi-supervised polarity lexicon induction”. In: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 675–682. Association for Computational Linguistics, 2009.
- [69] HASSAN, A., RADEV, D. “Identifying text polarity using random walks”. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 395–403. Association for Computational Linguistics, 2010.
- [70] TURNEY, P. D. “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews”. In: *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 417–424. Association for Computational Linguistics, 2002.
- [71] SCHÜTZE, H., MANNING, C. D., RAGHAVAN, P. “Introduction to information retrieval”. In: *Proceedings of the international communication of association for computing machinery conference*, v. 4, 2008.
- [72] VANDERPLAS, J. https://github.com/jakevdp/sklearn_pycon2015/blob/master/notebooks/03.1-Classification-SVMs.ipynb. acessado em 26 de Fevereiro 2020.

- [73] CORTES, C., VAPNIK, V. “Support-vector networks”, *Machine learning*, v. 20, n. 3, pp. 273–297, 1995.
- [74] JOACHIMS, T. “Text categorization with support vector machines: Learning with many relevant features”. In: *European conference on machine learning*, pp. 137–142. Springer, 1998.
- [75] HORNIK, K., STINCHCOMBE, M., WHITE, H. “Multilayer feedforward networks are universal approximators”, *Neural networks*, v. 2, n. 5, pp. 359–366, 1989.
- [76] WERBOS, P. J. “Applications of advances in nonlinear sensitivity analysis”. In: *System modeling and optimization*, Springer, pp. 762–770, 1982.
- [77] HOCHREITER, S. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 6, n. 02, pp. 107–116, 1998.
- [78] HINTON, G. E., OSINDERO, S., TEH, Y.-W. “A fast learning algorithm for deep belief nets”, *Neural computation*, v. 18, n. 7, pp. 1527–1554, 2006.
- [79] LECUN, Y., BENGIO, Y., OTHERS. “Convolutional networks for images, speech, and time series”, *The handbook of brain theory and neural networks*, v. 3361, n. 10, 1995.
- [80] KALCHBRENNER, N., GREFFENSTETTE, E., BLUNSOM, P. “A convolutional neural network for modelling sentences”, *arXiv preprint arXiv:1404.2188*, 2014.
- [81] YIH, W.-T., HE, X., MEEK, C. “Semantic parsing for single-relation question answering”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 643–648, 2014.
- [82] WILLIAMS, R. J., ZIPSER, D. “Gradient-based learning algorithms for recurrent”, *Backpropagation: Theory, architectures, and applications*, v. 433, 1995.
- [83] TAI, K. S., SOCHER, R., MANNING, C. D. “Improved semantic representations from tree-structured long short-term memory networks”, *arXiv preprint arXiv:1503.00075*, 2015.
- [84] SEN, P., NAMATA, G., BILGIC, M., et al. “Collective classification in network data”, *AI magazine*, v. 29, n. 3, pp. 93–93, 2008.

- [85] SHERVASHIDZE, N., SCHWEITZER, P., VAN LEEUWEN, E. J., et al. “Weisfeiler-lehman graph kernels.” *Journal of Machine Learning Research*, v. 12, n. 9, 2011.
- [86] GOYAL, P., FERRARA, E. “Graph embedding techniques, applications, and performance: A survey”, *Knowledge-Based Systems*, v. 151, pp. 78–94, 2018.
- [87] ROWEIS, S. T., SAUL, L. K. “Nonlinear dimensionality reduction by locally linear embedding”, *science*, v. 290, n. 5500, pp. 2323–2326, 2000.
- [88] BELKIN, M., NIYOI, P. “Laplacian eigenmaps and spectral techniques for embedding and clustering”. In: *Advances in neural information processing systems*, pp. 585–591, 2002.
- [89] OU, M., CUI, P., PEI, J., et al. “Asymmetric transitivity preserving graph embedding”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.
- [90] PEROZZI, B., AL-RFOU, R., SKIENA, S. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.
- [91] GROVER, A., LESKOVEC, J. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.
- [92] WANG, D., CUI, P., ZHU, W. “Structural deep network embedding”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234, 2016.
- [93] CAO, S., LU, W., XU, Q. “Deep neural networks for learning graph representations”. In: *Thirtieth AAAI conference on artificial intelligence*, 2016.
- [94] SCARSELLI, F., GORI, M., TSOI, A. C., et al. “The graph neural network model”, *IEEE Transactions on Neural Networks*, v. 20, n. 1, pp. 61–80, 2008.
- [95] YOU, J., YING, R., REN, X., et al. “GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models”. In: *ICML*, 2018.
- [96] ZHANG, Z., CUI, P., ZHU, W. “Deep learning on graphs: A survey”, *IEEE Transactions on Knowledge and Data Engineering*, 2020.

- [97] BRUNA, J., ZAREMBA, W., SZLAM, A., et al. “Spectral networks and locally connected networks on graphs”, *arXiv preprint arXiv:1312.6203*, 2013.
- [98] DEFFERRARD, M., BRESSON, X., VANDERGHEYNST, P. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*, pp. 3844–3852, 2016.
- [99] KIPF, T. N., WELLING, M. “Semi-supervised classification with graph convolutional networks”, *arXiv preprint arXiv:1609.02907*, 2016.
- [100] HAMILTON, W., YING, Z., LESKOVEC, J. “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems*, pp. 1024–1034, 2017.
- [101] VELIČKOVIĆ, P., FEDUS, W., HAMILTON, W. L., et al. “Deep Graph Infomax”. In: *International Conference on Learning Representations*, 2019. Disponível em: <<https://openreview.net/forum?id=rklz9iAcKQ>>.
- [102] CIODARO, T., DEVA, D., DE SEIXAS, J., et al. “Online particle detection with neural networks based on topological calorimetry information”. In: *Journal of physics: conference series*, v. 368, p. 012030. IOP Publishing, 2012.
- [103] SUYKENS, J. A., VANDEWALLE, J. “Least squares support vector machine classifiers”, *Neural processing letters*, v. 9, n. 3, pp. 293–300, 1999.
- [104] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.
- [105] ŘEHŮŘEK, R., SOJKA, P. “Software Framework for Topic Modelling with Large Corpora”. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta, 2010. ELRA. <http://is.muni.cz/publication/884893/en>.